PM 2.5 Emissions of various transportation modes in England of the early semi-decade of 2010.

STUDENT ID- 21025221

DATE- MAY 12TH/2023. WORD COUNT-3,165.

CC7184 Data Mining and Machine Learning.

Table of Contents

PM 2.5 Emissions of various transportation modes in England of the early semi-decade of 2010.

Samrat Rai
Student ID:21025221

Background

Particulate Matter is tiny particles or droplets in the air that are invisible to the naked eye. Particulate matter is classified into two categories i.e., PM2.5 and PM10. In this report, only the issues attributed to 2.5 will be discussed.

PM2.5 are particulate matters that have a width of fewer than 2.5 microns. Most common sources of Global PM2.5 to construction, energy production, and transportation (Meng et al., 2016; Tessum et al., 2022,). However, PM2.5 is also emitted through household activities such as cooking, smoking, burning candles, oils, etc (Massey et al., 2012).

PM2.5 is known to cause significant health impacts in occupants leaving in the vicinity of high concentrations. These effects include slow lung growth in children, heart diseases, lung cancer, and other various health issues (Yang et al., 2020; Huang et al., 2017; Alexeeff, 2021). In the United Kingdom, every year 28,000 to 36,000 mortalities are attributed to air pollution (GOV.UK, 2021).

The concentration of PM2.5 is one of the major variables that define the quality of air in any given area (Sahu and Kota, 2017). In the previous paragraphs, the impacts of PM2.5 in the health of the general population were established. Given these scenarios, monitoring, and observation of PM2.5 data are detrimental to public health.

To tackle the associated problems of air pollution, it is essential to identify the sources of emissions and emissions points. This allows policymakers and relevant authorities to observe the sectors and areas with high emissions and help them formulate project plans, policies, and legislations to mitigate or eliminate the emissions caused by specific sectors or emission points.

To identify the problem of PM2.5 emissions, various statistical modeling have been used by different authors across the globe. In one such study, the author nonlinear regression model to estimate single source concentrations of primarily and secondarily formed PM2.5 (Baker and Foley, 2011).

In another study, Linear Regression Model was used to predict short-period PM2.5 emissions in Taiwan (Zhao et al., 2018). Such data could help predict the air quality and can pre-warn authorities regarding various emission points which would allow them to apply mitigating measures.

Different Machine learning approaches have also been used to forecast the PM2.5 concentration in various geographical points across the globe. In one such study, Multiple Additive Regression trees to forecast PM2.5 concentration at different time intervals (Karimian et al., 2019). Similarly, Neural network model was also used to for hourly prediction of outdoor PM2.5 concentrations by using historical data sets (Liu et al., 2020). The emission of PM2.5 is attributed to various economic activities across every industry in the UK. So, far relevant authorities seem to have done good work in categorizing the sectors of emissions. Thus, integrating machine learning models with historical data sets gathered by various agencies regarding emission sectors could be used to understand emission patterns at the micro level and allow policymakers and other authorities to better implement the mitigating measures and policies.

Objectives of the project

- Summary of various transportation modes causing emissions in England.
- Evaluation of pm 2.5 emission.
- Finding out which transportation mode has the highest and the lowest pm 2.5 emission.
- Evaluation of pm 2.5 emission of public transportation system and private transportation sectors.
- Ways to reduce pm 2.5 emission in the transportation sector of England.

Description of Data

- Source: London Datastore, 2022. https://data.london.gov.uk/dataset/london-atmospheric-emissions-inventory--laei--2016
- Size: 100651 data points, 14 numerical variables and 4 categorical variables
- Type of data: an Excel spreadsheet containing data.
- Date Accessed: 22/02/2022.
- Years Of Data Collection: 2010,2013 & 2016.
- Information Contained: PM 2.5 emissions of various modes of transportation in UK like TfL buses, taxi, highway goods vehicles, petrol &diesel cars etc. known as minor roads emission in the dataset.

Potential machine learning methods

- Linear regression: a supervised machine learning model used to find the best fit line between independent and dependant variable. Independent variable is the variable used to predict another variables value and the dependant variable is the value we want to predict.it can be used to determine the character of dependant and independent variables, trend forecasting and predictions of events.

- K-means clustering: one of the most common unsupervised machine learning methods where all data points are independent variables. A centre point which is known as the centroid is present in each cluster which represents the mean of the cluster. Random data points are created from the dataset and clusters are formed with centroids. K-means can be used to cluster the different types of data points in a data set, data points like different types of industries like agriculture, minerals, electricity production sources etc can be grouped into clusters.

Change of dataset:

- I have changed my dataset as I was advised by professor to have at least 500 datapoints with at least 5 variables whereas, in the previous dataset I had submitted for the coursework proposal had contained less than 300 data points with 20 numerical variables and 14 categorical variables. I have found a new data set with 100651 data points, 14 numerical variables and 4 categorical variables and the topic remains the same being PM 2.5 emission.

Methodology:

Data preparation:

Data cleaning:

```
In [12]: import pandas as pd
         df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
         df
```

Out[12]:

	Grid_Id	Grid_Id_ExactCut	Location_ExactCut	Borough_ExactCut	Year	Pollutant	Emissions	Emissions Unit	Motorcycle	Taxi	...	Diesel Car	Elect
0	5910	1	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.789607	0.728706	...	28.659199	0.000(
1	5911	2	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.518391	0.478272	...	18.816494	0.000(
2	5912	3	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.795632	0.734195	...	28.882810	0.000(
3	5915	4	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.787536	0.726727	...	28.564638	0.000(
4	5916	5	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.710495	0.655763	...	25.790995	0.000(
...	...	...	...	...	...	...	...	...	...	...	...	...	
100645	10059	3351	Inner	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000000	0.000000	...	0.000000	0.000(
100646	10059	3352	Central	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000507	0.002053	...	0.003905	0.000(
100647	9714	3353	Central	Camden	2016	PM25_Tyre	COPERT	tonne/year	0.000745	0.002983	...	0.005739	0.000(
100648	9716	3354	Central	Islington	2016	PM25_Tyre	COPERT	tonne/year	0.000051	0.000200	...	0.000392	0.000(
100649	9716	3355	Central	City	2016	PM25_Tyre	COPERT	tonne/year	0.000623	0.002459	...	0.004804	0.000(

100650 rows × 21 columns

Figure 1

The above figure 1, is the first step of data cleaning process where we must load the dataset into a software. In this case, Jupyter notebook is the software used and the format of the dataset is in excel. Imported the library called pandas as it is a library used especially for

data cleaning and makes it possible to read files of different formats like CSV, JSON, Excel etc.

```
In [13]: df.drop(columns=['Grid_Id','Grid_Id_ExactCut'],axis=1 ,inplace=True)
         df
```

Out[13]:

	Location_ExactCut	Borough_ExactCut	Year	Pollutant	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	D
0	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.789607	0.728706	63.290727	28.659199	0.000000	0.526430	10.91
1	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.518391	0.478272	41.554182	18.816494	0.000000	0.345634	7.16
2	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.795632	0.734195	63.784549	28.882810	0.000000	0.530545	11.00
3	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.787536	0.726727	63.081901	28.564638	0.000000	0.524680	10.88
4	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.710495	0.655763	56.956611	25.790995	0.000000	0.473756	9.82
...	...	...	...	...	...	...	...	...	...	...	...	...	...
100645	Inner	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
100646	Central	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.00
100647	Central	Camden	2016	PM25_Tyre	COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.00
100648	Central	Islington	2016	PM25_Tyre	COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.00
100649	Central	City	2016	PM25_Tyre	COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.00

100650 rows × 19 columns

Figure 2

The second step of data cleaning where columns that were not used or of any numerical and categorical value to the project is dropped. The function, drop () is used to drop a column.

```
In [14]: df=df.rename(columns={'Total': 'Total PM 2.5 Emission'})
         df
```

Out[14]:

missions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Electric LGV	TfL Bus	Non-TfL Bus and Coach	Rigid HGV	Artic HGV	Total PM 2.5 Emission
DFT	tonne/year	0.789607	0.728706	63.290727	28.659199	0.000000	0.526430	10.919852	0.000000	1.489118	2.730038	6.708603	1.046861	116.889143
DFT	tonne/year	0.518391	0.478272	41.554182	18.816494	0.000000	0.345634	7.169557	0.000000	0.977660	1.792369	4.403879	0.686558	76.742997
DFT	tonne/year	0.795632	0.734195	63.784549	28.882810	0.000000	0.530545	11.005204	0.000000	1.500924	2.751682	6.761137	1.055737	117.802416
DFT	tonne/year	0.787536	0.726727	63.081901	28.564638	0.000000	0.524680	10.883539	0.000000	1.484490	2.721552	6.686387	1.039584	116.501034
DFT	tonne/year	0.710495	0.655763	56.956611	25.790995	0.000000	0.473756	9.827226	0.000000	1.338921	2.454678	6.037218	0.940247	105.185912
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000001	0.000000	0.000000	0.000000	0.000001
COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.003240	0.000017	0.000099	0.000338	0.001254	0.000301	0.016392
COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.004898	0.000026	0.001292	0.000000	0.001872	0.000449	0.024883
COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.000331	0.000002	0.000416	0.000000	0.000130	0.000031	0.002022
COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.004627	0.000025	0.000060	0.000533	0.001706	0.000409	0.021011

Figure 3

In fig 3, renaming columns with incorrect names has been done. In this case only one column was not correctly named, where the columns name was total and now has been changed to total PM 2.5 emissions.

```
In [4]: df.columns

Out[4]: Index(['Location_ExactCut', 'Borough_ExactCut', 'Year', 'Pollutant',
               'Emissions', 'Emissions Unit', 'Motorcycle', 'Taxi', 'Petrol Car',
               'Diesel Car', 'Electric Car', 'Petrol LGV', 'Diesel LGV',
               'Electric LGV', 'TfL Bus', 'Non-TfL Bus and Coach', 'Rigid HGV',
               'Artic HGV', 'Total PM 2.5 Emission'],
              dtype='object')
```

Figure 4

In fig 4, to check if all the columns have corrected names and if the name is correctly changed from total-to-total PM 2.5 emission.

```
In [5]: data_with_index = df.set_index("Pollutant")
        data_with_index.head()
```

Out[5]:

Pollutant	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Elec L
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.789607	0.728706	63.290727	28.659199	0.0	0.526430	10.919852	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.518391	0.478272	41.554182	18.816494	0.0	0.345634	7.169557	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.795632	0.734195	63.784549	28.882810	0.0	0.530545	11.005204	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.787536	0.726727	63.081901	28.564638	0.0	0.524680	10.883539	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.710495	0.655763	56.956611	25.790995	0.0	0.473756	9.827226	

Figure 5

In fig 5, set index is used to locate a specific column in a dataset and indexes it in the front of the dataset, where in the above figure is Pollutant.

```
In [6]: data_with_index = data_with_index.drop(["CO2", "PM10_Tyre", "NOx","PM10_Brake","PM10_Exhaust","PM10_Resusp","PM25_Re
        data_with_index
```

Out[6]:

Pollutant	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	El
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005894	0.002905	0.000000	0.000064	0.001482	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000042	0.000039	0.003870	0.001907	0.000000	0.000042	0.000973	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000065	0.000059	0.005940	0.002928	0.000000	0.000064	0.001493	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005875	0.002896	0.000000	0.000063	0.001477	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000058	0.000053	0.005305	0.002614	0.000000	0.000057	0.001333	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	
PM25_Tyre	Inner	Southwark	2016	COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
PM25_Tyre	Central	Southwark	2016	COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.003240	0.0
PM25_Tyre	Central	Camden	2016	COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.004898	0.0
PM25_Tyre	Central	Islington	2016	COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.000331	0.0
PM25_Tyre	Central	City	2016	COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.004627	0.0

30195 rows × 18 columns

```
In [7]: new_df=data_with_index
```

Figure 6

In fig 6, after assigning the variable name called data with index to the Pollutant column, I now can use the variable data with index to drop the rows inside the Pollutant column. In the research project, only PM 2.5 emission is going to be used so all the other types of pollutant types like CO2, NOx and PM 10 are dropped. After the rows have been dropped, data with index is converted into our new data frame by assigning it to a new variable called new df. This is done so it is easier to recall the new transformed data frame with less code.

```
In [19]: new_df.info()
<class 'pandas.core.frame.DataFrame'>
Index: 30195 entries, PM25_Brake to PM25_Tyre
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Location_ExactCut      30195 non-null  object
 1   Borough_ExactCut       30195 non-null  object
 2   Year                   30195 non-null  int64
 3   Emissions              30195 non-null  object
 4   Emissions Unit         30195 non-null  object
 5   Motorcycle             30195 non-null  float64
 6   Taxi                   30195 non-null  float64
 7   Petrol Car             30195 non-null  float64
 8   Diesel Car             30195 non-null  float64
 9   Electric Car           30195 non-null  float64
 10  Petrol LGV             30195 non-null  float64
 11  Diesel LGV             30195 non-null  float64
 12  Electric LGV           30195 non-null  float64
 13  TfL Bus                30195 non-null  float64
 14  Non-TfL Bus and Coach  30195 non-null  float64
 15  Rigid HGV              30195 non-null  float64
 16  Artic HGV              30195 non-null  float64
 17  Total PM 2.5 Emission  30195 non-null  float64
dtypes: float64(13), int64(1), object(4)
memory usage: 4.4+ MB
```

