```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'traffic-prediction-dataset:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F3860551%2F6696586%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3D

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading traffic-prediction-dataset, 28753 bytes compressed
[==================================================] 28753 bytes downloaded
Downloaded and uncompressed: traffic-prediction-dataset
Data source import complete.
```

# Traffic Prediction

## INDEX

- Initialisation & import
- Visualisation
- From scratch function
- Train
- Test
- Result

## Init Dataset & Import modules

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/traffic-prediction-dataset/Traffic.csv
```

```
df = pd.read_csv("/kaggle/input/traffic-prediction-dataset/Traffic.csv")
```

```
df
```

| | Time | Date | Day of the week | CarCount | BikeCount | BusCount | TruckCount | Total | Traffic Situation |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 12:00:00 AM | 10 | Tuesday | 31 | 0 | 4 | 4 | 39 | low |
| **1** | 12:15:00 AM | 10 | Tuesday | 49 | 0 | 3 | 3 | 55 | low |
| **2** | 12:30:00 AM | 10 | Tuesday | 46 | 0 | 3 | 6 | 55 | low |
| **3** | 12:45:00 AM | 10 | Tuesday | 51 | 0 | 2 | 5 | 58 | low |
| **4** | 1:00:00 AM | 10 | Tuesday | 57 | 6 | 15 | 16 | 94 | normal |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2971** | 10:45:00 PM | 9 | Thursday | 16 | 3 | 1 | 36 | 56 | normal |
| **2972** | 11:00:00 PM | 9 | Thursday | 11 | 0 | 1 | 30 | 42 | normal |

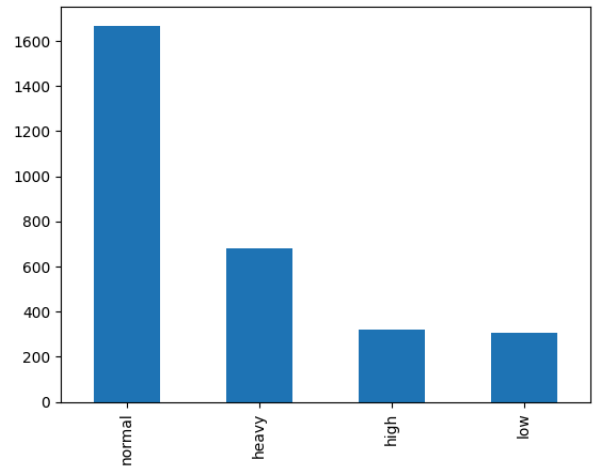
Next steps: Generate code with df View recommended plots

```
import random
import matplotlib.pyplot as plt
from tqdm import tqdm
import math
```

## VISUALISATION

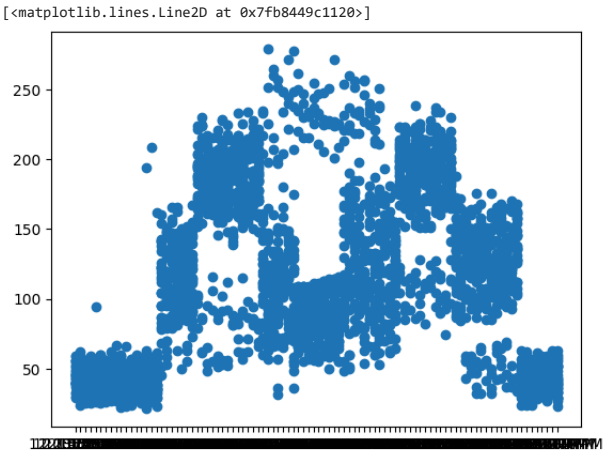
```
df['Traffic Situation'].value_counts().plot.bar()
```

```
<Axes: >
```

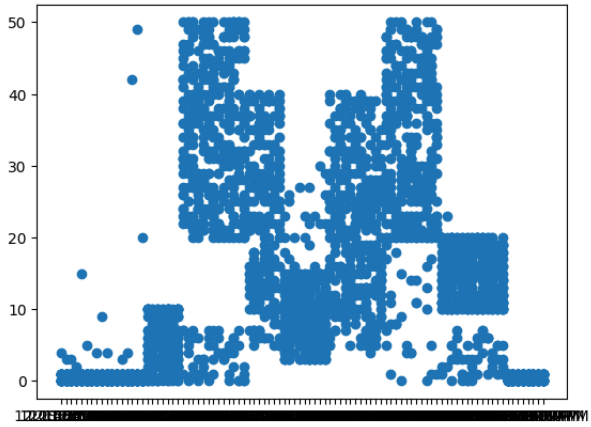


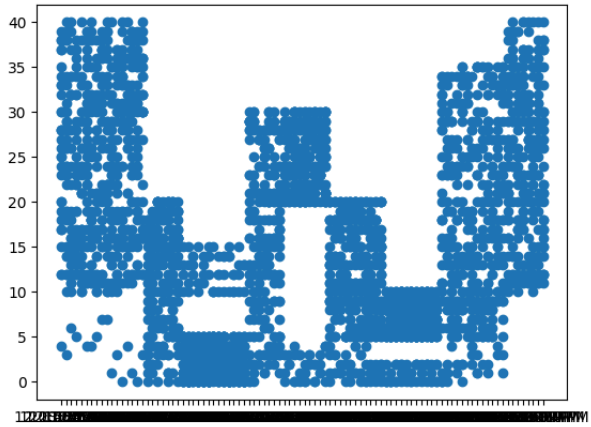

