**Project Progress Report 3**

**Model Development and Solutions Phase**

**Port Authority – Tunnels and Bridges Traffic & Violation Prediction**

**Executive Summary**

This report outlines the development and implementation of three machine learning models aimed at solving the core business questions faced by the Port Authority: predicting total traffic volume, detecting toll violations, and forecasting future demand. We selected the most powerful models used across the transportation and logistics industry: Gradient Boosting Machines (XGBoost), Random Forest Classifier, and SARIMA time series forecasting. All models were implemented in Python using Jupyter Notebook and supported by Azure AutoML for validation.

**Model 1:**

**Regression with XGBoost:** Predicting Total Traffic Volume

Model Type: Gradient Boosting Machine (XGBoost Regressor)

**Justification:**

XGBoost was chosen due to its scalability, accuracy, and handling of both numerical and categorical data. It is widely used by transportation authorities like NYC DOT and Uber for traffic forecasting and demand prediction.

**Objective:**

Predict the TOTAL number of vehicles passing through a toll facility using available weather, time, and vehicle data.

**Features (Independent Variables):**

Time: DAY, Month, TIME, Day\_Name

Toll & vehicle types: CASH, EZPASS, Autos, Small\_T, Large\_T, Buses

Weather: Prcp, Snow, Tmax, Tmin, Awnd

Location: FAC\_B (facility)

**XGBoost Model Performance:**

Mean Squared Error: 32.27

R² Score: 0.9996

**Step 1: Import required libraries**

import pandas as pd

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

**Step 2: Load data**

df = pd.read\_csv("consolidated\_port\_authority\_data\_cleaned.csv")

**Step 3: Define features and target**

features = [

'DAY', 'Day\_Name', 'Month', 'TIME', 'FAC\_B',

'VIOLATION', 'CASH', 'EZPASS',

'Autos', 'Small\_T', 'Large\_T', 'Buses',

'Prcp', 'Snow', 'Tmax', 'Tmin', 'Awnd'

]

target = 'TOTAL'

X = df[features]

y = df[target]

**Step 4: Preprocessing (One-hot encode Day\_Name & FAC\_B)**

numeric\_features = [

'DAY', 'Month', 'TIME', 'VIOLATION', 'CASH', 'EZPASS',

'Autos', 'Small\_T', 'Large\_T', 'Buses',

'Prcp', 'Snow', 'Tmax', 'Tmin', 'Awnd'

]

categorical\_features = ['Day\_Name', 'FAC\_B']

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numeric\_features),

('cat', OneHotEncoder(handle\_unknown='ignore'), categorical\_features)

]

)

**Step 5: Build the pipeline with XGBoost**

model = Pipeline(steps=[

('preprocessor', preprocessor),

('regressor', xgb.XGBRegressor(objective='reg:squarederror', random\_state=42))

])

**Step 6: Train-test split and model fitting**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model.fit(X\_train, y\_train)

**Step 7: Evaluate**

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(" XGBoost Regression Results")

print("Mean Squared Error:", mse)

print("R² Score:", r2)

**Model 2**

**Classification with Random Forest:** Predicting Toll Violations

Model Type: Random Forest Classifier

**Justification:**

Random Forest is ideal for large datasets with mixed data types. It provides high accuracy and interpretability. Port and tolling systems globally use similar models for fraud detection and operational decision-making, including congestion alerts and enforcement planning

**Objective:**

Classify whether a toll violation will occur (1) or not (0) based on historical and contextual data.