Figure 7

In fig 7, the dot info (), is used to obtain the meta data information of the dataset. This is the final up to date transformed data so far where the original raw dataset had 100650 rows with 21 columns whereas now the new transformed dataset has 30195 data entry rows with 18 columns.

```
In [21]: #sums up total null values of columns
         new_df.isna().sum()

Out[21]: Location_ExactCut          0
         Borough_ExactCut           0
         Year                       0
         Emissions                  0
         Emissions Unit             0
         Motorcycle                 0
         Taxi                       0
         Petrol Car                 0
         Diesel Car                 0
         Electric Car               0
         Petrol LGV                 0
         Diesel LGV                 0
         Electric LGV               0
         TfL Bus                    0
         Non-TfL Bus and Coach      0
         Rigid HGV                  0
         Artic HGV                  0
         Total PM 2.5 Emission      0
         dtype: int64
```

Figure 8

In fig 8, we check if our dataset still needs further cleaning by checking for any null values. In the figure above as you can see there seems be no null values so we don't need to remove any null values which could be removed by inputting dropna().

```
In [10]: new_df.head()
```

Out[10]:

Pollutant	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Elec
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005894	0.002905	0.0	0.000064	0.001482	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000042	0.000039	0.003870	0.001907	0.0	0.000042	0.000973	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000065	0.000059	0.005940	0.002928	0.0	0.000064	0.001493	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005875	0.002896	0.0	0.000063	0.001477	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000058	0.000053	0.005305	0.002614	0.0	0.000057	0.001333	

Figure 9

Fig 9, to check the top 5 values of the transformed data frame.

```
In [11]: new_df.tail()
```

Out[11]:

Pollutant	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Elec
PM25_Tyre	Inner	Southwark	2016	COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
PM25_Tyre	Central	Southwark	2016	COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.003240	0.000
PM25_Tyre	Central	Camden	2016	COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.004898	0.000
PM25_Tyre	Central	Islington	2016	COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.000331	0.000
PM25_Tyre	Central	City	2016	COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.004627	0.000

Figure 10

Fig 10, to check the last five rows of the transformed data frame.

```
In [23]: new_df.describe()
```

Out[23]:

	Year	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Electric LGV	TfL Bus	Non-Tf and C
count	30195.00000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	3.019500e+04	30195.0
mean	2013.00000	0.000198	0.000350	0.006850	0.006656	0.000007	0.000050	0.003353	0.000002	6.728246e-04	0.0
std	2.44953	0.000312	0.001132	0.009498	0.007048	0.000017	0.000085	0.003779	0.000004	2.479002e-03	0.0
min	2010.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.0
25%	2010.00000	0.000024	0.000026	0.000641	0.001120	0.000000	0.000003	0.000535	0.000000	2.578211e-08	0.0
50%	2013.00000	0.000092	0.000105	0.002628	0.004194	0.000000	0.000015	0.002013	0.000000	4.584893e-05	0.0
75%	2016.00000	0.000265	0.000315	0.009419	0.009964	0.000005	0.000058	0.004875	0.000001	3.663209e-04	0.0
max	2016.00000	0.006475	0.036213	0.076354	0.051548	0.000238	0.000831	0.032946	0.000095	8.498240e-02	0.0

Figure 11

Lastly fig 11 shows the mathematical computations of all columns like mean, median, mode, standard deviation, count, min, max, lower quartile, average quartile, and upper quartile.

Unsupervised learning method:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
df=pd.read_excel('minor roads emission 2010,13&16..xlsx')

#dropping columns for numerical value specification.
df.drop(columns=['Borough_ExactCut','Emissions','Emissions Unit','Pollutant','Location_ExactCut'],axis=1 ,inplace=Tr

#used for standardizing data.
scaler=StandardScaler()

#Data standardization.
df[['Motor-cycle','Cab','Diesel-Car','Petrol-Car','Petrol-LGV','Diesel-LGV','TfL-Bus','Non-TfL Bus & Coach','Rigid-H
```

```
#used for standardizing data.
scaler=StandardScaler()

issions','Electric-Car','Electric-LGV']]=scaler.fit_transform(df[['Motorcycle','Taxi','Diesel Car','Petrol Car','Petr
```

Figure 12

In the above figure panda's library has been used as it is one of the major libraries used in reading data files such as excel and is excellent for analysing, manipulating, transforming datasets. Then imported pandas as pd to assign it for further use. The second library used is Sklearn which is better known as scikit-learn, this library is used for statistical modelling and machine learning. For this project Sklearn is used for clustering to specifically graph k means hence using Sklearn cluster to import k means for further plotting. Matplotlib is a fantastic library used for visualization and pyplot is used for easier plotting format options like font style, control line style, formatting axes etc. and the package from scikit-learn called pre-processing has various utility function but in this project is used for scale numerical features

and to standardize data of the dataset, which is imported as StandardScaler. Pandas is used to read the dataset as seen in fig 12. After assigning Standardscaler as scaler, to standardize the dataset the code scaler fit transform is used to standardize the dataset. For this each column names had to be modified to new column names which were standardized. The reason for standardizing the dataset is to determine the similarity between datapoints as clustering methods like k-means use distance-based measurements.

```
#idetifying optimum number of clusters
#2
#create function to work out optimum number of clusters
def optimise_k_means(df,max_k):
    means=[]
    inertias=[]

    for k in range (1,max_k):
        kmeans=KMeans(n_clusters=k)
        kmeans.fit(df)

        means.append(k)
        inertias.append(kmeans.inertia_)

    #generate the elbow plot
    fig=plt.subplots(figsize=(10,5))
    plt.plot(means,inertias,'o-')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.grid(True)
    plt.show()
```

Figure 13

In Fig 13, Functions like Kmeans is used to optimize the number of clusters needed to plot. For loop is used to fit the number of clusters and plt is used to plot. Parameters like fit is used to fit kmeans in the dataset, parameter append is used to add an item to the end of the list, parameter plot is used to draw point or line, parameters x and y label is used to label the axis individually, parameter grid is used for the visibility of grids in the graph and lastly parameter show is used to plot the result. In this figure we are coding to plot an elbow curve which estimates the number of clusters along with inertia.

```
optimise_k_means(df[['Cab','Total PM 2.5 emissions']],10)
```

```
<IPython.core.display.Javascript object>
```

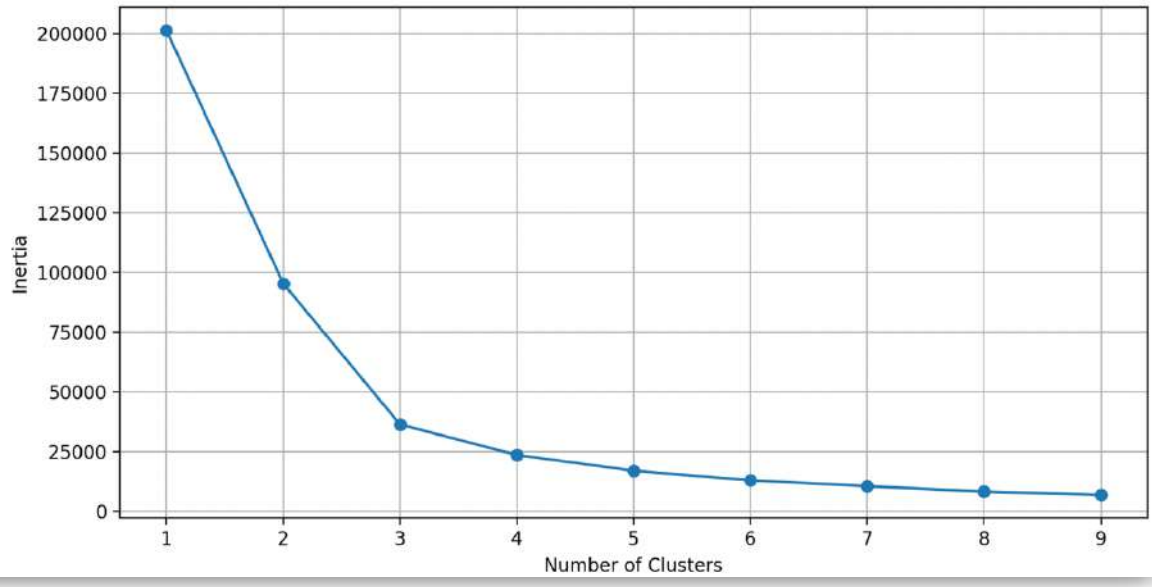


Figure 14

The function optimize k means plots an elbow curve of specific two columns chosen from the dataset which in fig 14 is cab and total PM 2.5 emissions.

```
kmeans=KMeans(n_clusters=3)
kmeans.fit(df[['Cab','Total PM 2.5 emissions']])
```

```
KMeans(n_clusters=3)
```

```
df['kmeans_3']=kmeans.labels_
```

```
centroids=kmeans.cluster_centers_
```

```
#plotting the results
%matplotlib notebook
x=df['Cab']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='purple=x(-0.097)&y(-0.1949),yellow=x(1.7166)&y(4.34638),green=x(29.6624
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(low),yellow mean(med),green mean(outli
plt.xlim(-1, 40)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Cab emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Cab PM 2.5 emission vs Total PM 2.5 emissions of all transportation modes')
plt.legend()
plt.show()
```

Figure 15

Here in fig 15, function K-Means is used again to determine the number of clusters and k means fit parameter is used to fit it into any chosen columns of the dataset. The parameter K means labels is used to colour the clusters. Parameter cluster centers is used to create the centroid points in clusters. %matplotlib notebook is used to zoom in and out of diagrams, parameter scatter is used to plot a scatter plot, xlim and ylim is used to zoom and out the

axises, label is used for labelling the x and y axis, parameter legend is used to describe any elements in the graph and show plot the result.

Supervised learning method:

```
In [21]: import matplotlib.pyplot as plt
         from scipy import stats
         import pandas as pd
         from sklearn.linear_model import LinearRegression


         %matplotlib notebook

         df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
         x=df['Artic HGV']
         y=df['Total']

         slope, intercept, r, p, std_err = stats.linregress(x, y)

         def myfunc(x):
           return slope * x + intercept

         mymodel = list(map(myfunc, x))

         plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
         plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
         plt.xlabel('PM 2.5 Artic HGV Emission')
         plt.ylabel('PM 2.5 Various Transportations of England Emission')
         plt.title('Total PM 2.5 Emission vs Artic HGV PM 2.5 Emission')
         plt.legend()
         plt.show()
```

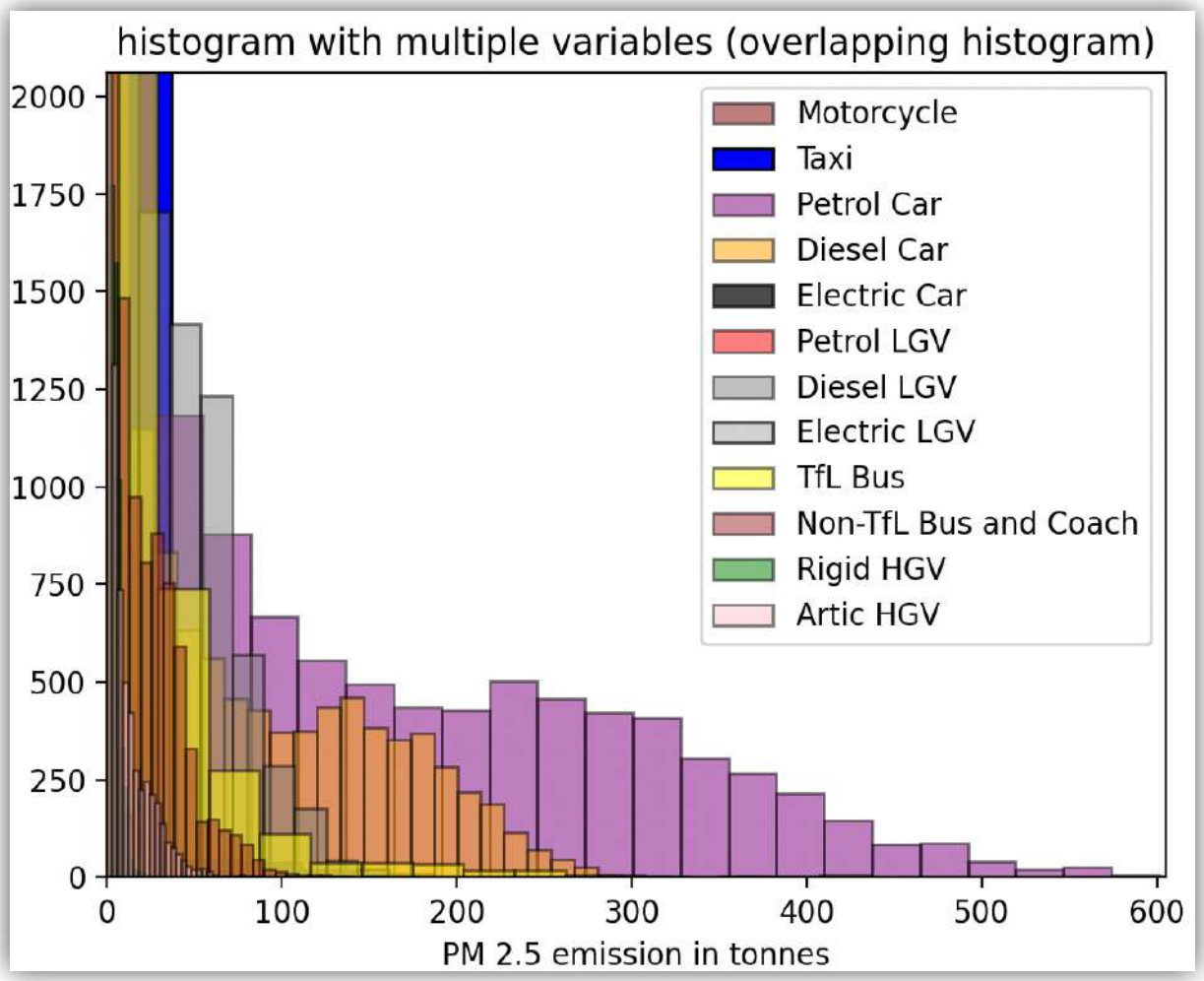
Figure 16

The library matplotlib is used for various types of visualization and the parameter pyplot is used as it offers many features that can be plotted inside a graph. Scipy library is used to solve mathematical and scientific problems, in this case it has been used to import the parameter stats for used of stats models like linear regression. Although I had coded in sklearn I did not have to use as the library of scipy was enough for me to graph linear regression. %matplotlib is used for a better graph visualization and for zooming in and out, pandas is used to read the dataset df, scatter is used to plot a scatter plot, plot is used to the line of best fit, edgecolor is used for colouring the edges of the clusters, color to colour the scatter points in the graph, my model function is used to plot linear regression, title to name the title of the graph, legend to show any elements in the graph, x and y label to label the axis's and show to plot the result.

Results and discussion:

Discuss the results of the unsupervised learning method; use figures to illustrate the results. Include the visual outputs of the methods that have been used such, clusters, dendrograms, PCA, etc.

Overview of all variables from histogram:



histogram 1

The above diagram is an overview of all variables used in and we can already see that petrol car labelled in pink graphs emit the most amount of PM 2.5 emission from 0 to a max of 600 emissions all in a span of combined 3 years data of 2010,13&16.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib notebook


df=pd.read_excel('minor roads emission 2010,13&16..xlsx')

# plotting three histograms on the same axis
plt.hist(df['Motorcycle'],bins = 5, alpha = 0.5,
         color = 'maroon',edgecolor='k')
plt.hist(df['Taxi'],bins = 5, alpha = 1,
         color = 'blue',edgecolor='k')
plt.hist(df['Petrol Car'],bins = 30, alpha = 0.5,
         color = 'purple',edgecolor='k')
plt.hist(df['Diesel Car'],bins = 30, alpha = 0.5,
         color = 'orange',edgecolor='k')
plt.hist(df['Electric Car'],bins = 1, alpha = 0.7,
         color = 'black',edgecolor='k')
plt.hist(df['Petrol LGV'],bins = 10, alpha = 0.5,
         color = 'red',edgecolor='k')
plt.hist(df['Diesel LGV'],bins = 10, alpha = 0.5,
         color = 'grey',edgecolor='k')
plt.hist(df['Electric LGV'],bins = 1, alpha = 0.7,
         color = 'silver',edgecolor='k')
plt.hist(df['TfL Bus'],bins = 30, alpha = 0.5,
         color = 'yellow',edgecolor='k')
plt.hist(df['Rigid HGV'],bins = 30, alpha = 0.5,
         color = 'brown',edgecolor='k')
plt.hist(df['Artic HGV'],bins = 15, alpha = 0.5,
         color = 'green',edgecolor='k')
plt.hist(df['Non-TfL Bus and Coach'],bins = 30, alpha = 0.5,
         color = 'pink',edgecolor='k')

plt.title("histogram with multiple \
variables (overlapping histogram)")
plt.legend(['Motorcycle','Taxi','Petrol Car','Diesel Car','Electric Car','Petrol LGV','Diesel LGV','Electric LGV','T
plt.xlabel('PM 2.5 emission in tonnes')
plt.show()
```

histogram 2

Pandas is used to read file, seaborn for visualization and matplotlib py-plot to plot.

Unsupervised learning method:

```
optimise_k_means(df[['Non-TfL Bus & Coach','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Non-TfL Bus & Coach','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```

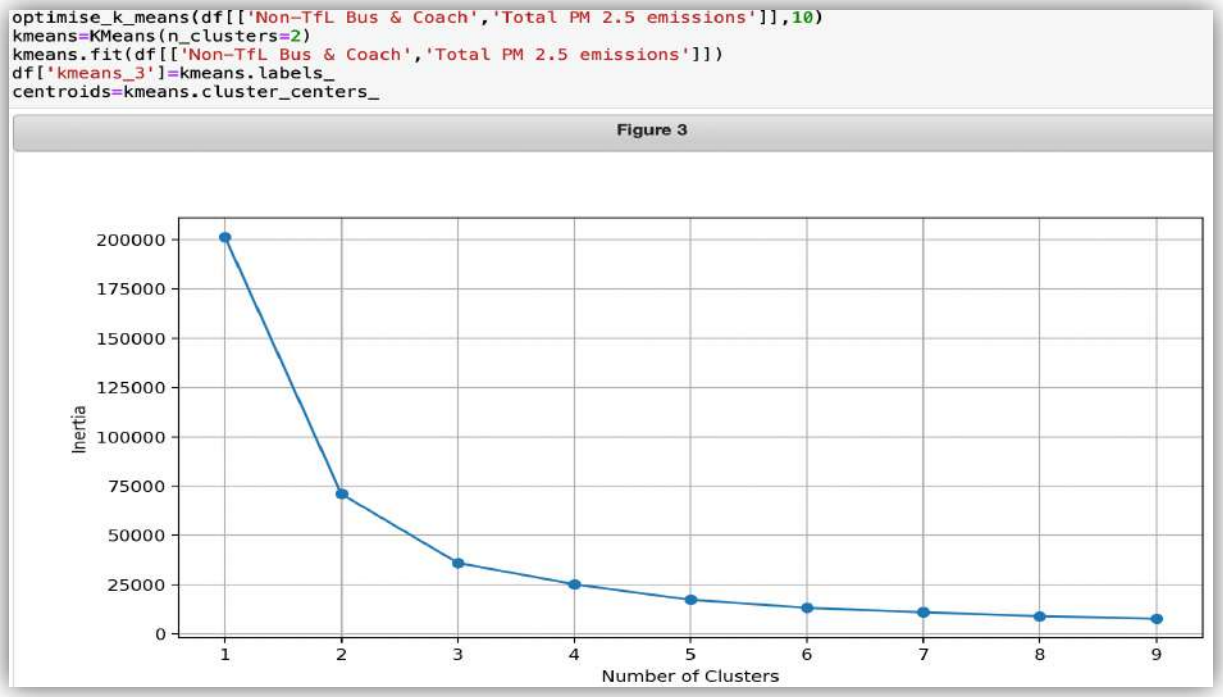


Figure 17

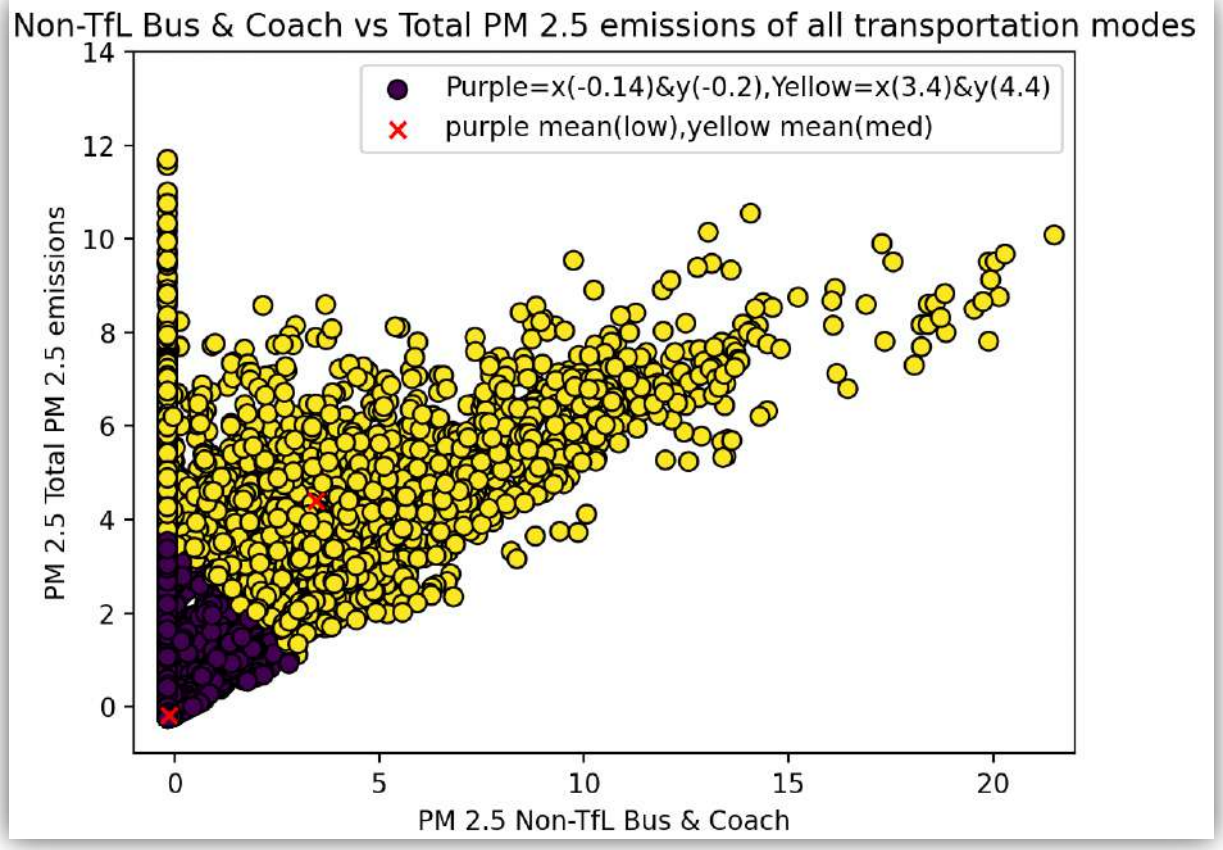


Figure 18

The elbow plot in fig 17, determines the number of clusters would be good for k-means. A good elbow plot is one with low number of clusters and low number of inertias. As number of clusters increases inertia decreases so we must find a point where it is balanced. As the elbow plot suggest we have taken 2 clusters for fig 18 and will be doing the same for the remaining clusters plotted in the figures below. The mean of purple cluster is x (-0.14) & y (-0.2), yellow cluster is x (3.4) & y (4.4). only two K-means models have been shown as example where the graph having the lowest number of centroids in the x and y axis of each cluster is the safest mode of transport in terms of PM 2.5 emission. x axis being the mode of transport and y axis being total PM 2.5 emitted by all the vehicles combined, y axis is the dependent variable used to predict which mode of transport causes the least amount of PM 2.5 emission. x axis being the dependent variable, which is various modes of transport like cab, TFL bus, electric car etc. is used to calculate the outcome.

