```
# Cell 1: Imports (run this first)

import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score
```

```
# Cell 2: Upload CSV from your computer

from google.colab import files

uploaded = files.upload()   # Choose your Urban Traffic CSV file

file_name = list(uploaded.keys())[0]
print("Uploaded file:", file_name)

# Read the CSV
df = pd.read_csv(file_name)
print("Shape:", df.shape)
df.head()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving urban_traffic_flow_with_target.csv to urban_traffic_flow_with_target (1).csv
Uploaded file: urban_traffic_flow_with_target (1).csv
Shape: (200, 7)
```

	Timestamp	Location	Vehicle_Count	Vehicle_Speed	Congestion_Level	Peak_Off_Peak	Target_Vehicle_Count
0	2024-01-01 00:00:00	Sensor_02	63	55.680777	1	Off-Peak	50
1	2024-01-01 00:15:00	Sensor_04	50	76.680379	0	Off-Peak	42
2	2024-01-01 00:30:00	Sensor_02	42	48.598038	2	Off-Peak	55
3	2024-01-01 00:45:00	Sensor_02	55	59.796931	1	Off-Peak	46
4	2024-01-01 01:00:00	Sensor_01	46	48.094884	2	Off-Peak	48

```
# Cell 3: Understand the data

print("Columns:\n", df.columns)
print("\nData types:\n", df.dtypes)
print("\nSample rows:")
display(df.head())
```

```
Columns:
 Index(['Timestamp', 'Location', 'Vehicle_Count', 'Vehicle_Speed',
       'Congestion_Level', 'Peak_Off_Peak', 'Target_Vehicle_Count'],
      dtype='object')

Data types:
 Timestamp               object
Location                 object
Vehicle_Count             int64
Vehicle_Speed           float64
Congestion_Level          int64
Peak_Off_Peak            object
Target_Vehicle_Count      int64
dtype: object

Sample rows:
```

	Timestamp	Location	Vehicle_Count	Vehicle_Speed	Congestion_Level	Peak_Off_Peak	Target_Vehicle_Count
0	2024-01-01 00:00:00	Sensor_02	63	55.680777	1	Off-Peak	50
1	2024-01-01 00:15:00	Sensor_04	50	76.680379	0	Off-Peak	42
2	2024-01-01 00:30:00	Sensor_02	42	48.598038	2	Off-Peak	55
3	2024-01-01 00:45:00	Sensor_02	55	59.796931	1	Off-Peak	46
4	2024-01-01 01:00:00	Sensor_01	46	48.094884	2	Off-Peak	48

```
# ✅ UPDATED Cell 4: Detect datetime + create best time features

datetime_col = None
for col in df.columns:
```

```
    col_lower = col.lower()
    if "time" in col_lower or "date" in col_lower or "datetime" in col_lower:
        datetime_col = col
        break

print("Detected datetime column:", datetime_col)

if datetime_col is None:
    raise ValueError("No datetime/time/date column detected automatically.")

# Convert to datetime
df[datetime_col] = pd.to_datetime(df[datetime_col], errors="coerce")

# Drop invalid datetime rows
df = df.dropna(subset=[datetime_col])

# Core time features
df["hour"] = df[datetime_col].dt.hour
df["day_of_week"] = df[datetime_col].dt.dayofweek   # 0=Mon ... 6=Sun

# ✅ Best weekend features
df["is_weekend"] = df["day_of_week"].isin([5, 6]).astype(int)
df["is_saturday"] = (df["day_of_week"] == 5).astype(int)
df["is_sunday"] = (df["day_of_week"] == 6).astype(int)

# Display output to verify
print(df[[datetime_col, "hour", "day_of_week", "is_weekend", "is_saturday", "is_sunday"]].head())
```

```
Detected datetime column: Timestamp
            Timestamp  hour  day_of_week  is_weekend  is_saturday  is_sunday
0 2024-01-01 00:00:00     0            0           0            0          0
1 2024-01-01 00:15:00     0            0           0            0          0
2 2024-01-01 00:30:00     0            0           0            0          0
3 2024-01-01 00:45:00     0            0           0            0          0
4 2024-01-01 01:00:00     1            0           0            0          0
```

```
# ✅ FINAL Cell 5: Select FUTURE traffic as target (Project-Correct)

# For your delivery routing project, we must predict FUTURE traffic
# So we directly set the correct target column:

target_col = "Target_Vehicle_Count"
y = df[target_col]

print("✅ Target column set for FUTURE traffic prediction:", target_col)
print("\nSample target values:")
y.head()
```

