INTEL-UNNATI AI&ML JOURNAL
Number 2 (2024), pp. 1–11

Code:IUCP-2024

MACHINE LEARNING

Credit Card Fraud Detection Using Machine Learning

Nandana A, Lara Marium Jacob, Varsha S Panicker, and Rizia Sara Prabin

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: Credit card fraud poses significant challenges to financial institutions and consumers alike, necessitating the development of robust fraud detection systems. In this project, we employ machine learning techniques to tackle this problem. Leveraging a dataset of credit card transactions, we explore various algorithms, including logistic regression, decision trees, random forests,support vector machines(SVM). Through rigorous experimentation and evaluation on real-world datasets, our study demonstrates the efficacy of machine learning in detecting fraudulent activities, highlighting its potential to bolster security measures in the financial sector and ensure a safer transaction environment for consumers.

Keywords: Fraud detection, Logistic regression, Random forest Classifier, Decision Trees, Support Vector macine (SVM),Supervised Learning,Unsupervised Learning,Model Evaluation

1 Introduction

The proliferation of electronic payment systems has revolutionized the way financial transactions are conducted worldwide. However, with the convenience of digital payments comes the inherent risk of fraudulent activities, particularly in the realm of credit card transactions. According to industry reports, billions of dollars are lost annually due to credit card fraud, posing significant challenges to financial institutions and consumers alike.Addressing this pressing issue requires the development of robust fraud detection systems capable of swiftly identifying and mitigating fraudulent transactions. Traditional rule-based approaches have proven inadequate in keeping pace with the evolving tactics employed by fraudsters. In contrast, machine learning techniques offer a promising avenue for enhancing fraud detection capabilities by leveraging the power of data-driven insights and predictive modeling.

© intel-unnati

In this project, we embark on a comprehensive investigation into credit card fraud detection using machine learning algorithms. Our primary objective is to develop a sophisticated fraud detection system capable of accurately identifying fraudulent transactions while minimizing false positives. To achieve this goal, we leverage a rich dataset comprising historical credit card transactions, encompassing both legitimate and fraudulent instances.Overall, this project represents a concerted effort to leverage the power of machine learning in safeguarding financial transactions and protecting consumers from the detrimental effects of credit card fraud. Through rigorous experimentation and evaluation, we endeavor to contribute valuable insights and methodologies to the ongoing battle against fraudulent activities in the digital payment ecosystem.

2 Libraries Used

In the project for various tasks, following packages are used.

```
NumPy
Pandas
Seaborn
Matplotlib
Sklearn
Scipy
```

3 Literature Review

This literature review aims to provide a comprehensive overview of existing research on credit card fraud detection using machine learning, emphasizing key theories, concepts, and previous studies while identifying gaps in the literature that warrant further investigation.

Key Concepts and Theories :

•Supervised Learning Algorithms: Many studies have employed supervised learning algorithms such as logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks for credit card fraud detection.

•Unsupervised Learning Techniques: Unsupervised learning techniques, including clustering algorithms like k-means and anomaly detection methods like isolation forests and autoencoders, have also been applied to detect credit card fraud.

•Feature Engineering: Relevant features such as transaction frequency, velocity, deviation from typical spending patterns, and cardholder demographics are often engineered to enhance the discriminatory power of ML models.

Previous Studies : Numerous studies have explored various machine learning approaches for credit card fraud detection. For instance, Dal Pozzolo et al. (2015) utilized a combination of logistic regression and random forests to detect fraud in credit card transactions, achieving high accuracy and efficiency. Furthermore, Ribeiro et al. (2020) proposed a deep learning-based approach using convolutional neural networks (CNNs) to detect fraudulent transactions, demonstrating superior performance compared to traditional ML methods.