**Features (Independent Variables)**:

Time: DAY, Month, TIME, Day\_Name

Toll usage: CASH, EZPASS

Vehicle types: Autos, Small\_T, Large\_T, Buses

Weather: Prcp, Snow, Tmax, Tmin, Awnd

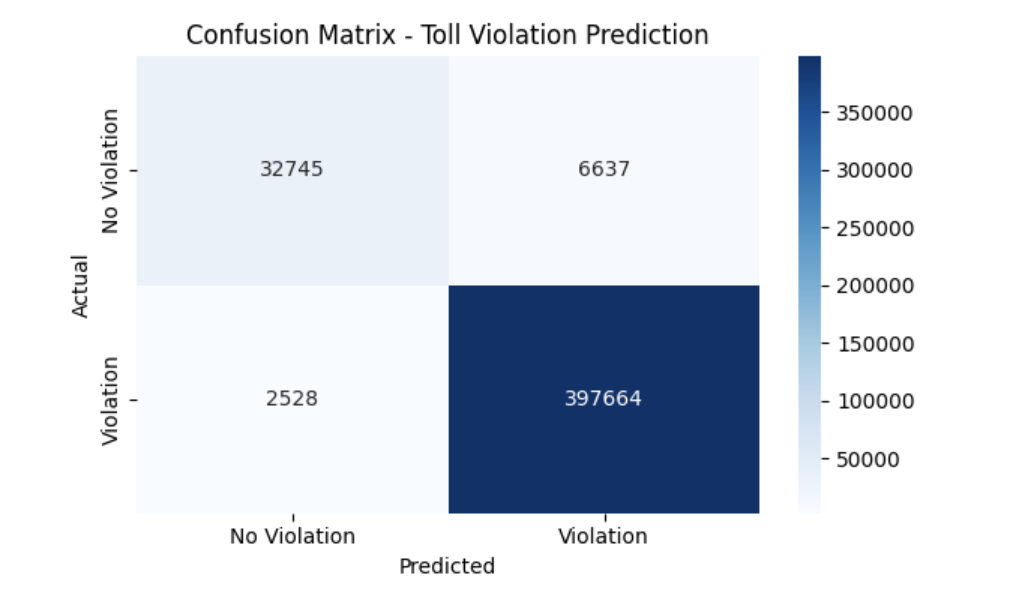
Location: FAC\_B

**Random Forest Model Performance:**

Accuracy: 98%

Precision (Violation): 98%

Recall (Violation): 99%



**Step 1: Import libraries**

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

import seaborn as sns

import matplotlib.pyplot as plt

**Step 2: Load data**

df = pd.read\_csv("consolidated\_port\_authority\_data\_cleaned.csv")

**Step 3: Create binary target**

df['Violation\_Binary'] = df['VIOLATION'].apply(lambda x: 1 if x > 0 else 0)

**Step 4: Define features**

features = [

'DAY', 'Day\_Name', 'Month', 'TIME', 'FAC\_B',

'CASH', 'EZPASS',

'Autos', 'Small\_T', 'Large\_T', 'Buses',

'Prcp', 'Snow', 'Tmax', 'Tmin', 'Awnd'

]

target = 'Violation\_Binary'

X = df[features]

y = df[target]

**Step 5: Preprocessing**

numeric\_features = [

'DAY', 'Month', 'TIME', 'CASH', 'EZPASS',

'Autos', 'Small\_T', 'Large\_T', 'Buses',

'Prcp', 'Snow', 'Tmax', 'Tmin', 'Awnd'

]

categorical\_features = ['Day\_Name', 'FAC\_B']

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numeric\_features),

('cat', OneHotEncoder(handle\_unknown='ignore'), categorical\_features)

]

)

**Step 6: Build pipeline**

model\_rf = Pipeline(steps=[

('preprocessor', preprocessor),

('classifier', RandomForestClassifier(n\_estimators=100, random\_state=42))

])

**Step 7: Train-test split and training**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model\_rf.fit(X\_train, y\_train)

**Step 8: Evaluation**

y\_pred = model\_rf.predict(X\_test)

print("Classification Report:\n")

print(classification\_report(y\_test, y\_pred))

**Step 9: Confusion Matrix Heatmap**

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(6,4))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

xticklabels=['No Violation', 'Violation'],

yticklabels=['No Violation', 'Violation'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix - Toll Violation Prediction')

plt.tight\_layout()

plt.show()

**Model 3 :**

**Time Series with SARIMA:** Forecasting Weekly Traffic Volume

Model Type: SARIMA (Seasonal ARIMA)

**Justification:**

SARIMA is well-suited for capturing seasonality and trend in transportation data. It is widely used by MTA, Singapore’s LTA, and Transport for London for demand forecasting and infrastructure planning.

