Applied Analytics Project - Week 12 Assignment
Save and Package Your Model for Deployment

Brady Levenson
ADAN 8888 - Applied Analytics Project
Professor Nurtekin Savas
November 24, 2024

Revisitation of Problem Statement for Motivation to Save and Package Model for Deployment

It is known that fraudulent financial transactions occur on a regular basis, but it is difficult to distinguish between them and legitimate financial transactions simply by viewing the transactions. The inability to correctly classify fraudulent financial transactions can promote uncontrolled and untraceable levels of financial losses for banks, investment firms, corporations, and numerous other entities. We are given a dataset of financial transactions and their unique properties. We will analyze this dataset using various machine learning techniques to determine effective measures to identify historical fraudulent transactions and prevent their occurrence in the future.

Saving Model for Deployment as a Pickle File

To save our final model for deployment as a pickle file, we fit the model to a training data set, tested it on a test dataset, and used the pickle.dump() function to serialize the model. The primary objective of using pickle to save our model is the ability to load the model in a variety of different environments while preserving the parameter settings of the entire model. This allows for the model to be deployed seamlessly by a wide range of companies or other stakeholders with minimal intervention.

Pickle Implementation

```
import pickle

# Fit the model to training data
kmeans.fit(X_train)

# Use the fitted model to predict cluster labels for test data
predicted_labels = kmeans.predict(X_test)

# Save model using pickle
with open('kmeans.pkl', 'wb') as f:  # 'wb' = writing in binary mode
    pickle.dump(kmeans, f)

# Step 4: Load the model back from the file
with open('kmeans.pkl', 'rb') as f:  # 'rb' = reading in binary mode
    loaded_model = pickle.load(f)

# Use the loaded model to predict cluster labels for the test data
predicted_labels = loaded_model.predict(X_test)

# Test the loaded model
silhouette_score = silhouette_score(X_test, predicted_labels)
print(f"Silhouette Score of the loaded model: {silhouette_score:.4f}")
```

```
Silhouette Score of the loaded model: 0.4719
```

Environment Dependency Documentation

This Python-based project was built in an environment with the below specifications.

```
Operating System: Windows
OS Version: 11
Full OS Version: 10.0.22631
Python Version: 3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.1929 64 bit (AMD64)]
```

I have identified all of the non-default packages (packages not automatically included in Python environments) that have been utilized in the construction of this project.

Sample Segment of Python Packages Used List and Their Versions

```
{'automat': '20.2.0',
 'babel': '2.11.0',
 'bottleneck': '1.3.7',
 'brotli': '1.0.9',
 'gitpython': '3.1.37',
 'heapdict': '1.0.1',
 'markdown': '3.4.1',
 'markupsafe': '2.1.3',
 'protego': '0.1.16',
 'pydispatcher': '2.0.5',
 'pyjwt': '2.8.0',
 'pynacl': '1.5.0',
 'pyqt5': '5.15.10',
 'pyqt5-sip': '12.13.0',
 'pyqtwebengine': '5.15.6',
 'pysocks': '1.7.1',
 'pyyaml': '6.0.1',
 'pygments': '2.15.1',
 'qdarkstyle': '3.2.3',
 'qtawesome': '1.2.2',
 'qtpy': '2.4.1',
 'rtree': '1.0.1',
 'sqlalchemy': '2.0.30',
 'scrapy': '2.11.1',
 'send2trash': '1.8.2',
 'unidecode': '1.2.0',
 'absl-py': '2.1.0',
 'aiobotocore': '2.12.3',
 'aiohttp': '3.9.5',
 'aioitertools': '0.7.1',
 'aiosignal': '1.2.0',
 'alabaster': '0.7.16',
 'altair': '5.0.1',
 'anaconda-anon-usage': '0.4.4',
 'anaconda-catalogs': '0.2.0',
 'anaconda-client': '1.12.3',
```

Batch Versus Real-Time Model Deployment

This model will be deployed using real-time inference and will consider batch mode applications when addressing data and concept drift over time. Given the time sensitivity of financial fraud detection, real time deployment is generally the most appropriate approach I could take with my model. The intent of the model is to flag down potentially fraudulent transactions and stop their complete execution in real time so they can be investigated further prior to becoming unrecoverable financial losses. While real time is the primary deployment method choice for this model, batch deployment may be used as a historical trend identifying tool to track large-scale fraud patterns and support the processes implemented to keep the model up to date over time.

