Traffictelligence:Advanced Traffic Volume Estimation With Machine Learning

By

(KRISHANMURTI KUMAR)

(RAJANI MUDIMADUGU)

(SHASHI KUMAR)

(VIKASH KUMAR UPADHAYA)

Guided by

Prof. Ms Raasha

A Dissertation Submitted to SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY, An Autonomous Institution affiliated to 'JNTU Ananthapur' in Partial Fulfilment of the Bachelor of Technology (***Computer Engineering***) with Specialization in ***Artificial Intelligence and Machine Learning.***

May 2024

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY
R.V.S. Nagar Tirupathi Road, Andhra Pradesh– 517127

DATA PREPROCESSING:

Data Pre-processing includes the following main tasks

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Feature Scaling.
- Splitting Data into Train and Test.

➢ Import Necessary Libraries:

It is important to import all the necessary libraries such as pandas, NumPy,matplotlib.

- **Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
- **Pandas-** It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Matplotlib**- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python
- **Sklearn** – which contains all the modules required for model building.

```
# importing the necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as sk
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
import xgboost
```

➢ Importing the dataset:

- You might have your data in .csv files, .excel files
- Let's load a .csv data file into pandas using read_csv() function.We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).
- If your dataset is in some other location, Then
- **Data=pd.read_csv(r"File_location/datasetname.csv")**

Note: r stands for "raw" and will cause backslashes in the string to be interpreted as actual backslashes rather than special characters.

- If the dataset is in the same directory of your program, you can directly read it, without giving raw as r.
- Our Dataset weatherAus.csv contains the following Columns

- Holiday - working day or holiday
- Temp- temperature of the day
- Rain and snow – whether it is raining or snowing on that day or not
- Weather = describes the weather conditions of the day
- Date and time = represents the exact date and time of the day
- Traffic volume – output column

The output column to be predicted is Traffic volume.Based on the input variables we predict the volume of the traffic. The predicted output gives them a fair idea of the count of traffic

➢ Analyse the data:

head() method is used to return top n (5 by default) rows of a DataFrame orseries.

```
# displaying first 5 columns of the data
data.head()
```

	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	None	288.28	0.0	0.0	Clouds	02-10-2012	09:00:00	5545
1	None	289.36	0.0	0.0	Clouds	02-10-2012	10:00:00	4516
2	None	289.58	0.0	0.0	Clouds	02-10-2012	11:00:00	4767
3	None	290.13	0.0	0.0	Clouds	02-10-2012	12:00:00	5026
4	None	291.14	0.0	0.0	Clouds	02-10-2012	13:00:00	4918

describe() method computes a summary of statistics like count, mean, standard deviation, min, max, and quartile values.

```
data.describe()
```

The output is as shown below

```
# used to understand the descriptive analysis of the data
data.describe()
```

	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

From the data, we infer that there are only decimal values and no categorical values.

info() gives information about the data - paste the image here.

```
# used to display the basic information of the data
data.info()
```

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

➢ Handling Missing Values:

1. The Most important step in data pre-processing is dealing with missing data, the presence of missing data in the dataset can lead to low accuracy.

2. Check whether any null values are there or not. if it is present then the following can be done.

```
# used to display the null values of the data
data.isnull().sum()
```

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

There are missing values in the dataset, we will fill the missing values in the columns.

3. We are using mean and mode methods for filling the missing values

- Columns such as temp, rain, and snow are the numeric columns, when there is a numeric column you should fill the missing values with the mean/median method. so here we are using the mean method to fill the missing values.
- Weather column has a categorical data type, in such case missing data needs to be filled with the most repeated/ frequent value. Clouds are the most repeated value in the column, so imputing with cloudsvalue.

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

```
print(Counter(data['weather']))
```

```
Counter({'Clouds': 15144, 'Clear': 13383, 'Mist': 5942, 'Rain': 5665, 'Snow': 2875, 'Drizzle': 1818, 'H
'Fog': 912, nan: 49, 'Smoke': 20, 'Squall': 4})
```

```
data['weather'].fillna('Clouds',inplace=True)
```

➢ Data visualization:

Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data.

Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.

- To visualize the dataset we need libraries called Matplotlib and Seaborn.
- The Matplotlib library is a Python 2D plotting library that allows you to generate plots, scatter plots, histograms, bar charts etc.

Let’s visualize our data using Matplotlib and seaborn library.