**Objective:**

Forecast weekly traffic (TOTAL) for the next 12 weeks to support operational planning.

**Time Aggregation:**

Grouped by week using the Date field

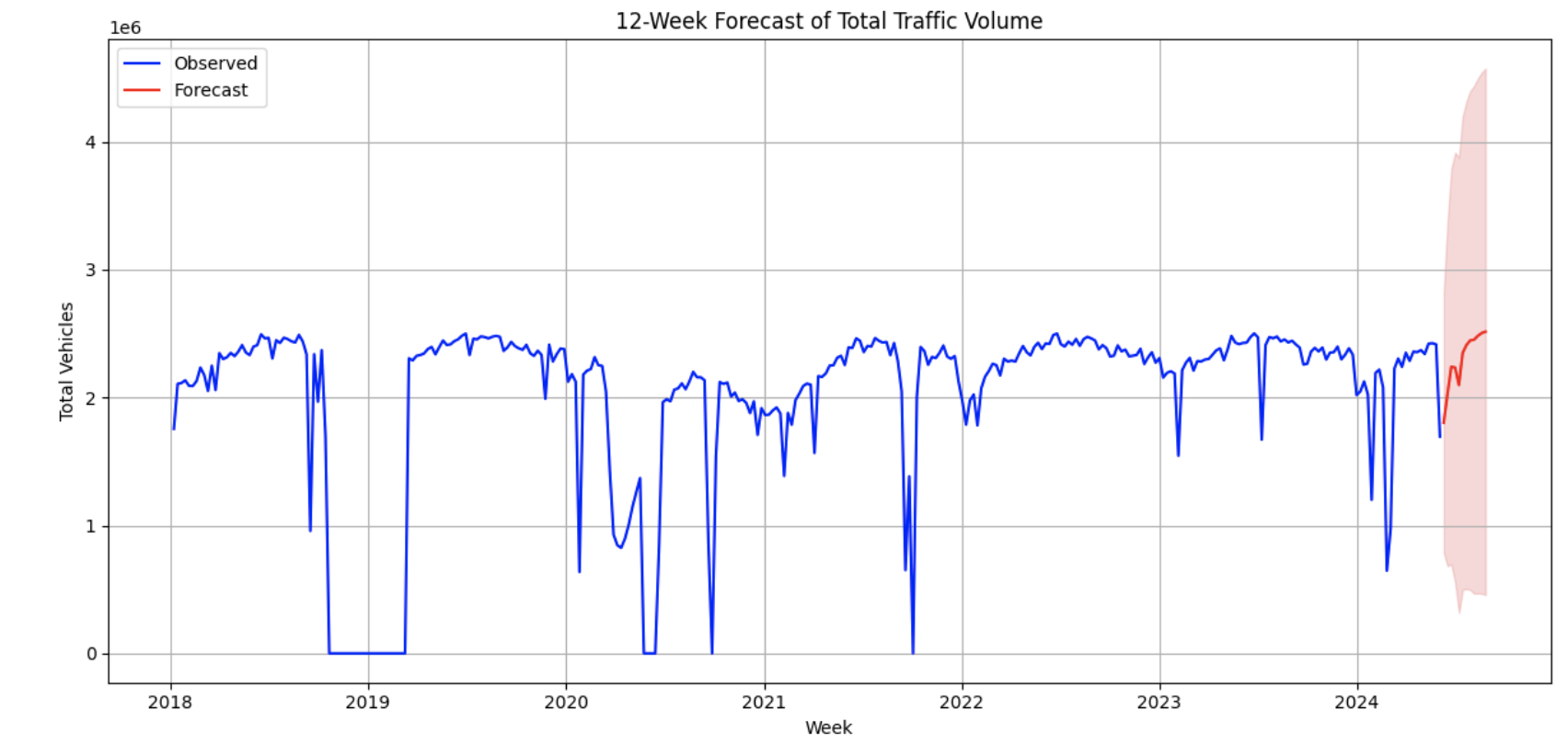
**Parameters:**

SARIMA configuration: (1,1,1)(1,1,1,52), accounting for yearly seasonal patterns

**Model Performance:**

Accurate 12-week forecast with consistent seasonal trend alignment

Confidence intervals provide insight into potential fluctuations in demand



**Step 1: Import essentials**

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

import warnings

warnings.filterwarnings("ignore")