```
optimise_k_means(df[['TfL-Bus','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=3)
kmeans.fit(df[['TfL-Bus','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```

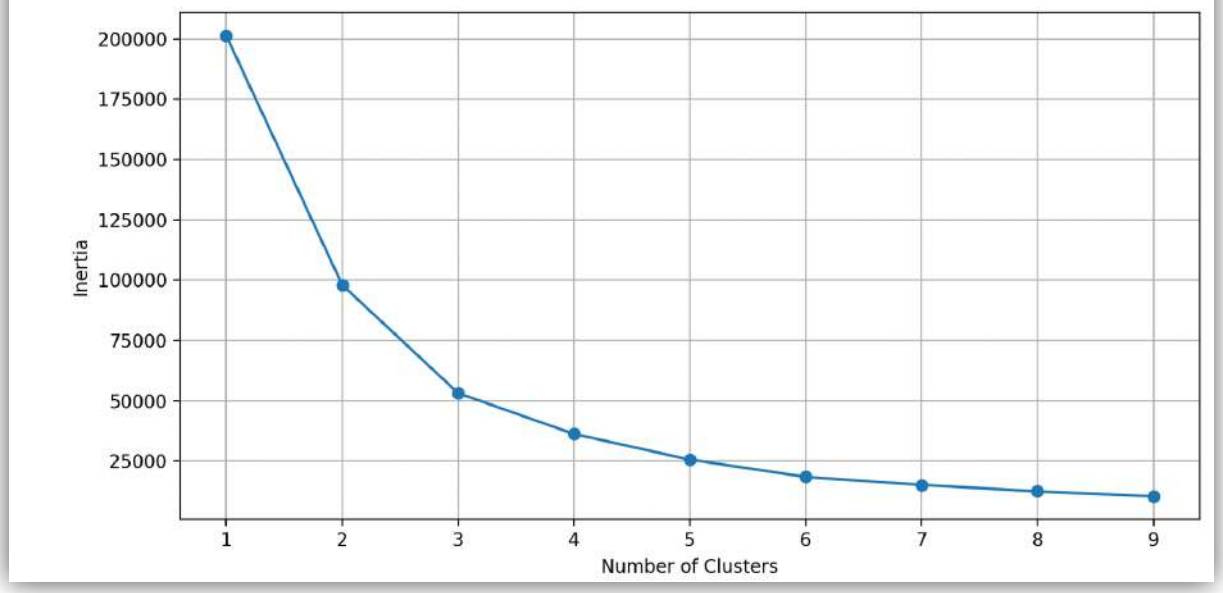


Figure 19

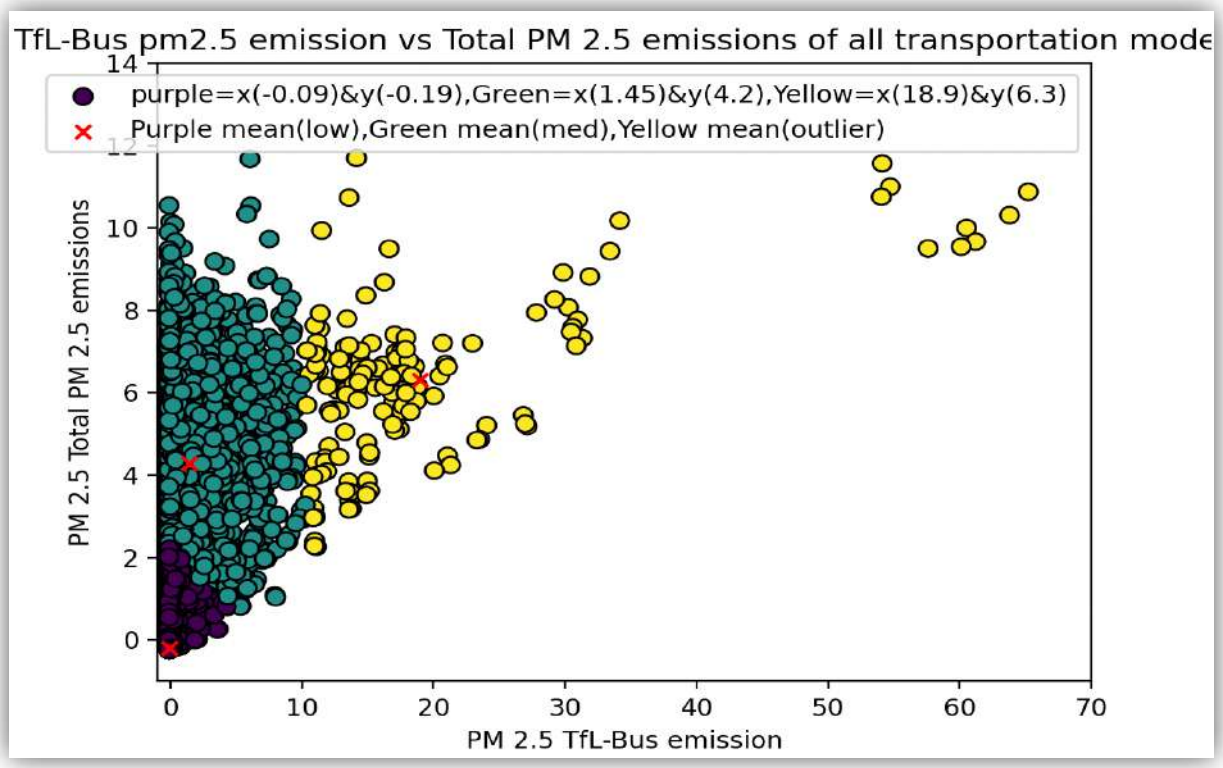


Figure 20

For TfL bus emission and total PM 2.5 emission, the mean of purple cluster is x (-0.09) &y (-0.19), the mean of green cluster is x (1.45) &y (4.2), the mean of yellow cluster is x (18.9) &y (6.3). since yellow cluster is comprised of less data points containing outliers only the mean of purple and green clusters will be considered. The best fit would have been 2 clusters according to elbow plot in fig 19 for the TFL bus emission but as an example, 3 clusters have been plotted where the yellow cluster datapoints consists mainly of outliers, hence taking 2 clusters would suit best as advised by the elbow plot.

Supervised learning:

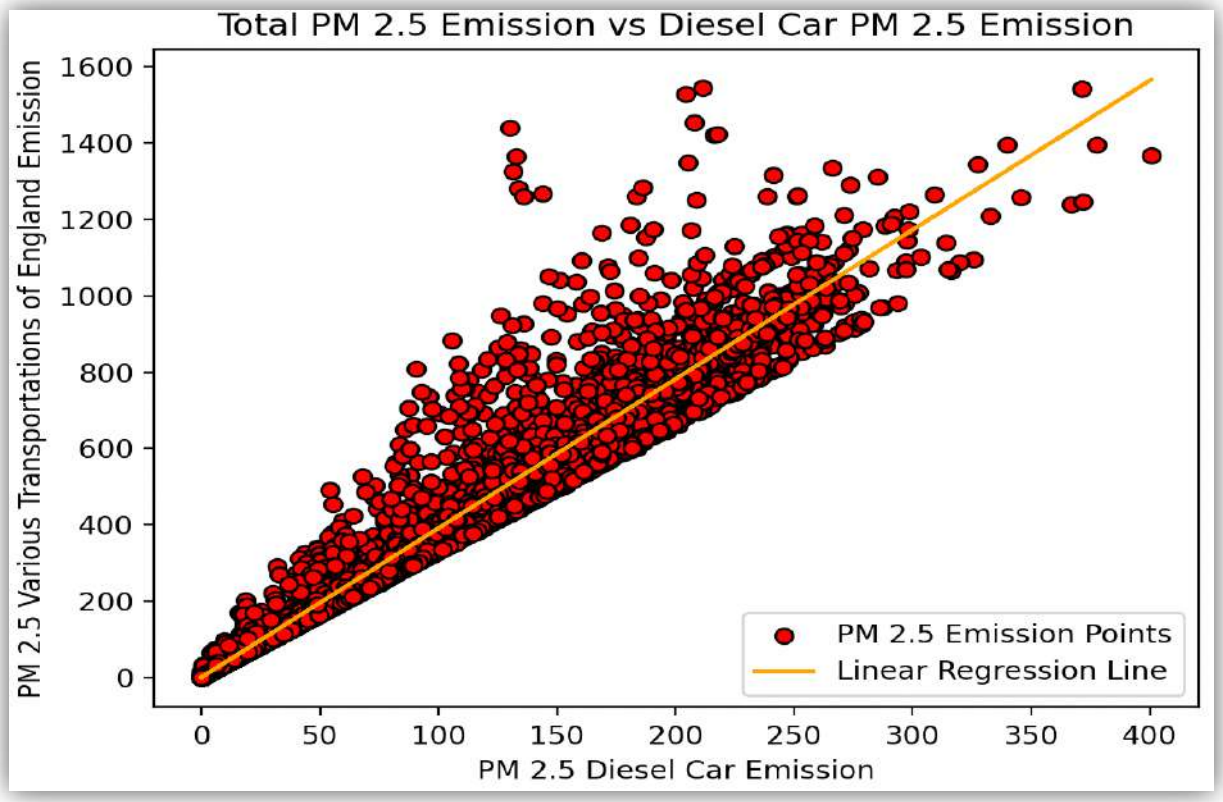


Figure 21

Diesel car Emissions ranges densely from 0 to 300 tonnes in 3 years, with 300 to 400 having data points with outliers for x axis. As total emission values rise in Y axis so does the emission values in X axis rises. The mode of transport having the lowest range of emission in x-axis in comparison to y-axis will be the safest mode of transport for PM 2.5 emissions.

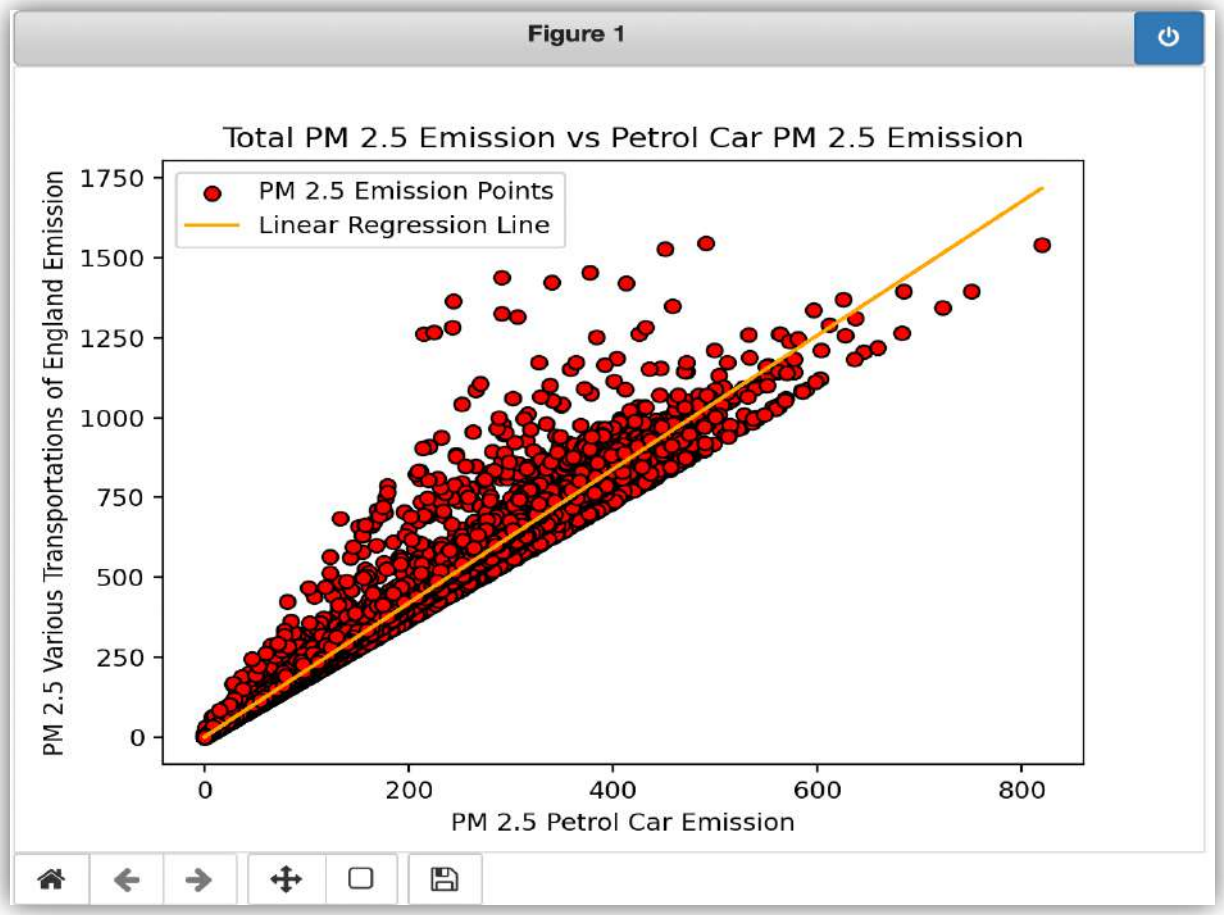


Figure 22

Petrol car emission ranges densely from 0 to 600 tonnes in 3 years. Datapoints are heavily dense below the best fit line that comprises mostly of petrol car emissions. This linear regression graph shows petrol cars emitted more PM 2.5 than diesel cars.

