# *Traffic Intelligence: Advanced Traffic Volume Estimation with Machine Learning*

*MENTOR: SRI **L.LAKSHMI NARAYANA***

***TEAM ID**: **LTVIP2025TMID40409***

*TEAM LEADER* : ***VAIBOYINA DEVI***

MAIL ID: deviyaiboina@gmail.com

*TEAM MEMBERS*:

1) **MARIPI SAI**

MAIL ID: saimaripi3@gmail.com

2) **SAPPA KIRAN MAI**

MAIL ID: kiransappa5@gmail.com

3) **YELETI SUDHARANI**

MAIL ID: yeletisudharani08@gmail.com

4) **AINAPARTHI SUJI**

MAIL ID: sujia0099@gmail.com

# ***INTRODUCTION TO AIML***

## Artificial Intelligence (AI)

AI refers to the simulation of **human intelligence by machines**. It enables computers and systems to perform tasks that typically require human intelligence, such as:

- Learning
- Reasoning
- Problem-solving
- Understanding language
- Perception

**Types of AI:**

1. **Narrow AI:** Focused on a specific task (e.g., Siri, Google Maps).
2. **General AI:** Human-level cognitive abilities (still theoretical).
3. **Super AI:** Exceeds human intelligence (not yet developed).

## Machine Learning (ML)

ML is a **subset of AI** that enables machines to **learn from data** and improve their performance over time **without being explicitly programmed**.

**Types of ML:**

1. **Supervised Learning:** Uses labeled data.
    - Examples: Regression, Classification
2. **Unsupervised Learning:** Uses unlabeled data.
    - Examples: Clustering, Association
3. **Reinforcement Learning:** Learning by trial and error with feedback.
    - Example: Game playing bots, self-driving cars

## Deep Learning (DL)

A subset of ML that uses **neural networks** with multiple layers (like the human brain) to analyze data and make decisions.

- Common in image processing, speech recognition, etc.
- Tools: TensorFlow, PyTorch

## What is Machine Learning?

**Machine Learning** is a branch of **Artificial Intelligence (AI)** that allows computers to **learn from data** and make decisions or predictions **without being explicitly programmed**.

In simple words:

"ML teaches machines to learn from experience (data), just like humans learn from practice."

---

## How Machine Learning Works

1. **Collect Data** – Historical data (e.g., past sales, weather data)
2. **Train a Model** – Use algorithms to learn patterns from the data
3. **Make Predictions** – Apply the model to new, unseen data
4. **Evaluate Accuracy** – Check how well the model performs

---

## Types of Machine Learning

| Type | Description | Example |
|---|---|---|
| ☑ **Supervised Learning** | Learn from labeled data | Email spam detection |
| **Unsupervised Learning** | Find hidden patterns in unlabeled data | Customer segmentation |
| **Reinforcement Learning** | Learn by trial & error with feedback | Self-driving cars, game bots |

---

## Common Algorithms

| Algorithm | Use Case |
|---|---|
| Linear Regression | Predicting prices/sales |
| Decision Trees | Making logical decisions |
| K-Nearest Neighbors | Pattern recognition |
| Support Vector Machines | Classification problems |
| K-Means Clustering | Grouping similar data |
| Random Forest | Complex predictions |

## Traffic Intelligence: Advanced Traffic Volume Estimation with Machine Learning

Traffic Intelligence is an advanced system that uses machine learning algorithms to estimate and predict traffic volume with precision. By analyzing historical traffic data, weather patterns, events, and other relevant factors, TrafficTelligence provides accurate forecasts and insights to enhance traffic management, urban planning, and commuter experiences

## 1) *Data Collection*:

ML depends heavily on data, without data, it is impossible for an “AI” model to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