✅ Target column set for FUTURE traffic prediction: Target_Vehicle_Count

Sample target values:

	Target_Vehicle_Count
0	50
1	42
2	55
3	46
4	48

dtype: int64

```
# ✅✅ FINAL, PROJECT-SAFE CELL 6 (NO DATA LEAKAGE)

# Best time-based features (from Cell 4)
feature_cols = [
    "hour",
    "day_of_week",
    "is_weekend"
]

# Add other numeric features EXCEPT the target
for col in df.select_dtypes(include=["int64", "float64"]).columns:
    if col not in feature_cols and col != target_col:
        feature_cols.append(col)

print("✅ Numeric feature columns used for ML:", feature_cols)
```

```
# Detect categorical columns
cat_cols = df.select_dtypes(include=["object", "category"]).columns.tolist()
print("Categorical columns:", cat_cols)

# One-hot encode categorical columns
df_encoded = pd.get_dummies(df, columns=cat_cols, drop_first=True)

# Build final X WITHOUT the target column
X = df_encoded[feature_cols + [c for c in df_encoded.columns if c not in df.columns]]

print("✅ Final feature shape (no leakage):", X.shape)
X.head()
```

```
✅ Numeric feature columns used for ML: ['hour', 'day_of_week', 'is_weekend', 'Vehicle_Count', 'Vehicle_Speed', 'Congestion
Categorical columns: ['Location', 'Peak_Off_Peak']
✅ Final feature shape (no leakage): (200, 13)
```

	hour	day_of_week	is_weekend	Vehicle_Count	Vehicle_Speed	Congestion_Level	is_saturday	is_sunday	Location_Sensor_02
0	0	0	0	63	55.680777	1	0	0	True
1	0	0	0	50	76.680379	0	0	0	False
2	0	0	0	42	48.598038	2	0	0	True
3	0	0	0	55	59.796931	1	0	0	True
4	1	0	0	46	48.094884	2	0	0	False

```
# Cell 7: Train-test split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Train size:", X_train.shape, "Test size:", X_test.shape)
```

```
Train size: (160, 11) Test size: (40, 11)
```

```
# Cell 8: Train RandomForestRegressor (traffic volume prediction)

model = RandomForestRegressor(
    n_estimators=200,
    random_state=42,
    n_jobs=-1
)

model.fit(X_train, y_train)

print("Model trained ✅")
```

```
Model trained ✅
```

```
# Cell 9: Evaluate model using MAE and R^2

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae:.3f}")
print(f"R^2 Score: {r2:.3f}")

# Show a few actual vs predicted values
comparison = pd.DataFrame({
    "Actual": y_test.values[:10],
    "Predicted": y_pred[:10]
})
comparison
```

```
Mean Absolute Error (MAE): 5.303
R^2 Score: -0.270
```

	Actual	Predicted
0	59	50.360
1	53	53.730
2	50	51.225
3	54	50.385
4	51	50.825
5	44	51.685
6	60	51.610
7	49	50.130
8	56	52.495
9	61	47.885

```
# Cell 10: Demo – Predict traffic for a specific time & decide congestion

def predict_traffic(hour, day_of_week, is_weekend=0):
    # Build a single-row dataframe compatible with X
    sample = {col: 0 for col in X.columns}  # default zeros

    # Set time-based features
    if "hour" in sample:
        sample["hour"] = hour
    if "day_of_week" in sample:
        sample["day_of_week"] = day_of_week
    if "is_weekend" in sample:
        sample["is_weekend"] = is_weekend

    sample_df = pd.DataFrame([sample])
    pred = model.predict(sample_df)[0]
    return pred

# Example: Saturday 8 PM (hour=20, day_of_week=5, weekend=1)
predicted_volume = predict_traffic(hour=20, day_of_week=5, is_weekend=1)
print("Predicted traffic volume:", predicted_volume)

# Simple rule: if traffic volume > threshold → congested
threshold = y_train.median()  # you can tune this
print("Threshold (median traffic):", threshold)

if predicted_volume > threshold:
    print("⚠️ Road is likely CONGESTED. Suggest alternate route.")
else:
    print("✅ Road seems OK. Continue on current route.")
```

```
Predicted traffic volume: 51.005
Threshold (median traffic): 50.0
⚠️ Road is likely CONGESTED. Suggest alternate route.
```