Final Project Report

Team A

End-to-End Process of Machine Learning Models to Enhance Operational Efficiencies for Delhivery

Table of Contents

Executive Summary

This project successfully optimized Delhivery's logistics operations by deploying machine learning models that predict delivery times and enhance operational efficiency. Key steps included robust data preparation, feature engineering, algorithm evaluation, and cloud-based deployment. Results showed significant improvements in prediction accuracy, cost optimization, and route planning.

Introduction

Delhivery's challenges in logistics management, such as delivery accuracy and cost minimization, require innovative solutions. This project leveraged machine learning to build predictive models that address these challenges by offering precise delivery time forecasts and actionable operational insights.

Data Preparation

- **Data Cleaning**
  - Identify columns with missing values and Missing values were handled using imputation techniques
  - Treated outliers using statistical techniques.
- **Feature Engineering**
  - Extracted temporal features (e.g., year, month, day).
  - Derived geographic features from addresses.
  - Encoded categorical data using one-hot encoding.
- **Feature Selection**
  - Applied multicollinearity handling techniques.
  - Used feature importance metrics like Random Forest and mutual information.
- **Normalization**
  - Ensured numerical features were scaled uniformly.

Model Development

- **Algorithms Explored**
    - Linear Regression, Ridge Regression, and Lasso.
- **Model Tuning**
    - Applied grid search, random search.
- **Cross-Validation**
    - K-Fold Cross-Validation ensured model robustness.
- **Purpose**
    - Delivery time predictions.

Algorithms Explored:

• **Ridge Regression**:

•Time Discrepancy Model:

•Mean Squared Error: 1.4040894112965502e-13

•R-squared: 1.0

•Distance Discrepancy Model:

•Mean Squared Error: 3.685967764280214e-14

•R-squared: 1.0

•Linear Regression:

•Time Discrepancy Model:

•Mean Squared Error: 2.016010862972678e-06

•R-squared: 0.9999999999783656

•Distance Discrepancy Model:

•Mean Squared Error: 1.2246722165409683e-07

•R-squared: 0.9999999999814942

Final Model Selection:

- Lasso Regression

Deployment Process

Prerequisites

1. **System Requirements:** Ensure you have a system with the following:

   - Python 3.8 or above
   - Docker installed (optional for containerized deployment)
   - pip (Python package manager)

2. **Libraries/Dependencies:**

   - Install the required Python packages listed in the `requirements.txt` file using:
     `pip install -r requirements.txt`

3. **Model Files:**

   - Ensure the following model files are available in the project directory:
     - `linear_regression_model.pkl`
     - `rfe_data_model.pkl`

4. **Datasets:**

   - Include the following datasets in the project directory:
     - `lasso_selected_data.csv`
     - `rfe_selected_features.csv`
     - `test.csv`

Deployment Steps

Option 1: Local Deployment

1. **Clone the Repository:**

   ```
   git clone <repository_url>
   cd <repository_name>
   ```

2. **Run the Application:**

   ```
   python app.py
   ```

   This will start the server on `http://127.0.0.1:5000` by default.

Option 2: Dockerized Deployment

1. **Build the Docker Image:**

   ```
   docker build -t flask-regression-app.
   ```

2. **Run the Docker Container:**

```
docker run -p 5000:5000 flask-regression-app
```

Access the application at `http://localhost:5000`.

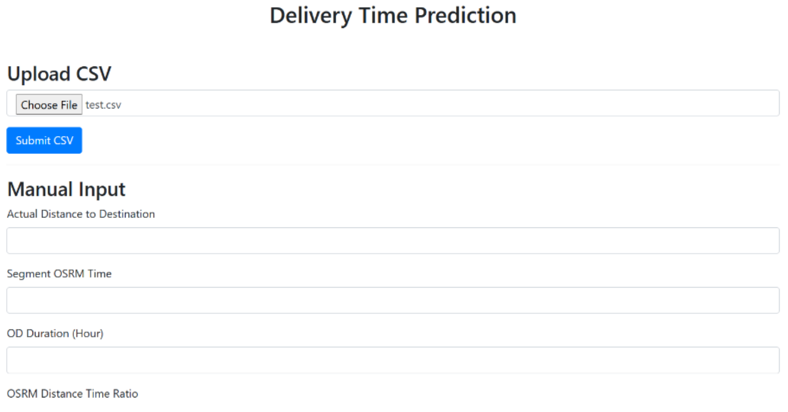
Usage Instructions

Interacting with the Web Application

1. **Access the Web App:** Open a browser and navigate to `http://127.0.0.1:5000` or the appropriate Docker container address.

2. **Upload Dataset:**
   - Use the upload feature to input datasets (e.g., `test.csv`).
3. **Run Predictions:**
   - Click the "Predict" button to analyze the dataset.
4. **View Results:**
   - View predictions and insights directly on the dashboard.

