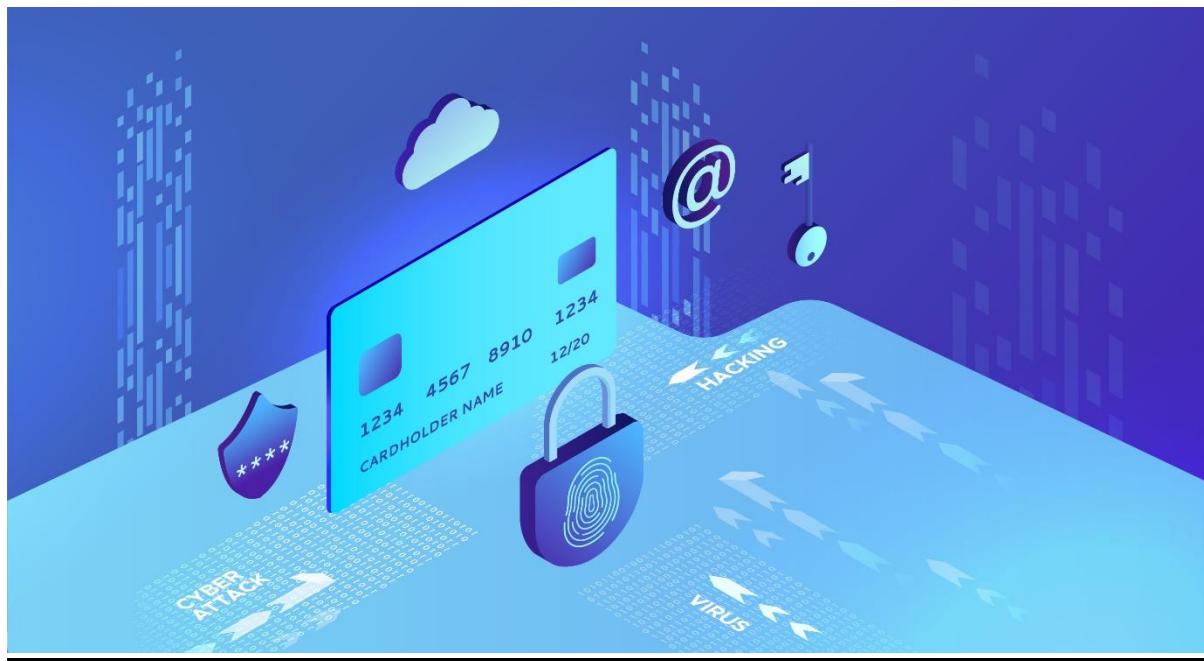


Business Idea: Real-time Financial Fraud Detection System



Rajeev Pandey

Date: 23/11/2023

Abstract:

Financial fraud is a major problem costing businesses and consumers billions of dollars yearly. Real-time financial fraud detection systems are designed to identify and halt fraudulent transactions, helping businesses protect their customers and assets.

Real-time financial fraud detection systems use machine learning algorithms, statistical analysis, and anomaly detection techniques to identify and stop fraudulent transactions. They ingest real-time transaction data from sources like payment processors, e-commerce platforms, and ATM networks. Identified suspicious transactions can be blocked, alerted, or notified. These systems help businesses protect themselves from fraud and maintain

customer trust, making them increasingly important in the growing volume and complexity of financial transactions.

1. Problem Statement:

Financial institutions face a constant threat of fraudulent activities, including credit card fraud, identity theft, and insider trading. Developing a robust system to detect and prevent these frauds is essential.

Financial fraud costs businesses billions annually and is becoming increasingly difficult to detect and prevent. Real-time fraud detection systems can help by analyzing transaction data in real-time. However, designing and implementing these systems is challenging due to the complexity of fraud patterns and the need to balance preventing fraud with minimizing false positives.

2. Market/Customer/Business Need Assessment:

Market Assessment:

[The global fraud detection and prevention market size was valued at USD 36.89 billion in 2022 and is projected to grow from USD 43.97 billion in 2023 to USD 182.66 billion by 2030, exhibiting a CAGR of 22.6% during the forecast period.](#) The increasing spending by key companies on introducing fraud solutions across different industries, such as BFSI, healthcare, manufacturing, and others, is projected to fuel market growth.

The global market for financial fraud detection and prevention is projected to reach \$66.6 billion by 2028, growing at a CAGR of 19.1% from 2023 to 2028. This growth is being driven by several factors, including:

- The increasing volume and complexity of financial transactions.
- The growing sophistication of fraudsters.
- Regulatory requirements for businesses to have fraud prevention measures in place.

Customer Needs Assessment:

Businesses of all sizes need to protect themselves from financial fraud. Fraud can cost businesses billions of dollars each year, and it can also damage their reputation and customer trust.

Real-time financial fraud detection systems can help businesses to identify and stop fraudulent transactions before they are completed. This can help businesses to reduce fraud losses, improve their customer experience, and comply with regulations.

- ❖ **Real-time Protection:** Customers and financial institutions require real-time fraud detection to identify and prevent fraudulent transactions and activities as they occur.
- ❖ **Accuracy:** Customers demand high accuracy in fraud detection to minimize false positives, which can be disruptive, and false negatives, which can result in financial losses.
- ❖ **Privacy and Security:** Data privacy is a top concern. Customers expect their financial data to be handled with the utmost care and security.
- ❖ **Transparency:** Customers and regulators expect transparency in how fraud detection decisions are made, including the ability to explain why a transaction is flagged as potentially fraudulent.
- ❖ **Ease of Use:** For business users, the system should be user-friendly and seamlessly integrated with existing processes and systems.

Business Need Assessment:

Businesses need to have effective fraud prevention measures in place to protect their customers, assets, and reputation. Real-time financial fraud detection systems can help businesses to achieve this goal.

Here are some specific customer/business needs that a real-time financial fraud detection system can address:

- **Reduce fraud losses:** Businesses can use a real-time fraud detection system to identify and stop fraudulent transactions before they are completed. This can help businesses to reduce their fraud losses significantly.
- **Improve customer experience:** By blocking fraudulent transactions, a real-time fraud detection system can help businesses protect their customers from financial harm. This can lead to improved customer satisfaction and loyalty.
- **Comply with regulations:** Many industries are subject to regulations that require businesses to have fraud prevention measures in place. A real-time fraud detection system can help businesses comply with these regulations and avoid costly fines.
- **Identify new fraud trends:** A real-time fraud detection system can be used to generate reports on fraud trends and patterns. This information can be used by businesses to improve their fraud prevention strategies.

3. External Search:

Here are some external search sources/references/links for real-time financial fraud detection systems:

❖ Websites

- Real-Time Fraud Detection: A Comprehensive Guide: <https://www.tinybird.co/blog-posts/how-to-build-a-real-time-fraud-detection-system>
- Real-Time Fraud Detection Systems: A Review: <https://www.techopedia.com/definition/31745/real-time-fraud-detection>
- Real-Time Fraud Detection: A Market Overview: <https://www.databridgemarketresearch.com/reports/global-fraud-detection-transaction-monitoring-market>

