Holly Cohan, Yesh Onipede, Yu Xia

Week 12 Report: Saving and Packing Model for Deployment

The objective of this week's assignment is to deploy the model and develop a monitoring plan, discussing whether data drift and concept drift impact the data.

Saving Model for Deployment as a Pickle File

The model is serialized using Python's pickle library to enable consistent saving and loading for deployment. The 'save_model' function writes the model object to a binary file (.pkl format), while the 'load_model' function reads it back into memory. This ensures the trained model can be efficiently shared or reused without retraining, saving computational resources.

Documenting Dependencies

The dependencies listed below are critical for recreating the development environment, ensuring the model and related scripts run smoothly in production. The following environment details were used to ensure compatibility during deployment:

- Operating System: Windows 10
- Python Version: 3.10
- Python Packages and Versions:
- pandas: 1.3.3
- numpy: 1.21.2
- scikit-learn: 1.0
- pickle (built into Python's standard library)

Model Deployment: Real Time Inferences vs Batch Deployment

Real-time inference is better suited for the rideshare price prediction model because the predictions are highly time-sensitive. Rideshare pricing fluctuates dynamically based on factors like demand, weather conditions, traffic, and ongoing events, which can change rapidly within minutes. For example, a sudden surge in demand due to an unexpected rainstorm or the end of a sports event requires immediate adjustments to predictions to remain accurate. With real-time inference, the model can process live data and deliver instant predictions, enabling timely and actionable insights for users and businesses. This approach ensures that pricing predictions remain relevant and aligned with current conditions, unlike batch deployment, which relies on pre-processed data and would fail to account for sudden, short-term changes. Real-time inference provides a competitive edge by allowing rideshare companies to optimize pricing strategies in real-time while improving customer satisfaction by offering accurate fare estimates.

Performance Metrics for Monitoring

Monitoring the following metrics ensures model performance aligns with expectations, identifies drift or inaccuracies, and maintains optimal system health. Model metrics ensure that the model performs as expected and continues to meet accuracy requirements. Business metrics are important for ensuring that the model's performance aligns with company goals. Regular tracking enables proactive intervention, maintaining reliability and effectiveness in production. In production, the following metrics will be tracked:

Model Metrics:

These metrics track the consistency of the model's predictions, particularly for mid-range prices, where accuracy is generally easier to achieve. It is monitored ideally in real time to evaluate the model's performance continuously and helps identify patterns in prediction errors. It is also important to monitor this metric over longer time intervals—such as daily, weekly, and monthly—to assess the model's ability to generalize across different time periods and adapt to seasonal patterns. This broader monitoring approach ensures that the model maintains consistent performance despite changes in factors like demand fluctuations, weather, or traffic patterns that may vary over time.

- Mean Squared Error (MSE): Tracks the average squared error to evaluate model precision.
- Root Mean Squared Error (RMSE): Provides an interpretable metric in the same units as the target variable. Reflects how well the model generalizes to unseen data.
- R^2 (Coefficient of Determination): Measures the proportion of variance in the price data explained by the model. A declining R^2 may indicate reduced model accuracy.

Business Metrics:

- **Revenue Trend Deviation(%):** Revenue Trend Deviation quantifies the percentage difference between the model's predicted revenue trends (e.g., hourly, daily, or weekly) and historical revenue trends observed under similar conditions, such as traffic, weather, or day of the week.
    - Defined as: (Predicted Trend−Historical Trend)/Historical Trend *100

Performance Metrics for Original Data and Enhanced Data

Model Type	Metrics	Training Set Values	Validation Set Values	Testing Set Values
Pruned Decision Tree Model on Enhanced Data (with updated coordinates and eta_minutes_rh) – Original model	**MSE**	2.6845	3.0536	**3.0240**
	RMSE	1.6385	1.7474	**1.7390**
	R^2	0.9675	0.9627	**0.9639**
Pruned Decision Tree Model on Enhanced Data (with updated coordinates and eta_minutes_rh) with price outliers removed	**MSE**	2.4186	3.9584	**3.8990**
	RMSE	1.5552	1.9896	**1.9746**
	R^2	0.9668	0.9517	**0.9534**

Thresholds for Key Metrics:
MSE & RMSE Thresholds:

- Green Zone (Good Performance):
    - **Threshold: MSE ≤ 4.0; RMSE ≤ 2.0**

    - This threshold allows a small buffer above the lowest test MSE/RMSE observed for the original and cleaned models
- Yellow Zone**:** (Needs Monitoring):
    - **Threshold: 4.0 < MSE ≤ 6.0; 2.0 < RMSE ≤ 3**
    - MSE threshold values are higher than typical testing values but not drastically different. The threshold is set by calculating about 1.5x the highest test MSE/RMSE observed as a benchmark.
- Red Zone (Critical Issue)
    - **Threshold: MSE > 8.0; RMSE > 4.0**
    - This threshold indicates severe degradation where the model's predictions deviate far beyond what was observed during testing, where MSE/RMSE exceeds 2x the highest MSE/RMSE observed.
    - This could indicate potential data or concept drift, or model degradation and indicates that the model needs immediate intervention.

R^2 Thresholds

- Green Zone (Good Performance):
    - **Threshold: $R^2 \geq 0.95$**
    - R^2 values from testing were around 0.9639 (original) and 0.9534 (cleaned), so 0.95 is a reasonable lower limit for "good" performance.
- Yellow Zone**:** (Needs Monitoring):
    - **Threshold: $0.90 \leq R^2 < 0.95$**
    - This range suggests the model is performing adequately but has lost some explanatory power.
- Red Zone (Critical Issue)
    - **Threshold: $R^2 < 0.90$**
    - Below 0.90, the model is likely failing to capture key relationships in the data and may require retraining.

Revenue Trend Deviation

This metric ensures that the model supports business objectives, especially profitability.

- Green Zone:
    - **Threshold: Deviation ≤ ±5%**
    - The model closely follows historical trends, indicating that it is well-calibrated to seasonal and temporal patterns.
- Yellow Zone:
    - **Threshold: Deviation between ±5% and ±10%**
    - Moderate deviation; further investigation is needed to determine if the model struggles with specific time periods, locations, or external factors like extreme weather.
- Red Zone:
    - **Threshold: Deviation > ±10%**
    - Significant deviation; immediate recalibration or retraining may be necessary to improve model alignment with revenue patterns.

Risk Mitigation Strategies for Green, Yellow, and Red Thresholds

Green Zone (The model is working well)

- Status: The model is performing as expected and there are no significant issues.
- Actions:
    - Regular Monitoring: Continue tracking metrics like RMSE, MSE, R^2, and accuracy periodically (weekly) to ensure that the model remains in the green zone.
    - Drift Detection: run periodic checks for data drift and concept drift to catch potential changes early before they impact performance.
    - Versioning: Maintain a version control system for the model and data to compare historical and current performance.
    - Model Validation: Perform periodic validation on new data to ensure that the model's performance is consistent across datasets.

Yellow Zone (Errors are increasing, but the model is still usable)

- Status: The model's performance is deteriorating, but the predictions are still moderately accurate.
- Actions:
    - Immediate Diagnostics:
        - Analyze input data for signs of data drift such as shifts in the distribution of features like time of day, weather, or ride distances.
        - Check for concept drift where the relationship between features and price may have changed. This could be new pricing policies from the rideshare services or external events like holidays.
    - Recalibration:
        - Retrain the model using more recent data from the past few weeks or months in order to improve performance.
        - Incorporate new features that could better explain changes in the data. For example, introducing more granular time variables or accounting for special events like festivals could contribute to a stronger model.
    - Increase Monitoring Frequency: switch from weekly to daily monitoring to catch any rapid degradation in performance.
    - Communicate with Stakeholders: Notify stakeholders about the degradation and discuss potential short-term fixes or adjustments in decision-making that relies on the model.

Red Zone (Critical Issues Detected)

- Status: The model is no longer reliable. Predictions are highly inaccurate and could cause significant business risk if used.
- Actions:
    - Immediate Pullback:
        - Remove the model from production immediately to prevent inaccurate predictions from impacting decisions.
        - Fall back to a previous and more stable version of the model temporarily.
    - Root Cause Analysis:
        - Investigate the cause of failure:
            - Investigate if there was data drift like new user behaviors or changes in weather patterns.
            - Investigate if concept drift occurred. For example, investigate if the relationship between variables like weather and pricing has fundamentally changed.

