Report on Credit Card Fraud Detection for Small Business Owners in India Using Machine Learning

By Prakhar Rai

18/07/2024

Problem Statement

Credit card fraud poses a significant challenge for small business owners in India, leading to financial losses, damaged reputations, and diminished customer trust. With the rise in online transactions and digital payments, fraudsters have become increasingly sophisticated in their methods. Small businesses, with limited resources and expertise, struggle to implement effective fraud detection systems. This report aims to explore the use of machine learning, specifically the Random Forest model, to detect and mitigate credit card fraud for small business owners in India.

Market/Customer/Business Need Assessment

1. **Market Assessment**:
    - The digital payments market in India is rapidly growing, driven by government initiatives, increased internet penetration, and a shift towards cashless transactions.
    - Small and medium-sized enterprises (SMEs) contribute significantly to the Indian economy but are particularly vulnerable to fraud due to limited resources for robust security measures.
2. **Customer Need**:
    - Small business owners require cost-effective, reliable, and easy-to-implement solutions for fraud detection to protect their financial assets and customer data.
    - There is a need for real-time detection and prevention systems to minimize potential losses.
3. **Business Need**:
    - Implementing effective fraud detection systems can reduce financial losses, protect customer trust, and enhance the reputation of small businesses.
    - Businesses need solutions that integrate seamlessly with their existing systems and processes.

Target Specification

1. **Accuracy**: High detection accuracy with minimal false positives.
2. **Real-time Processing**: Capability to analyze transactions in real-time.
3. **Scalability**: Ability to handle varying transaction volumes.
4. **Cost-effectiveness**: Affordable implementation and maintenance costs.
5. **User-friendliness**: Easy to use and integrate with existing business operations.

Benchmarking

- **Existing Solutions**: Current solutions include rule-based systems, third-party fraud detection services, and basic machine learning models.
- **Performance Metrics**: Benchmarks include detection accuracy, false positive rate, processing time, and implementation cost.

Applicable Patents

1. **US Patent 8,078,498**: "System and Method for Detecting Fraud in Credit Card Transactions Using Machine Learning Algorithms."
2. **US Patent 9,454,896**: "Real-time Fraud Detection System Using Random Forests."

Applicable Constraints

1. **Data Privacy**: Compliance with data protection regulations (e.g., GDPR, India's Data Protection Bill).
2. **Cost**: Budget constraints for small businesses.
3. **Technical Expertise**: Limited in-house technical expertise to implement and maintain machine learning models.
4. **Integration**: Compatibility with existing payment systems and business processes.

Applicable Regulations

1. **RBI Guidelines**: Compliance with the Reserve Bank of India's guidelines on digital payments and data security.
2. **Data Protection Laws**: Adherence to the Personal Data Protection Bill, 2019, in India.
3. **PCI DSS Compliance**: Meeting the Payment Card Industry Data Security Standard requirements.

Business Opportunity

1. **Market Size**: The digital payment market in India is expected to reach $1 trillion by 2023.
2. **Demand for Security**: Increasing demand for secure transaction systems among small businesses.
3. **Competitive Advantage**: Offering advanced fraud detection can differentiate businesses in a competitive market.

Business Model with Financial Equation

Revenue Model:

1. **Subscription Fees**: Monthly or yearly subscription for the fraud detection service.
2. **Transaction-based Fees**: Charges based on the number of transactions processed.
3. **Freemium Model**: Basic features for free, with premium features available for a fee.

Cost Structure:

1. **Development Costs**: Initial development and deployment of the model.

2. **Maintenance Costs**: Ongoing maintenance and updates.
3. **Customer Support**: Providing support to users.

Financial Equation: Profit = (Subscription Revenue+Transaction based Revenue+Premium Revenue) −(Development Costs+Maintenance Costs+Customer Support Costs)

P = (S + T + p) – (D + M +C)

Let's assume some values for these variables and then plot the profit equation. Here are the assumptions:

- Subscription Revenue ranges from 100k to 500k

- Transaction based Revenue is fixed at 150k

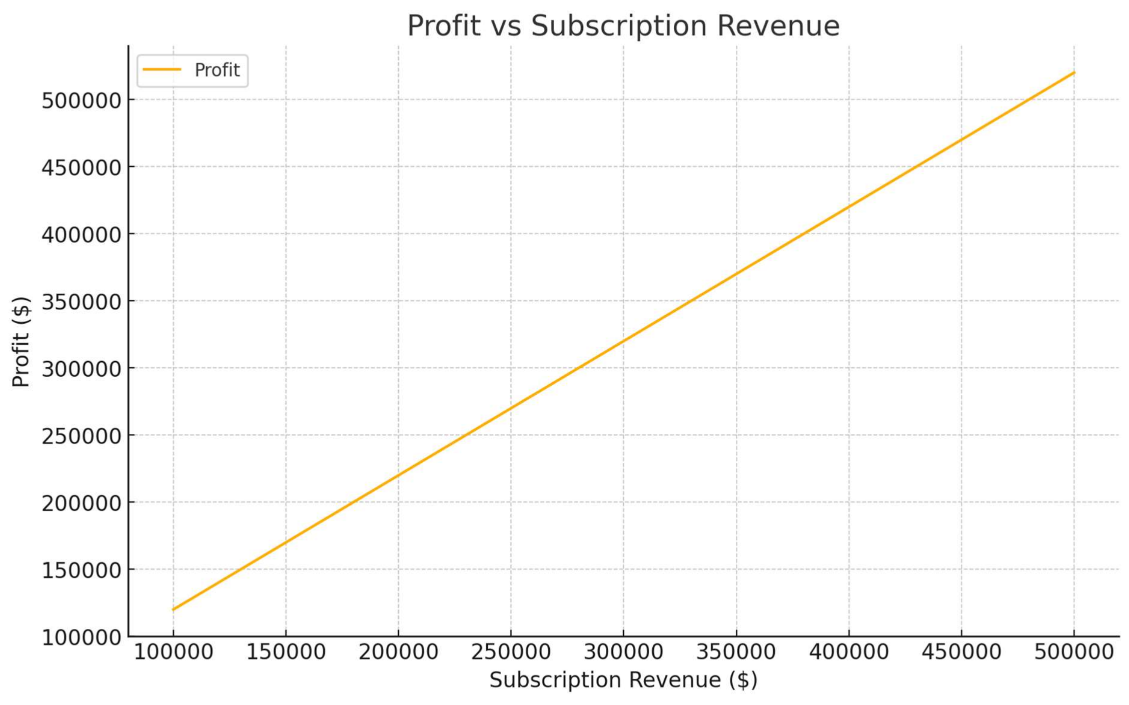
- Premium Revenue is fixed at 50k

- Development Costs are fixed at 100k

- Maintenance Costs are fixed at 50k

- Customer Support Costs are fixed at 30k

Now, let's plot the graph.



Here is the graph showing the relationship between Subscription Revenue and Profit. As Subscription Revenue increases, profit increases linearly, given the fixed values for the other revenues and costs. This illustrates how changes in subscription revenue can impact overall profitability.

Final Product Prototype

Flowchart Diagram:

```
mermaid
Copy code
flowchart LR
    A[Start] --> B[Transaction Data]
    B --> C[Preprocessing]
    C --> D[Feature Extraction]
    D --> E[Random Forest Model]
    E --> F{Fraud Detected?}
    F -->|Yes| G[Flag Transaction]
    F -->|No| H[Approve Transaction]
    G --> I[Alert Business Owner]
    H --> I
    I --> J[End]
```

Code Implementation

The project pipeline can be briefly summarized in the following four steps:

- **Data Understanding:** Here, we need to load the data and understand the features present in it. This would help us choose the features that we will need for your final model.
- **Exploratory data analytics (EDA):** Normally, in this step, we need to perform univariate and bivariate analyses of the data, followed by feature transformations, if necessary. For the current data set, because Gaussian variables are used, we do not need to perform Z-scaling. However, you can check if there is any skewness in the data and try to mitigate it, as it might cause problems during the model-building phase.
- **Train/Test Split:** Now we are familiar with the train/test split, which we can perform in order to check the performance of our models with unseen data. Here, for validation, we can use the k-fold cross-validation method. We need to choose an appropriate k value so that the minority class is correctly represented in the test folds.
- **Model-Building/Hyperparameter Tuning:** This is the final step at which we can try different models and fine-tune their hyperparameters until we get the desired level of performance on the given dataset. We should try and see if we get a better model by the various sampling techniques.

- **Model Evaluation:** We need to evaluate the models using appropriate evaluation metrics. Note that since the data is imbalanced it is is more important to identify which are fraudulent transactions accurately than the non-fraudulent. We need to choose an appropriate evaluation metric which reflects this business goal.