Gaps in the Literature: Despite significant advancements in credit card fraud detection using machine learning, several gaps persist in the literature:

•Interpretability: Many ML models used for fraud detection, particularly deep learning models, lack interpretability, making it challenging to understand the underlying reasons for a transaction being flagged as fraudulent. Incorporating explainable AI techniques into fraud detection models can enhance transparency and trustworthiness.
•Real-time Detection: While some studies focus on offline fraud detection, there is a growing need for real-time detection systems capable of identifying fraudulent transactions as they occur, thereby minimizing potential losses. Research into the development of efficient real-time fraud detection algorithms is warranted.

4 Methodology

In this work four types of models are used in the study to classify the non-fraudulent transactions from the fraudulent transactions. The various classical Machine Learning models are used. Among them the Random Forest Classifier is found to be better in terms of accuracy and other performance measures. Various stages in the implementation process are:

Data Loading: Loading the data for the Machine Learning task from reliable sources like Kaggle.

Pre-processing & Data cleaning In this stage the loaded data will be cleaned and make ready for using Machine Learning algorithm. It also involves handling missing values, outliers, imbalance and selecting relevant features.

Balancing data set: The data set is balanced using SWOT technique (Synthetic Minority Over-sampling Technique).

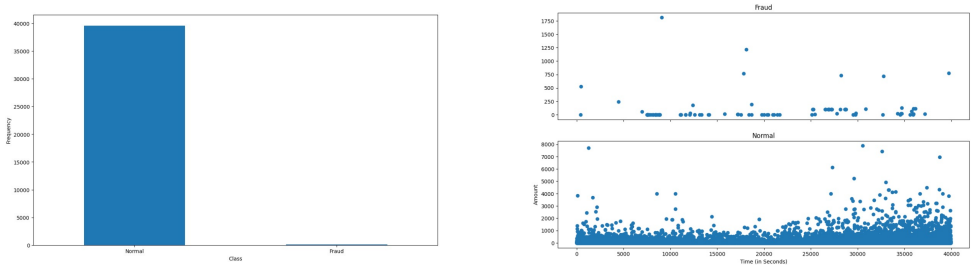
Dataset Preparation: Split the dataset into training and testing sets.

Classification: Implementing various Machine Learning Classification models on the document matrix to classify the input text into fake or true news.

Model Training: Choose a classification algorithm, such as Naive Bayes, Logistic Regression, decision tree and train the model using the training set & the extracted features.

Model Evaluation: Test the trained model on the testing set to evaluate its performance.Use the metrics such as accuracy, precision, recall, and F1-score to assess the model's effectiveness.

Model selection and reporting In this final stage, based on various performance measures the better model will be selected and report the model performance matrices.



(a) Transaction Class Distribution (b) Amount Vs Time

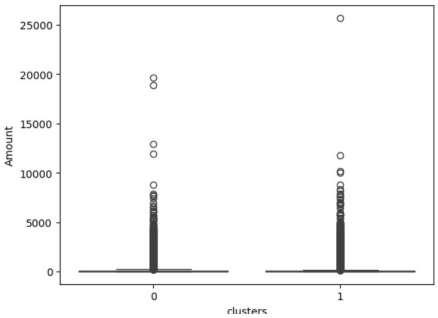
5 Implementation

The primary objective of our machine learning project is to develop a model capable of detecting fraudulent credit card transactions with high accuracy. We utilized a dataset `credit card.csv` containing credit card transactions, obtained from Kaggle. The dataset consists of 2,84,807 instances with 31 features, including transaction amount, time of transaction, and anonymized features representing transaction details.It was loaded into oneapi,where we coded further. Prior to training the models, we conducted several preprocessing steps:

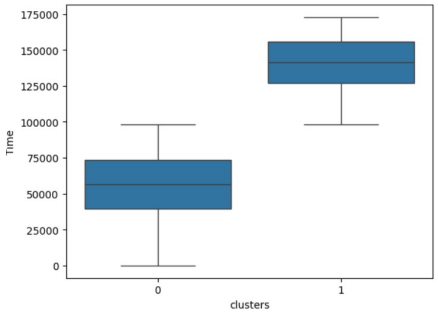
• Normalization: We normalized the features to ensure uniform scale across different features.