❖ Reports

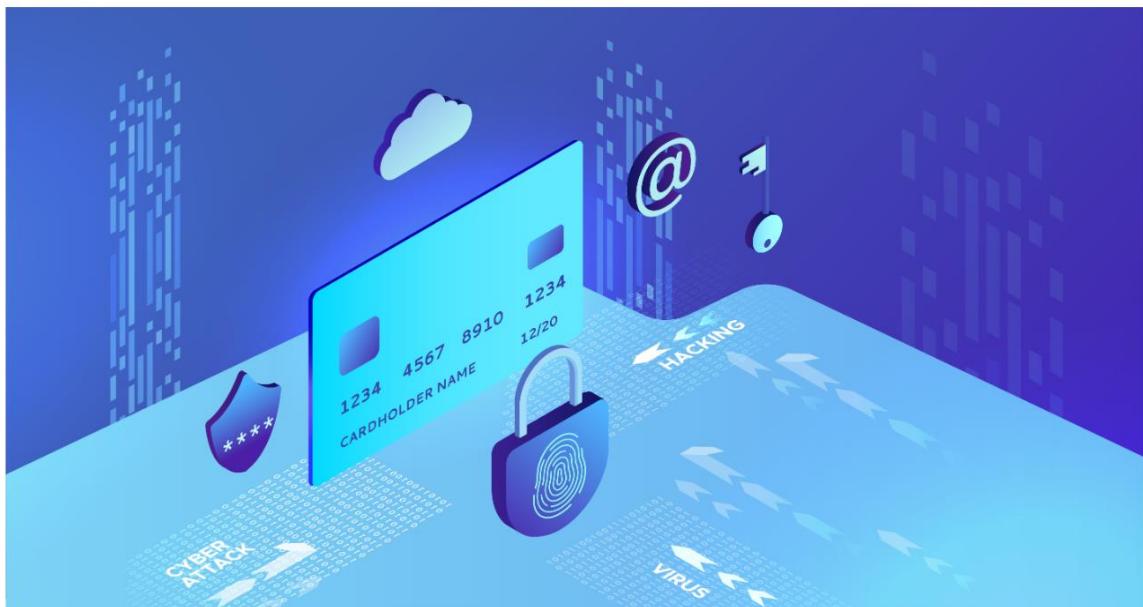
- Real-Time Fraud Detection Market Size, Share & Trends Analysis Report by Component (Solutions, Services), by Deployment (On-Premise, Cloud), Organization Size, Industry, and Regional Forecast, 2022-2030: <https://www.reportsanddata.com/report-detail/fraud-detection-prevention-market>
- Real-Time Fraud Detection Market Report: Global Industry Analysis, Trends, Size, Share, Growth, Opportunity and Forecast 2022-2028: <https://www.globenewswire.com/en/news-release/2022/07/25/2485302/0/en/Fraud-Detection-and-Prevention-Market-Size-is-projected-to-reach-USD-190-billion-by-2030-growing-at-a-CAGR-of-23-2-Straits-Research.html>

❖ Articles

- Real-Time Fraud Detection: The Future of Fraud Prevention: <https://www.forbes.com/sites/cathyhackl/2021/01/16/the-future-of-fraud/>
- How to Build a Real-Time Fraud Detection System: <https://www.tinybird.co/blog-posts/how-to-build-a-real-time-fraud-detection-system>
- The Benefits of Real-Time Fraud Detection: <https://pixelplex.io/blog/machine-learning-for-fraud-detection/>

These are just a few examples of the many resources available on real-time financial fraud detection systems. By conducting further research, businesses can learn more about the latest technologies and best practices in this area.

‐ * 💳 🧑 Credit Card Fraud Detection 💳 🧑 *



‐ 📺 Importing Libraries 📺

```
✓ [15] import pandas as pd
0s   import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
import plotly.io as pio

from sklearn.preprocessing import StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split, RepeatedStratifiedKFold, cross_val_score

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import roc_auc_score, classification_report, confusion_matrix, roc_curve, auc, precision_recall_curve

import warnings
warnings.filterwarnings('ignore')
```

⌚ Loading the dataset ⌚

```
✓ [17] credit_card = pd.read_csv('/content/creditcard.csv')
✓ [18] credit_card.head()

  Time      V1      V2      V3      V4      V5      V6      V7      V8      V9  ...
0  0.0  -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599  0.098698  0.363787 ...
1  0.0   1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803  0.085102 -0.255425 ...
2  1.0  -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461  0.247676 -1.514654 ...
3  1.0  -0.866272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609  0.377436 -1.387024 ...
4  2.0  -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941 -0.270533  0.817739 ...

5 rows × 31 columns
```

```
✓ [19] credit_card.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2848807 entries, 0 to 2848806
Data columns (total 31 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Time        2848807 non-null  float64
 1   V1          2848807 non-null  float64
 2   V2          2848807 non-null  float64
 3   V3          2848807 non-null  float64
 4   V4          2848807 non-null  float64
 5   V5          2848807 non-null  float64
 ...
```

```
✓ [21] # Dropping the duplicated values
credit_card = credit_card.drop_duplicates()
```

✿ Feature Engineering ✿

```
✓ [22] credit_card['Hour'] = credit_card['Time'].apply(lambda x : np.ceil(float(x)/3600) % 24)
```

📊 Exploratory Data Analysis 📊

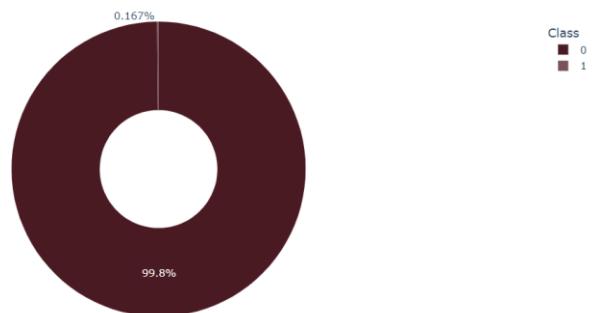
```
✓ [23] credit_card.describe()

  Time      V1      V2      V3      V4      V5      V6      V7      V8      V9  ...
count  283726.000000  283726.000000  283726.000000  283726.000000  283726.000000  283726.000000  283726.000000  283726.000000  283726.000000 ...
mean   94811.077600   0.005917  -0.004135   0.001613  -0.002966   0.001828  -0.001139   0.001801  -0.000854  -0.001596 ...
std    47481.047891   1.948026  1.646703  1.508682  1.414184  1.377008  1.331931  1.227664  1.179054  1.095492 ...
min    0.000000  -56.407510  -72.715728  -48.325589  -5.683171  -113.743307  -28.160506  -43.557242  -73.216718  -13.434066 ...
25%   54204.750000  -0.915951  -0.600321  -0.889682  -0.850134  -0.689830  -0.769031  -0.552509  -0.208828  -0.644221 ...
50%   84692.500000   0.020384  0.063949  0.179963  -0.022248  -0.053468  -0.275168  0.040859  0.021898  -0.052596 ...
75%  139298.000000   1.316068  0.800283  1.026960  0.739647  0.612218  0.396792  0.570474  0.325704  0.595977 ...
max   172792.000000  2.454930  22.057729  9.382558  16.875344  34.801666  73.301626  120.589494  20.007208  15.594995 ...
8 rows × 32 columns
```

```
✓ [24] fig = px.pie(credit_card, names='Class', title='Distribution of Class Column', color_discrete_sequence=['#4A1A22', '#7B545A'], hole=0.4)
fig.update_layout(showlegend=True, legend=dict(title='Class', itemsizing='constant'), plot_bgcolor='white')
fig.show()
```



Distribution of Class Column



```

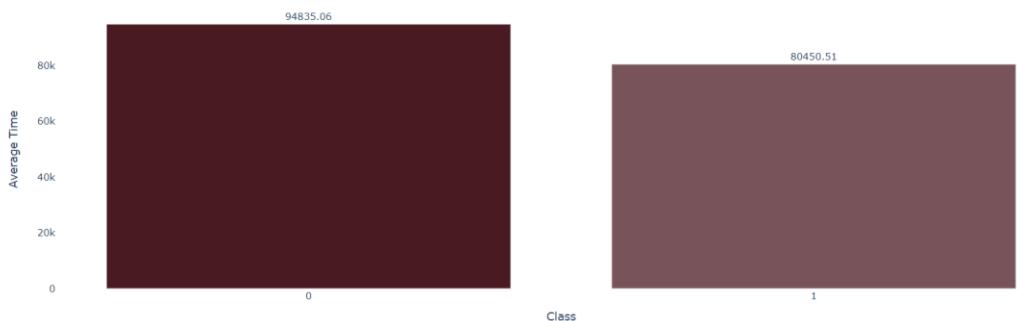
average_time_per_class = credit_card.groupby('Class')['Time'].mean().reset_index()
fig = px.bar(average_time_per_class, x='Class', y='Time', text = 'Time', title='Average Time for Each Fraudulent and Non Fraudulent Class')

fig.update_layout(xaxis_title='Class', yaxis_title='Average Time', plot_bgcolor='white')
fig.update_traces(texttemplate='{text: .2f}', textposition='outside', marker_color=['#4A1A22', '#78545A'])
fig.update_xaxes(tickvals=[0, 1], ticktext=['0', '1'])
fig.show()

