**Credit Card Fraud Detection(Development Part-2)**

Detecting credit card fraud using data science typically involves building a machine learning model to identify potentially fraudulent transactions. Here’s a high-level guide to developing such a system, along with some code examples in Python. Please note that this is a simplified example, and real-world credit card fraud detection systems are much more complex and use extensive datasets.

1. **Data Collection:**

Gather a dataset of credit card transactions. You can use public datasets like Kaggle’s Credit Card Fraud Detection dataset.

Python code

Import pandas as pd

Df = pd.read\_csv(“creditcard.csv”)

1. **Data Preprocessing**:

Explore and clean the data. Check for missing values and outliers.

Normalize numerical features, like ‘Amount’ and ‘Time’.

From sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()

Df[‘normalized\_Amount’] = scaler.fit\_transform(df[‘Amount’].values.reshape(-1, 1))

1. **Feature Selection:**

Select relevant features for the model. In this case, all features except ‘Time’ and ‘Amount’ may be relevant.

X = df.drop([‘Time’, ‘Amount’, ‘Class’], axis=1)

Y = df[‘Class’]

1. **Split Data:**

Split the data into training and testing sets.

Python code

From sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. **Model Selection:**

Choose an appropriate machine learning algorithm for classification. Random Forest and Gradient Boosting are often used for this task.

**Python code**

From sklearn.ensemble import RandomForestClassifier

Model = RandomForestClassifier()

1. **Model Training:**

Train the model on the training data.

Python

Copy code

Model.fit(X\_train, y\_train)

1. **Model Evaluation:**

Evaluate the model’s performance using metrics like accuracy, precision, recall, F1-score, and ROC AUC.

**Python code**

Print(confusi From sklearn.metrics import classification\_report, confusion\_matrix

Y\_pred = model.predict(X\_test)

Print(classification\_report(y\_test, y\_pred))

on\_matrix(y\_test, y\_pred))

1. **Hyperparameter Tuning:**

Optimize the model’s hyperparameters to improve performance.

1. **Deployment**:

Once satisfied with the model’s performance, deploy it in a production environment where it can continuously monitor incoming transactions for fraud.

Please note that a real-world system would also include ongoing monitoring, retraining, and integration with a transaction processing system. Additionally, dealing with imbalanced datasets (few fraud cases compared to non-fraud) requires advanced techniques like oversampling, undersampling, or using anomaly detection algorithms.

**Done by:**

**Jeripiti Venkata Tarun**

**Reg no:720921244023**