***TSK-338833*:**

# Download the dataset

traffic volume.csv

Op vith Share

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | holiday | temp | rain | snow | weather | date | Time |
| 2 | None | 288.28 | 0 | 0 | Clouds | 02-10-2012 | 9 |
| 3 | None | 289.36 | 0 | 0 | Clouds | 02-10-2012 | 10 |
| 4 | None | 289.58 | 0 | 0 | Clouds | 02-10-2012 | 11 |
| 5 | None | 290.13 | 0 | 0 | Clouds | 02-10-2012 | 12 |
| 6 | None | 291.14 | 0 | 0 | Clouds | 02-10-2012 | 13 |
| 7 | None | 291.72 | 0 | 0 | Clear | 02-10-2012 | 14 |
| 8 | None | 293.17 | 0 | 0 | Clear | 02-10-2012 | 15 |
| 9 | None | 293.86 | 0 | 0 | Clear | 02-10-2012 | 16 |
| 10 | None | 294.14 | 0 | 0 | Clouds | 02-10-2012 | 17 |
| 11 | None | 293.1 | 0 | 0 | Clouds | 02-10-2012 | 18 |
| 12 | None | 290.97 | 0 | 0 | Clouds | 02-10-2012 | 19 |
| 13 | None | 289.38 | 0 | 0 | Clear | 02-10-2012 | 20 |
| 14 | None | 288.61 | 0 | 0 | Clear | 02-10-2012 | 21 |
| 15 | None | 287.16 | 0 | 0 | Clear | 02-10-2012 | 22 |
| 16 | None | 285.45 | 0 | 0 | Clear | 02-10-2012 | 23 |
| 17 | None | 284.63 | 0 | 0 | Clear | 03-10-2012 | 0 |
| 18 | None | 283.47 | 0 | 0 | Clear | 03-10-2012 | 1 |
| 19 | None | 281.18 | 0 | 0 | Clear | 03-10-2012 | 2 |
| 20 | None | 281.09 | 0 | 0 | Clear | 03-10-2012 | 3 |
| 21 | None | 279.53 | 0 | 0 | Clear | 03-10-2012 | 4 |
| 22 | None | 278.62 | 0 | 0 | Clear | 03-10-2012 | 5 |
| 23 | None | 278.23 | 0 | 0 | Clear | 03-10-2012 | 6 |

## 2) *Data Pre-processing*:

***TSK-338834*:**

# Import Necessary Libraries

```
[1]: import pandas as pd
     import numpy as np
     import sklearn as sk
     from sklearn import linear_model
     from sklearn import tree
     from sklearn import ensemble
     from sklearn import svm
     import xgboost
```

```
[2]: !pip install seaborn
```

***TSK-338835:***

# Importing the Dataset

```
Requirement already satisfied: six>=1.5 in
c:\users\dell\appdata\local\programs\python\python313\lib\site-packages (from
python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
```

[3]:
```
df = pd.read_csv('traffic volume - traffic volume.csv')
```

[4]:
```
df.head()
```

[4]:

| | holiday | temp | rain | snow | weather | date | Time | traffic_volume |
|---|---|---|---|---|---|---|---|---|
| 0 | NaN | 288.28 | 0.0 | 0.0 | Clouds | 02-10-2012 | 9:00:00 | 5545 |
| 1 | NaN | 289.36 | 0.0 | 0.0 | Clouds | 02-10-2012 | 10:00:00 | 4516 |
| 2 | NaN | 289.58 | 0.0 | 0.0 | Clouds | 02-10-2012 | 11:00:00 | 4767 |
| 3 | NaN | 290.13 | 0.0 | 0.0 | Clouds | 02-10-2012 | 12:00:00 | 5026 |
| 4 | NaN | 291.14 | 0.0 | 0.0 | Clouds | 02-10-2012 | 13:00:00 | 4918 |

***TSK-338836:***

# Analyse the data

[3]:
```
df = pd.read_csv('traffic volume - traffic volume.csv')
```

[4]:
```
df.head()
```

[4]:

| | holiday | temp | rain | snow | weather | date | Time | traffic_volume |
|---|---|---|---|---|---|---|---|---|
| 0 | NaN | 288.28 | 0.0 | 0.0 | Clouds | 02-10-2012 | 9:00:00 | 5545 |
| 1 | NaN | 289.36 | 0.0 | 0.0 | Clouds | 02-10-2012 | 10:00:00 | 4516 |
| 2 | NaN | 289.58 | 0.0 | 0.0 | Clouds | 02-10-2012 | 11:00:00 | 4767 |
| 3 | NaN | 290.13 | 0.0 | 0.0 | Clouds | 02-10-2012 | 12:00:00 | 5026 |
| 4 | NaN | 291.14 | 0.0 | 0.0 | Clouds | 02-10-2012 | 13:00:00 | 4918 |