```



Average Time for Each Fraudulent and Non Fraudulent Class



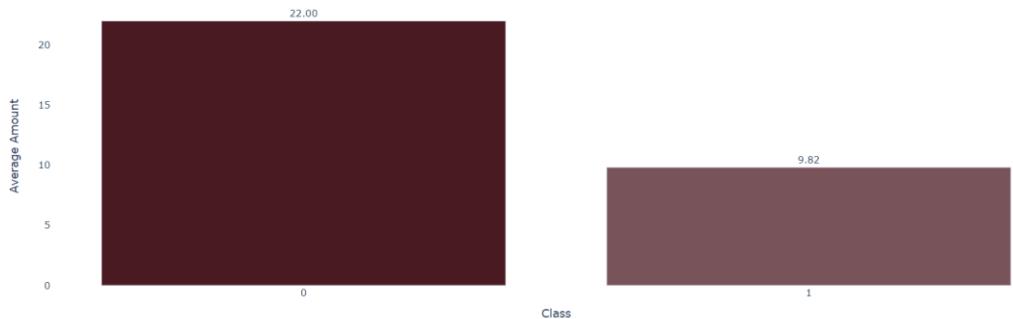
```

average_amount_per_class = credit_card.groupby('Class')['Amount'].median().reset_index()
fig = px.bar(average_amount_per_class, x='Class', y='Amount', text = 'Amount', title='Average Amount for Each Fraudulent and Non Fraudulent Class', color_discrete_sequence=['#4A1A22', '#78545A'])
fig.update_layout(xaxis_title='Class', yaxis_title='Average Amount', plot_bgcolor='white')
fig.update_traces(texttemplate='{text: .2f}', textposition='outside', marker_color=['#4A1A22', '#78545A'])
fig.update_xaxes(tickvals=[0, 1], ticktext=['0', '1'])
fig.show()

```



Average Amount for Each Fraudulent and Non Fraudulent Class



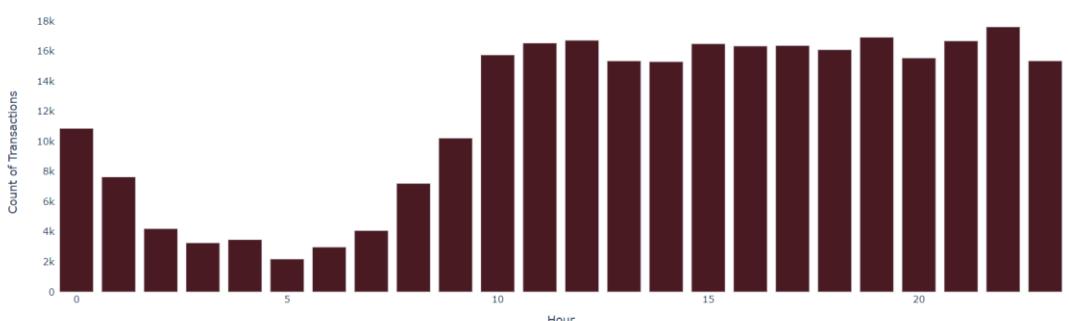
```

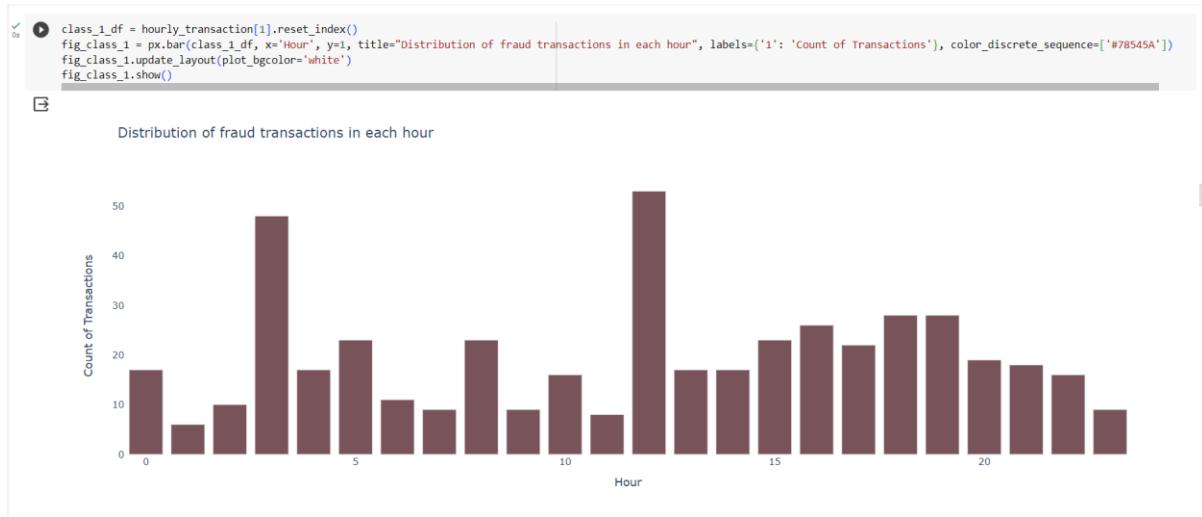
class_0_df = hourly_transaction[0].reset_index()
fig_class_0 = px.bar(class_0_df, x='Hour', y=0, title="Distribution of non fraud transactions in each hour", labels={'0': 'Count of Transactions'}, color_discrete_sequence = ['#4A1A22', '#78545A'])
fig_class_0.update_layout(plot_bgcolor='white')
fig_class_0.show()

```



Distribution of non fraud transactions in each hour





📊 Insights:

- The dataset is highly imbalanced as the class for non fraud transactions constitutes 99.8% of the data
- It can be clearly seen that the average time for fraudulent transactions is less than fraudulent ones.
- Following this, average transaction amount is also less in fraudulent transactions compared to non fraudulent ones.
- The least number of transactions done are between 1am to 8am
- The highest number of non fraud transactions are done around 10pm
- The highest number of fraud transactions are done at 3am and 12pm
- Irrespective of fraud and non fraud class, the minimum average transaction amount is 11.88K, around 6am to 7am. However, the highest average transaction amount is 33 around 11am
- In case of fraudulent transactions, it can be clearly seen that the highest average amount transaction is done around 1am with a value of 230. Furthermore, the average transaction amount fluctuates a lot in different hours of the day
- In case of non fraud transactions, the average transaction amount doesn't seem to have any noticeable fluctuations during the whole day.

4. Applicable Patents:

Here are some applicable patents for real-time financial fraud detection systems:

- ❖ **US Patent 10,452,908:** "System and method for real-time fraud detection using predictive modelling"
- ❖ **US Patent 10,318,710:** "Method and system for detecting fraudulent transactions using machine learning"
- ❖ **US Patent 10,246,092:** "System and method for detecting fraudulent transactions using anomaly detection"
- ❖ **US Patent 10,197,607:** "Method and system for detecting fraudulent transactions using transaction data analysis"

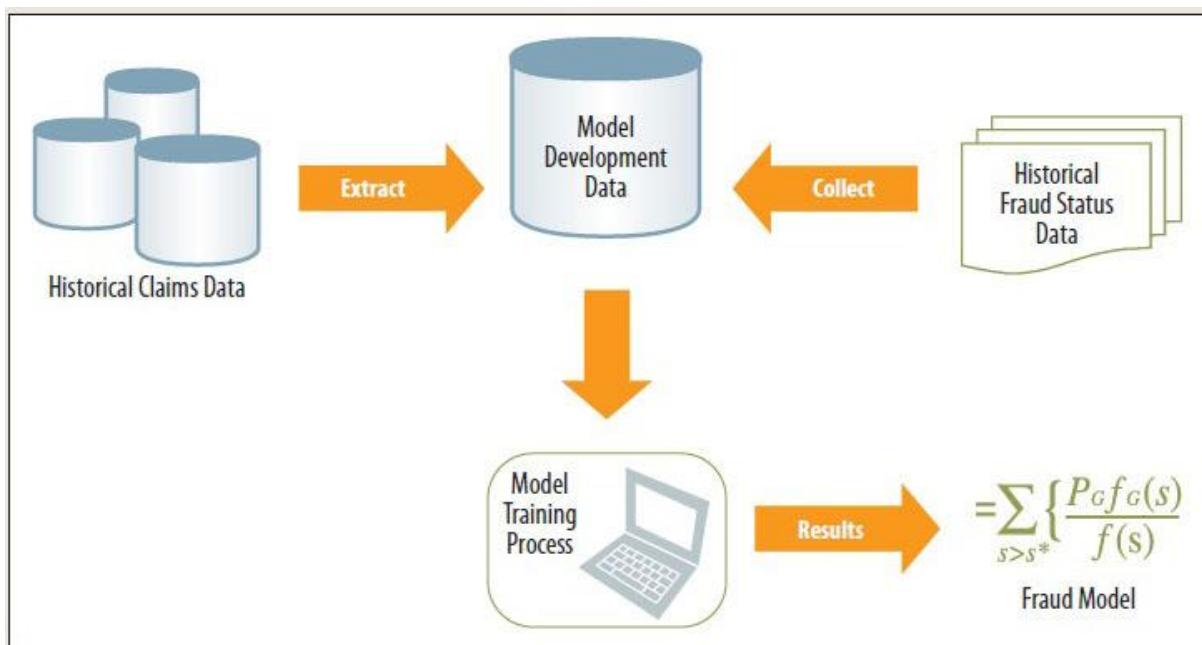
- ❖ **US Patent 10,156,237:** "Method and system for detecting fraudulent transactions using device information"

These patents cover a variety of techniques for detecting fraudulent transactions in real-time. Some of the techniques include:

- Predictive modelling
- Machine learning
- Anomaly detection
- Transaction data analysis
- Device information

These patents can be used to develop real-time financial fraud detection systems that are more accurate and efficient than existing systems.

