CREDIT CARD FRAUD DETECTION USING DATA SCIENCE

TEAM MEMBER

511821104034: SEETHANA.A

PODHIGAI

COLLEGE OF ENGINEERING ANDTECHNOLOGY

Phase 2: Document Submission

Name of the Project

- **CREDIT CARD FRAUD DETECTION**

Problems in existing:

1. Financial Loss: Victims of credit card fraud can suffer financial losses when unauthorized transactions are made using their cards. These losses can be particularly damaging if the victim is not promptly reimbursed by their bank or credit card issuer.

2. Identity Theft: Credit card fraud often involves stolen personal information, which can lead to broader identity theft issues. Criminals may use the stolen data to open new accounts, apply for loans, or commit other types of fraud.

3. Security Risks: Weaknesses in payment systems and data breaches can expose consumers' sensitive information to

hackers and criminals. This can erode trust in online and offline payment methods.

4. Inconvenience: Dealing with the aftermath of credit card fraud can be time-consuming and frustrating. Victims may need to cancel cards, dispute charges, and update their information with various merchants and financial institutions.

5. Impact on Businesses: Credit card fraud can harm businesses by leading to chargebacks, loss of revenue, and damage to their reputation. Merchants may also incur costs related to fraud prevention measures.

6. Legal and Regulatory Challenges: Financial institutions and businesses must comply with laws and regulations related to fraud prevention and data security. Failing to do so can result in legal and financial penalties.

7. Technological Advancements: Fraudsters are constantly developing new techniques and technologies to commit credit card fraud. Keeping up with these evolving threats can be a challenge for security professionals.

8. International Nature: Credit card fraud can Ad national borders, making it difficult to track and apprehend perpetrators who may operate from countries with lax law enforcement.

Solutions for the problems:

- EMV Chip Technology: EMV (Europay, Mastercard, and Visa) chip cards provide better security compared to traditional magnetic stripe cards. The chips generate unique transaction codes for

each purchase, making it harder for fraudsters to create counterfeit cards.

- Two-Factor Authentication (2FA): Implement 2FA for online transactions, requiring an additional verification step beyond just entering the card details. This can include SMS codes, biometrics, or authentication a

- Regular Account Monitoring: Review your credit card statements and transactions frequently, either through online banking or mobile apps. Report any suspicious or unauthorized transactions immediately.

- Card Activation and Deactivation: Enable the ability to activate and deactivate your card temporarily through your bank's mobile app. This feature can help protect your card if it's misplaced or stolen.

- Secure Online Shopping: Only use your credit card on reputable and secure websites. Look for HTTPS in the URL, use virtual credit cards for online purchases when possible, and avoid saving card details on websites.

- Strong Passwords and PINs: Use complex, unique passwords for online accounts associated with your credit cards. Avoid using easily guessable PINs, such as your birthdate or "1234."Secure Your Physical Card: Keep your credit card in a secure wallet or cardholder to prevent theft or skimming. Don't leave it in plain sight in your car or home.

- Beware of Phishing: Be cautious of emails, messages, or phone calls asking for your credit card information. Legitimate organizations will not request such information through unsolicited communication.

- Software Updates: Ensure your devices, including smartphones and computers, are updated with the latest security patches and antivirus software.

- Shredding Documents: Properly dispose of sensitive documents containing credit card information by shredding them before disposal.

- Credit Monitoring Services: Consider using credit monitoring services that can alert you to any unusual activity or new accounts opened in your name.

- Report Lost or Stolen Cards: Immediately report lost or stolen credit cards to your bank or credit card issuer. They can deactivate the card to prevent unauthorized use.

- Educate Yourself: Stay informed about the latest fraud tactics and scams. Awareness is key to avoiding potential traps.

- Use Wallet Apps: Mobile wallet apps like Apple Pay, Google Pay, or Samsung Pay provide an extra layer of security by replacing your card details with tokenized data during transactions.

- Employ AI and Machine Learning: Credit card companies and banks often use AI and machine learning algorithms to detect unusual patterns and transactions, helping to identify potential fraud.

Platform Needed:

✞ Machine Learning and Data Analytics Platforms:

Python: Python is a popular programming language for building machine learning models and data analysis. Libraries like scikit-learn and TensorFlow can be useful.

R: R is another language commonly used for statistical analysis and machine learning.

Jupyter Notebooks: Jupyter provides an interactive environment for data exploration and model development.

Language Needed:

○ Python ○
R: R ○ Java ○ C++ ○ SQL ○
Scala ○ Javascript **Front End:**

Web-Based Interfaces: HTML/CSS:.

HTML is used for structuring web content, while CSS is used for styling and layout.

JavaScript: JavaScript is essential for adding interactivity to web interfaces. Frameworks like React, Angular, or Vue.js can be used for building dynamic front-end applications.