[5]:
```
df.describe()
```

[5]:

| | temp | rain | snow | traffic_volume |
|---|---|---|---|---|
| count | 48151.000000 | 48202.000000 | 48192.000000 | 48204.000000 |
| mean | 281.205351 | 0.334278 | 0.000222 | 3259.818355 |
| std | 13.343675 | 44.790062 | 0.008169 | 1986.860670 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 272.160000 | 0.000000 | 0.000000 | 1193.000000 |
| 50% | 282.460000 | 0.000000 | 0.000000 | 3380.000000 |
| 75% | 291.810000 | 0.000000 | 0.000000 | 4933.000000 |
| max | 310.070000 | 9831.300000 | 0.510000 | 7280.000000 |

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48204 entries, 0 to 48203
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```



```
 0   holiday         61 non-null     object
 1   temp            48151 non-null  float64
 2   rain            48202 non-null  float64
 3   snow            48192 non-null  float64
 4   weather         48155 non-null  object
 5   date            48204 non-null  object
 6   Time            48204 non-null  object
 7   traffic_volume  48204 non-null  int64
dtypes: float64(3), int64(1), object(4)
memory usage: 2.9+ MB
```

***TSK-338837:***

# Handling Missing Values

[7]:
```
df.isnull().sum()
```

[7]:
```
holiday          48143
temp                53
rain                 2
snow                12
weather             49
date                 0
Time                 0
traffic_volume       0
dtype: int64
```

[8]:
```
df['temp'].fillna(df['temp'].mean(),inplace=True)
df['rain'].fillna(df['rain'].mean(),inplace=True)
df['snow'].fillna(df['snow'].mean(),inplace=True)
```

```
df['snow'].fillna(df['snow'].mean(),inplace=True)
```

[9]:
```
df['weather'].fillna(df['weather'].mode()[0],inplace=True)
df['holiday'].fillna(df['holiday'].mode()[0],inplace=True)
```

[10]:
```
df
```

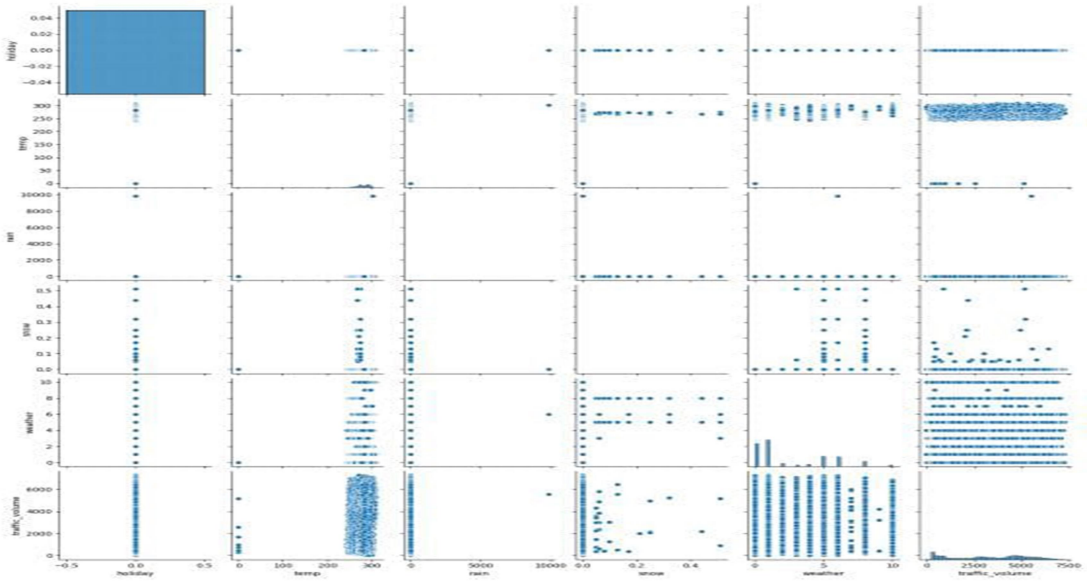
[10]:

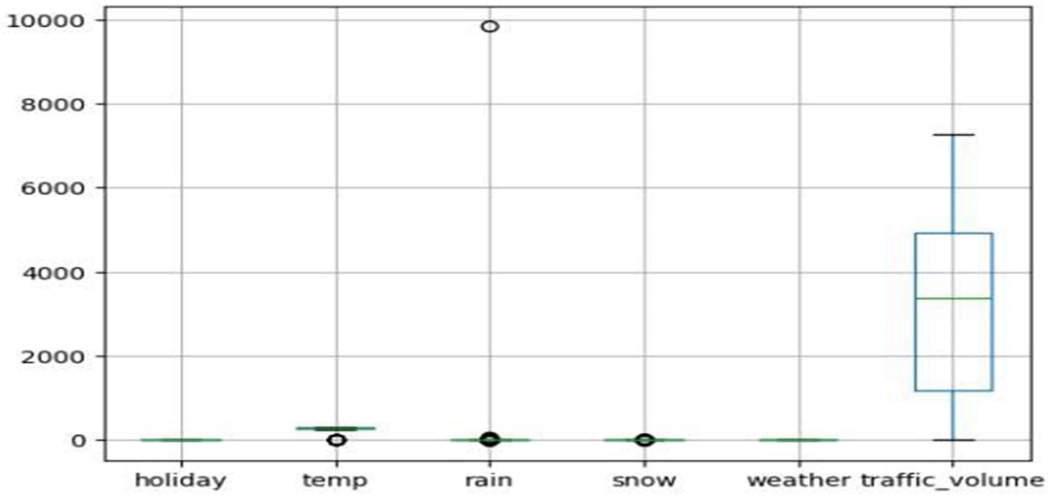
| | holiday | temp | rain | snow | weather | date | Time | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | Labor Day | 288.28 | 0.0 | 0.0 | Clouds | 02-10-2012 | 9:00:00 | |
| 1 | Labor Day | 289.36 | 0.0 | 0.0 | Clouds | 02-10-2012 | 10:00:00 | |
| 2 | Labor Day | 289.58 | 0.0 | 0.0 | Clouds | 02-10-2012 | 11:00:00 | |
| 3 | Labor Day | 290.13 | 0.0 | 0.0 | Clouds | 02-10-2012 | 12:00:00 | |
| 4 | Labor Day | 291.14 | 0.0 | 0.0 | Clouds | 02-10-2012 | 13:00:00 | |
| … | … | … | … | … | … | … | … | |
| 48199 | Labor Day | 283.45 | 0.0 | 0.0 | Clouds | 30-09-2018 | 19:00:00 | |
| 48200 | Labor Day | 282.76 | 0.0 | 0.0 | Clouds | 30-09-2018 | 20:00:00 | |
| 48201 | Labor Day | 282.73 | 0.0 | 0.0 | Thunderstorm | 30-09-2018 | 21:00:00 | |
| 48202 | Labor Day | 282.09 | 0.0 | 0.0 | Clouds | 30-09-2018 | 22:00:00 | |
| 48203 | Labor Day | 282.12 | 0.0 | 0.0 | Clouds | 30-09-2018 | 23:00:00 | |

| | traffic_volume |
|---|---|
| 0 | 5545 |
| 1 | 4516 |
| 2 | 4767 |
| 3 | 5026 |

***TSK-338838:***

# Data Visualization






***TSK-338839:***

# Splitting the Dataset into Dependent and Independent variable

[24]:
```
# Target variable
y = df['traffic_volume']

# Feature set (drop traffic_volume)
x = df.drop(columns=['traffic_volume'], axis=1)
```

***TSK-338840:***

# Feature Scaling

[24]:
```
# Target variable
y = df['traffic_volume']

# Feature set (drop traffic_volume)
x = df.drop(columns=['traffic_volume'], axis=1)
```

[25]:
```
import pandas as pd
from sklearn.preprocessing import scale

# Step 1: Separate target variable
y = df['traffic_volume']

# Step 2: Drop target from features
x = df.drop(columns=['traffic_volume'], axis=1)

# Step 3: Save column names before scaling
names = x.columns
```

```
# Step 4: Apply scaling (Z-score normalization)
x = scale(x)  # This returns a NumPy array

# Step 5: Convert scaled data back to DataFrame with original column names
x = pd.DataFrame(x, columns=names)

# Step 6: Preview the result
x.head()
```