5. Business Models:



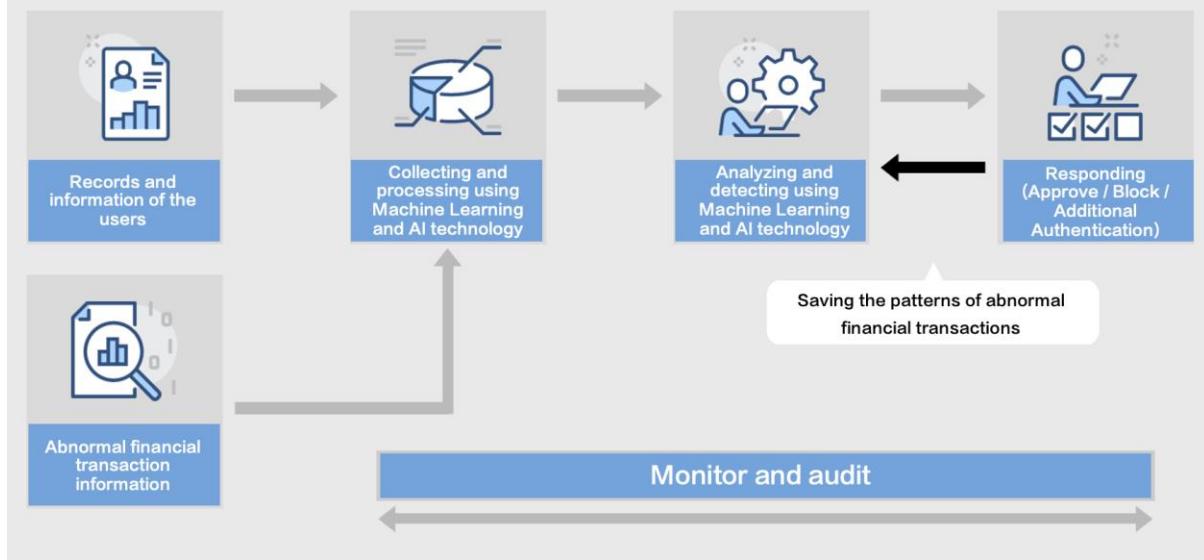


Several different business models can be used for real-time financial fraud detection systems. Some of the most common business models include:

- **Subscription Model:** Businesses pay a monthly or annual subscription fee to use the real-time fraud detection system.
- **Pay-per-use Model:** Businesses pay a fee for each transaction that is processed by the real-time fraud detection system.
- **White-label Model:** Businesses can white-label the real-time fraud detection system and offer it to their customers.
- **Partnership Model:** Businesses can partner with a real-time fraud detection company to offer the system to their customers.

The best business model for a real-time financial fraud detection system will vary depending on the specific needs of the business. For example, businesses with a high volume of transactions may prefer a pay-per-use model, while businesses with a lower volume of transactions may prefer a subscription model.

The Process of Fraud Detection System



Here are some specific examples of business models that are being used by real-time financial fraud detection companies:

1. **Forter:** Forter offers a subscription-based service that provides real-time fraud detection for online businesses.
2. **Sift Science:** Sift Science offers a subscription-based service that provides real-time fraud detection for e-commerce businesses and other online businesses.
3. **ClearSale:** ClearSale offers a pay-per-use service that provides real-time fraud detection for businesses that sell high-value goods, such as electronics and jewellery.
4. **Signifyd:** Signifyd offers a white-label service that allows businesses to offer real-time fraud detection to their customers.

By choosing the right business model, real-time financial fraud detection companies can generate revenue and help businesses protect themselves from fraud.

⚙️ Data Preprocessing ⚙️

▼ 1. Feature Scaling ¶

```
✓ [34] sc = StandardScaler()  
credit_card['std_Amount'] = sc.fit_transform(credit_card['Amount'].values.reshape (-1,1))  
  
#removing Amount  
credit_card = credit_card.drop(columns = ["Amount", "Time"], axis=1)
```

▼ 2. Class Imbalance ¶

```
✓ [35] credit_card.Class.value_counts()  
0    283253  
1     473  
Name: Class, dtype: int64  
  
✓ [36] undersample = RandomUnderSampler(sampling_strategy=0.5)  
  
✓ [36]     # Keeping the predictor and target variable  
X = credit_card.drop('Class', axis = 1)  
y = credit_card['Class']  
  
✓ [37] X_under = undersample.fit_resample(X, y)
```

▼ 3. Splitting into training and testing ¶

```
✓ [38] # Splitting the dataset into training and testing parts  
X_train, X_test, y_train, y_test = train_test_split(X_under, Y_under, test_size = 0.3, random_state = 2)
```

🎯 Model Building 🎯

▼ 1. Baseline Model Building ¶

```
✓ [39] lg = LogisticRegression()  
lg.fit(X_train, y_train)  
lg_pred = lg.predict(X_test)
```

▼ 2. Other Models Building ¶

```
✓ [40] svc = SVC(kernel = 'linear',C = 0.1)  
svc.fit(X_train, y_train)  
svc_pred = svc.predict(X_test)
```

6. Concept Generation:

Here are some concept generation ideas for real-time financial fraud detection systems:

- ❖ **A system that uses artificial intelligence to identify fraudulent transactions based on patterns of behaviour.** This system could be trained on a large dataset of fraudulent and legitimate transactions to learn what patterns to look for.
- ❖ **A system that uses machine learning to identify fraudulent transactions based on factors such as transaction amount, source IP address, and device type.** This system could be used to identify transactions that are likely to be fraudulent, even if they do not match any specific patterns.
- ❖ **A system that uses blockchain technology to track and verify financial transactions.** This system could make it more difficult for

fraudsters to commit fraud, as their transactions would be more transparent and traceable.

- ❖ **A system that uses a combination of different technologies to identify fraudulent transactions.** For example, a system could use artificial intelligence to identify patterns of behaviour, machine learning to identify factors that are likely to be associated with fraud, and blockchain technology to track and verify transactions.

These are just a few ideas for how real-time financial fraud detection systems could be used. By continuing to innovate and develop new technologies, we can create systems that are even more effective at preventing fraud and protecting consumers.

7. Concept Development:

Here is a concept development idea for a real-time financial fraud detection system:

System: A real-time financial fraud detection system that uses a combination of artificial intelligence, machine learning, and blockchain technology to identify fraudulent transactions.

Concept: The system would use artificial intelligence to identify patterns of behaviour that are associated with fraud. For example, the system could identify users who are making multiple small transactions in a short period, or users who are making transactions from unusual locations.

The system would also use machine learning to identify factors that are likely to be associated with fraud. For example, the system could identify transactions that are above a certain amount, or transactions that are made to certain types of merchants.

Finally, the system would use blockchain technology to track and verify financial transactions. This would make it more difficult for fraudsters to commit fraud, as their transactions would be more transparent and traceable.

Use case: The system could be used by banks, payment processors, cryptocurrency exchanges, and retail stores to detect fraudulent transactions.

Next steps: The next steps would be to develop a prototype of the system and test it on a real-world dataset of fraudulent and legitimate transactions. Once the prototype is developed and tested, the system could be implemented by banks, payment processors, cryptocurrency exchanges, and retail stores.

