# Last-Mile ETA Predictor

Presented By: BALAJI ENNAWAR

Project Type: Individual

# Project Summary

This project creates a smart prediction system for Amazon deliveries.

The main goal is to accurately forecast how long a specific order will take to reach the customer (Estimated Time of Arrival or ETA) before the delivery agent even leaves the store.

What is the System?

The system is a machine learning (ML) model—a program that learns from thousands of past delivery records. Instead of guessing, it uses history to make a precise prediction.

How it Works

(The 3 Simple Steps)

Input: We feed the system all the details of a new order:

The route: Where the store is and where the customer lives (distance).

The conditions: Current traffic, weather (e.g., rain, sun), and the time of day (e.g., rush hour).

The agent: The delivery agent's rating and the vehicle they are using (motorcycle or van).

Prediction: A powerful algorithm called Random Forest (which acts like a committee of 100 experienced delivery managers) instantly analyzes all these inputs. It knows, for example, that a 10 km trip in heavy traffic at 5:00 PM usually takes 160 minutes.

Output: The result is delivered through a simple website interface (built with Streamlit), giving the exact estimated delivery time in minutes (e.g., "155 minutes").

Why this Matters (Business Value)

This predictor is crucial because it allows the company to:

Improve Customer Happiness: Give customers highly accurate delivery windows, avoiding frustration.

Optimize Operations: Better manage agent schedules and resources by knowing exactly which deliveries will take longer.

In short, we built a reliable digital brain to turn messy real-world variables into consistent, trustworthy delivery time forecasts.

Double-click (or enter) to edit

```
# ==============================================================================
# AMZ Project: Model Training Pipeline (UPDATED with EDA)
# ==============================================================================

# --- Colab Setup: Install Required Libraries ---
# Uncomment and run this cell in Colab first
!pip install pandas numpy scikit-learn xgboost mlflow joblib geopy matplotlib s
# ----------------------------------------------------------------------------

import pandas as pd
import numpy as np
import joblib
import mlflow
import mlflow.sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from math import radians, sin, cos, sqrt, asin

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


# --- 1. Load Data ---
try:
    # Ensure 'amazon_delivery.csv' is uploaded to your Colab environment
    df = pd.read_csv('amazon_delivery.csv')
except FileNotFoundError:
    print("ERROR: 'amazon_delivery.csv' not found. Please upload the file.")
    exit()


# --- 2. Data Cleaning and Feature Engineering (Same as Part 1) ---

# Clean up string columns by stripping whitespace
for col in ['Weather', 'Traffic', 'Vehicle', 'Area', 'Category']:
    if col in df.columns:
        df[col] = df[col].astype(str).str.strip()
```

```
# Handle Missing Values
df['Agent_Rating'] = df['Agent_Rating'].fillna(df['Agent_Rating'].mean())
df['Weather'] = df['Weather'].fillna(df['Weather'].mode()[0])

# Haversine function
def haversine(lat1, lon1, lat2, lon2):
    R = 6371
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = np.sin(dlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2.0)**2
    c = 2 * np.arcsin(np.sqrt(a))
    return R * c

df['Delivery_Distance_km'] = haversine(
    df['Store_Latitude'], df['Store_Longitude'],
    df['Drop_Latitude'], df['Drop_Longitude']
)

# Time Calculations
df['Order_Datetime'] = pd.to_datetime(df['Order_Date'] + ' ' + df['Order_Time']
df['Pickup_Datetime'] = pd.to_datetime(df['Order_Date'] + ' ' + df['Pickup_Time
df.dropna(subset=['Order_Datetime', 'Pickup_Datetime'], inplace=True)
df.loc[df['Pickup_Datetime'] < df['Order_Datetime'], 'Pickup_Datetime'] += pd.T
df['Time_to_Pickup_min'] = (df['Pickup_Datetime'] - df['Order_Datetime']).dt.tc
df['Order_Hour'] = df['Order_Datetime'].dt.hour
df['Order_DayOfWeek'] = df['Order_Datetime'].dt.dayofweek

# --- 3. Final Dataset Preparation (for EDA and Modeling) ---

columns_to_drop = [
    'Order_ID', 'Store_Latitude', 'Store_Longitude', 'Drop_Latitude',
    'Drop_Longitude', 'Order_Date', 'Order_Time', 'Pickup_Time',
    'Order_Datetime', 'Pickup_Datetime'
]
df_cleaned = df.drop(columns=columns_to_drop, errors='ignore').copy()

print("Data Cleaning and Feature Engineering Complete.")


# ==============================================================================
# --- 4. Exploratory Data Analysis (EDA) and Visualization ---
# ==============================================================================
print("\n" + "="*50)
print("STARTING EDA AND STATISTICAL ANALYSIS")
print("="*50)

# --- A. Descriptive Statistics ---
print("\n--- A. Descriptive Statistics for Numerical Features ---")
print(df_cleaned.describe().T)

# --- B. Visualization Setup ---
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)
```

```
# --- C. Univariate Analysis: Target Variable ---
plt.figure(figsize=(10, 4))
sns.histplot(df_cleaned['Delivery_Time'], bins=30, kde=True)
plt.title('Distribution of Delivery Time (Target Variable)')
plt.xlabel('Delivery Time (minutes)')
plt.savefig('eda_delivery_time_distribution.png')
plt.show()

# --- D. Correlation Heatmap ---
numerical_cols = df_cleaned.select_dtypes(include=np.number).columns
plt.figure(figsize=(10, 8))
sns.heatmap(df_cleaned[numerical_cols].corr(), annot=True, fmt=".2f", cmap='vir
plt.title('Correlation Heatmap of Numerical Features')
plt.savefig('eda_correlation_heatmap.png')
plt.show()

# --- E. Bivariate Analysis: Categorical vs. Delivery Time ---

categorical_cols_for_eda = ['Traffic', 'Weather', 'Vehicle', 'Area', 'Category'

for col in categorical_cols_for_eda:
    plt.figure(figsize=(12, 6))

    # Calculate median delivery time for sorting
    median_time = df_cleaned.groupby(col)['Delivery_Time'].median().sort_values

    sns.boxplot(x=col, y='Delivery_Time', data=df_cleaned, order=median_time)
    plt.title(f'Delivery Time Distribution by {col}')
    plt.xlabel(col)
    plt.ylabel('Delivery Time (minutes)')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.savefig(f'eda_delivery_time_vs_{col.lower()}.png')
    plt.show()

print("\nEDA Visualizations Generated and Saved as PNG files.")

# ==============================================================================
# --- 5. Modeling and MLflow Tracking (Same as Part 2) ---
# ==============================================================================

print("\n" + "="*50)
print("STARTING MODELING AND MLFLOW TRACKING")
print("="*50)

# Preprocessing Setup
X = df_cleaned.drop('Delivery_Time', axis=1)
y = df_cleaned['Delivery_Time']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

categorical_features = ['Weather', 'Traffic', 'Vehicle', 'Area', 'Category']
numerical_features = X.select_dtypes(include=np.number).columns.tolist()

numerical_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')
```