Model assessment:

Unsupervised learning method:

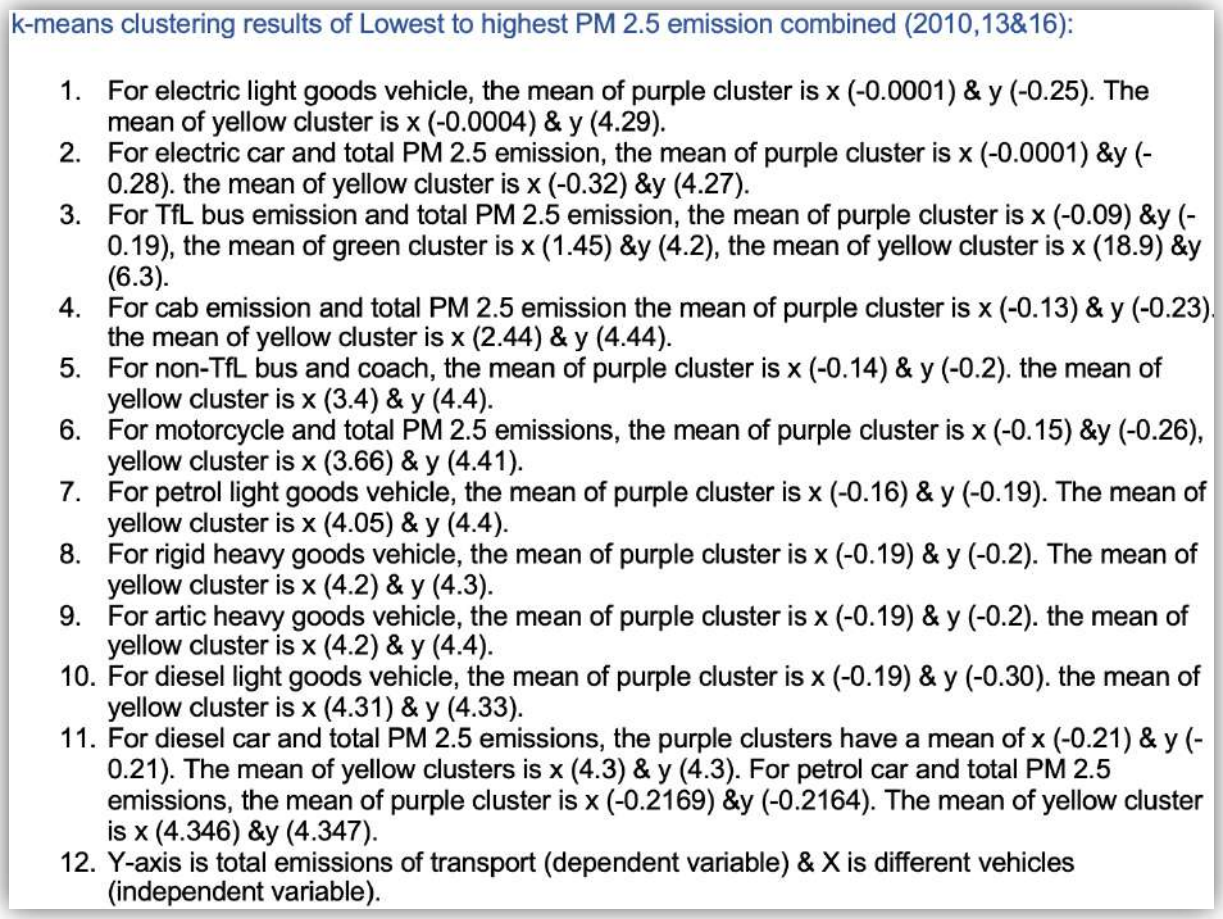


Figure 23

As we see in fig 42, electric light good vehicle and car has the top two lowest PM 2.5 emission K centroids which implies that electric modes of transport cause the least amount of PM 2.5 emission during the span of 3years, to be specific in the early semi-decade of 2010. In terms of lower mean emissions of PM 2.5, Public transport sector vehicles like TFL bus comes in 3rd, non-TFL buses and coaches in 5th and taxi comes in 4th which implies that public sector transport tend to cause lesser emission than private sector vehicles. Industrial sector vehicles come in 7th to 10th in low to high PM 2.5 emission. lastly private sector vehicles like petrol and diesel car tend to cause to most amount of PM 2.5 with an identical mean emission point. they both are responsible for emitting most of the PM 2.5 emission caused in England as their mean point in lower gradient cluster and higher gradient cluster is identical to the mean point of total PM 2.5 emission of various transport vehicles used in England. the mean point as shown in the fig42 being when y-axis which represents the emission of all transport emits 4.3 tonnes of PM 2.5 emissions in 3 years, the x-axis which

represent only petrol car also emits 4.3 tonnes of emission during the span of 3 years. The same is true for diesel cars.

Supervised learning method:

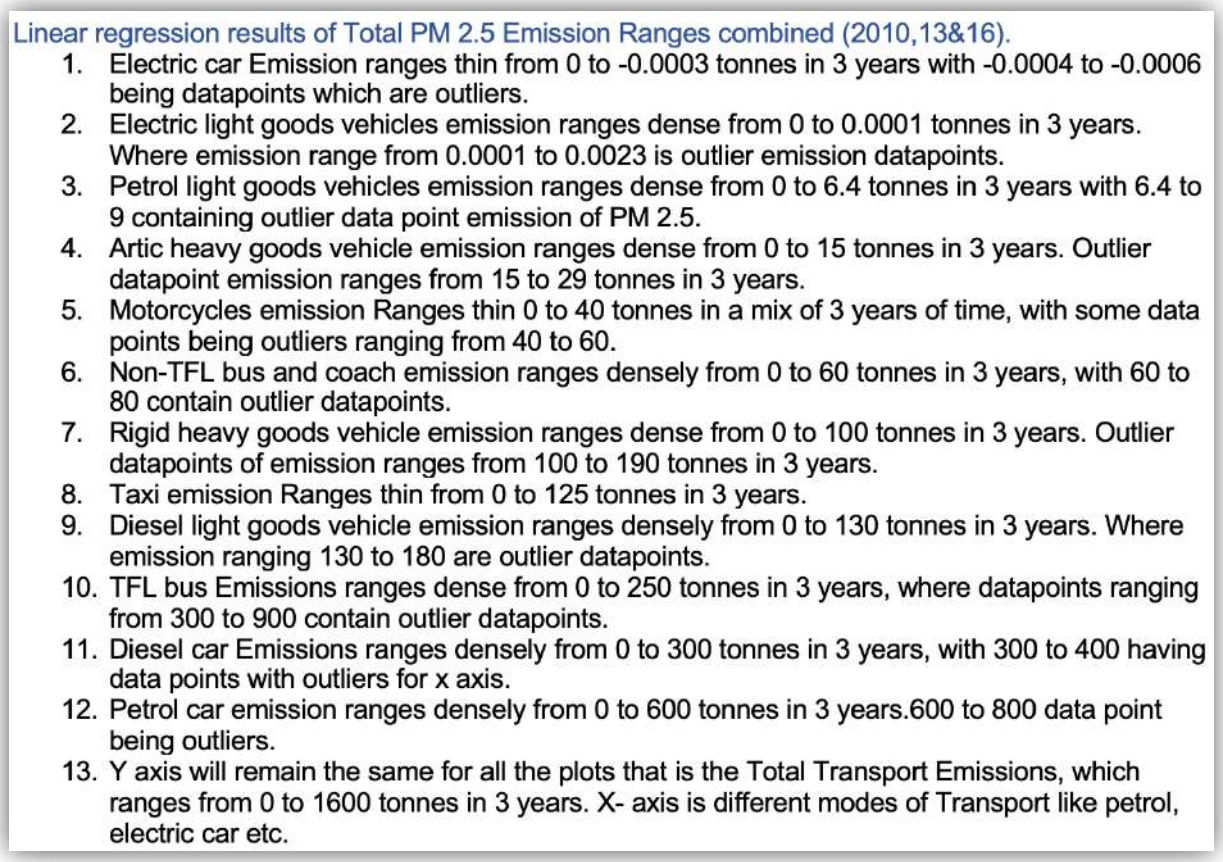


Figure 24

Linear regression estimates how much PM 2.5 emission has been emitted during a mix of 3 years. Electric car has the lowest emission rate as PM 2.5 emission in y-axis increases from 0 to 1600 tonnes, the emission rate in decreases in x-axis of electric car drop from 0 to -0.0003 tonnes. Petrol cars emits the most PM 2.5 emission. as y-axis emissions increases from 0 to 1600 tonnes, the x-axis petrol car emission increases densely from 0 to 600 tonnes which means that is the range where the datapoints are tightly compacted and hence is most accurate, whereas the datapoints ranging from 600 to 800 PM 2.5 emissions for petrol car when Y-axis emits 1190 to 1600 PM 2.5 emission serves as an outlier data points which are not accurate but can be used as a possible prediction range of petrol car emission going up to 800 PM 2.5 emissions in the future. in fig 43, private sector vehicles like petrol and diesel cause the most emission. other sectors like transportation and industrial sector cause 300 tonnes less PM2.5 emission than private sector vehicles in England.

Conclusion and perspectives:

Limitations of the project:

One of the major limitations of the study is the number of vehicles counts of each type of transport like the total number of motorcycles, taxi etc. as one of the reasoning of more or less PM 2.5 emission caused by a particular mode of transport could be linked to having more or less number of vehicles of a specific mode of transport like motorcycle or taxi. As I could only examine which mode of transport is better suited for England.

Recommendations:

Through the models used like k-means clustering and linear regression it was found out that PM 2.5 emissions are caused less by electric mode of transport in all three sectors which are industrial, private, and public transport. Hence in England, Electric vehicles should be used more for environmental safety. In terms of private and public transport, diesel cars and petrol cars emit the most PM 2.5 emission so it would be more health/eco-friendly to use public transport like TFL busses or non-TFL buses and coach services often.

Link to Jupyter Notebook project:

https://drive.google.com/drive/folders/15fyKJn_hzHP02yKVWMNVxMLQV9SpMJtP?usp=sharing

Appendix Of Code Screenshots:

Appendix for data cleaning:

```
In [12]: import pandas as pd
         df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
         df
```

Out[12]:

	Grid_Id	Grid_Id_ExactCut	Location_ExactCut	Borough_ExactCut	Year	Pollutant	Emissions	Emissions Unit	Motorcycle	Taxi	...	Diesel Car	Elect
0	5910	1	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.789607	0.728706	...	28.659199	0.000
1	5911	2	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.518391	0.478272	...	18.816494	0.000
2	5912	3	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.795632	0.734195	...	28.882810	0.000
3	5915	4	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.787536	0.726727	...	28.564638	0.000
4	5916	5	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.710495	0.655763	...	25.790995	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...	
100645	10059	3351	Inner	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000000	0.000000	...	0.000000	0.000
100646	10059	3352	Central	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000507	0.002053	...	0.003905	0.000
100647	9714	3353	Central	Camden	2016	PM25_Tyre	COPERT	tonne/year	0.000745	0.002983	...	0.005739	0.000
100648	9716	3354	Central	Islington	2016	PM25_Tyre	COPERT	tonne/year	0.000051	0.000200	...	0.000392	0.000
100649	9716	3355	Central	City	2016	PM25_Tyre	COPERT	tonne/year	0.000623	0.002459	...	0.004804	0.000

100650 rows × 21 columns

data cleaning 1

```
In [13]: df.drop(columns=['Grid_Id','Grid_Id_ExactCut'],axis=1 ,inplace=True)
         df
```

Out[13]:

	Location_ExactCut	Borough_ExactCut	Year	Pollutant	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	D
0	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.789607	0.728706	63.290727	28.659199	0.000000	0.526430	10.91
1	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.518391	0.478272	41.554182	18.816494	0.000000	0.345634	7.16
2	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.795632	0.734195	63.784549	28.882810	0.000000	0.530545	11.00
3	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.787536	0.726727	63.081901	28.564638	0.000000	0.524680	10.88
4	NonGLA	NonGLA	2010	CO2	DFT	tonne/year	0.710495	0.655763	56.956611	25.790995	0.000000	0.473756	9.82
...	...	...	...	...	...	...	...	...	...	...	...	...	
100645	Inner	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
100646	Central	Southwark	2016	PM25_Tyre	COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.00
100647	Central	Camden	2016	PM25_Tyre	COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.00
100648	Central	Islington	2016	PM25_Tyre	COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.00
100649	Central	City	2016	PM25_Tyre	COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.00

