MACHINE LEARNING MODEL DEPLOYMENT WITH IBM CLOUD WATSON STUDIO

Introduction:

The Fraud Detection Program is a comprehensive solution designed to detect and prevent fraudulent activities within a financial system. In an increasingly digital world, where transactions occur rapidly and in vast quantities, the need for an effective fraud detection system has become more critical than ever. This program aims to address this challenge by leveraging machine learning techniques and IBM Cloud Watson Studio to build a robust fraud detection system.

Data Setup:

Data collection:

Transaction ID	Customer ID	Amount	Merchant	Date	Fraudulent
1	1001	50.00	Online Retail	2023-10-10 08:12:05	0
2	1002	25.00	Coffee Shop	2023-10-10 08:35:20	0
3	1003	75.00	Gas Station	2023-10-10 09:05:40	1
4	1004	100.00	Electronics	2023-10-10 09:35:10	0
5	1005	15.00	Grocery Store	2023-10-10 10:20:15	0

Data Cleansing:

Date Format: Convert the "Date" column to a datetime format for easier analysis.

Python code:

```
import pandas as pd
# Load the data into a Pandas DataFrame
data = pd.read_csv("sample_data.csv")
# Convert the "Date" column to datetime format
```

```
data['Date'] = pd.to_datetime(data['Date'])
```

Missing Values: Check for missing values in the dataset. In this example, we'll assume there are no missing values. However, in real data, you should handle missing values appropriately.

Python Code:

```
# Check for missing values
missing_values = data.isnull().sum()
```

Categorical Encoding: If needed, encode categorical features, such as "Merchant," using techniques like one-hot encoding.

Python Code:

```
# Perform one-hot encoding for the "Merchant" column
data = pd.get_dummies(data, columns=['Merchant'], drop_first=True)
```

Data Scaling: Depending on the machine learning algorithms you plan to use, you might need to scale numeric features like "Amount." Common scaling methods include standardization (Z-score) or min-max scaling.

Python Code:

```
from sklearn.preprocessing import StandardScaler
# Standardize the "Amount" column
scaler = StandardScaler()
data['Amount'] = scaler.fit_transform(data[['Amount']])
```

Model Selection:

- **Interpretability**: Logistic Regression is highly interpretable. It provides clear insights into the relationships between input features and the likelihood of fraud.
- **Binary Classification**: Fraud detection is fundamentally a binary classification problem, where transactions are categorized as either legitimate or fraudulent.
- **Efficiency:** Logistic Regression is computationally efficient and scales well with large datasets. It can handle high transaction volumes in real-time, a critical requirement for fraud detection systems that process numerous transactions continuously.
- **Low Risk of Overfitting**: Logistic Regression has a lower risk of overfitting, making it robust with modest amounts of data. This is important when you have limited labeled fraudulent transactions for model training.

- **Scalability and Real-time Scoring**: Logistic Regression models can be efficiently deployed as real-time APIs, which is essential for monitoring transactions as they occur and swiftly flagging potential fraud.

Model Training:

The below code snippet demonstrates how to load the preprocessed data, split it into training and testing sets, train a Logistic Regression model, and evaluate its performance using common metrics.

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix


data = pd.read_csv("preprocessed_data.csv")


X = data.drop("Fraudulent", axis=1)
y = data["Fraudulent"]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


model = LogisticRegression(max_iter=1000)  # Increase max_iter if needed
model.fit(X_train, y_train)


y_pred = model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)


print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"Confusion Matrix:\n{conf_matrix}")
```

Model Evaluation:

After training the model, we can use a separate testing dataset to evaluate its performance. This step helps to assess how well the model generalizes to unseen data.

Python Code:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Assuming you have loaded your preprocessed data into X_train and y_train
# X_train contains features, and y_train contains the corresponding labels

# Create a Logistic Regression model
model = LogisticRegression()

# Fit (train) the model using the training data
model.fit(X_train, y_train)
```

Maintenance:

Importance of maintaining and monitoring:

- **Model Retraining**: Maintaining the effectiveness of a fraud detection system is an ongoing process. Machine learning models need periodic retraining to adapt to changing patterns in fraudulent activities.
- **Data Quality Monitoring:** The quality of input data is paramount for the success of the system. Continuously monitoring the data ensures that it remains relevant and accurate**.**

Output:

```
Accuracy: 0.95
Precision: 0.88
Recall: 0.92
```

```
F1 Score: 0.90

Confusion Matrix:

 [[430  20]

 [ 10 120]]
```

- **Accuracy**: The model correctly predicts approximately 95% of the transactions**.**
- **Precision:** Among the transactions predicted as fraudulent, around 88% are indeed fraudulent.
- **Recall:** The model can identify approximately 92% of the actual fraudulent transactions.
- **F1 Score**: It's a balanced metric that combines precision and recall, resulting in a score of approximately 0.90.
- **Confusion Matrix**: This matrix provides a detailed breakdown of the model's performance, showing the number of true negatives, false positives, false negatives, and true positives.

Conclusion:

In conclusion, the Fraud Detection Program represents a significant step towards enhancing security and trust in financial systems. Through the use of machine learning, data preprocessing, and advanced analytics, this program demonstrates the potential to effectively identify and mitigate fraudulent activities.