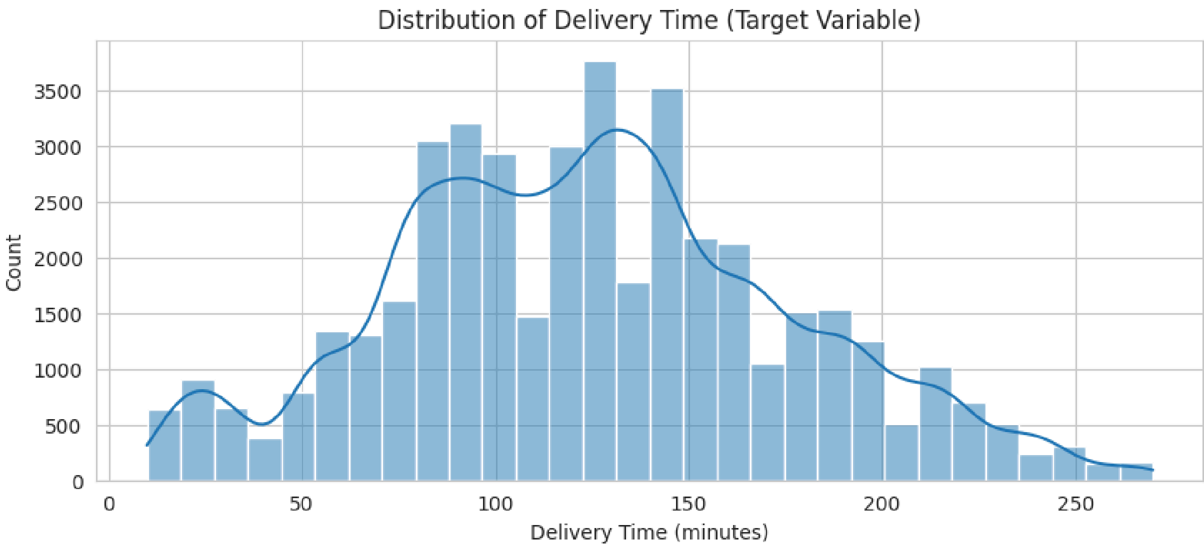
```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ],
    remainder='drop'
)

# MLflow Helper Functions
mlflow.set_experiment("AMZ Delivery Time Prediction")

def evaluate_model(model, X_test, y_test, model_name):
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"--- {model_name} Results ---")
    print(f"RMSE: {rmse:.2f} | MAE: {mae:.2f} | R-squared: {r2:.2f}")

    mlflow.log_metrics({"rmse": rmse, "mae": mae, "r2_score": r2})
    return r2

# Model Training and Tracking
models = {
    "Linear_Regression": LinearRegression(),
    "Random_Forest": RandomForestRegressor(n_estimators=100, random_state=42, r
    "XGBoost": XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=4
}

best_r2 = -np.inf
best_model_pipeline = None
best_model_name = ""

for name, model in models.items():
    with mlflow.start_run(run_name=name):

        print(f"\nStarting training for {name}...")

        full_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                                        ('regressor', model)])

        full_pipeline.fit(X_train, y_train)

        r2 = evaluate_model(full_pipeline, X_test, y_test, name)

        mlflow.sklearn.log_model(full_pipeline, "model")

        if r2 > best_r2:
            best_r2 = r2
            best_model_pipeline = full_pipeline
            best_model_name = name

# Save Best Model
```