Back End:

1. Programming Languages:
   - ✞ Java: Java is a popular choice for building the back end of fraud detection systems due to its performance, scalability, and extensive libraries.
   - ✞ Python: Python is commonly used for data processing, machine learning model training, and data analysis in the back end.

- Scala: Scala is useful for building highly scalable and concurrent backend systems, often used with technologies like Apache Spark.

Database:

1.Database Management Systems (DBMS):

a. Relational Databases: MySQL, PostgreSQL, Oracle, or Microsoft SQL Server are used for storing transaction data and fraud-related information.

b.NoSQL Databases: MongoDB, Cassandra, or Redis may be used for handling unstructured or semi-structured data, such as logs and event streams.

2.Big Data Processing:

a. Apache Hadoop: Hadoop is used for distributed storage and batch processing of large datasets.

b.Apache Spark: Spark is ideal for real-time and batch data processing, as well as machine learning tasks.

3.Stream Processing

a. Apache Kafka: Kafka is used for real-time data streaming, enabling the system to ingest and process transaction data as it arrives.

b. Apache Flink: Flink is used for stream processing and complex event processing (CEP).

4.Machine Learning Frameworks:

a. Scikit-Learn, TensorFlow, and PyTorch: These libraries are used for building, training, and deploying machine learning models to detect fraud patterns.

Advantages:

Reduced Financial Losses: The primary benefit of credit card fraud detection is the ability to identify and prevent unauthorized transactions promptly. This helps financial institutions and

cardholders avoid substantial financial losses resulting from fraudulent activities.

Enhanced Security: These systems provide an additional layer of security by continuously monitoring transactions for suspicious patterns or anomalies. This contributes to a safer and more secure payment environment.

Real-Time Detection: Many credit card fraud detection systems operate in real-time, allowing for the immediate identification of fraudulent transactions as they occur. This quick response can prevent further unauthorized activity.

Minimized Impact on Cardholders: Prompt detection means that cardholders are less likely to be inconvenienced by the need to dispute charges and wait for reimbursement. They can also receive alerts about suspicious activities, helping them take action swiftly.

Preservation of Trust: Effective fraud detection helps preserve trust in payment systems and financial institutions. Customers are more likely to continue using credit cards if they feel their funds are secure.

Data Analysis for Pattern Recognition: These systems analyze historical transaction data to identify trends and patterns associated with fraudulent activities. This data-driven approach enables more accurate detection.

Adaptability to New Fraud Schemes: As fraud tactics evolve, fraud detection systems can adapt and learn from new patterns, helping stay ahead of emerging threats.

Reduced False Positives: While preventing fraud is crucial, minimizing false positives (legitimate transactions mistakenly flagged as fraudulent) is equally important. Modern systems use advanced

algorithms to reduce false positives, reducing inconvenience for cardholders.

Efficient Investigation: Fraud detection systems often provide tools for investigators to review flagged transactions and gather evidence efficiently. This streamlines the investigation process.

Customization and Configurability: Financial institutions can customize the rules and parameters of their fraud detection systems to align with their risk tolerance and specific needs.

Compliance with Regulations: Credit card fraud detection systems help financial institutions meet regulatory requirements related to fraud prevention and customer protection.

Cost Savings: While implementing and maintaining fraud detection systems incurs costs, these are often outweighed by the savings resulting from prevented fraud losses and reduced administrative expenses related to fraud resolution.

Consumer Confidence: The presence of robust fraud detection systems can boost consumer confidence in credit card usage, leading to increased cardholder adoption and usage.

Reference:

- Academic Journals: Academic journals often contain research articles and papers on various aspects of credit card fraud detection. Search databases like PubMed, IEEE Xplore, or Google Scholar using keywords like “credit card fraud detection” or “credit card fraud prevention.”

- Books: Look for books related to fraud detection, machine learning, or data analytics in your local library or online retailers. Some books may specifically focus on credit card fraud detection.

- Online Courses and Tutorials: Online learning platforms like Coursera, edX, and Udemy offer courses on data science, machine learning, and fraud detection, which may include sections on credit card fraud detection.

- Research Organizations and Government Agencies: Websites of organizations like the Federal Trade Commission (FTC) or the Payment Card Industry Security Standards Council (PCI SSC) often provide reports and resources related to credit card fraud detection.

- Industry Reports: Market research firms and financial institutions sometimes publish reports on fraud trends and detection methods. Look for reports from companies like Experian, TransUnion, or Deloitte.

- Blogs and News Articles: Stay updated on the latest developments in credit card fraud detection by following industry-specific blogs and news outlets. Websites like KrebsOnSecurity and Dark Reading often cover cybersecurity topics, including fraud detection.

- Whitepapers: Many cybersecurity and financial technology companies publish whitepapers on fraud detection techniques and best practices. These can often be found on the websites of these companies.