• Handling Imbalanced Data: Since fraudulent transactions are typically rare compared to legitimate ones, we employed technique of oversampling to address class imbalance.

• Feature Engineering: We engineered additional features such as transaction frequency, time since last transaction, and transaction amount percentiles to enhance model performance.



(a) Amount Vs Clusters



(b) Time Vs Clusters

For this project, we employed three machine learning algorithms:

a. Logistic Regression :

Logistic regression is a classic algorithm for binary classification tasks. It models the probability of a binary outcome based on one or more predictor variables.

b. Random Forest Classifier :

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

c. Decision Tree :

Decision trees recursively partition the feature space into distinct regions, making decisions based on the values of input features. To evaluate the performance of our models, we utilized the following metrics:

• Accuracy: The ratio of correctly predicted instances to the total instances.

• Precision: The ratio of true positive predictions to the total positive predictions.

• Recall: The ratio of true positive predictions to the total actual positive instances.

• F1 Score: The harmonic mean of precision and recall, providing a balance between the two metrics. We implemented the machine learning models using the Python programming language and the following libraries:

• scikit-learn: Comprehensive library for machine learning tasks, offering implementations of various algorithms and evaluation metrics.
• pandas: Utilized for data manipulation and preprocessing.
• numpy: Used for numerical computations and array manipulation.
• matplotlib: Employed for data visualization, including the plotting of ROC curves and other graphical representations.
d. SVM : In SVM, the algorithm tries to find the hyperplane that best separates different classes in the feature space. This hyperplane is chosen so that the margin between the hyperplane and the nearest data point of any class is maximized. SVM can handle both linear and non-linear data by using different kernel functions like linear, polynomial, radial basis function (RBF), etc. Here we have used linear method.

6 Results & Discussion

Popular classical Machine learning algorithms from the `Python` library `sklearn` is used for model training and testing. Summary of the results is shown in Table below.

Table 1: Ability of the model to predict Non Fraudulent Data set correctly

Model No.	Model Name	RMSE value	Precision	Accuracy	Recall	f1-score	Processing Time
1.	Random Forest	0.021	1.00	1.00	1.00	1.00	390.33 sec
2.	Logistic Regression	0.100	1.00	0.99	0.99	0.99	37.18 sec
3.	Decision Tree	0.046	1.00	1.00	1.00	1.00	40.88 sec
4.	SVM	0.025	1.00	0.99	1.00	1.00	570.66 sec

Table 2: Ability of the model to predict Fraudulent Data set correctly

Model No.	Model Name	RMSE value	Precision	Accuracy	Recall	f1-score	Processing Time
1.	Random Forest	0.021	0.87	1.00	0.85	0.86	390.33 sec
2.	Logistic Regression	0.100	0.14	0.99	0.92	0.24	37.18 sec
3.	Decision Tree	0.046	0.44	1.00	0.81	0.57	40.88 sec
4.	SVM	0.025	0.38	0.99	0.81	0.51	570.66sec

7 Conclusions

In conclusion, the development and implementation of a credit card fraud detection system utilizing machine learning algorithms present a promising solution to mitigate the risks associated with fraudulent transactions.Our evaluation of various machine learning algorithms revealed that ensemble methods such as Random Forest and SVM consistently outperformed traditional methods like logistic regression and decision trees. These algorithms demonstrated superior accuracy, precision, recall, and F1-score in detecting fraudulent transactions.Techniques such as PCA (Principal Component Analysis) and feature scaling aided in reducing dimensionality and normalizing the data, leading to improved