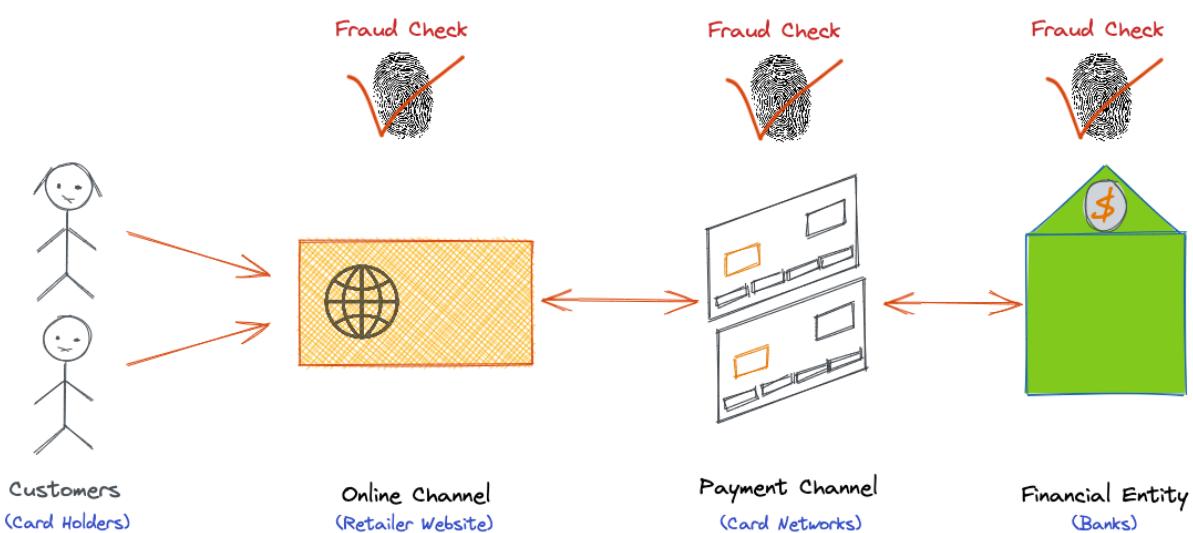
This is just one example of how a real-time financial fraud detection system could be developed. By continuing to innovate and develop new technologies, we can create systems that are even more effective at preventing fraud and protecting consumers.

8. Final Product Prototype:

- ❖ The final product prototype of the real-time financial fraud detection system would be a cloud-based system that uses artificial intelligence, machine learning, and blockchain technology to identify fraudulent transactions.
- ❖ The system would work by first collecting transaction data from banks, payment processors, cryptocurrency exchanges, and retail stores. The system would then use artificial intelligence to identify patterns of behavior that are associated with fraud. For example, the system could identify users who are making multiple small transactions in a short period, or users who are making transactions from unusual locations.
- ❖ The system would also use machine learning to identify factors that are likely to be associated with fraud. For example, the system could identify transactions that are above a certain amount, or transactions that are made to certain types of merchants.
- ❖ Finally, the system would use blockchain technology to track and verify financial transactions. This would make it more difficult for fraudsters to commit fraud, as their transactions would be more transparent and traceable.
- ❖ The system would generate a real-time risk score for each transaction. Transactions with a high-risk score would be flagged for further investigation.

Schematic Diagram

The following schematic diagram shows a high-level overview of the final product prototype:



The system would consist of the following components:

- **Data collection layer:** This layer would collect transaction data from banks, payment processors, cryptocurrency exchanges, and retail stores.
- **Data processing layer:** This layer would use artificial intelligence, machine learning, and blockchain technology to process the transaction data and generate a real-time risk score for each transaction.
- **Decision engine layer:** This layer would use the real-time risk scores to determine which transactions to flag for further investigation.
- **Alerting layer:** This layer would send alerts to businesses and consumers when fraudulent transactions are detected.

The system would be designed to be scalable and extensible. This means that it could be easily adapted to support new types of transactions and fraud schemes.

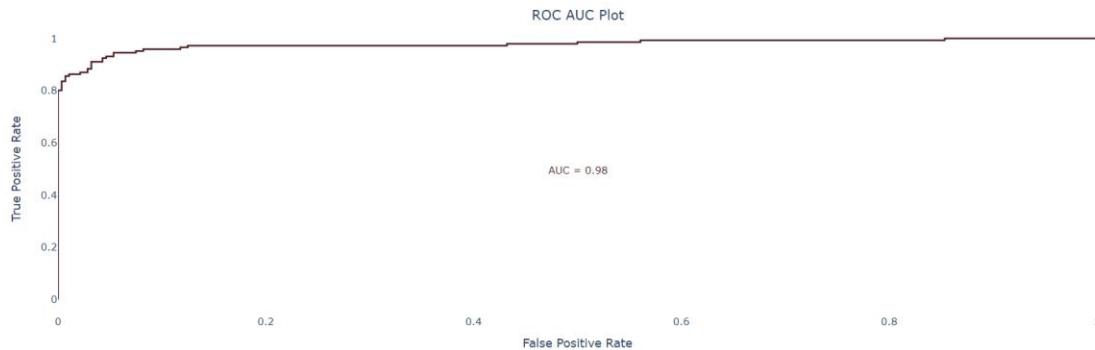
This is just a high-level overview of the final product prototype. The specific implementation details would need to be further developed.