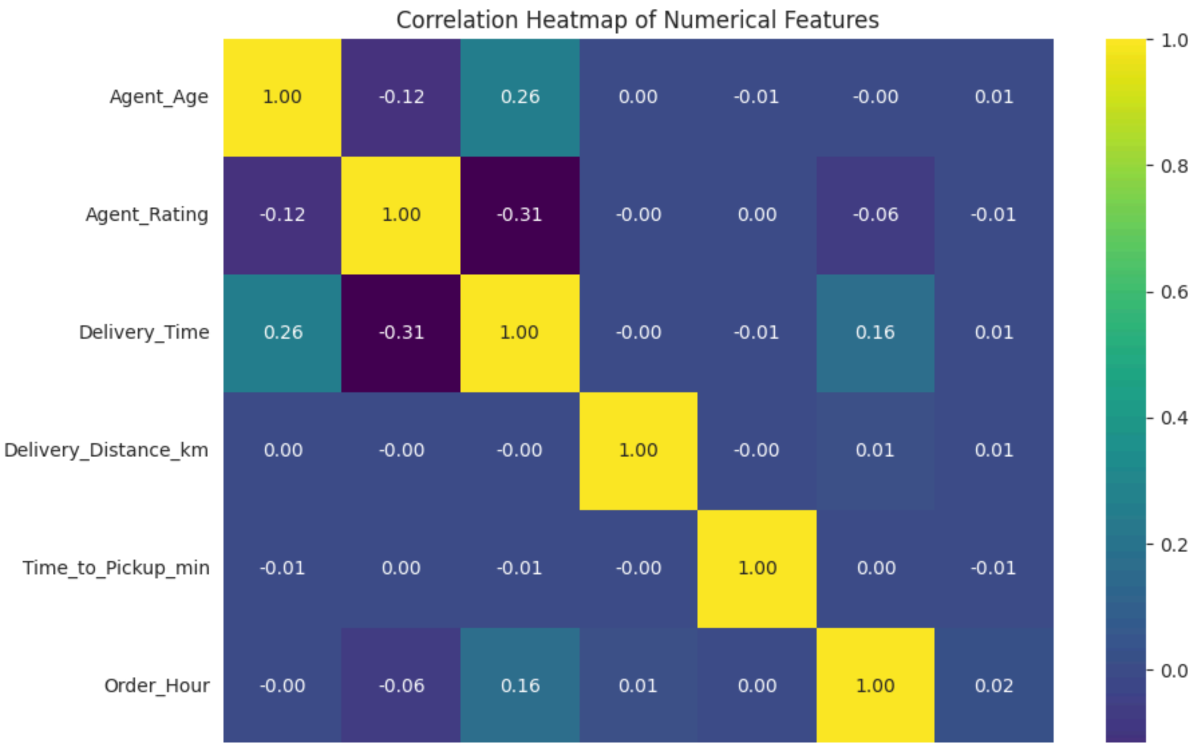
```
if best_model_pipeline:
    joblib.dump(best_model_pipeline, 'best_model_pipeline.pkl')
    print(f"\nTraining and MLflow Tracking Complete. Best model saved is: {best
    print("Best performing model pipeline saved as 'best_model_pipeline.pkl' fc
```