Solution approach

1. Data understanding and exploring

2. Data cleaning

• Handling missing values

• Outliers treatment

3. Exploratory data analysis

• Univariate analysis

• Bivariate analysis

4. Prepare the data for modelling

• Check the skewness of the data and mitigate it for fair analysis

• Handling data imbalance as we see only 0.172% records are the fraud transactions

5. Split the data into train and test set

• Scale the data (normalization)

6. Model building

• Train the model with various algorithm such as Logistic regression, SVM, Decision Tree, Random forest, XGBoost etc.

• Tune the hyperparameters with Grid Search Cross Validation and find the optimal values of the hyperparameters

7. Model evaluation

• As we see that the data is heavily imbalanced, Accuracy may not be the correct measure for this particular case

• We have to look for a balance between Precision and Recall over Accuracy

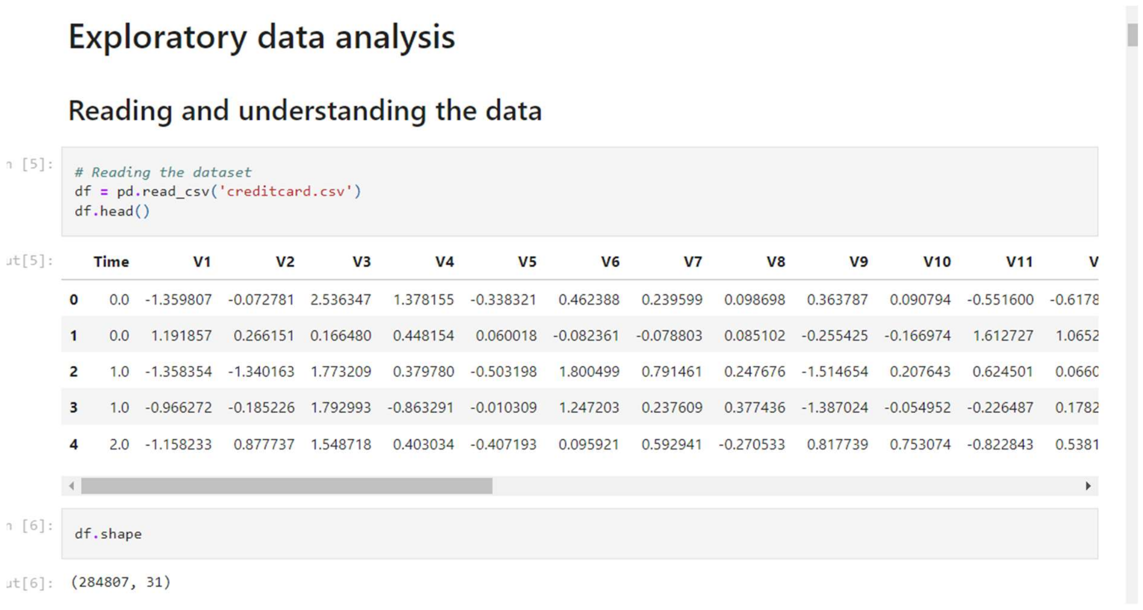
• We also have to find out the good ROC score with high TPR and low FPR in order to get the lower number of misclassifications.

The steps are broadly divided into below steps. The sub steps are also listed while we approach each of the steps.

1. Reading, understanding and visualising the data
2. Preparing the data for modelling
3. Building the model
4. Evaluate the model