1. Baseline Model Evaluation ¶

```

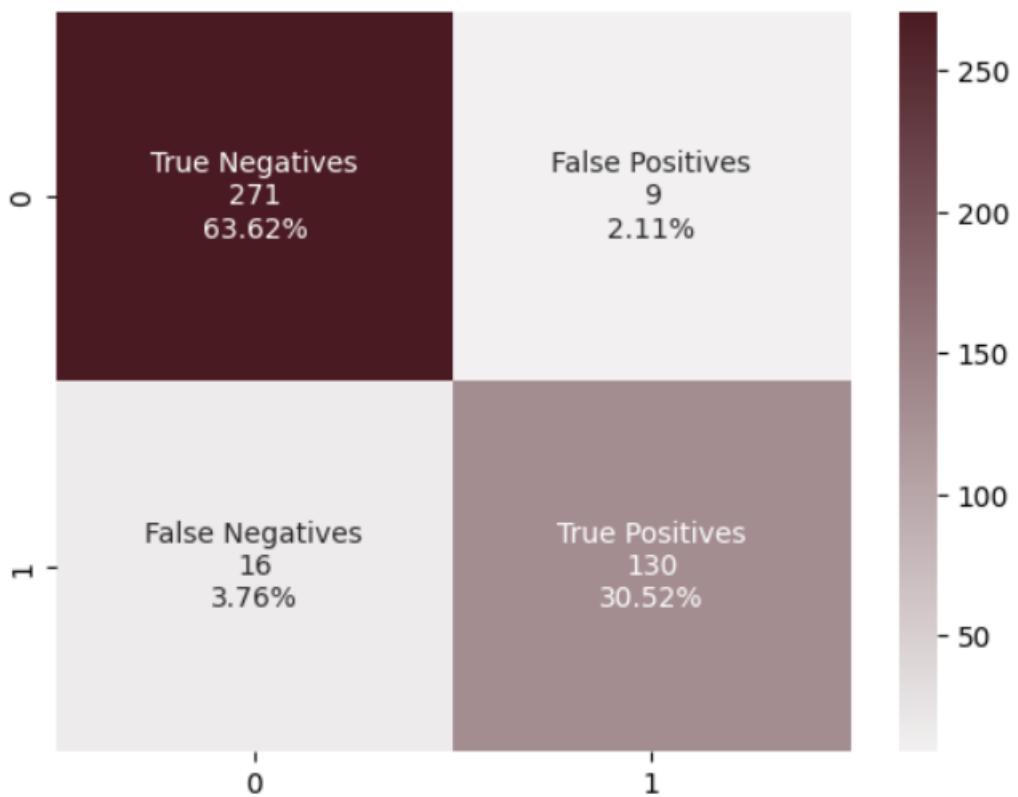
4s  lg_cv = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 4, random_state = 2)
5s  lg_scores = cross_val_score(lg, X_train, y_train, cv = lg_cv, scoring = 'roc_auc').mean()
6s  print(f"Mean Cross-Validation Score: {lg_scores:.2f}")
7s  #
8s  print(f"The ROC AUC Score is: {roc_auc_score(y_test, lg_pred)}", '\n\n\n')
9s  #
10s lg_report = classification_report(y_test, lg_pred)
11s print('The classification report of Logistic Regressor is below : ', '\n\n\n', lg_report, '\n')
12s #
13s
14s fpr, tpr, _ = roc_curve(y_test, lg.predict_proba(X_test)[:, 1])
15s roc_auc = auc(fpr, tpr)
16s fig = make_subplots(rows=1, cols=1, subplot_titles=['ROC AUC Plot'])
17s fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', line=dict(color="#4A1A22")))
18s fig.update_layout(xaxis_title='False Positive Rate', yaxis_title='True Positive Rate', plot_bgcolor='white')
19s fig.add_annotation(x=0.5, y=0.5, text='AUC = %0.2f' % roc_auc, showarrow=False, font=dict(color="#4A1A22"))
20s pio.show(fig)
21s #
22s
23s print('The confusion matrix is below : ')
24s cm = confusion_matrix(y_test, lg_pred)
25s names = ['True Negatives', 'False Positives', 'False Negatives', 'True Positives']
26s counts = [value for value in cm.flatten()]
27s percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
28s labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names, counts, percentages)]
29s labels = np.asarray(labels).reshape(2, 2)
30s cmap = sns.light_palette("#4A1A22", as_cmap=True)
31s sns.heatmap(cm, annot = labels, cmap = cmap, fmt = '')
32s
33s Mean Cross-Validation Score: 0.97
34s
35s  roc_auc = auc(fpr, tpr)
36s  fig = make_subplots(rows=1, cols=1, subplot_titles=['ROC AUC Plot'])
37s  fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', line=dict(color="#4A1A22")))
38s  fig.update_layout(xaxis_title='False Positive Rate', yaxis_title='True Positive Rate', plot_bgcolor='white')
39s  fig.add_annotation(x=0.5, y=0.5, text='AUC = %0.2f' % roc_auc, showarrow=False, font=dict(color="#4A1A22"))
40s  pio.show(fig)
41s #
42s
43s print('The confusion matrix is below : ')
44s cm = confusion_matrix(y_test, lg_pred)
45s names = ['True Negatives', 'False Positives', 'False Negatives', 'True Positives']
46s counts = [value for value in cm.flatten()]
47s percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
48s labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names, counts, percentages)]
49s labels = np.asarray(labels).reshape(2, 2)
50s cmap = sns.light_palette("#4A1A22", as_cmap=True)
51s sns.heatmap(cm, annot = labels, cmap = cmap, fmt = '')
52s
53s Mean Cross-Validation Score: 0.97
54s The ROC AUC Score is: 0.93
55s
56s The classification report of Logistic Regressor is below :
57s
58s
59s
60s
61s
62s
63s
64s
65s
66s
67s
68s
69s
70s
71s
72s
73s
74s
75s
76s
77s
78s
79s
80s
81s
82s
83s
84s
85s
86s
87s
88s
89s
90s
91s
92s
93s
94s
95s
96s
97s
98s
99s
100s
101s
102s
103s
104s
105s
106s
107s
108s
109s
110s
111s
112s
113s
114s
115s
116s
117s
118s
119s
120s
121s
122s
123s
124s
125s
126s
127s
128s
129s
130s
131s
132s
133s
134s
135s
136s
137s
138s
139s
140s
141s
142s
143s
144s
145s
146s
147s
148s
149s
150s
151s
152s
153s
154s
155s
156s
157s
158s
159s
160s
161s
162s
163s
164s
165s
166s
167s
168s
169s
170s
171s
172s
173s
174s
175s
176s
177s
178s
179s
180s
181s
182s
183s
184s
185s
186s
187s
188s
189s
190s
191s
192s
193s
194s
195s
196s
197s
198s
199s
200s
201s
202s
203s
204s
205s
206s
207s
208s
209s
210s
211s
212s
213s
214s
215s
216s
217s
218s
219s
220s
221s
222s
223s
224s
225s
226s
227s
228s
229s
230s
231s
232s
233s
234s
235s
236s
237s
238s
239s
240s
241s
242s
243s
244s
245s
246s
247s
248s
249s
250s
251s
252s
253s
254s
255s
256s
257s
258s
259s
260s
261s
262s
263s
264s
265s
266s
267s
268s
269s
270s
271s
272s
273s
274s
275s
276s
277s
278s
279s
280s
281s
282s
283s
284s
285s
286s
287s
288s
289s
290s
291s
292s
293s
294s
295s
296s
297s
298s
299s
300s
301s
302s
303s
304s
305s
306s
307s
308s
309s
310s
311s
312s
313s
314s
315s
316s
317s
318s
319s
320s
321s
322s
323s
324s
325s
326s
327s
328s
329s
330s
331s
332s
333s
334s
335s
336s
337s
338s
339s
340s
341s
342s
343s
344s
345s
346s
347s
348s
349s
350s
351s
352s
353s
354s
355s
356s
357s
358s
359s
360s
361s
362s
363s
364s
365s
366s
367s
368s
369s
370s
371s
372s
373s
374s
375s
376s
377s
378s
379s
380s
381s
382s
383s
384s
385s
386s
387s
388s
389s
390s
391s
392s
393s
394s
395s
396s
397s
398s
399s
400s
401s
402s
403s
404s
405s
406s
407s
408s
409s
410s
411s
412s
413s
414s
415s
416s
417s
418s
419s
420s
421s
422s
423s
424s
425s
426s
427s
428s
429s
430s
431s
432s
433s
434s
435s
436s
437s
438s
439s
440s
441s
442s
443s
444s
445s
446s
447s
448s
449s
450s
451s
452s
453s
454s
455s
456s
457s
458s
459s
460s
461s
462s
463s
464s
465s
466s
467s
468s
469s
470s
471s
472s
473s
474s
475s
476s
477s
478s
479s
480s
481s
482s
483s
484s
485s
486s
487s
488s
489s
490s
491s
492s
493s
494s
495s
496s
497s
498s
499s
500s
501s
502s
503s
504s
505s
506s
507s
508s
509s
510s
511s
512s
513s
514s
515s
516s
517s
518s
519s
520s
521s
522s
523s
524s
525s
526s
527s
528s
529s
530s
531s
532s
533s
534s
535s
536s
537s
538s
539s
540s
541s
542s
543s
544s
545s
546s
547s
548s
549s
550s
551s
552s
553s
554s
555s
556s
557s
558s
559s
559s
560s
561s
562s
563s
564s
565s
566s
567s
568s
569s
570s
571s
572s
573s
574s
575s
576s
577s
578s
579s
580s
581s
582s
583s
584s
585s
586s
587s
588s
589s
589s
590s
591s
592s
593s
594s
595s
596s
597s
598s
599s
599s
600s
601s
602s
603s
604s
605s
606s
607s
608s
609s
609s
610s
611s
612s
613s
614s
615s
616s
617s
618s
619s
619s
620s
621s
622s
623s
624s
625s
626s
627s
628s
629s
629s
630s
631s
632s
633s
634s
635s
636s
637s
638s
639s
639s
640s
641s
642s
643s
644s
645s
646s
647s
648s
649s
649s
650s
651s
652s
653s
654s
655s
656s
657s
658s
659s
659s
660s
661s
662s
663s
664s
665s
666s
667s
668s
669s
669s
670s
671s
672s
673s
674s
675s
676s
677s
678s
678s
679s
680s
681s
682s
683s
684s
685s
686s
687s
687s
688s
689s
689s
690s
691s
692s
693s
694s
695s
696s
697s
697s
698s
699s
699s
700s
701s
702s
703s
704s
705s
706s
707s
708s
709s
709s
710s
711s
712s
713s
714s
715s
716s
717s
718s
719s
719s
720s
721s
722s
723s
724s
725s
726s
727s
728s
729s
729s
730s
731s
732s
733s
734s
735s
736s
737s
738s
738s
739s
739s
740s
741s
742s
743s
744s
745s
746s
747s
747s
748s
749s
749s
750s
751s
752s
753s
754s
755s
756s
757s
757s
758s
759s
759s
760s
761s
762s
763s
764s
765s
766s
767s
767s
768s
769s
769s
770s
771s
772s
773s
774s
775s
776s
777s
777s
778s
779s
779s
780s
781s
782s
783s
784s
785s
785s
786s
787s
787s