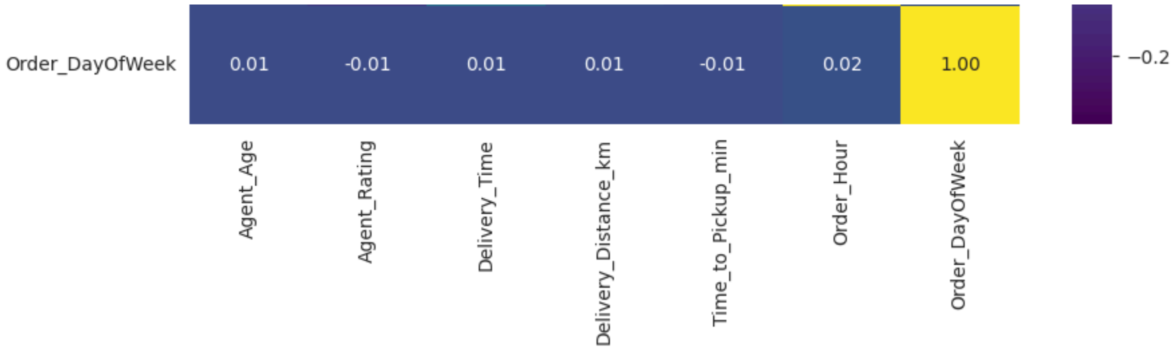
```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: xgboost in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: mlflow in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: geopy in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.12/dis
Requirement already satisfied: mlflow-skinny==3.4.0 in /usr/local/lib/python3.12
Requirement already satisfied: mlflow-tracing==3.4.0 in /usr/local/lib/python3.1
Requirement already satisfied: Flask<4 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: alembic!=1.10.0,<2 in /usr/local/lib/python3.12/d
Requirement already satisfied: cryptography<46,>=43.0.0 in /usr/local/lib/python
Requirement already satisfied: docker<8,>=4.0.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: fastmcp<3,>=2.0.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: graphene<4 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: gunicorn<24 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: pyarrow<22,>=4.0.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: sqlalchemy<3,>=1.4.0 in /usr/local/lib/python3.12
Requirement already satisfied: cachetools<7,>=5.0.0 in /usr/local/lib/python3.12
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: cloudpickle<4 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: databricks-sdk<1,>=0.20.0 in /usr/local/lib/pytho
Requirement already satisfied: fastapi<1 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: gitpython<4,>=3.1.9 in /usr/local/lib/python3.12/
Requirement already satisfied: importlib_metadata!=4.7.0,<9,>=3.7.0 in /usr/loca
Requirement already satisfied: opentelemetry-api<3,>=1.9.0 in /usr/local/lib/pyt
Requirement already satisfied: opentelemetry-proto<3,>=1.9.0 in /usr/local/lib/p
Requirement already satisfied: opentelemetry-sdk<3,>=1.9.0 in /usr/local/lib/pyt
Requirement already satisfied: packaging<26 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: protobuf<7,>=3.12.0 in /usr/local/lib/python3.12/
Requirement already satisfied: pydantic<3,>=1.10.8 in /usr/local/lib/python3.12/
Requirement already satisfied: python-dotenv<2,>=0.19.0 in /usr/local/lib/python
Requirement already satisfied: pyyaml<7,>=5.1 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: requests<3,>=2.17.3 in /usr/local/lib/python3.12/
Requirement already satisfied: sqlparse<1,>=0.4.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: typing-extensions<5,>=4.0.0 in /usr/local/lib/pyt
Requirement already satisfied: uvicorn<1 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: geographiclib<3,>=1.52 in /usr/local/lib/python3.
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: Mako in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: authlib>=1.5.2 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: cyclopts>=3.0.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: exceptiongroup>=1.2.2 in /usr/local/lib/python3.1
Requirement already satisfied: httpx>=0.28.1 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: mcp<2.0.0,>=1.12.4 in /usr/local/lib/python3.12/d
```