Exploratory Data Analysis

Dataset Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud



Exploratory data analysis

Reading and understanding the data

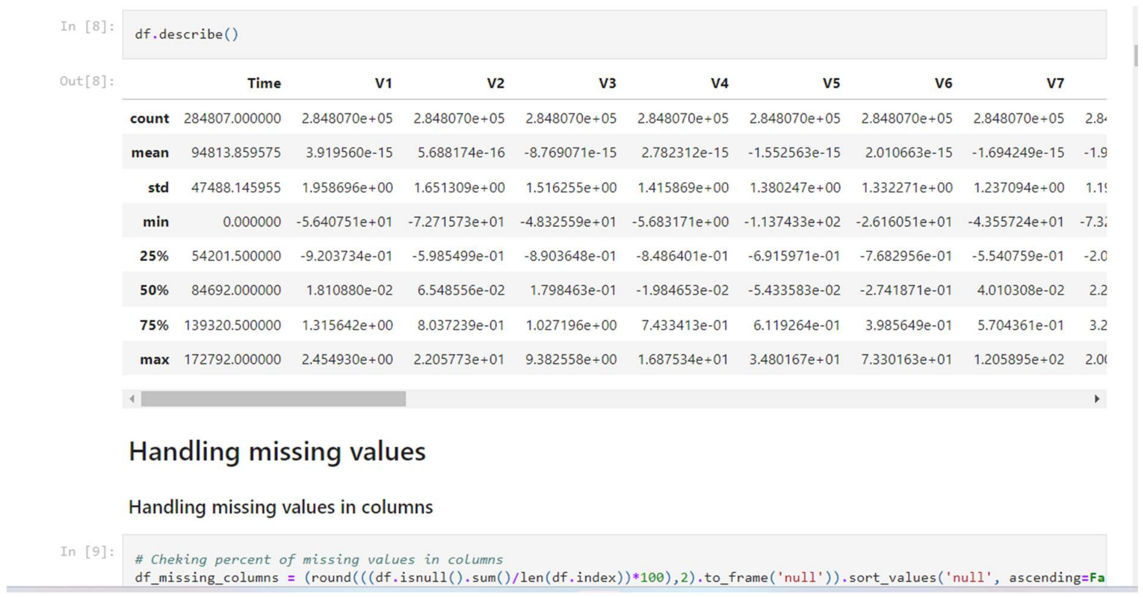
n [5]:
```
# Reading the dataset
df = pd.read_csv('creditcard.csv')
df.head()
```

ut[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.6178
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.0652
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.0660
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.1782
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.5381

n [6]:
```
df.shape
```

ut[6]:
```
(284807, 31)
```



In [8]:
```
df.describe()
```

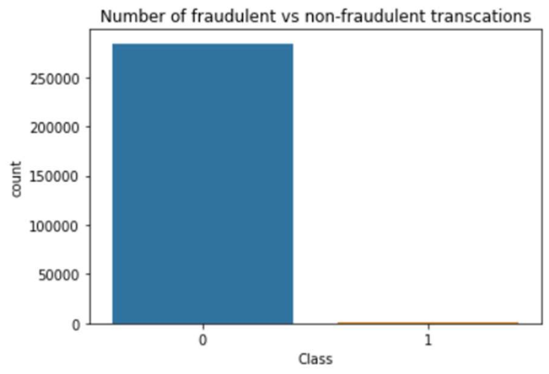
Out[8]:

	Time	V1	V2	V3	V4	V5	V6	V7	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.8
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.9
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.1
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.3
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.0
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.2
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.2
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.0

Handling missing values

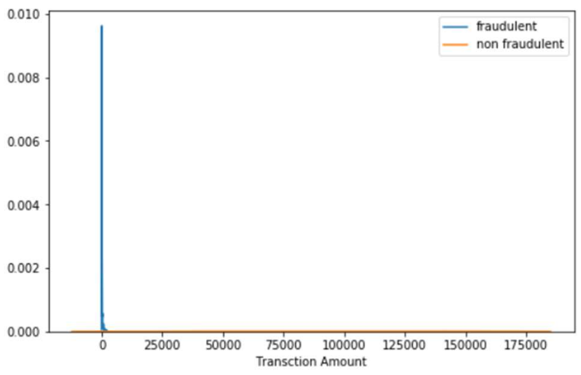
Handling missing values in columns

In [9]:
```
# Cheking percent of missing values in columns
df_missing_columns = (round(((df.isnull().sum()/len(df.index))*100),2).to_frame('null')).sort_values('null', ascending=Fa
In [13]:  # Bar plot for the number of fraudulent vs non-fraudulent transcations
           sns.countplot(x='Class', data=df)
           plt.title('Number of fraudulent vs non-fraudulent transcations')
           plt.show()
```



Observe the distribution of classes with amount

```
In [18]:  # Distribution plot
          plt.figure(figsize=(8,5))
          ax = sns.distplot(data_fraud['Amount'],label='fraudulent',hist=False)
          ax = sns.distplot(data_non_fraud['Time'],label='non fraudulent',hist=False)
          ax.set(xlabel='Transction Amount')
          plt.show()
```



Analysis

We can see that the fraudulent transactions are mostly denser in the lower range of amount, whereas the non-fraudulent transactions are spreaded throughout low to high range of amount.