- Conference Proceedings: Look for conference proceedings related to cybersecurity, data analytics, or machine learning. Conferences like RSA Conference and Black Hat often feature presentations and research on fraud detection.

Data Models:

- Supervised Learning Models:
  Logistic Regression: It's a commonly used model for binary classification problems like fraud detection.

- Random Forest: Effective in handling imbalanced datasets and capturing complex patterns.
- Support Vector Machines (SVM): Useful for separating data into fraudulent and non-fraudulent classes
- Deep Learning Models:
  Convolutional Neural Networks (CNNs): For detecting patterns in images on credit cards.
- Recurrent Neural Networks (RNNs): Can be used for time-series analysis when considering the sequence of transactions.
- Anomaly Detection Models:
  Isolation Forest: Identifies anomalies by isolating them in the data.
- One-Class SVM: Learns the distribution of non-fraudulent transactions and detects outliers.
- Ensemble Models:
  XGBoost:
  Often used for ensemble learning to improve the model's performance.
  Gradient
- Boosting: Combines multiple models to create a robust fraud detection system
- Unsupervised Learning Models:
  K-Means Clustering: Can be used to group transactions into clusters and detect unusual clusters.
- DBSCAN: Useful for density-based clustering of transactions.
- Time-Series Analysis:
  Using models like ARIMA (AutoRegressive Integrated Moving Average) to analyze and forecast time-series data of transactions.
- Feature Engineering:
- Creating relevant features such as transaction amount, location, time, and historical behavior of the cardholder.
- Imbalanced Data Handling:

Techniques like oversampling, undersampling, or Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance in the dataset.

- Real-time Monitoring:
  Implementing rules-based systems that trigger alerts based on predefined thresholds or patterns
- Behavioral Analysis:
  Analyzing the behavioral patterns of cardholders to detect anomalies in their spending habits
- Feature Selection:
  Using techniques like Recursive Feature Elimination (RFE) to select the most important features for the model.
- Model Evaluation and Validation:
  Using metrics like precision, recall, F1-score, and ROC-AUC to assess the model's performance.

Sample source code:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul  4 18:05:05 2019

@author: Ravi
"""



# import the required packages


Import numpy as np
Import pandas as pd
```

```
Import matplotlib.pyplot as plt

Import seaborn as sns

Import sys

Import scipy


# load the dataset using pandas

Data =
pd.read_csv(r'C:\Users\Ravi\Downloads\creditcardfraud\creditcard.c
sv')


# dataset exploring

Print(data.columns)


# Print the shape of the data

Data = data.sample(frac=0.1, random_state = 1)

Print(data.shape)

Print(data.describe())


# V1 – V28 are the results of a PCA Dimensionality reduction to
protect user identities and sensitive features


# Plot histograms of each parameter

Data.hist(figsize = (20, 20))

Plt.show()
```

```
# Determine number of fraud cases in dataset

Fraud = data[data['Class'] == 1]
Valid = data[data['Class'] == 0]

Outlier_fraction = len(Fraud)/float(len(Valid))
Print(outlier_fraction)

Print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
Print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))

# Correlation matrix
Corrmat = data.corr()
Fig = plt.figure(figsize = (12, 9))

Sns.heatmap(corrmat, vmax = .8, square = True)
Plt.show()

# Get all the columns from the dataFrame
Columns = data.columns.tolist()

# Filter the columns to remove data we do not want
Columns = [c for c in columns if c not in ["Class"]]
```

```
# Store the variable we'll be predicting on
Target = "Class"

X = data[columns]
Y = data[target]

# Print shapes
Print(X.shape)
Print(Y.shape)

From sklearn.metrics import classification_report, accuracy_score
From sklearn.ensemble import IsolationForest
From sklearn.neighbors import LocalOutlierFactor

# define random states
State = 1

# define outlier detection tools to be compared
Classifiers = {
"Isolation Forest": IsolationForest(max_samples=len(X),
Contamination=outlier_fraction,
Random_state=state),
"Local Outlier Factor": LocalOutlierFactor(
N_neighbors=20,
```

```
Contamination=outlier_fraction)}

Plt.figure(figsize=(9, 7))
N_outliers = len(Fraud)


For I, (clf_name, clf) in enumerate(classifiers.items()):

# fit the data and tag outliers
If clf_name == “Local Outlier Factor”:
Y_pred = clf.fit_predict(X)
Scores_pred = clf.negative_outlier_factor_
Else:
Clf.fit(X)
Scores_pred = clf.decision_function(X)
Y_pred = clf.predict(X)

# Reshape the prediction values to 0 for valid, 1 for fraud.
Y_pred[y_pred == 1] = 0
Y_pred[y_pred == -1] = 1

N_errors = (y_pred != Y).sum()

# Run classification metrics
```

```
Print('{}: {}'.format(clf_name, n_errors))
Print(accuracy_score(Y, y_pred))
Print(classification_report(Y, y_pred
```