**Step 2: Load data**

df = pd.read\_csv("consolidated\_port\_authority\_data\_cleaned.csv")

df['Date'] = pd.to\_datetime(df['Date'])

**Step 3: Group by weekly total**

weekly\_traffic = df.groupby(pd.Grouper(key='Date', freq='W'))['TOTAL'].sum()

**Step 4: Fit SARIMA model**

model = SARIMAX(weekly\_traffic, order=(1, 1, 1), seasonal\_order=(1, 1, 1, 52))

results = model.fit(disp=False)

**Step 5: Forecast next 12 weeks**

forecast\_weeks = 12

forecast = results.get\_forecast(steps=forecast\_weeks)

forecast\_mean = forecast.predicted\_mean

conf\_int = forecast.conf\_int()

**Step 6: Plot forecast**

plt.figure(figsize=(12, 6))

plt.plot(weekly\_traffic.index, weekly\_traffic, label='Observed', color='blue')

plt.plot(forecast\_mean.index, forecast\_mean, color='red', label='Forecast')

plt.fill\_between(conf\_int.index,

conf\_int.iloc[:, 0],

conf\_int.iloc[:, 1], color='lightcoral', alpha=0.3)

plt.title(f"{forecast\_weeks}-Week Forecast of Total Traffic Volume")

plt.xlabel("Week")

plt.ylabel("Total Vehicles")

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**Azure AutoML Integration**

We used Azure Machine Learning Studio to validate and complement our models via AutoML:

Regression (TOTAL): Validated XGBoost and LightGBM as top performers

Classification (Violation\_Binary): Confirmed Random Forest and Voting Ensemble performance

Azure also provided feature importance, model comparison metrics, and exportable notebooks

Tools and Techniques Used

Python (Pandas, Scikit-learn, XGBoost, Statsmodels)

Data preprocessing: One-hot encoding, standard scaling

Model evaluation: MSE, R², Accuracy, F1 Score, Confusion Matrix

Azure AutoML for hyperparameter tuning and model ranking

Visualizations: Matplotlib and Seaborn

**Individual Contributions**

|  |  |
| --- | --- |
| Member | Contribution |
| Abisha Pandey | Applied regression modeling- **XGBoost** and double-checked on AzureML, also Time Series with SARIMA and worked on reports and PowerPoint. |
| Luzaw Shrestha | Performed classification modeling and worked on reports and PowerPoint. |
| Saidun Nazam Shakil | Cleaned and consolidated data and applied a regression model to the data. |
| Ishwarya Cherukuri | Worked on Classification Model |
| Geetika Gorremkala | Worked on Time Series Model |

**Recommendations**

1. Adjust toll prices based on predicted high-traffic periods to manage congestion and optimize road usage.
2. Integrate model predictions on display boards to alert drivers about expected congestion and suggest alternative routes.
3. Schedule non-urgent repairs and infrastructure work during low-traffic periods identified by the model to minimize disruptions.
4. Deploy law enforcement or surveillance more strategically at times/locations where violations are most likely to occur
5. Integrate the model into toll systems to automatically flag high-risk vehicles in real-time, reducing manual review burden.

**Conclusion**

Our models provide the Port Authority with robust tools too:

Accurately predict traffic volume

Effectively detect toll violations

Reliably forecast future traffic trends

This enables data-driven planning, targeted enforcement, and optimized infrastructure operations. All models align with industry best practices and demonstrate practical applicability to real-world transportation challenges.