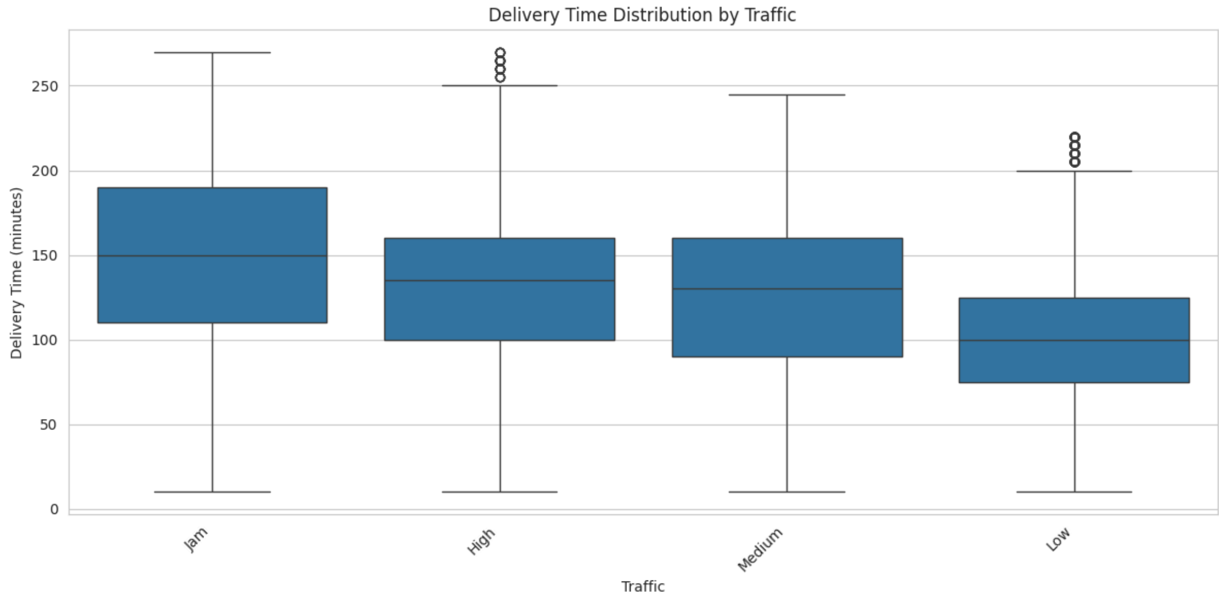
```
Requirement already satisfied: openapi-core>=0.19.5 in /usr/local/lib/python3.12/
Requirement already satisfied: openapi-pydantic>=0.5.1 in /usr/local/lib/python3
Requirement already satisfied: pyperclip>=1.9.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: rich>=13.9.4 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: itsdangerous>=2.2.0 in /usr/local/lib/python3.12/
Requirement already satisfied: jinja2>=3.1.2 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: werkzeug>=3.1.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: graphql-core<3.3,>=3.1 in /usr/local/lib/python3.
Requirement already satisfied: graphql-relay<3.3,>=3.1 in /usr/local/lib/python3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: attrs>=23.1.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: docstring-parser>=0.15 in /usr/local/lib/python3.
Requirement already satisfied: rich-rst<2.0.0,>=1.3.1 in /usr/local/lib/python3.
Requirement already satisfied: google-auth~=2.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: starlette<0.49.0,>=0.40.0 in /usr/local/lib/pytho
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.12/dist
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: httpx-sse>=0.4 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: jsonschema>=4.20.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: pydantic-settings>=2.5.2 in /usr/local/lib/python
Requirement already satisfied: python-multipart>=0.0.9 in /usr/local/lib/python3
Requirement already satisfied: sse-starlette>=1.6.1 in /usr/local/lib/python3.12
Requirement already satisfied: isodate in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: jsonschema-path<0.4.0,>=0.3.1 in /usr/local/lib/p
Requirement already satisfied: more-itertools in /usr/local/lib/python3.12/dist-
Requirement already satisfied: openapi-schema-validator<0.7.0,>=0.6.0 in /usr/lo
Requirement already satisfied: openapi-spec-validator<0.8.0,>=0.7.1 in /usr/loca
Requirement already satisfied: parse in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: opentelemetry-semantic-conventions==0.58b0 in /us
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.1
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python
Requirement already satisfied: email-validator>=2.0.0 in /usr/local/lib/python3.
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: dnspython>=2.0.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.1
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/loca
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.12/
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: pathable<0.5.0,>=0.4.1 in /usr/local/lib/python3.
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: rfc3339-validator in /usr/local/lib/python3.12/di
Requirement already satisfied: lazy-object-proxy<2.0.0,>=1.7.1 in /usr/local/lib
Requirement already satisfied: docutils in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.12
Data Cleaning and Feature Engineering Complete.
```