[25]:

| | holiday | temp | rain | snow | weather | year | month |
|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.530485 | -0.007463 | -0.027235 | -0.566452 | -1.855294 | 1.02758 |
| 1 | 0.0 | 0.611467 | -0.007463 | -0.027235 | -0.566452 | -1.855294 | 1.02758 |
| 2 | 0.0 | 0.627964 | -0.007463 | -0.027235 | -0.566452 | -1.855294 | 1.02758 |
| 3 | 0.0 | 0.669205 | -0.007463 | -0.027235 | -0.566452 | -1.855294 | 1.02758 |
| 4 | 0.0 | 0.744939 | -0.007463 | -0.027235 | -0.566452 | -1.855294 | 1.02758 |

| | day | hours | minutes | seconds |
|---|---|---|---|---|
| 0 | -1.574903 | -0.345548 | 0.0 | 0.0 |
| 1 | -1.574903 | -0.201459 | 0.0 | 0.0 |
| 2 | -1.574903 | -0.057371 | 0.0 | 0.0 |
| 3 | -1.574903 | 0.086718 | 0.0 | 0.0 |
| 4 | -1.574903 | 0.230807 | 0.0 | 0.0 |

***TSK-338841:***

# Splitting the data into Train and Test

[26]:
```
from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=0
)
```

## 3) *Model Building*:

### *TSK-338842:*

# Training and Testing the Model

[27]:
```
from sklearn import linear_model, tree, ensemble, svm
import xgboost
from sklearn.metrics import r2_score, mean_squared_error

# Initialize models
lin_reg = linear_model.LinearRegression()
Dtree = tree.DecisionTreeRegressor()
Rand = ensemble.RandomForestRegressor()
svr = svm.SVR()
XGB = xgboost.XGBRegressor()


# Train all models
lin_reg.fit(x_train, y_train)
Dtree.fit(x_train, y_train)
Rand.fit(x_train, y_train)
```

```
svr.fit(x_train, y_train)
XGB.fit(x_train, y_train)


# Predict on training set
p1 = lin_reg.predict(x_train)
p2 = Dtree.predict(x_train)
p3 = Rand.predict(x_train)
p4 = svr.predict(x_train)
p5 = XGB.predict(x_train)
```

### *TSK-338843:*

# Model Evaluation

[28]:
```
from sklearn import metrics

# R² Score on training predictions
print("Linear Regression R² on training set:", metrics.r2_score(y_train, p1))
print("Decision Tree R² on training set:", metrics.r2_score(y_train, p2))
print("Random Forest R² on training set:", metrics.r2_score(y_train, p3))
print("SVR R² on training set:", metrics.r2_score(y_train, p4))
print("XGBoost R² on training set:", metrics.r2_score(y_train, p5))
```

```
Linear Regression R² on training set: 0.13296715651758817
Decision Tree R² on training set: 1.0
Random Forest R² on training set: 0.9774641542381987
SVR R² on training set: 0.25417555904090483
XGBoost R² on training set: 0.8734947443008423
```

[29]:
```
p1=lin_reg.predict(x_test)
p2=Dtree.predict(x_test)
p3=Rand.predict(x_test)
p4=svr.predict(x_test)
p5=XGB.predict(x_test)

print(metrics.r2_score(y_test, p1))
print(metrics.r2_score(y_test, p2))
print(metrics.r2_score(y_test, p3))
print(metrics.r2_score(y_test, p4))
print(metrics.r2_score(y_test, p5))
```

```
0.13558012821266197
0.6881461887460081
0.8391783560539203
0.25808608029636915
0.8377184271812439
```

[30]:
```
MSE=metrics.mean_squared_error(p3,y_test)
np.sqrt(MSE)
```

```
[30]: np.float64(793.0544564468826)
```

***TSK-338844:***

## Save the Model

```
[31]: import pickle

      pickle.dump(Rand,open("model.pkl",'wb'))
      pickle.dump(le,open("encoder.pkl",'wb'))

[ ]:
```

## *4) Application Building:*

we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

***TSK-338845:***

## Build HTML Code

***TSK-338846*:**

# Main Python Script


***TSK-338847* & *TSK-338848*:**

# Run the App & Output

## COMMENTS:

**I have learn so many concepts about AIML(Artificial Intelligence & Machine Learning) me and my team can learn how to tackle do it DATA by Machine Learning and it is very interesting to do it in step to step and finally in project we will do it project**

**-------- VAIBOYINA DEVI**

**Team Leader**