```
plt.plot_date(df['Time'], df['Total'])
```

```
[<matplotlib.lines.Line2D at 0x7fb8449c1120>]
```



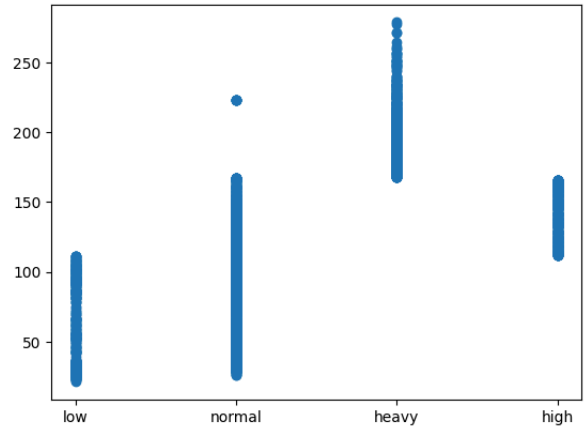

```
plt.plot_date(df['Time'], df['CarCount'])
plt.show()
plt.plot_date(df['Time'], df['BikeCount'])
plt.show()
plt.plot_date(df['Time'], df['BusCount'])
plt.show()
plt.plot_date(df['Time'], df['TruckCount'])
plt.show()
```

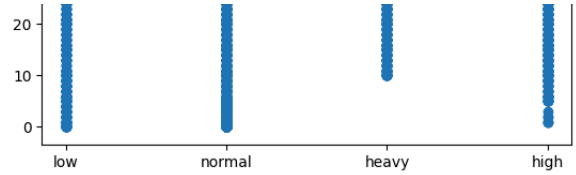



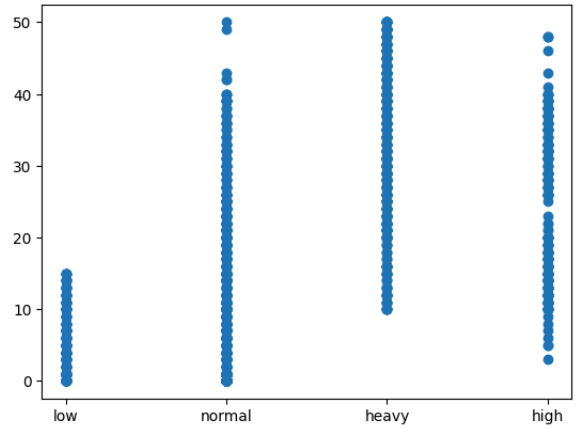

```
plt.scatter(df['Traffic Situation'], df['Total'])
```

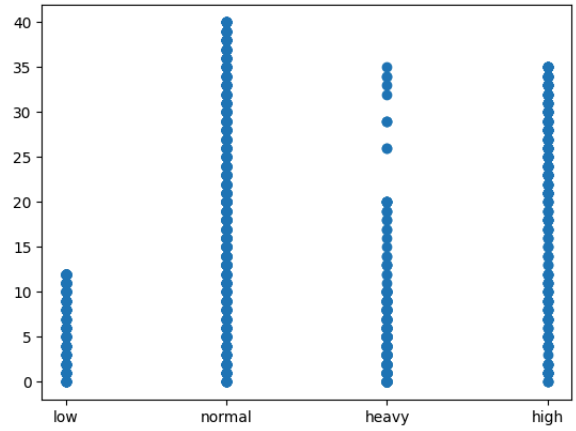
```
<matplotlib.collections.PathCollection at 0x7fb844668dc0>
```




```
plt.scatter(df['Traffic Situation'], df['CarCount'])
plt.show()
plt.scatter(df['Traffic Situation'], df['BikeCount'])
plt.show()
plt.scatter(df['Traffic Situation'], df['BusCount'])
plt.show()
plt.scatter(df['Traffic Situation'], df['TruckCount'])
plt.show()
```




## FROM SCRATCH TRAINING FUNCTION

```
def sigmoid(x: float) -> float:
        return 1 / (1 + math.exp(-x))


def predict(input: list,w: list) -> float:
        output: float = 0
        for i in range(len(input)):
            output += w[i] * input[i]
        return sigmoid(output)
```