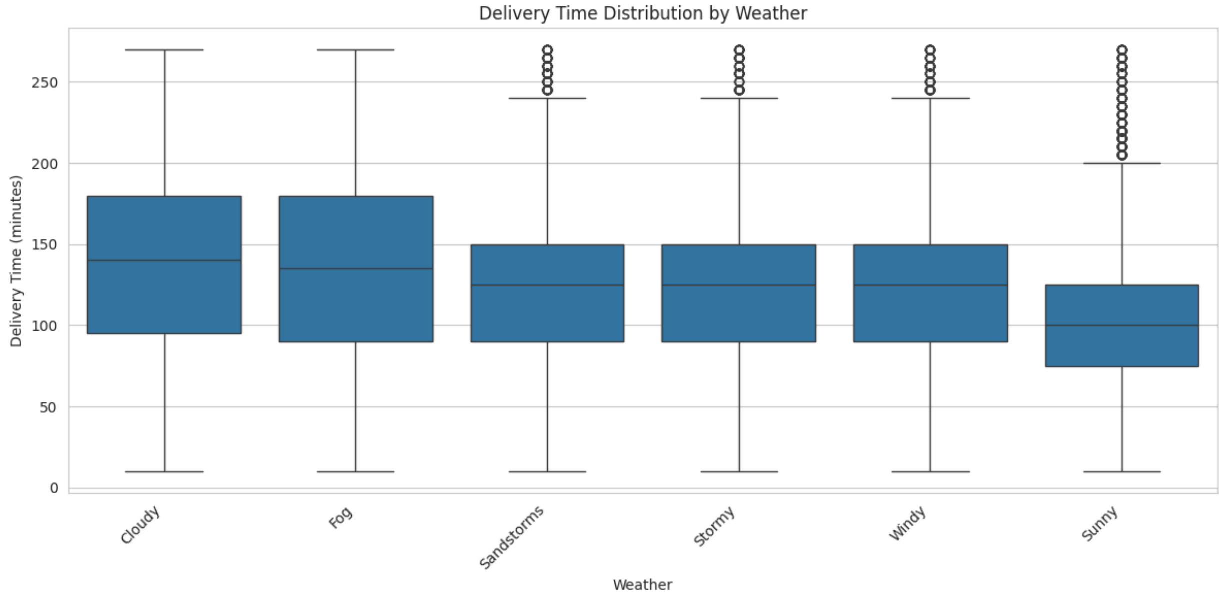
```
==================================================
STARTING EDA AND STATISTICAL ANALYSIS
==================================================

--- A. Descriptive Statistics for Numerical Features ---
                       count        mean         std        min        25%  \
Agent_Age            43648.0   29.555008    5.761482  20.000000  25.000000
Agent_Rating         43648.0    4.635285    0.313632   2.500000   4.500000
Delivery_Time        43648.0  124.914475   51.933163  10.000000  90.000000
Delivery_Distance_km 43648.0   27.255432  303.815765   1.465067   4.663432
Time_to_Pickup_min   43648.0    9.991294    4.086680   5.000000   5.000000
Order_Hour           43648.0   17.425976    4.818494   0.000000  15.000000
Order_DayOfWeek      43648.0    3.000275    1.969665   0.000000   1.000000

                            50%         75%          max
Agent_Age             30.000000   35.000000    39.000000
Agent_Rating           4.700000    4.900000     5.000000
Delivery_Time        125.000000  160.000000   270.000000
Delivery_Distance_km   9.220419   13.682165  6884.726399
Time_to_Pickup_min    10.000000   15.000000    15.000000
Order_Hour            19.000000   21.000000    23.000000
Order_DayOfWeek        3.000000    5.000000     6.000000
```