100650 rows × 19 columns

data cleaning 2

```
In [14]: df=df.rename(columns={'Total': 'Total PM 2.5 Emission'})
         df
```

Out[14]:

missions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Electric LGV	TfL Bus	Non-TfL Bus and Coach	Rigid HGV	Artic HGV	Total PM 2.5 Emission
DFT	tonne/year	0.789607	0.728706	63.290727	28.659199	0.000000	0.526430	10.919852	0.000000	1.489118	2.730038	6.708603	1.046861	116.889143
DFT	tonne/year	0.518391	0.478272	41.554182	18.816494	0.000000	0.345634	7.169557	0.000000	0.977660	1.792369	4.403879	0.686558	76.742997
DFT	tonne/year	0.795632	0.734195	63.784549	28.882810	0.000000	0.530545	11.005204	0.000000	1.500924	2.751682	6.761137	1.055737	117.802416
DFT	tonne/year	0.787536	0.726727	63.081901	28.564638	0.000000	0.524680	10.883539	0.000000	1.484490	2.721552	6.686387	1.039584	116.501034
DFT	tonne/year	0.710495	0.655763	56.956611	25.790995	0.000000	0.473756	9.827226	0.000000	1.338921	2.454678	6.037218	0.940247	105.185912
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000001	0.000000	0.000000	0.000000	0.000001
COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.003240	0.000017	0.000099	0.000338	0.001254	0.000301	0.016392
COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.004898	0.000026	0.001292	0.000000	0.001872	0.000449	0.024883
COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.000331	0.000002	0.000416	0.000000	0.000130	0.000031	0.002022
COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.004627	0.000025	0.000060	0.000533	0.001706	0.000409	0.021011

data cleaning 3

```
In [4]: df.columns
```

Out[4]:
```
Index(['Location_ExactCut', 'Borough_ExactCut', 'Year', 'Pollutant',
       'Emissions', 'Emissions Unit', 'Motorcycle', 'Taxi', 'Petrol Car',
       'Diesel Car', 'Electric Car', 'Petrol LGV', 'Diesel LGV',
       'Electric LGV', 'TfL Bus', 'Non-TfL Bus and Coach', 'Rigid HGV',
       'Artic HGV', 'Total PM 2.5 Emission'],
      dtype='object')
```

data cleaning 4

```
In [5]: data_with_index = df.set_index("Pollutant")
        data_with_index.head()
```

Out[5]:

	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Elec L
Pollutant													
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.789607	0.728706	63.290727	28.659199	0.0	0.526430	10.919852	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.518391	0.478272	41.554182	18.816494	0.0	0.345634	7.169557	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.795632	0.734195	63.784549	28.882810	0.0	0.530545	11.005204	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.787536	0.726727	63.081901	28.564638	0.0	0.524680	10.883539	
CO2	NonGLA	NonGLA	2010	DFT	tonne/year	0.710495	0.655763	56.956611	25.790995	0.0	0.473756	9.827226	

data cleaning 5

```
In [6]: data_with_index = data_with_index.drop(["CO2", "PM10_Tyre", "NOx","PM10_Brake","PM10_Exhaust","PM10_Resusp","PM25_Re
        data_with_index
```

Out[6]:

Pollutant	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	El
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005894	0.002905	0.000000	0.000064	0.001482	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000042	0.000039	0.003870	0.001907	0.000000	0.000042	0.000973	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000065	0.000059	0.005940	0.002928	0.000000	0.000064	0.001493	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005875	0.002896	0.000000	0.000063	0.001477	0.0
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000058	0.000053	0.005305	0.002614	0.000000	0.000057	0.001333	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	
PM25_Tyre	Inner	Southwark	2016	COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
PM25_Tyre	Central	Southwark	2016	COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.003240	0.0
PM25_Tyre	Central	Camden	2016	COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.004898	0.0
PM25_Tyre	Central	Islington	2016	COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.000331	0.0
PM25_Tyre	Central	City	2016	COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.004627	0.0

30195 rows × 18 columns

```
In [7]: new_df=data_with_index
```

data cleaning 6

```
In [19]: new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 30195 entries, PM25_Brake to PM25_Tyre
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Location_ExactCut      30195 non-null  object
 1   Borough_ExactCut       30195 non-null  object
 2   Year                   30195 non-null  int64
 3   Emissions              30195 non-null  object
 4   Emissions Unit         30195 non-null  object
 5   Motorcycle             30195 non-null  float64
 6   Taxi                   30195 non-null  float64
 7   Petrol Car             30195 non-null  float64
 8   Diesel Car             30195 non-null  float64
 9   Electric Car           30195 non-null  float64
 10  Petrol LGV             30195 non-null  float64
 11  Diesel LGV             30195 non-null  float64
 12  Electric LGV           30195 non-null  float64
 13  TfL Bus                30195 non-null  float64
 14  Non-TfL Bus and Coach  30195 non-null  float64
 15  Rigid HGV              30195 non-null  float64
 16  Artic HGV              30195 non-null  float64
 17  Total PM 2.5 Emission  30195 non-null  float64
dtypes: float64(13), int64(1), object(4)
memory usage: 4.4+ MB
```

data cleaning 7

```
In [21]: #sums up total null values of columns
         new_df.isna().sum()

Out[21]: Location_ExactCut          0
         Borough_ExactCut           0
         Year                       0
         Emissions                  0
         Emissions Unit             0
         Motorcycle                 0
         Taxi                       0
         Petrol Car                 0
         Diesel Car                 0
         Electric Car               0
         Petrol LGV                 0
         Diesel LGV                 0
         Electric LGV               0
         TfL Bus                    0
         Non-TfL Bus and Coach      0
         Rigid HGV                  0
         Artic HGV                  0
         Total PM 2.5 Emission      0
         dtype: int64
```

data cleaning 8

```
In [10]: new_df.head()
```

Out[10]:

Pollutant	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Elec
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005894	0.002905	0.0	0.000064	0.001482	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000042	0.000039	0.003870	0.001907	0.0	0.000042	0.000973	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000065	0.000059	0.005940	0.002928	0.0	0.000064	0.001493	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000064	0.000059	0.005875	0.002896	0.0	0.000063	0.001477	
PM25_Brake	NonGLA	NonGLA	2010	COPERT	tonne/year	0.000058	0.000053	0.005305	0.002614	0.0	0.000057	0.001333	

data cleaning 9

```
In [11]: new_df.tail()
```

Out[11]:

Pollutant	Location_ExactCut	Borough_ExactCut	Year	Emissions	Emissions Unit	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Elec
PM25_Tyre	Inner	Southwark	2016	COPERT	tonne/year	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
PM25_Tyre	Central	Southwark	2016	COPERT	tonne/year	0.000507	0.002053	0.004595	0.003905	0.000043	0.000041	0.003240	0.000
PM25_Tyre	Central	Camden	2016	COPERT	tonne/year	0.000745	0.002983	0.006754	0.005739	0.000064	0.000062	0.004898	0.000
PM25_Tyre	Central	Islington	2016	COPERT	tonne/year	0.000051	0.000200	0.000461	0.000392	0.000004	0.000004	0.000331	0.000
PM25_Tyre	Central	City	2016	COPERT	tonne/year	0.000623	0.002459	0.005653	0.004804	0.000054	0.000059	0.004627	0.000

data cleaning 10

```
In [23]: new_df.describe()
```

Out[23]:

	Year	Motorcycle	Taxi	Petrol Car	Diesel Car	Electric Car	Petrol LGV	Diesel LGV	Electric LGV	TfL Bus	Non-Tf and C
count	30195.00000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	30195.000000	3.019500e+04	30195.0
mean	2013.00000	0.000198	0.000350	0.006850	0.006656	0.000007	0.000050	0.003353	0.000002	6.728246e-04	0.0
std	2.44953	0.000312	0.001132	0.009498	0.007048	0.000017	0.000085	0.003779	0.000004	2.479002e-03	0.0
min	2010.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.0
25%	2010.00000	0.000024	0.000026	0.000641	0.001120	0.000000	0.000003	0.000535	0.000000	2.578211e-08	0.0
50%	2013.00000	0.000092	0.000105	0.002628	0.004194	0.000000	0.000015	0.002013	0.000000	4.584893e-05	0.0
75%	2016.00000	0.000265	0.000315	0.009419	0.009964	0.000005	0.000058	0.004875	0.000001	3.663209e-04	0.0
max	2016.00000	0.006475	0.036213	0.076354	0.051548	0.000238	0.000831	0.032946	0.000095	8.498240e-02	0.0

data cleaning 11

Appendix for k-means clustering:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
```

```
#dropping columns for numerical value specification.
df.drop(columns=['Borough_ExactCut','Emissions','Emissions Unit','Pollutant','Location_ExactCut'],axis=1 ,inplace=Tr
```

```
#used for standardizing data.
scaler=StandardScaler()
```

```
#Data standardization.
df[['Motor-cycle','Cab','Diesel-Car','Petrol-Car','Petrol-LGV','Diesel-LGV','TfL-Bus','Non-TfL Bus & Coach','Rigid-H
```

K-means clustering step 1

```
#used for standardizing data.
scaler=StandardScaler()
```

```
issions','Electric-Car','Electric-LGV']]=scaler.fit_transform(df[['Motorcycle','Taxi','Diesel Car','Petrol Car','Petr
```

K-means clustering step 2

```
#idetifying optimum number of clusters
#2
#create function to work out optimum number of clusters
def optimise_k_means(df,max_k):
    means=[]
    inertias=[]

    for k in range (1,max_k):
        kmeans=KMeans(n_clusters=k)
        kmeans.fit(df)

        means.append(k)
        inertias.append(kmeans.inertia_)

    #generate the elbow plot
    fig=plt.subplots(figsize=(10,5))
    plt.plot(means,inertias,'o-')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.grid(True)
    plt.show()
```

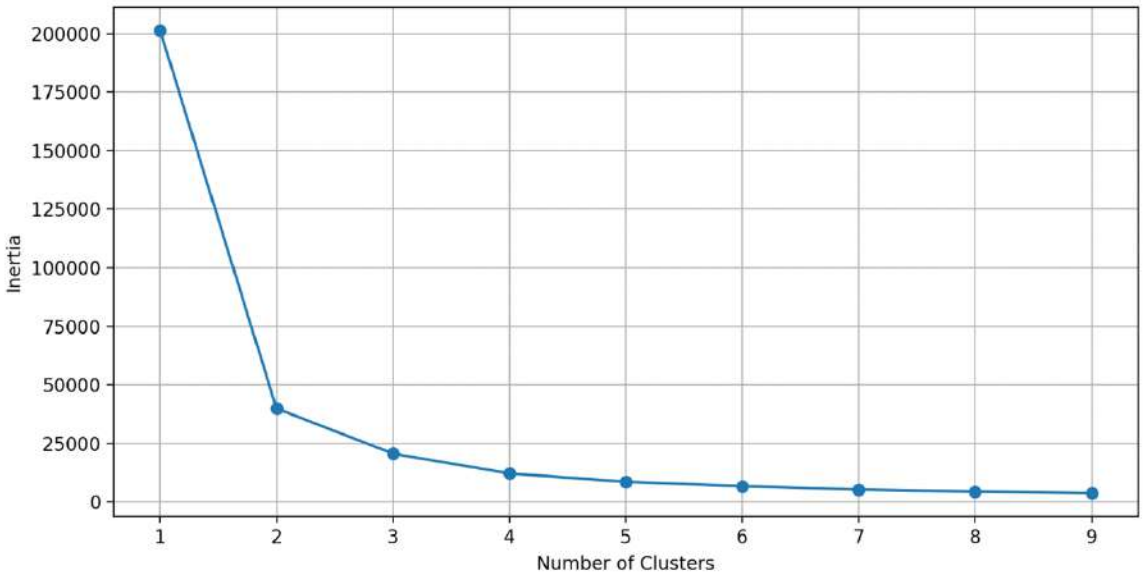
K-means clustering step 3