Input:



Output:

The output will have the entered data with the data that has been predicted which here is the delivery time

data	route_type	start_scan_to_end_scan	actual_distance_to_destination	segment_actual_time	segment_osrm_time	od_duration_dirr_hour	osrm_distance_time_ratio	distance_time_ratio	segment_actual_time_sum	predicted_delivery_time
2024-12-27	Urban	10:00-10:15	15.000000	12.5	13.2	0.800000	1.050000	0.900000	14.0	13.765838
2024-12-27	Rural	10:15-10:30	25.500000	20.0	22.0	1.000000	1.100000	1.000000	23.0	24.134693
2024-12-27	Suburban	10:30-10:45	12.000000	10.5	11.0	0.600000	0.980000	0.950000	11.5	10.702691
2024-12-27	Urban	10:45-11:00	8.500000	6.8	7.0	0.500000	1.000000	1.100000	7.5	7.064030
2024-12-27	Rural	11:00-11:15	18.000000	15.5	16.2	0.900000	1.080000	0.980000	17.0	16.691859
1.0	0.0	86.0	10.435660	14.0	11.0	1.436894	1.087755	-3.564340	14.0	13.732366
1.0	0.0	86.0	18.936842	10.0	9.0	1.436894	1.086215	-5.063158	24.0	23.548231
1.0	0.0	86.0	27.637279	16.0	7.0	1.436894	1.162125	-12.362721	40.0	39.391393
1.0	0.0	86.0	36.118028	21.0	12.0	1.436894	1.139050	-25.881972	61.0	61.292089
1.0	0.0	86.0	39.386040	6.0	5.0	1.436894	1.232230	-28.613960	67.0	67.169446
1.0	0.0	109.0	10.403038	15.0	11.0	1.819553	1.101555	-4.596962	15.0	14.741021
1.0	0.0	109.0	18.045481	28.0	6.0	1.819553	1.252294	-25.954519	43.0	43.580471
1.0	0.0	109.0	28.061896	21.0	11.0	1.819553	1.235352	-36.938104	64.0	64.458772

API Details

Base URL

- Local: `http://127.0.0.1:5000`
- Docker: `http://localhost:5000`

Endpoints

1. **`/`**

   - **Method:** GET
   - **Description:** Renders the homepage of the web application.

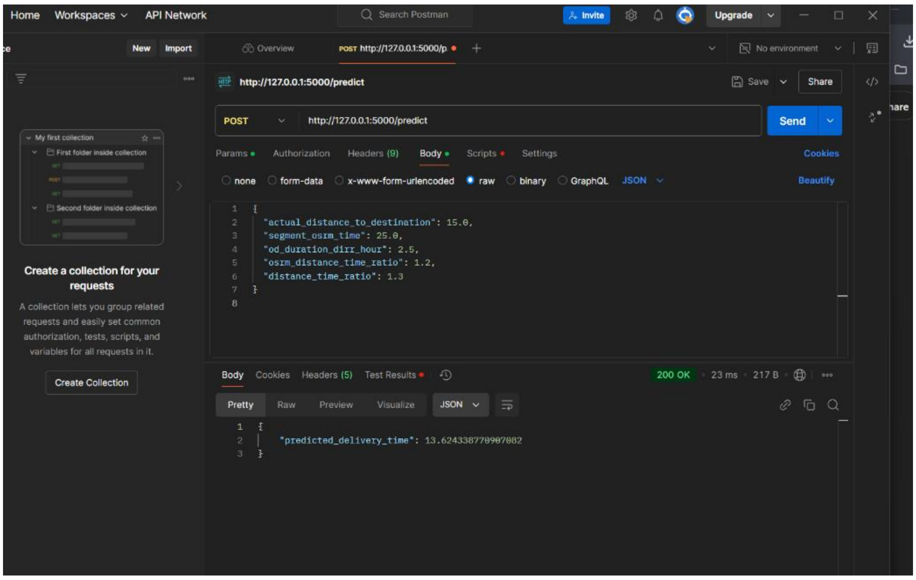
2. **`/predict`**

   - **Method:** POST
   - **Description:** Processes the uploaded dataset and returns predictions.
   - **Payload:**

   ```
   {
     Json format data
   }
   ```

   - **Response:**

   ```
   {
     "predictions": [<list_of_predictions>]
   }
   ```



User Guide

Steps for Interacting with the Web Application

1. **Uploading a File:**

   - Navigate to the "Upload" section.
   - Browse and select a CSV file (e.g., `test.csv`).
   - A person can input the prefered data manually.

2. **Submit for Prediction:**

   - Click on the "Predict" button to initiate the analysis.

3. **View and Download Results:**

   - Results are displayed in a table format on the dashboard.
   - Use the "Download" button to save the results.

Troubleshooting

1. **Common Issues:**

   - **Error:** Missing file or dataset.
     - **Solution:** Ensure required files (e.g., `linear_regression_model.pkl`) are in the correct directory.
   - **Error:** Dependency issues.
     - **Solution:** Reinstall dependencies using `pip install -r requirements.txt`.

Challenges and Solutions

- **Challenges and Solutions**
  • Data Issues: Data inconsistencies and missing values posed challenges during the initial stages.

  Solution: Implemented advanced imputation techniques and automated data validation pipelines to ensure data quality.

  • Model Performance: Certain models underperformed on unseen data.

  Solution: Enhanced models using ensemble techniques and retrained them with augmented datasets.

  • Deployment Hurdles: Integration with legacy systems caused compatibility issues.

  Solution: Built middleware layers to ensure smooth data flow between new and existing systems.

Recommendations and Future Work

- **Advanced Techniques**: Explore deep learning methods, such as Recurrent Neural Networks (RNNs), to better capture temporal patterns in logistics data.
  • **Additional Data Sources**: Integrate external data like weather conditions, traffic patterns, and regional holidays for more robust predictions.
  • **New Use Cases**: Investigate dynamic pricing models and predictive maintenance for vehicles to further enhance operational efficiency.
  • **System Enhancements**: Develop real-time dashboards for visualizing predictions and their impacts on key performance metrics.

Conclusion

This project delivered impactful machine learning solutions that enhanced Delhivery's logistics operations. The models have established a strong foundation for further innovation and efficiency in logistics management.