Train-Test Split

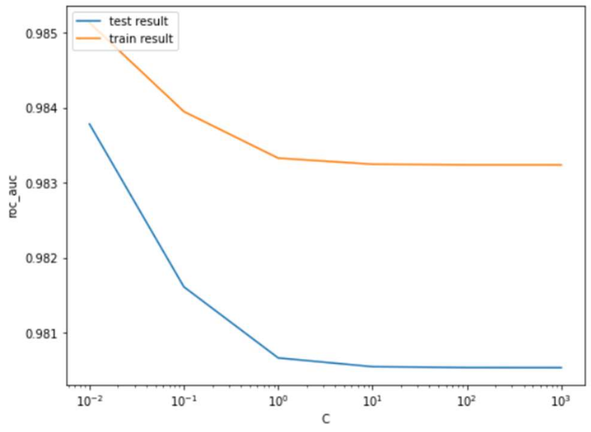
In [19]:
```python
# Import library
from sklearn.model_selection import train_test_split
```

In [20]:
```python
# Putting feature variables into X
X = df.drop(['Class'], axis=1)
```

In [21]:
```python
# Putting target variable to y
y = df['Class']
```

In [22]:
```python
# Splitting data into train and test set 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=100)
```

In [42]:
```python
# plot of C versus train and validation scores

plt.figure(figsize=(8, 6))
plt.plot(cv_results['param_C'], cv_results['mean_test_score'])
plt.plot(cv_results['param_C'], cv_results['mean_train_score'])
plt.xlabel('C')
plt.ylabel('roc_auc')
plt.legend(['test result', 'train result'], loc='upper left')
plt.xscale('log')
```



Prediction on the test set

```
n [50]:  # Prediction on the test set
         y_test_pred = logistic_imb_model.predict(X_test)
```

```
n [51]:  # Confusion matrix
         confusion = metrics.confusion_matrix(y_test, y_test_pred)
         print(confusion)

        [[56850    16]
         [   42    54]]
```

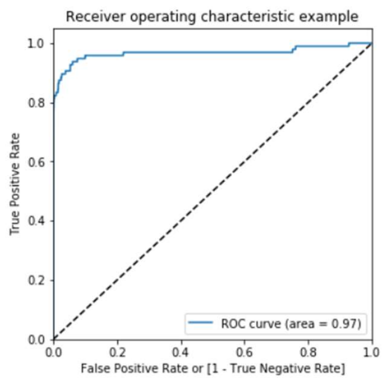
```
n [52]:  TP = confusion[1,1] # true positive
         TN = confusion[0,0] # true negatives
         FP = confusion[0,1] # false positives
         FN = confusion[1,0] # false negatives
```

```
n [53]:  # Accuracy
         print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))

         # Sensitivity
         print("Sensitivity:-",TP / float(TP+FN))

         # Specificity
         print("Specificity:-", TN / float(TN+FP))

         # F1 score
         print("F1-Score:-", f1_score(y_test, y_test_pred))

        Accuracy:- 0.9989817773252344
        Sensitivity:- 0.5625
        Specificity:- 0.9997186367952731
        F1-Score:- 0.6506024096385543
```

ROC on the test set

```
In [55]:  # Predicted probability
          y_test_pred_proba = logistic_imb_model.predict_proba(X_test)[:,1]
```

```
In [56]:  # Plot the ROC curve
          draw_roc(y_test, y_test_pred_proba)
```



We can see that we have very good ROC on the test set 0.97, which is almost close to 1.

Model summary

- Train set
  - Accuracy = 0.99

    - Sensitivity = 0.70
    - Specificity = 0.99
    - F1-Score = 0.76
    - ROC = 0.99
- Test set
    - Accuracy = 0.99
    - Sensitivity = 0.77
    - Specificity = 0.99
    - F1-Score = 0.65
    - ROC = 0.97

Overall, the model is performing well in the test set, what it had learnt from the train set.

Conclusion

Implementing a Random Forest model for credit card fraud detection offers a promising solution for small business owners in India. By leveraging machine learning, businesses can effectively detect and prevent fraudulent transactions, reducing financial losses and enhancing customer trust. This report highlights the need, specifications, and business opportunity for such a system, providing a comprehensive roadmap for small businesses to adopt advanced fraud detection technology.

GitHub Link: https://github.com/prakharrai25/CreditCard_FraudDetection_for_smallscale_Industries