**TASK #5**

**\*Build a machine learning model to identify fraudulent credit card**

**transactions.**

**\*Preprocess and normalize the transaction data, handle class**

**imbalance issues, and split the dataset into training and testing sets.**

**\*Train a classification algorithm, such as logistic regression or random**

**forests, to classify transactions as fraudulent or genuine.**

**\*Evaluate the model'**

s performance using metrics like precision, recall,

and F1-score, and consider techniques like oversampling or

undersampling for improving results.

Building a machine learning model to identify fraudulent credit card transactions involves several steps:

1. Data Preprocessing:

- Handle missing values (if any)

- Normalize transaction amount and other numerical features

- Convert categorical features (e.g., card type, country) into numerical variables

2. Class Imbalance Handling:

- Fraudulent transactions are typically a small percentage of the total transactions (class imbalance)

- Techniques to handle class imbalance:

- Oversampling the minority class (fraudulent transactions)

- Undersampling the majority class (genuine transactions)

- Synthetic data generation (e.g., SMOTE)

3. Data Split:

- Split the preprocessed data into training (70-80%) and testing sets (20-30%)

4. Model Training:

- Train a classification algorithm (e.g., logistic regression, random forests, or more advanced models like XGBoost or LightGBM)

5. Model Evaluation:

- Metrics for evaluation:

- Precision (fraudulent transactions correctly identified)

- Recall (genuine transactions correctly identified)

- F1-score (balance between precision and recall)

- Consider using techniques like cross-validation to ensure robust evaluation

Let's implement this using Python and scikit-learn:

import pandas as pd

from sklearn.preprocessing import StandardScaler

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import precision\_score, recall\_score, f1\_score

from imblearn.over\_sampling import SMOTE

# Load the dataset

df = pd.read\_csv('transactions.csv')

# Preprocess data

scaler = StandardScaler()

df[['amount', 'other\_numerical\_features']] = scaler.fit\_transform(df[['amount', 'other\_numerical\_features']])

# Handle class imbalance using SMOTE

smote = SMOTE(random\_state=42)

X = df.drop('is\_fraud', axis=1)

y = df['is\_fraud']

X\_smote, y\_smote = smote.fit\_resample(X, y)

# Split data into training and testing sets

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

# Train a logistic regression model

model = LogisticRegression()

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

# Evaluate the model

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

print("Precision:", precision\_score(y\_test, y\_pred))

print("Recall:", recall\_score(y\_test, y\_pred))

print("F1-score:", f1\_score(y\_test, y\_pred))

This is a basic example, and you may need to adjust the preprocessing steps and model parameters based on your specific dataset. Additionally, consider using more advanced techniques like feature engineering, hyperparameter tuning, and ensemble methods to improve the model's performance.