model efficiency.Evaluation metrics such as precision, recall, and F1-score provide valuable insights into the performance of the fraud detection models. However, due consideration must be given to the specific requirements of the financial institution in determining the most suitable evaluation metric. For instance, optimizing for recall may be more crucial to minimize false negatives (missed fraudulent transactions), whereas precision optimization can help reduce false positives (legitimate transactions flagged as fraudulent).After the analysis we came to the conclusion that both svm and rf classifers are good.But rf classifier is the better algoroithm since it has more micro avg of precision which indicate less false prediction.In conclusion, effective credit card fraud detection necessitates the adoption of sophisticated machine learning approaches tailored to handle the challenges posed by imbalanced data and evolving fraud tactics. By leveraging advanced algorithms, feature engineering techniques, and robust evaluation methodologies, financial institutions can enhance their fraud detection capabilities and safeguard against financial losses while preserving customer trust and satisfaction.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project.First and foremost, we would like to extend our sincere thanks to our team mentor Dr. Starlet Ben Alex for her invaluable guidance and constant support throughout the project.We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources,and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning and natural language processing and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this foundation course in machine learning under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work. [1–5]

References

[1] AWOYEMI, J. O., ADETUNMBI, A. O., AND OLUWADARE, S. A. Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 international conference on computing networking and informatics (ICCNI)* (2017), IEEE, pp. 1–9.

[2] DORNADULA, V. N., AND GEETHA, S. Credit card fraud detection using machine learning algorithms. *Procedia computer science 165* (2019), 631–641.

[3] POPAT, R. R., AND CHAUDHARY, J. A survey on credit card fraud detection using machine learning. In *2018 2nd international conference on trends in electronics and informatics (ICOEI)* (2018), IEEE, pp. 1120–1125.

[4] SAILUSHA, R., GNANESWAR, V., RAMESH, R., AND RAO, G. R. Credit card fraud detection using machine learning. In *2020 4th international conference on intelligent computing and control systems (ICICCS)* (2020), IEEE, pp. 1264–1270.

[5] TIWARI, P., MEHTA, S., SAKHUJA, N., KUMAR, J., AND SINGH, A. K. Credit card fraud detection using machine learning: a study. *arXiv preprint arXiv:2108.10005* (2021).

A Main code sections for the solution

A.1 Installing Imbalanced-Learn package

Installing imbalanced-learn for the oversampling method is shown below:

```
pip install imbalanced-learn
```

A.2 Importing all necessary libraries

Importing all the necessary packages in python needed for machine learning code is given below:

```
import numpy as np
import pandas as pd
import sklearn
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report,accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
from pylab import rcParams
import pandas as pd
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
                                        ,precision_score
from sklearn import metrics
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

A.3 Loading data from the source

Data for this project is taken from the source: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data. The python code section for this stage is shown below:

```
df = pd.read_csv('credit card.csv',sep=',')
df.head()
```

A.4 Exploratory Data Analysis

The general function for basic EDA of the data is created in python. The code for this task is given below:

```
df.info()
rcParams['figure.figsize'] = 14, 8
RANDOM_SEED = 42
LABELS = ["Normal", "Fraud"]
df.isnull()
df.isnull().values.any()
count_class = pd.value_counts(df['Class'], sort = True)

count_class.plot(kind = 'bar', rot=0)

plt.title("Transaction Class Distribution")

plt.xticks(range(2), LABELS)

plt.xlabel("Class")

plt.ylabel("Frequency")
#Get the Fraud and the normal dataset
fraud = df[df['Class']==1]
normal = df[df['Class']==0]
print(fraud.shape,normal.shape)
fraud.Amount.describe()
normal.Amount.describe()
f, (axis1, axis2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Amount per transaction by class')
bins = 50
axis1.hist(fraud.Amount, bins = bins)
axis1.set_title('Fraud')
axis2.hist(normal.Amount, bins = bins)
axis2.set_title('Normal')
plt.xlabel('Amount (\$)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show()
f, (axis1, axis2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Time of transaction vs Amount by class')
axis1.scatter(fraud.Time, fraud.Amount)
axis1.set_title('Fraud')
axis2.scatter(normal.Time, normal.Amount)
axis2.set_title('Normal')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```