788s
789s
789s
790s
791s
792s
793s
794s
794s
795s
796s
796s
797s
798s
798s
799s
799s
800s
801s
802s
803s
804s
805s
806s
807s
808s
809s
809s
810s
811s
811s
812s
813s
813s
814s
815s
815s
816s
817s
817s
818s
819s
819s
820s
821s
821s
822s
823s
823s
824s
825s
825s
826s
827s
827s
828s
829s
829s
830s
831s
831s
832s
833s
833s
834s
835s
835s
836s
837s
837s
838s
839s
839s
840s
841s
841s
842s
843s
843s
844s
845s
845s
846s
847s
847s
848s
849s
849s
850s
851s
851s
852s
853s
853s
854s
855s
855s
856s
857s
857s
858s
859s
859s
860s
861s
861s
862s
863s
863s
864s
865s
865s
866s
867s
867s
868s
869s
869s
870s
871s
871s
872s
873s
873s
874s
875s
875s
876s
877s
877s
878s
879s
879s
880s
881s
881s
882s
883s
883s
884s
885s
885s
886s
887s
887s
888s
889s
889s
890s
891s
891s
892s
893s
893s
894s
895s
895s
896s
897s
897s
898s
899s
899s
900s
901s
901s
902s
903s
903s
904s
905s
905s
906s
907s
907s
908s
909s
909s
910s
911s
911s
912s
913s
913s
914s
915s
915s
916s
917s
917s
918s
919s
919s
920s
921s
921s
922s
923s
923s
924s
925s
925s
926s
927s
927s
928s
929s
929s
930s
931s
931s
932s
933s
933s
934s
935s
935s
936s
937s
937s
938s
939s
939s
940s
941s
941s
942s
943s
943s
944s
945s
945s
946s
947s
947s
948s
949s
949s
950s
951s
951s
952s
953s
953s
954s
955s
955s
956s
957s
957s
958s
959s
959s
960s
961s
961s
962s
963s
963s
964s
965s
965s
966s
967s
967s
968s
969s
969s
970s
971s
971s
972s
973s
973s
974s
975s
975s
976s
977s
977s
978s
979s
979s
980s
981s
981s
982s
983s
983s
984s
985s
985s
986s
987s
987s
988s
989s
989s
990s
991s
991s
992s
993s
993s
994s
995s
995s
996s
997s
997s
998s
999s
999s
1000s
1001s
1001s
1002s
1003s
1003s
1004s
1005s
1005s
1006s
1007s
1007s
1008s
1009s
1009s
1010s
1011s
1011s
1012s
1013s
1013s
1014s
1015s
1015s
1016s
1017s
1017s
1018s
1019s
1019s
1020s
1021s
1021s
1022s
1023s
1023s
1024s
1025s
1025s
1026s
1027s
1027s
1028s
1029s
1029s
1030s
1031s
1031s
1032s
1033s
1033s
1034s
1035s
1035s
1036s
1037s
1037s
1038s
1039s
1039s
1040s
1041s
1041s
1042s
1043s
1043s
1044s
1045s
1045s
1046s
1047s
1047s
1048s
1049s
1049s
1050s
1051s
1051s
1052s
1053s
1053s
1054s
1055s
1055s
1056s
1057s
1057s
1058s
1059s
1059s
1060s
1061s
1061s
1062s
1063s
1063s
1064s
1065s
1065s
1066s
1067s
1067s
1068s
1069s
1069s
1070s
1071s
1071s
1072s
1073s
1073s
1074s
1075s
1075s
1076s
1077s
1077s
1078s
1079s
1079s
1080s
1081s
1081s
1082s
1083s
1083s
1084s
1085s
1085s
1086s
1087s
1087s
1088s
1089s
1089s
1090s
1091s
1091s
1092s
1093s
1093s
1094s
1095s
1095s
1096s
1097s
1097s
1098s
1099s
1099s
1100s
1101s
1101s
1102s
1103s
1103s
1104s
1105s
1105s
1106s
1107s
1107s
1108s
1109s
1109s
1110s
1111s
1111s
1112s
1113s
1113s
1114s
1115s
1115s
1116s
1117s
1117s
1118s
1119s
1119s
1120s
1121s
1121s
1122s
1123s
1123s
1124s
1125s
1125s
1126s
1127s
1127s
1128s
1129s
1129s
1130s
1131s
1131s
1132s
1133s
1133s
1134s
1135s
1135s
1136s
1137s
1137s
1138s
1139s
1139s
1140s
1141s
1141s
1142s
1143s
1143s
1144s
1145s
1145s
1146s
1147s
1147s
1148s
1149s
1149s
1150s
1151s
1151s
1152s
1153s
1153s
1154s
1155s
1155s
1156s
1157s
1157s
1158s
1159s
1159s
1160s
1161s
1161s
1162s
1163s
1163s
1164s
1165s
1165s
1166s
1167s
1167s
1168s
1169s
1169s
1170s
1171s
1171s
1172s
1173s
1173s
1174s
1175s
1175s
1176s
1177s
1177s
1178s
1179s
1179s
1180s
1181s
1181s
1182s
1183s
1183s
1184s
1185s
1185s
1186s
1187s
1187s
1188s
1189s
1189s
1190s
1191s
1191s
1192s
1193s
1193s
1194s
1195s
1195s
1196s
1197s
1197s
1198s
1199s
1199s
1200s
1201s
1201s
1202s
1203s
1203s
1204s
1205s
1205s
1206s
1207s
1207s
1208s
1209s
1209s
1210s
1211s
1211s
1212s
1213s
1213s
1214s
1215s
1215s
1216s
1217s
1217s
1218s
1219s
1219s
1220s
1221s
1221s
1222s
1223s
1223s
1224s
1225s
1225s
1226s
1227s
1227s
1228s
1229s
1229s
1230s
1231s
1231s
1232s
1233s
1233s
1234s
1235s
1235s
1236s
1237s
1237s
1238s
1239s
1239s
1240s
1241s
1241s
1242s
1243s
1243s
1244s
1245s
1245s
1246s
1247s
1247s
1248s
1249s
1249s
1250s
1251s
1251s
1252s
1253s
1253s
1254s
1255s
1255s
1256s
1257s
1257s
1258s
1259s
1259s
1260s
1261s
1261s
1262s
1263s
1263s
1264s
1265s
1265s
1266s
1267s
1267s
1268s
1269s
1269s
1270s
1271s
1271s
1272s
1273s
1273s
1274s
1275s
1275s
1276s
1277s
1277s
1278s
1279s
1279s
1280s
1281s
1281s
1282s
1283s
1283s
1284s
1285s
1285s
1286s
1287s
1287s
1288s
1289s
1289s
1290s
1291s
1291s
1292s
1293s
1293s
1294s
1295s
1295s
1296s
1297s
1297s
1298s
1299s
1299s
1300s
1301s
1301s
1302s
1303s
1303s
1304s
1305s
1305s
1306s
1307s
1307s
1308s
1309s
1309s
1310s
1311s
1311s
1312s
1313s
1313s
1314s
1315s
1315s
1316s
1317s
1317s
1318s
1319s
1319s
1320s
1321s
1321s
1322s
1323s
1323s
1324s
1325s
1325s
1326s
1327s
1327s
1328s
1329s
1329s
1330s
1331s
1331s
1332s
1333s
1333s
1334s
1335s
1335s
1336s
1337s
1337s
1338s
1339s
1339s
1340s
1341s
1341s
1342s
1343s
1343s
1344s
1345s
1345s
1346s
1347s
1347s
1348s
1349s
1349s
1350s
1351s
1351s
1352s
1353s
1353s
1354s
1355s
1355s
1356s
1357s
1357s
1358s
1359s
1359s
1360s
1361s
1361s
1362s
1363s
1363s
1364s
1365s
1365s
1366s
1367s
1367s
1368s
1369s
1369s
1370s
1371s
1371s
1372s
1373s
1373s
1374s
1375s
1375s
1376s
1377s
1377s
1378s
1379s
1379s
1380s
1381s
1381s
1382s
1383s
1383s
1384s
1385s
1385s
1386s
1387s
1387s
1388s
1389s
1389s
1390s
1391s
1391s
1392s
1393s
1393s
1394s
1395s
1395s
1396s
1397s
1397s
1398s
1399s
1399s
1400s
1401s
1401s
1402s
1403s
1403s
1404s
1405s
1405s
1406s
1407s
1407s
1408s
1409s
1409s
1410s
1411s
1411s
1412s
1413s
1413s
1414s
1415s
1415s
1416s
1417s
1417s
1418s
1419s
1419s
1420s
1421s
1421s
1422s
1423s
1423s
1424s
1425s
1425s
1426s
1427s
1427s
1428s
1429s
1429s
1430s
1431s
1431s
1432s
1433s
1433s
1434s
1435s
1435s
1436s
1437s
1437s
1438s
1439s
1439s
1440s
1441s
1441s
1442s
1443s
1443s
1444s
1445s
1445s
1446s
1447s
1447s
1448s
1449s
1449s
1450s
1451s
1451s
1452s
1453s
1453s
1454s
1455s
1455s
1456s
1457s
1457s
1458s
1459s
1459s
1460s
1461s
1461s
1462s
1463s
1463s
1464s
1465s
1465s
1466s
1467s
1467s
1468s
1469s
1469s
1470s
1471s
1471s
1472s
1473s
1473s
1474s
1475s
1475s
1476s
1477s
1477s
1478s
1479s
1479s
1480s
1481s
1481s
1482s
1483s
1483s
1484s
1485s
1485s
1486s
1487s
1487s
1488s
1489s
1489s
1490s
1491s
1491s
1492s
1493s
1493s
1494s
1495s
1495s
1496s
1497s
1497s
1498s
1499s
1499s
1500s
1501s
1501s
1502s
1503s
1503s
1504s
1505s
1505s
1506s
1507s
1507s
1508s
1509s
1509s
1510s
1511s
1511s
1512s
1513s
1513s
1514s
1515s
1515s
1516s
1517s
1517s
1518s
1519s
1519s
1520s
1521s
1521s
1522
```

```
45  labels = np.asarray(labels).reshape(2, 2)
46  cmap = sns.light_palette("#4DAE4D", as_cmap=True)
47  sns.heatmap(cm, annot = labels, cmap = cmap, fmt = '')
48
49  macro avg      0.94      0.93      0.93      426
50  weighted avg   0.94      0.94      0.94      426
```