Performance Metric Tracking in Monitoring Plan

Silhouette score is the primary performance metric that we will be tracking in our monitoring plan for multiple reasons. Over long periods of time, large variances in the model's silhouette scores on similar transaction datasets may indicate that the model is not clustering the data points well and hence should be evaluated to be adjusted in some appropriate manner. Implementing this checkpoint for our model periodically will help us ensure that the model is performing optimally at all times. In our real-time inference application of our model, silhouette score can also be used to identify transactions that are outliers in the local dataset under consideration. If a new financial transaction is made and instantly added to a recently occurring transaction dataset and significantly reduces the model's silhouette score on this dataset, this may indicate that said transaction is an outlier of the dataset and should be flagged for further fraudulent transaction consideration.
The silhouette score performance metric's importance to track in production lies in its ability to capture distinct patterns in data similarity amongst large groups of transactions. We can leverage this tool to dynamically adjust our modeling approach or flag high priority transactions in real time, potentially preventing large financial losses.

Green/ Yellow/ Red Thresholds

Identifying threshold values for green (nothing is wrong with the model), yellow (errors should be tracked closely), and red (model should be pulled out of production) is essential for determining the best course of action when faced with varying levels of model performance over time. For the purpose of this anomaly detection task, silhouette score values above 0.5 indicate strong model performance. When the model is achieving clustering scores above 0.5, it is clustering the data points very clearly, making it easy to identify when new, fraudulent transactions enter the dataset.

Silhouette scores ranging from 0.3 to 0.5 indicate that the model is clustering data points moderately well but not optimally. In cases such as this, there may be some ambiguity around decision boundaries that determine whether a financial transaction is classified as an anomaly or as a normal data point. When the model enters this score range it should be monitored closely for error tracking and potential data and concept drift considerations may be worth further consideration. Silhouette scores under 0.3 are indicative of the model not producing easily discernible clusters. This circumstance implies that the model is having significant trouble identifying whether transactions are normal or anomalies and it would be unsafe to continue using the model until this issue is effectively addressed.

Threshold Values:

Green: silhouette_score > 0.5

Yellow: $0.3 \leq$ silhouette_score ≤ 0.5

Red: silhouette_score < 0.3

Mitigation Strategies

Having risk mitigation strategies in place for varying levels of model performance is essential to ensuring that the model is operating in a productive and responsible manner at any given time. While the model is operating in the green zone, it is performing as it should. Even though models under these circumstances are performing as desired, real-time feedback loops should be put in place to evaluate the effectiveness of the model in real time and allow for minor adjustments to be made according to the outputs of these loops that may aid in the continuance of model performance in the green zone. When the model is performing in the yellow zone, alert systems should be used to notify model managers of when the model begins performing at dangerously low levels. Additionally, retraining and hyperparameter fine-tuning are options to attempt to return the model to green zone performance levels. When the model begins performing in the red zone, it should be immediately removed from production. This strategy stops the negative impacts that a red zone model being used in production may cause and provides an opportunity for developers to identify the root causes of the model's disparities and resolve them without negatively impacting involved businesses.

Model Retraining Frequency

Model retraining is a necessary component of this model's monitoring plan. Considering the nature of numerous new financial transactions being made daily, model retraining in regular intervals such as in monthly or quarterly intervals is an advisable model retraining frequency. While this represents a generally acceptable frequency to maintain strong model performance that promotes the swift identification of data or concept changes, model retraining may also be required immediately or with little notice if the model's performance decays at a greater than expected rate. By retraining our model periodically, we can identify periods of time when the model is outputting accurate predictions and adjust our approach as the retraining uncovers newfound flaws with the model.

Data and Concept Drift

Data and concept drift are pivotal topics to consider with respect to my model. The advent of new technologies constantly changes the manner with which financial transactions are executed. In consideration of this factor, the way that financial transaction data is tracked may also evolve with time, potentially creating discrepancies between live-market financial transaction data and outdated training data. Likewise, concept drift may involve fraud patterns changing over time, rendering outdated models unsophisticated at identifying true fraudulent transactions. Mitigation strategies for the risks posed by data and concept drift involve continual monitoring of fraud patterns and model retraining strategies are essential parts of this financial fraud detection model as well as of nearly any fraud detection model. By incorporating batch mode analysis of financial transaction data over certain intervals of time, new fraud patterns or techniques may become discoverable, further allowing for our model to adapt and prevent the execution of more fraudulent transactions.