Before diving into the code, let's look at some of the basic properties we will be using when plotting.

xlabel: Set the label for the x-axis.

ylabel: Set the label for the y-axis.

title: Set a title for the axes.

Legend: Place a legend on the axes.

1. data.corr() gives the correlation between the columns

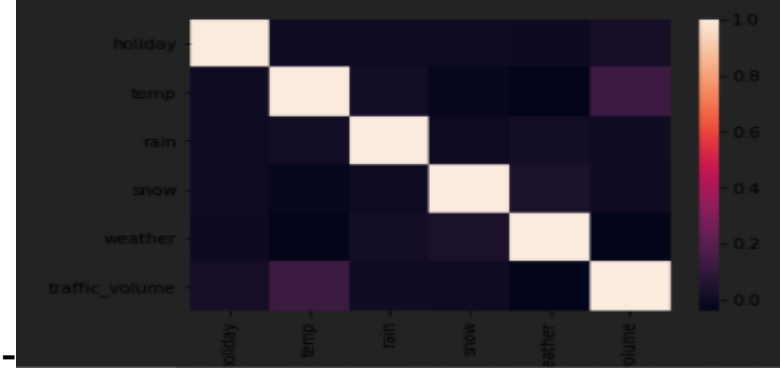
Correlation is a statistical term describing the degree to which two variables move in coordination with one another. If the two variables move in the same direction, then those variables are said to have a positive correlation. If they move in opposite directions, then they have a negativecorrelation.

```
cor
```

	holiday	temp	rain	snow	weather	traffic_volume
holiday	1.000000	-0.000472	0.000066	0.000432	-0.004328	0.018676
temp	-0.000472	1.000000	0.009070	-0.019758	-0.033559	0.130034
rain	0.000066	0.009070	1.000000	-0.000090	0.009542	0.004714
snow	0.000432	-0.019758	-0.000090	1.000000	0.036662	0.000735
weather	-0.004328	-0.033559	0.009542	0.036662	1.000000	-0.040035
traffic_volume	0.018676	0.130034	0.004714	0.000735	-0.040035	1.000000

```
sns.heatmap(cor)
```

```
<AxesSubplot:>
```



- Correlation strength varies based on colour, lighter the colour between two variables, more the strength between the variables, darker the colour displays the weaker correlation
- We can see the correlation scale values on the left side of the above image

2. Pair Plot: Plot pairwise relationships in a dataset.

A pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or making a linear separation in our data-set.

- By default, this function will create a grid of Axes such that each numeric variable in data will be shared across the y-axes across a single row and the x-axes across a single column. The diagonal

plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.

- We implement this using the below code.

Code:- sns.pairplot(data)

The output is as shown below-

Pair plot usually gives pairwise relationships of the columns in the dataset

From the above pair plot, we infer that

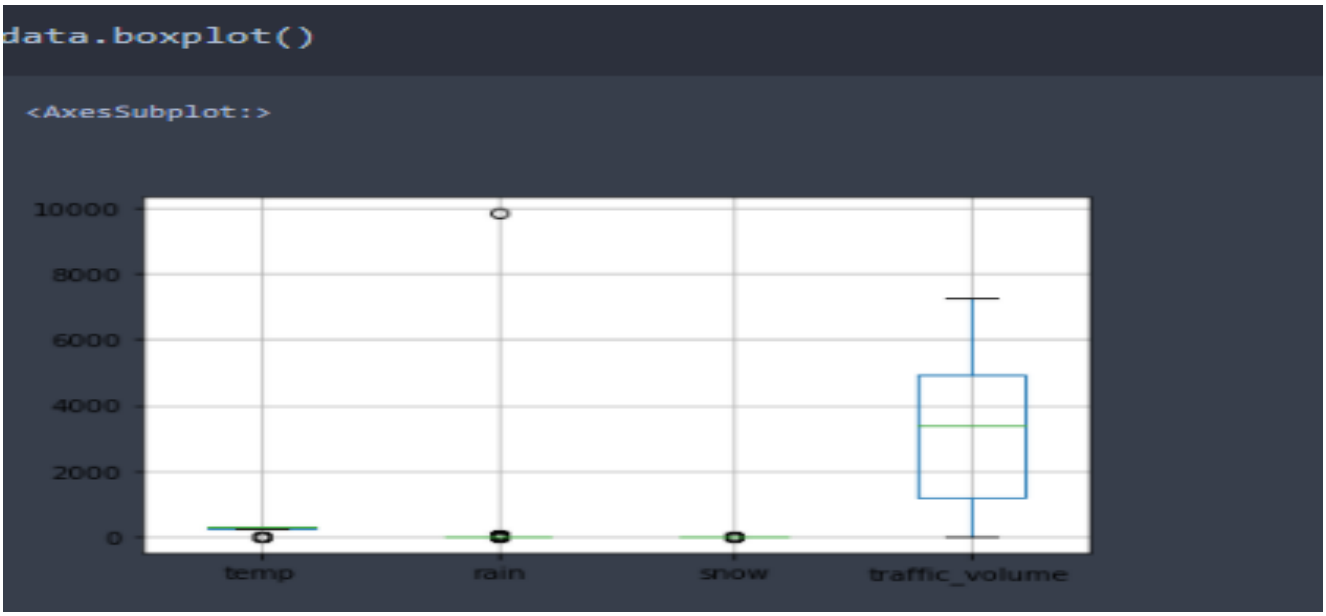
1.From the above plot we can draw inferences such as linearity and strength between the variables. how features are correlated(positive, neutral and negative)