```
%matplotlib notebook
x=df['Artic-HGV']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='purple=x(-0.19)&y(-0.2),yellow=x(4.2)&y(4.4)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='Purple mean(low),Yellow mean(med with outliers)')
plt.xlim(-1, 22)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Artic-HGV')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Artic-HGV vs Total PM 2.5 emissions of all transport modes')
plt.legend()
plt.show()
```

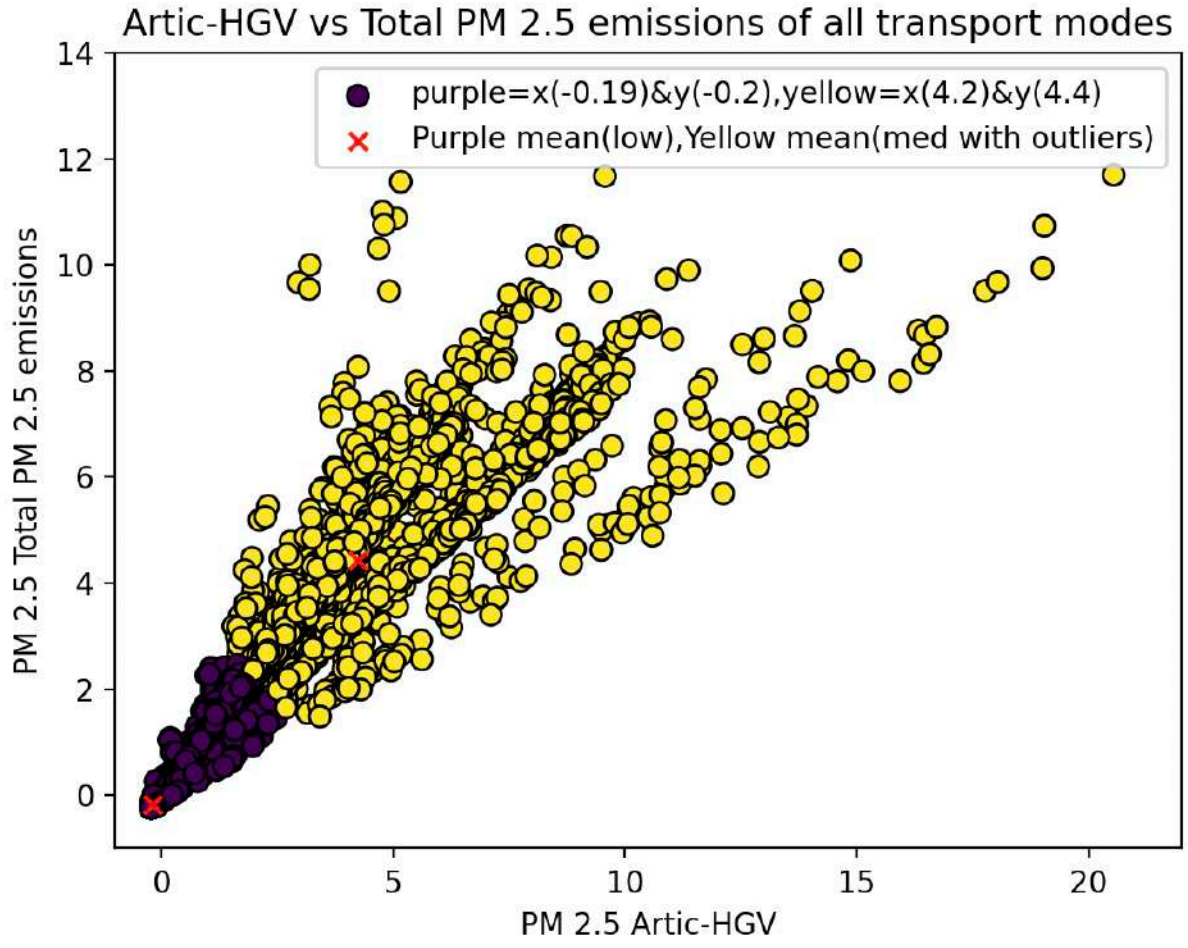
K-means clustering models 1

```
optimise_k_means(df[['Artic-HGV','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Artic-HGV','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 2



K-means clustering models 3

```
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Cab','Total PM 2.5 emissions']])
```

```
KMeans(n_clusters=2)
```

```
df['kmeans_3']=kmeans.labels_
```

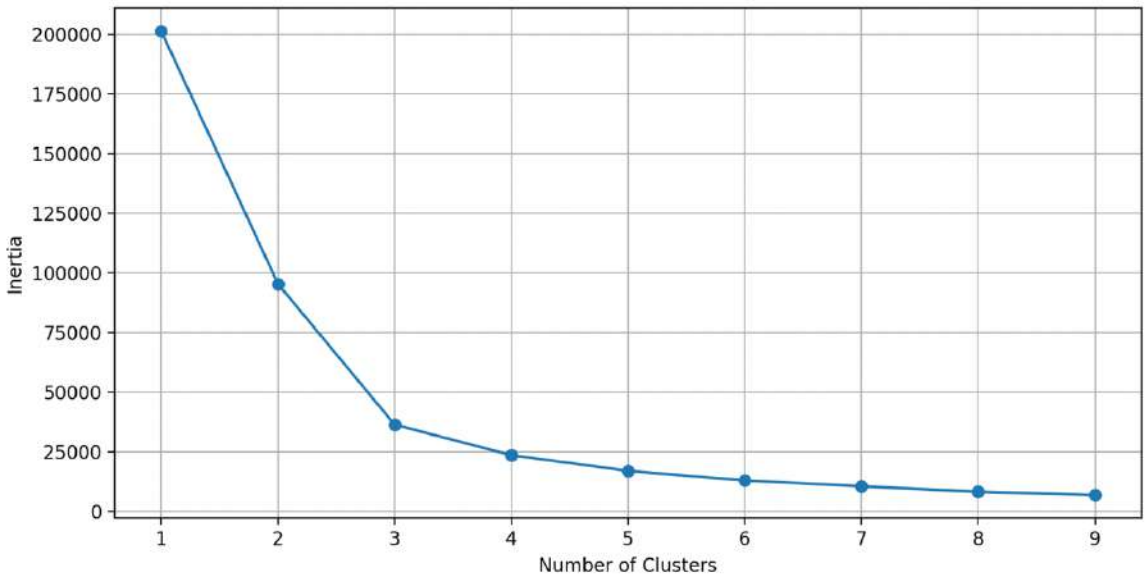
```
centroids=kmeans.cluster_centers_
```

```
#plotting the results
%matplotlib notebook
x=df['Cab']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='p=x(-0.13)&y(-0.23),y=x(2.44)&y(4.44)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(low),yellow mean(mid with outliers)')
plt.xlim(-1, 40)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Cab emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Cab PM 2.5 emission vs Total PM 2.5 emissions of all transportation modes')
plt.legend()
plt.show()
```

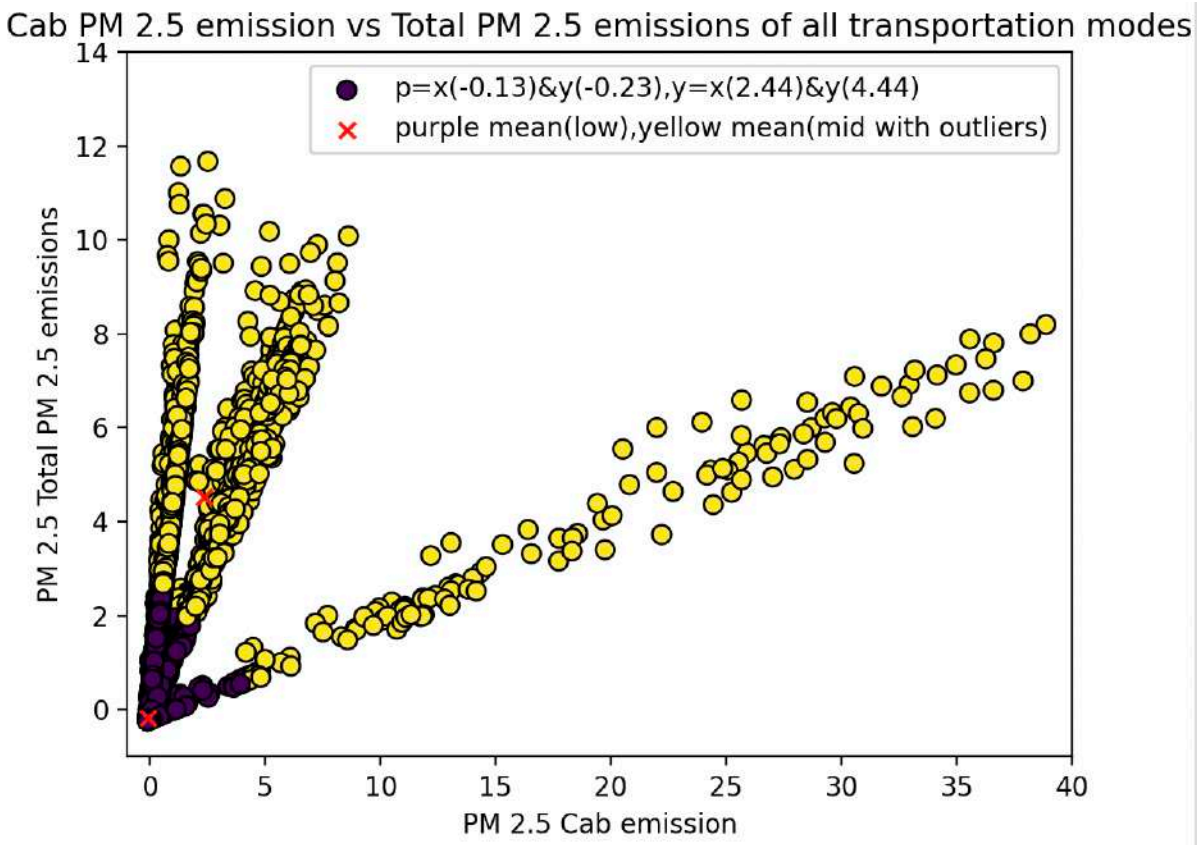
K-means clustering models 4

```
optimise_k_means(df[['Cab','Total PM 2.5 emissions']],10)
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 5



K-means clustering models 6

```
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Diesel-Car','Total PM 2.5 emissions']])
```

```
KMeans(n_clusters=2)
```

```
df['kmeans_3']=kmeans.labels_
```

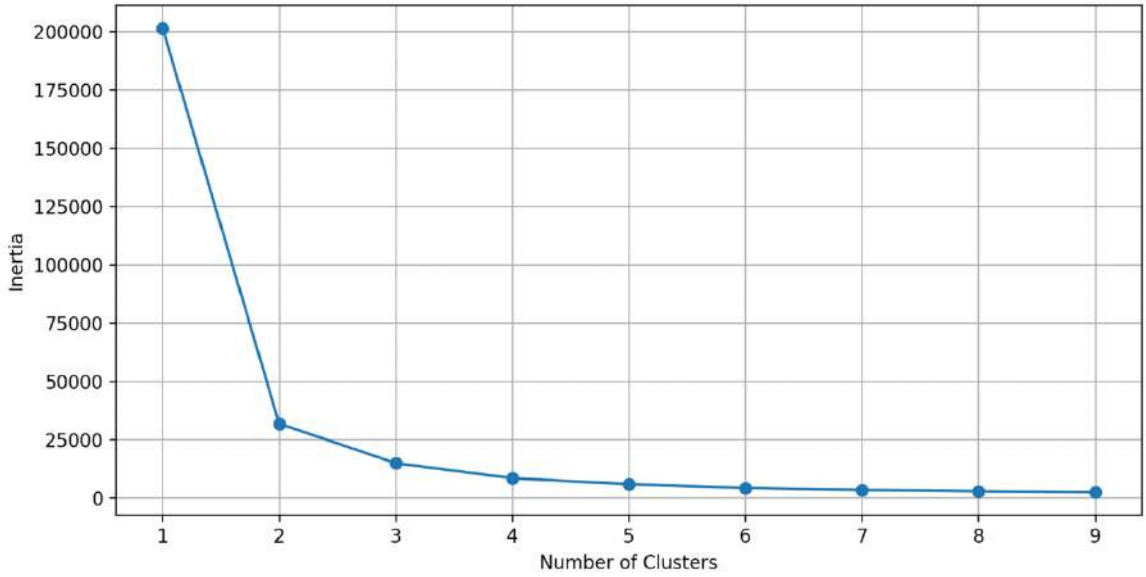
```
centroids=kmeans.cluster_centers_
```