        - Asses external factors such as changes in business practices (i.e. Uber/Lyft adjusting their pricing algorithms)/
    - Retrain the model:
        - Use fresh, updated data to retrain the model to reflect the current environment.
        - Consider increasing the training frequency to retrain weekly instead of monthly.
    - Improve Model Robustness:
        - Build a monitoring system to detect drift in real time and trigger automatic retraining or at least alerts.
        - Potentially introduce ensemble techniques or hybrid models that can adapt better to changing data patterns.

Retraining Frequency

The dynamic nature of rideshare pricing makes it essential to keep the prediction model updated. A weekly or biweekly retraining schedule provides the optimal balance between maintaining model accuracy and managing computational resources.

- Dynamic Input Variables
    - Rideshare pricing is influenced by rapidly changing factors like weather conditions and customer behaviors. Weather conditions such as sudden snowstorms or heavy rain can cause unexpected surges in demand and pricing. Similarly, holidays and special events like concerts and festivals may increase demand, which could lead to pricing changes.
- Holiday and Event Seasonality
    - During specific months, such as November and December (our dataset timeframe), holiday-related travel increases significantly. Bi-weekly retraining ensures that the model captures various trends.
        - Increased demand during Thanksgiving, Christmas, and New Years.
        - Longer travel distances to family gatherings or events.
        - The possibility of unusual patterns caused by limited availability of rides.
- Weather Sensitivity
    - Weather changes, such as snowstorms or heavy rain can alter pricing dynamics. A model that is trained bi-weekly will be better equipped to adapt to recent weather patterns and reflect the real-time impact of weather on ride demand and pricing.
- User Behavior Changes:
    - Bi-weekly retraining ensures the model reflects recent customer preferences, such as:
        - Choosing more affordable ride options during certain weeks.
        - Switching to premium services like SUVs or shared rides during peak times.

Balancing costs and accuracy is crucial when determining the retraining schedule for the rideshare price prediction model. Weekly or biweekly retraining strikes an effective balance between computational cost and maintaining model accuracy. Weekly retraining is ideal for capturing fast-changing trends, such as demand surges, weather-related impacts, or promotional offers, ensuring the model reflects recent data. For example, in Boston during December, a snowstorm could cause sudden price hikes for both Uber and Lyft; retraining the model weekly would allow it to adapt quickly to such events, improving predictions during similar occurrences in the future. On the other hand, biweekly retraining is more suitable for periods with less variability, such as mid-January when holiday travel subsides, or when computational resources need to be optimized. Choosing a

retraining schedule depends on the level of variability in features like weather, surge multipliers, and demand. Weekly retraining is preferred when significant variability and rapid pricing changes occur, while biweekly retraining works well during more stable periods, balancing accuracy and resource efficiency.

Data Drift and Concept Drift

- **Data drift** refers to the changes in the statistical properties of the input data distribution over time. This could be observed in our use case through a shift in typical weather patterns due to climate change, which may influence both demand and supply in the rideshare market. Additionally, as Boston grows and develops, factors like population density, infrastructure changes, or seasonal events could cause fluctuations in rideshare prices.
    - To mitigate the risk of data drift, a regular monitoring system could be implemented to monitor the distribution of the independent variables such as weather conditions and location data. This system would help detect significant changes in these distributions, signaling potential drift. For instance, if the system detects a shift in average temperature patterns or a steady increase in rides during non-peak hours, it could trigger a re-evaluation of the model's training data. Retraining the model on more recent data would help incorporate these new trends, ensuring that it continues to capture the evolving patterns in Boston's rideshare market.
- **Concept drift** refers to the change in the relationship between the predictive features and the target variable over time. For example, the influence of weather patterns on rideshare pricing may evolve. A colder winter season may initially result in higher prices, but as more drivers opt into the market during the winter months, the price may stabilize or even decrease. This dynamic change in the influence of various features on pricing signals concept drift.
    - Concept drift could be mitigated by setting up a monitoring system to track the model's performance in real-time. By comparing the model's current predictions against actual prices and assessing error metrics, it would be possible to detect when performance begins to degrade. If the model consistently performs poorly on new data, retraining the model with updated data that better reflects these changing relationships would be necessary.