3. Box Plot:

Box-plot is a type of chart often used in explanatory data analysis. Box plots visually show the distribution of numerical data and skewness through displaying the data quartiles (or percentiles) and averages.

Box plots are useful as they show the average score of a data set. The median is the average value from a set of data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less.

jupyter has a built-in function to create a boxplot called boxplot(). A boxplot plot is a type of plot that shows the spread of data in all the quartiles.



From the above box plot, we infer how the data points are spread and the existence of the outliers

4. Data and time columns need to be split into columns so that analysis and training of the model can be done in an easy way, so we use the split function to convert date into the year, month and day. time column into hours, minutes and seconds.

```
# spliiting the date column into year,month,day
data[["day", "month", "year"]] = data["date"].str.split("-", expand = True)

# spliiting the date column into year,month,day
data[["hours", "minutes", "seconds"]] = data["Time"].str.split(":", expand = True)

data.drop(columns=['date','Time'],axis=1,inplace=True)

data.head()
```

	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds
0	7	288.28	0.0	0.0	1	5545	02	10	2012	09	00	00
1	7	289.36	0.0	0.0	1	4516	02	10	2012	10	00	00
2	7	289.58	0.0	0.0	1	4767	02	10	2012	11	00	00
3	7	290.13	0.0	0.0	1	5026	02	10	2012	12	00	00
4	7	291.14	0.0	0.0	1	4918	02	10	2012	13	00	00

➢ Splitting the Dataset into Dependent And Independent Variable:

· In machine learning, the concept of the dependent variable (y) and independent variables(x) is important to understand. Here, the Dependent variable is nothing but output in dataset and the independent variable is all inputs in the dataset.

· With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

To read the columns, we will use iloc of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

Let’s split our dataset into independent and dependent variables.

y = data[traffic_volume] - independent

x = data.drop(traffic_volume,axis=1)

```
y = data['traffic_volume']

x = data.drop(columns=['traffic_volume'],axis=1)
```

➢ Feature Scaling:

There is a huge disparity between the x values so let us use feature scaling.

Feature scaling is a method used to normalize the range of independent variables or features of data.

```
y = data['traffic_volume']
x = data.drop(columns=['traffic_volume'],axis=1)

names = x.columns

from sklearn.preprocessing import scale

x = scale(x)

x   = pd.DataFrame(x,columns=names)

x.head()
```

	holiday	temp	rain	snow	weather	day	month	year	hours	minutes	seconds
0	0.015856	0.530485	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.345548	0.0	0.0
1	0.015856	0.611467	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.201459	0.0	0.0
2	0.015856	0.627964	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.057371	0.0	0.0
3	0.015856	0.669205	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	0.086718	0.0	0.0
4	0.015856	0.744939	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	0.230807	0.0	0.0

- After scaling the data will be converted into an array form
- Loading the feature names before scaling and converting them back to data frame after standard scaling is applied

➢ Splitting The Data Into Train And Test:

When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test datasets. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.
- In general you can allocate 80% of the dataset to the training set and the remaining 20% to test.
- Now split our dataset into train set and test using train_test_split class from sci-kit learn library.

from sklearn import model_selection
x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.2,random_state=0)