The confusion matrix is below :

<Axes: >



2. SVC Model Evaluation

```
✓ 1 svc_cv = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 4, random_state = 2)
  svc_scores = cross_val_score(svc, X_train, y_train, cv = svc_cv, scoring = 'roc_auc').mean()
  print(f"Mean Cross-Validation Score: {lg_scores:.2f}")
#-----
  print(f"The ROC AUC Score is: {roc_auc_score(y_test, lg_pred):.2f}", '\n\n\n')
#-----

  svc_report = classification_report(y_test, svc_pred)
  print('The classification report of Logistic Regressor is below : ', '\n\n\n', svc_report, '\n')
#-----

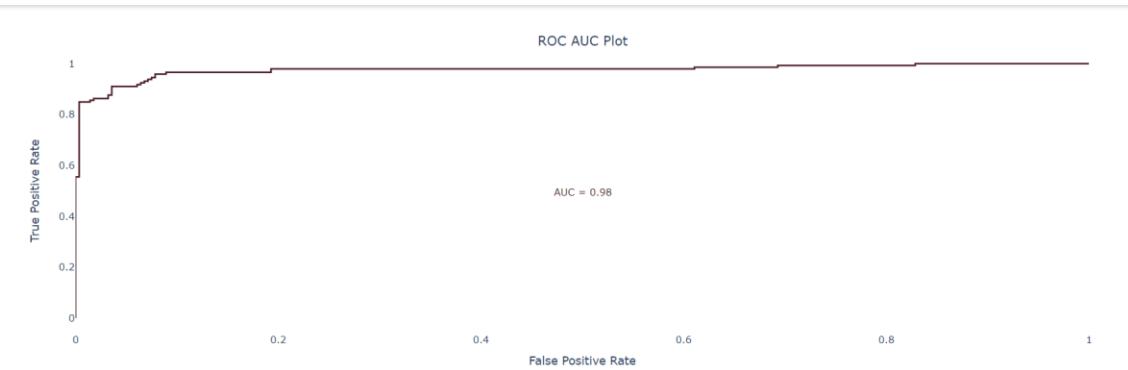
  fpr, tpr, _ = roc_curve(y_test, svc.decision_function(X_test))
  roc_auc = auc(fpr, tpr)
  fig = make_subplots(rows=1, cols=1, subplot_titles=['ROC AUC Plot'])
  fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', line=dict(color="#4A1A22")))
  fig.update_layout(xaxis_title='False Positive Rate', yaxis_title='True Positive Rate', plot_bgcolor='white')
  fig.add_annotation(x=0.5, y=0.5, text=f'AUC = {roc_auc:.2f}', showarrow=False, font=dict(color="#4A1A22"))
  fig.show(fig)
#-----
```

Mean Cross-Validation Score: 0.97

The ROC AUC Score is: 0.93

The classification report of Logistic Regressor is below :

	precision	recall	f1-score	support
0	0.93	0.98	0.95	280
1	0.95	0.86	0.91	146
accuracy			0.94	426
macro avg	0.94	0.92	0.93	426
weighted avg	0.94	0.94	0.94	426

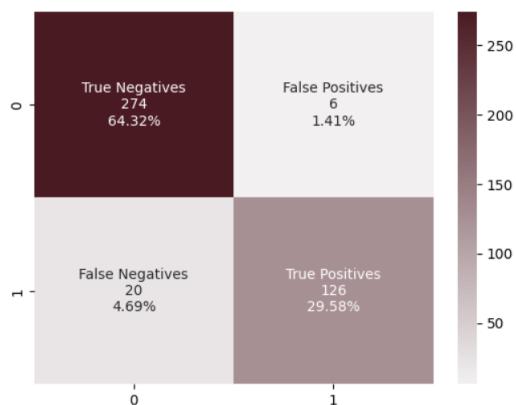


✓ [42]

False Positive Rate

The confusion matrix is below :

<Axes: >



💡 Insights:

- The performance evaluation revealed that the Logistic Regression performed well and gave an f1-score of 0.96. The precision-recall curve is indication a great performance with almost touching the top right corner having recall on x axis and precision on y axis

9. Financial equation:

Total Cost = Detection Costs + Fraud Prevention and Loss Mitigation costs

Total Benefits = Fraud Avoidance + Reduced Chargebacks + Customer Experience and Reputation Benefits

Net Financial Impact = Total Benefits – Total Cost

In the context of the financial equation for a credit card fraud detection system, if we want to represent it in the form of $(y = mx + c)$, where (y) is the dependent variable (Net Financial Impact), (x) is an independent variable (Total Benefits), and (c) is a constant term representing the intercept, we can rearrange the equation as follows:

Let:

- y represents the Net Financial Impact.

- x represents the Total Benefits.
- m represents the slope or coefficient associated with the Total Benefits.
- c represents the constant term or intercept.

The equation in the form of ($y = mx + c$) would be:

$$y = mx + c$$

Given:

$$\{\text{Total Cost}\} = C_d + C_p$$

$$\{\text{Total Benefits}\} = B_f + B_c + B_e$$

$$\{\text{Net Financial Impact}\} = \{\text{Total Benefits}\} - \{\text{Total Cost}\}$$

We rearrange the equation to fit the form ($y = mx + c$):

$$\{\text{Net Financial Impact}\} = (B_f + B_c + B_e) - (C_d + C_p)$$

This equation can be represented as ($y = mx + c$) by defining:

- y as Net Financial Impact.
- x as Total Benefits.
- m as 1 (since it's a 1:1 relationship between Total Benefits and Net Financial Impact).
- c as $-(C_d + C_p)$ (the constant term or intercept).

So, in the form ($y = mx + c$), the equation for the financial impact of a credit card fraud detection system could be written as:

$$y = x - (C_d + C_p)$$

This representation reflects the relationship between the Net Financial Impact and the Total Benefits of implementing the fraud detection system, considering the costs associated with detection and prevention.

10. Conclusion:

A real-time financial fraud detection system, using artificial intelligence, machine learning, and blockchain technology, could significantly protect businesses and consumers from financial losses. It would be scalable and extensible, allowing for easy adaptation to new transactions and fraud schemes. However, the system's development and deployment require careful consideration of cost, complexity, and implementation challenges.

Real-time financial fraud detection systems are crucial for safeguarding financial transactions and protecting organizations and consumers from fraudulent activities. These systems use machine learning algorithms and rules-based logic to process data from various sources and generate alerts when suspicious activities are detected. A diverse team of experts is needed to develop and deploy the system, considering factors like data sources, technology stack, team composition, and cost estimation. Continuous improvement and adaptation are essential for staying ahead of emerging fraud tactics.