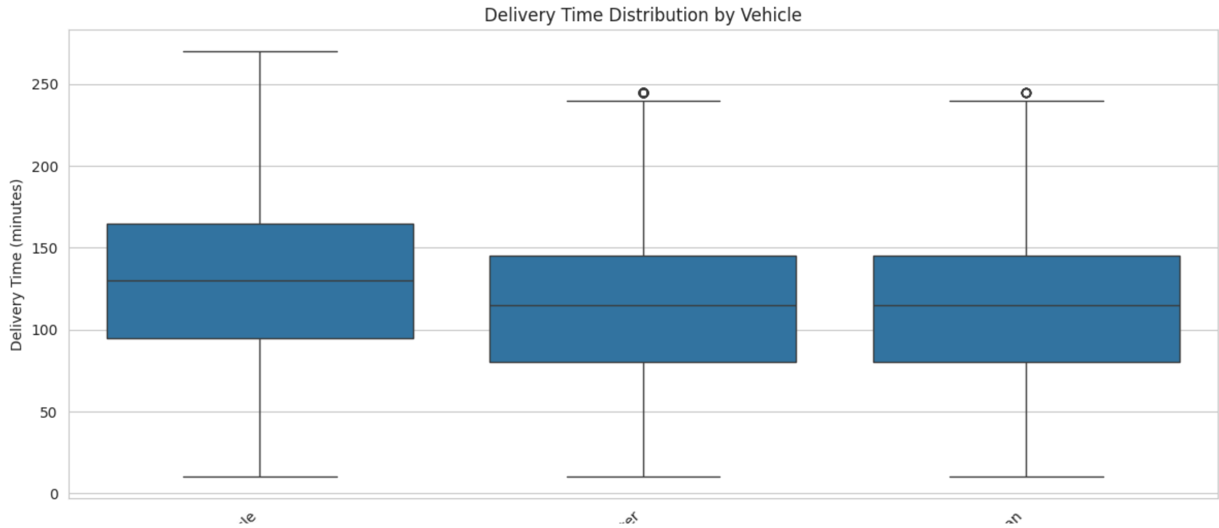


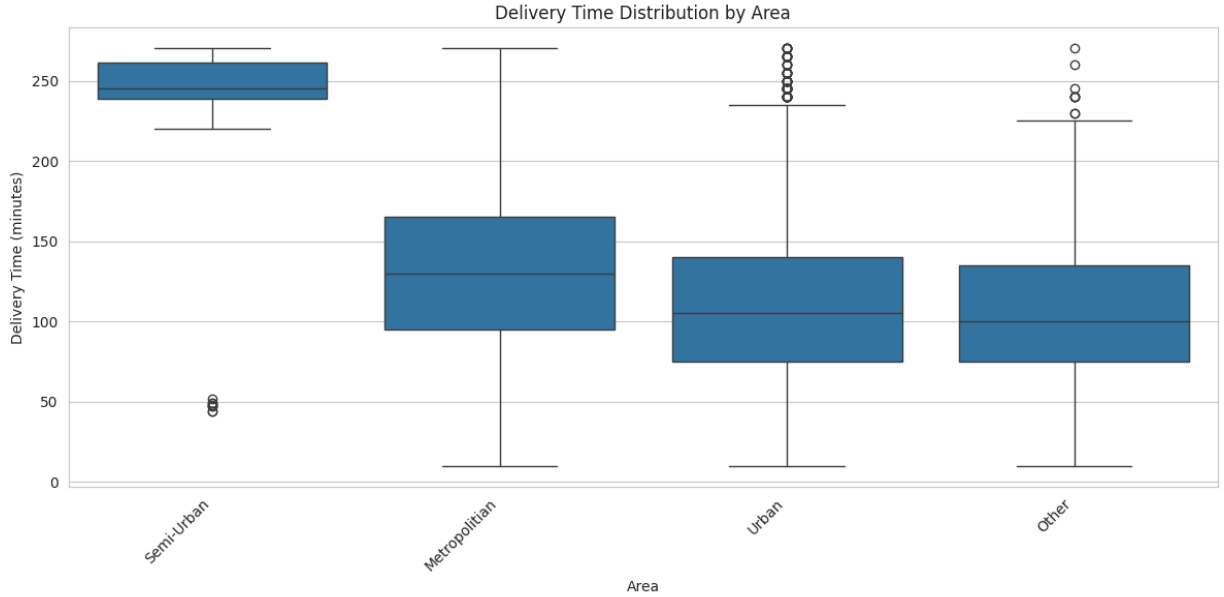



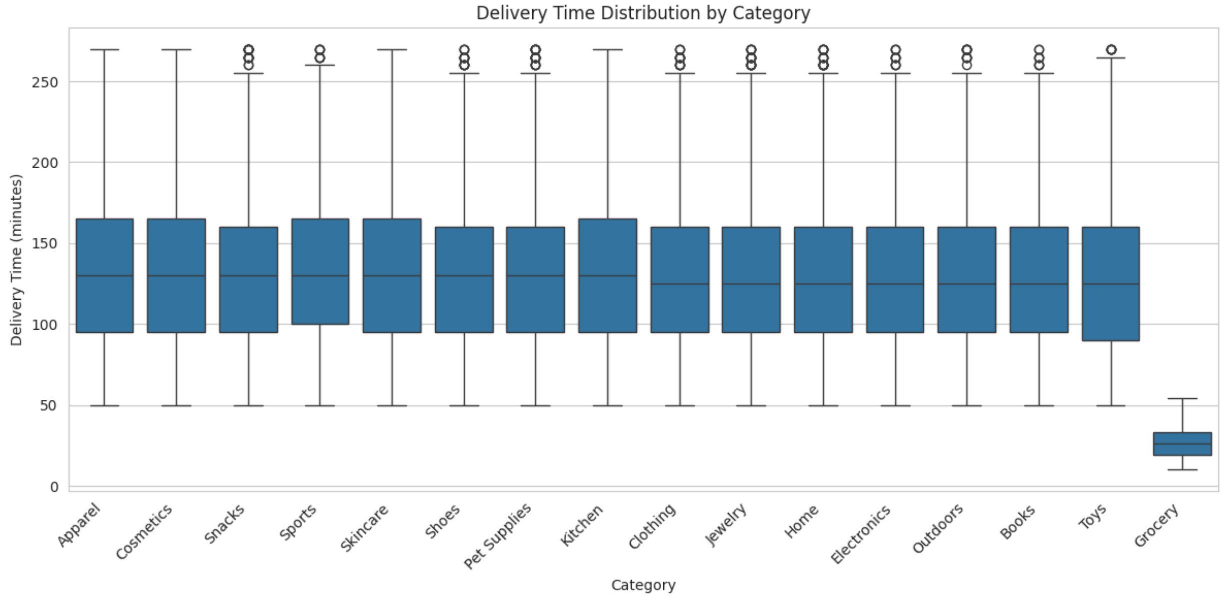

Order_DayOfWeek
0.01
-0.01
0.01
0.01
-0.01
0.02
1.00
−0.2
Agent_Age
Agent_Rating
Delivery_Time
Delivery_Distance_km
Time_to_Pickup_min
Order_Hour
Order_DayOfWeek


Delivery Time Distribution by Traffic
Delivery Time (minutes)
0
50
100
150
200
250
Jam
High
Medium
Low
Traffic


Delivery Time Distribution by Weather
Delivery Time (minutes)
0
50
100
150
200
250
Cloudy
Fog
Sandstorms
Stormy
Windy
Sunny
Weather


Delivery Time Distribution by Vehicle
Delivery Time (minutes)
0
50
100
150
200
250


```
EDA Visualizations Generated and Saved as PNG files.

==================================================
STARTING MODELING AND MLFLOW TRACKING
==================================================

Starting training for Linear_Regression...
2025/10/04 17:18:02 WARNING mlflow.models.model: `artifact_path` is deprecated. I
--- Linear_Regression Results ---
RMSE: 33.48 | MAE: 26.40 | R-squared: 0.59
2025/10/04 17:18:06 WARNING mlflow.models.model: Model logged without a signatur

Starting training for Random_Forest...
2025/10/04 17:22:20 WARNING mlflow.models.model: `artifact_path` is deprecated. I
