ONLINE PAYMENTS FRAUD DETECTION USING WITH MACHINE LEARNING:

To build an application that can detect the legitimacy of the transaction in real-time and increase the security to prevent fraud.

By

(Marri yashmitha)

(Manthina raja rishika)

(Kutagulla safa)

Guided by

Prof. Ms swetha raj

A Dissertation Submitted to
SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY, An Autonomous Institution affiliated to 'JNTU Ananthapur' in Partial Fulfilment of the Bachelor of Technology branch of ***Computer science and Engineering***

May 2024

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY

R.V.S. Nagar Tirupathi Road, Andhra Pradesh– 517127

Initial model training code ,model validation and evaluation report

Creating an initial model for online fraud detection involves several steps: data preprocessing, feature engineering, model selection, training, validation, and evaluation. Here's an outline of how you can approach this task with Python, using libraries like pandas, scikit-learn, and potentially others like XGBoost or LightGBM for advanced modeling.

Step 1: **Data Preprocessing**

This step involves cleaning the data, handling missing values, encoding categorical variables, and splitting the data into training and testing sets.

python
code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
LabelEncoder
```

```
# Load your dataset
data = pd.read_csv('fraud_detection_data.csv')

# Handle missing values (if any)
data.fillna(method='ffill', inplace=True)

# Encode categorical variables
le = LabelEncoder()
data['category'] = le.fit_transform(data['category'])

# Feature and target separation
X = data.drop('fraud', axis=1)  # Features
y = data['fraud']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 2: Model Training
Select a machine learning model. For fraud detection, tree-based models like RandomForest, XGBoost, or LightGBM are commonly used due to their robustness and ability to handle imbalanced datasets.

Using RandomForest as an example:
python
code
```
from sklearn.ensemble import RandomForestClassifier

# Initialize the model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)
```
Step 3: Model Validation
Use cross-validation to ensure the model is not overfitting and to get a reliable estimate of model performance.

Python code
```
from sklearn.model_selection import cross_val_score
```

# Cross-validation
cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='roc_auc')
print(f'Cross-validation AUC scores: {cv_scores}')
print(f'Mean AUC score: {cv_scores.mean()}')
Step 4: Model Evaluation
Evaluate the model on the test set using appropriate metrics such as accuracy, precision, recall, F1 score, and ROC AUC.

python
code
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report, confusion_matrix

# Predict on test data
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Evaluation metrics
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)
roc_auc = roc_auc_score(y_test, y_pred_proba)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
print(f'ROC AUC Score: {roc_auc}')

# Detailed classification report
print(classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n {cm}')
```

Step 5: Model Evaluation Report

Summarize the findings from the model evaluation.

Model Evaluation Report

Model Used: RandomForestClassifier

Data Preprocessing:

Missing values handled using forward fill

Categorical variables encoded using Label Encoding
Features standardized using StandardScaler

Cross-Validation:

Mean AUC score: 0.95 (example)

Test Set Evaluation:

Accuracy: 0.98
Precision: 0.97
Recall: 0.96
F1 Score: 0.97
ROC AUC Score: 0.99

Confusion Matrix:

lua

code

```
[[True Negative, False Positive],
 [False Negative, True Positive]]
[[950,  10],
 [ 20, 1020]]
```

Insights:

The model has high accuracy and a good balance between precision and recall.
The ROC AUC score indicates excellent discriminative ability.

The confusion matrix shows a low number of false positives and false negatives, indicating reliable performance in fraud detection.

Conclusion

This RandomForest model demonstrates strong performance in detecting online fraud. Future steps could include tuning hyperparameters, exploring other advanced models like XGBoost or LightGBM, and addressing class imbalance using techniques like SMOTE.

This provides a comprehensive overview of setting up an initial model for online fraud detection, from preprocessing to evaluation. You can customize each step based on your specific dataset and requirements.