A.5 Correlation

```
corrmatrix = df.corr()
top_corr_features = corrmatrix.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

A.6 Data preprocessing

Python Code for checking for missing values,and other preprocessing task are shown below :

```
X = df.drop('Class', axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)
nan_indices = y_train.index[y_train.isnull()]
X_train_cleaned = X_train.drop(index=nan_indices)
y_train_cleaned = y_train.drop(index=nan_indices)
print("Class distribution before SMOTE:")
print(y_train.value_counts())
```

A.7 Balancing the dataset

Since the number of fraudulent transaction is less,we are balancing the amount of data using up sampling technique by using SMOTE.

```
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
print("\nClass distribution after SMOTE:")
print(pd.Series(y_train_balanced).value_counts())
```

A.8 Model Training and Evaluation

Model Training is done on the basis of machine learning algorithms such as logistic regression,Decision Tree,SVM,and rf classifier using the training set and extracted features.The model is then evaluated using metrics such as accuracy,precision,recall,and F1-score. Splitting your dataset into training and testing sets.It helps to avoid overfitting and to accurately evaluate your model.

Training the model using Logistic Regression

```
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(X_train_balanced, y_train_balanced)
y_pred = logreg.predict(X_test)
logistic_accuracy = accuracy_score(y_test, y_pred)
print("Logistic Regression Accuracy:", logistic_accuracy)
logistic_precision = precision_score(y_test, y_pred)
print("Logistic Regression Precision:", logistic_precision)
#Confusion Matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

logistic_conf_matrix = confusion_matrix(y_test, y_pred)
print("Logistic Regression Confusion Matrix:")
print(logistic_conf_matrix)

%matplotlib inline
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
```

```
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Classification report of Logistic Regression

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Model training using Random Forest Classifier

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_balanced, y_train_balanced)
y_pred3 = rf_classifier.predict(X_test)
rf_accuracy = accuracy_score(y_test, y_pred3)
print("RandomForestClassifier Accuracy:",rf_accuracy)
rf_precision = precision_score(y_test, y_pred3)
print("RandomForestClassifier Precision:", rf_precision)
#Confusion Matrix
print("Confusion Matrix:")
con_mat=confusion_matrix(y_test, y_pred3)
print(con_mat)
%matplotlib inline
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(con_mat), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Classification report of RF Classifier

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred3))
```

Training Model using decision Tree

```
clf = DecisionTreeClassifier(criterion="entropy")
clf.fit(X_train_balanced, y_train_balanced)
y_pred1 = clf.predict(X_test)
print('Training set score: {:.4f}'.format(clf.score(X_train_balanced,
                                          y_train_balanced)))
print('Test set score: {:.4f}'.format(clf.score(X_test, y_test)))
plt.figure(figsize=(12,8))
tree.plot_tree(clf.fit(X_train_balanced, y_train_balanced))
d_accuracy = accuracy_score(y_test, y_pred1)
print("decision tree Accuracy:", d_accuracy)
d_precision = precision_score(y_test, y_pred1)
print("decision tree Precision:", d_precision)
```

```
#Confusion Matrix
cnf1_matrix=confusion_matrix(y_test, y_pred1)
print("Confusion Matrix:")
print(cnf1_matrix)
%matplotlib inline
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf1_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Classification report of Decision Tree

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred2))
```

Training model using SVM

```
svm_classifier = SVC(kernel='linear', random_state=42)
svm_classifier.fit(X_train_balanced, y_train_balanced)
y_pred2 = svm_classifier.predict(X_test)
#Confusion Matrix
conf1_matrix = confusion_matrix(y_test, y_pred2)
print("Confusion Matrix:")
print(conf1_matrix)
%matplotlib inline
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(conf1_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Classification Report of SVM

```
print("\nClassification Report of SVM:")
print(classification_report(y_test, y_pred2))
```