--- Random_Forest Results ---
RMSE: 22.73 | MAE: 17.45 | R-squared: 0.81
2025/10/04 17:22:25 WARNING mlflow.models.model: Model logged without a signatur

Starting training for XGBoost...
2025/10/04 17:22:27 WARNING mlflow.models.model: `artifact_path` is deprecated. I
--- XGBoost Results ---
RMSE: 22.43 | MAE: 17.42 | R-squared: 0.81
```

```
2025/10/04 17:22:32 WARNING mlflow.models.model: Model logged without a signatur

Training and MLflow Tracking Complete. Best model saved is: XGBoost
Best performing model pipeline saved as 'best_model_pipeline.pkl' for Streamlit
```

```
import pandas as pd
import numpy as np

# Load the original dataset (ensure 'amazon_delivery.csv' is uploaded)
try:
    df = pd.read_csv('amazon_delivery.csv')
except FileNotFoundError:
    print("ERROR: 'amazon_delivery.csv' not found. Please upload the original f
    exit()

# --- Data Cleaning and Feature Engineering Logic ---

# 1. Cleaning
for col in ['Weather', 'Traffic', 'Vehicle', 'Area', 'Category']:
    if col in df.columns:
        df[col] = df[col].astype(str).str.strip()

df['Agent_Rating'] = df['Agent_Rating'].fillna(df['Agent_Rating'].mean())
```