```
#plotting the results
%matplotlib notebook
x=df['Diesel-Car']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='Green=x(-0.219)&y(-0.217),Yellow=x(4.37)&y(4.34)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='Green mean(lower gradient),Yellow mean(med gradein
plt.xlim(-1, 14)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Diesel-Car emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Diesel-Car PM 2.5 emission vs Total PM 2.5 emissions of all modes')
plt.legend()
plt.show()
```

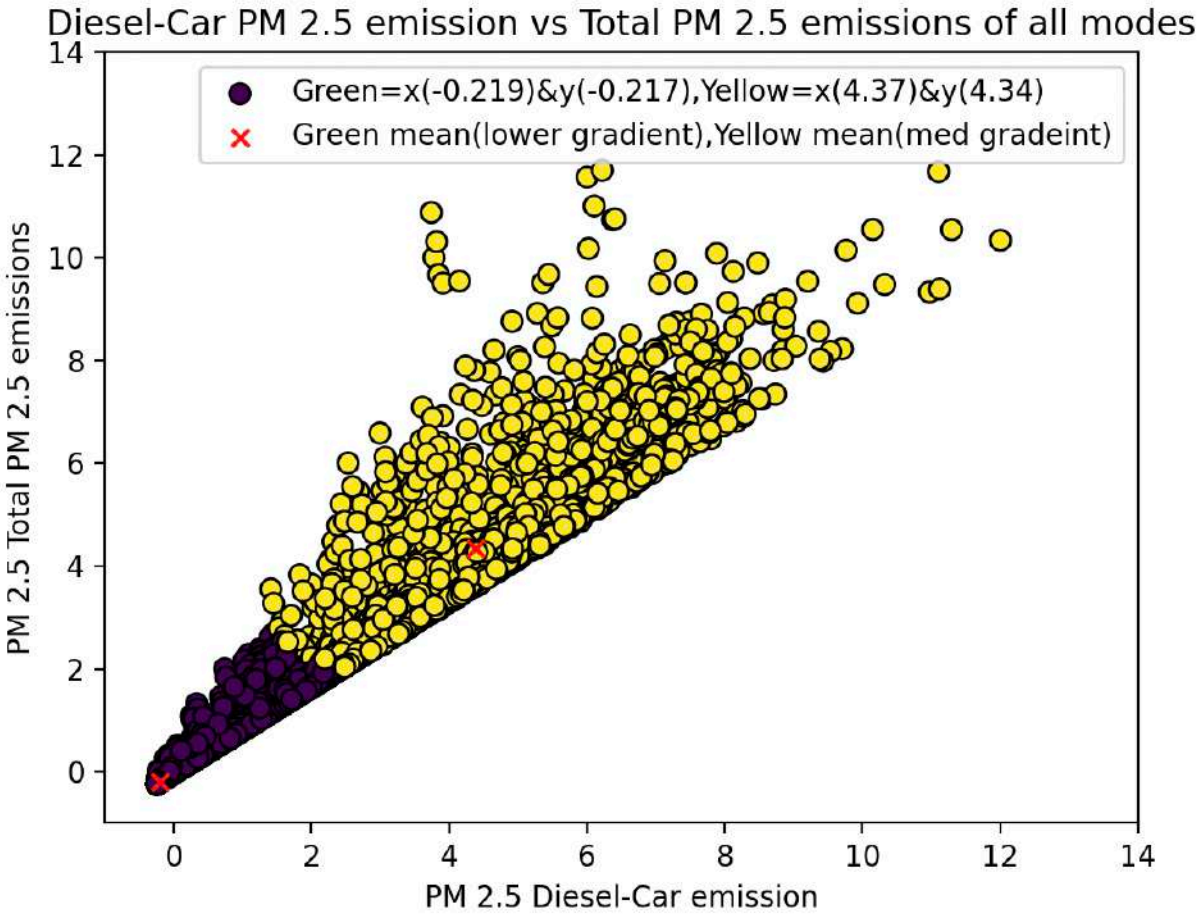
K-means clustering models 7

```
optimise_k_means(df[['Diesel-Car','Total PM 2.5 emissions']],10)
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 8



K-means clustering models 9

```
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Diesel-LGV','Total PM 2.5 emissions']])
```

```
KMeans(n_clusters=2)
```

```
df['kmeans_3']=kmeans.labels_
```

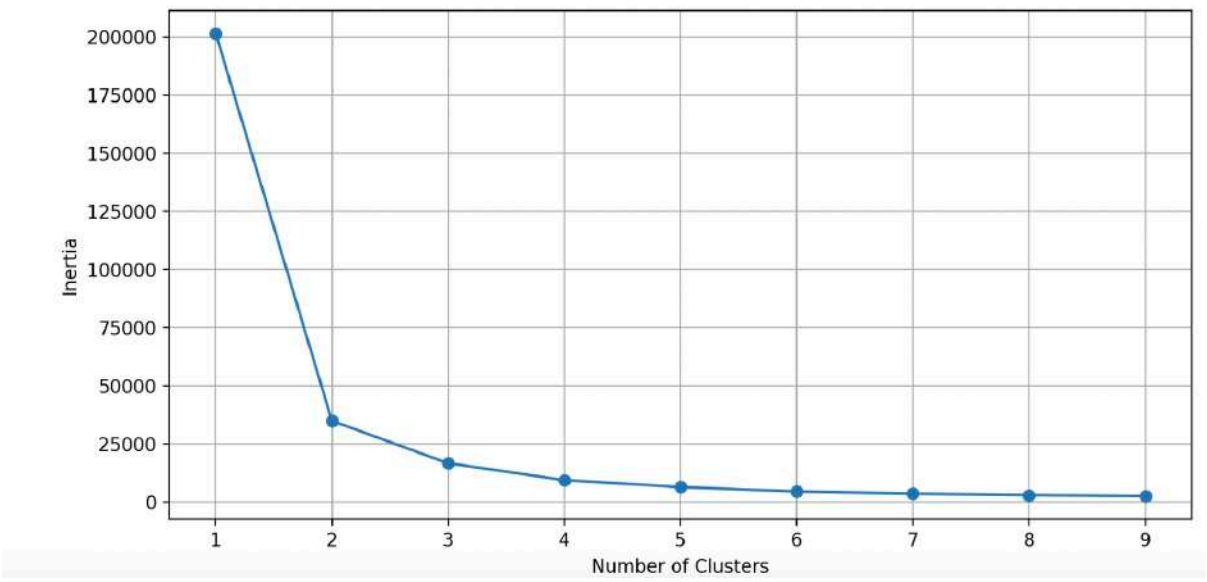
```
centroids=kmeans.cluster_centers_
```

```
#plotting the results
%matplotlib notebook
x=df['Diesel-LGV']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='Purple=x(-0.19)&y(-0.30),Yellow=x(4.31)&y(4.33)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='Purple mean(low),Yellow mean(mid with outliers)')
plt.xlim(-1, 22)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Diesel-LGV emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Diesel-LGV PM 2.5 emission vs Total PM 2.5 emissions of all transportation modes')
plt.legend()
plt.show()
```

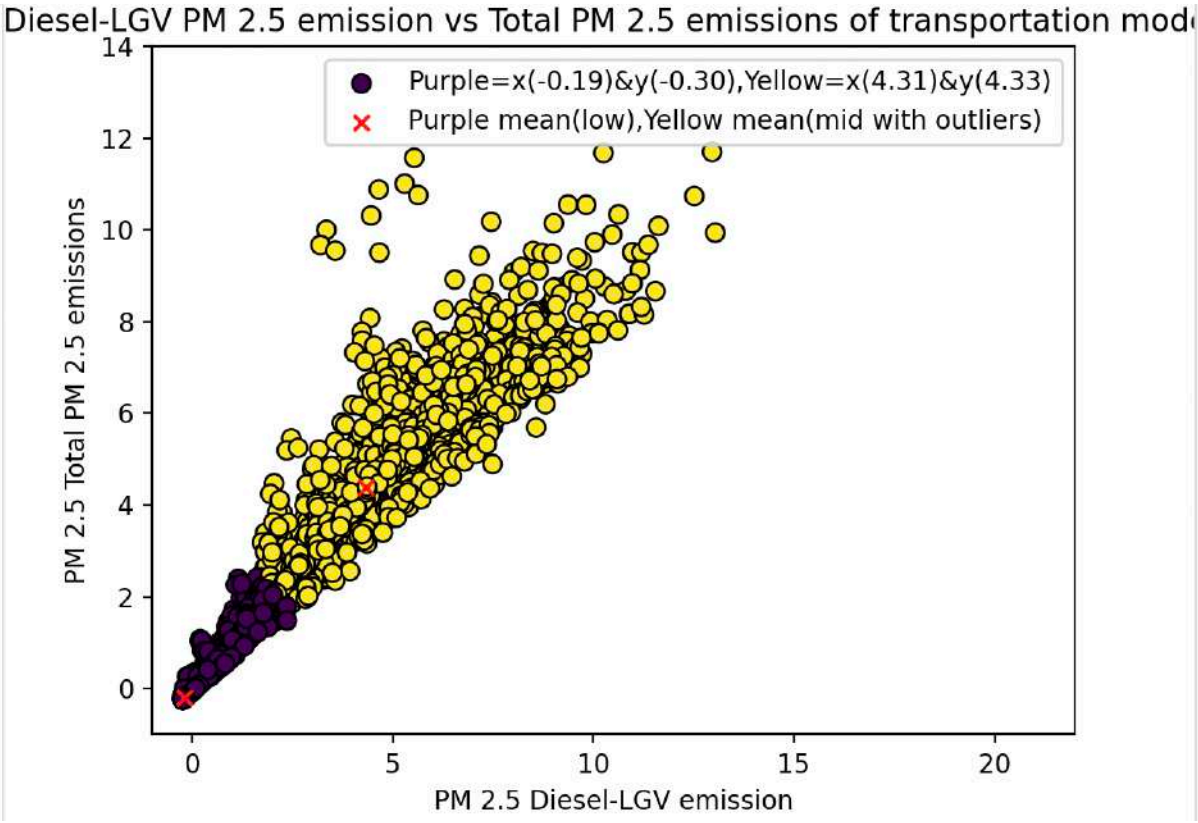
K-means clustering models 10

```
optimise_k_means(df[['Diesel-LGV','Total PM 2.5 emissions']],10)
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 11



K-means clustering models 12

```
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Motor-cycle','Total PM 2.5 emissions']])
```

```
KMeans(n_clusters=2)
```

```
df['kmeans_3']=kmeans.labels_
```

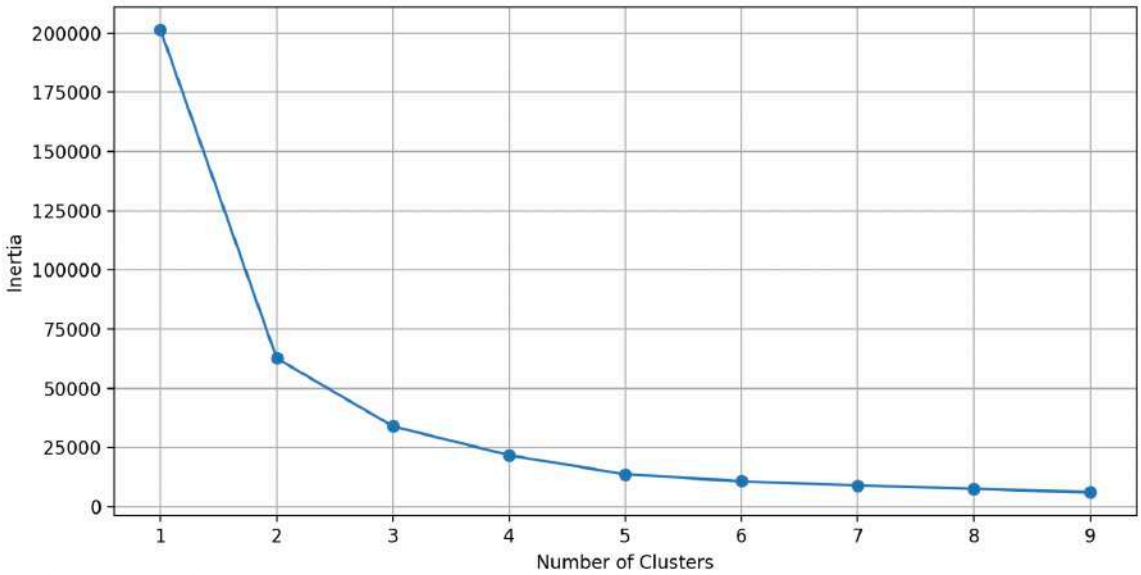
```
centroids=kmeans.cluster_centers_
```

```
#plotting the results
%matplotlib notebook
x=df['Motor-cycle']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='purple=x(-0.15)&y(-0.26),yellow=x(3.66)&y(4.41)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(low),yellow mean(med with outliers)')
plt.xlim(-1, 30)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Motor-cycle emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Motor-cycle PM 2.5 emission vs Total PM 2.5 emissions of all modes')
plt.legend()
plt.show()
```

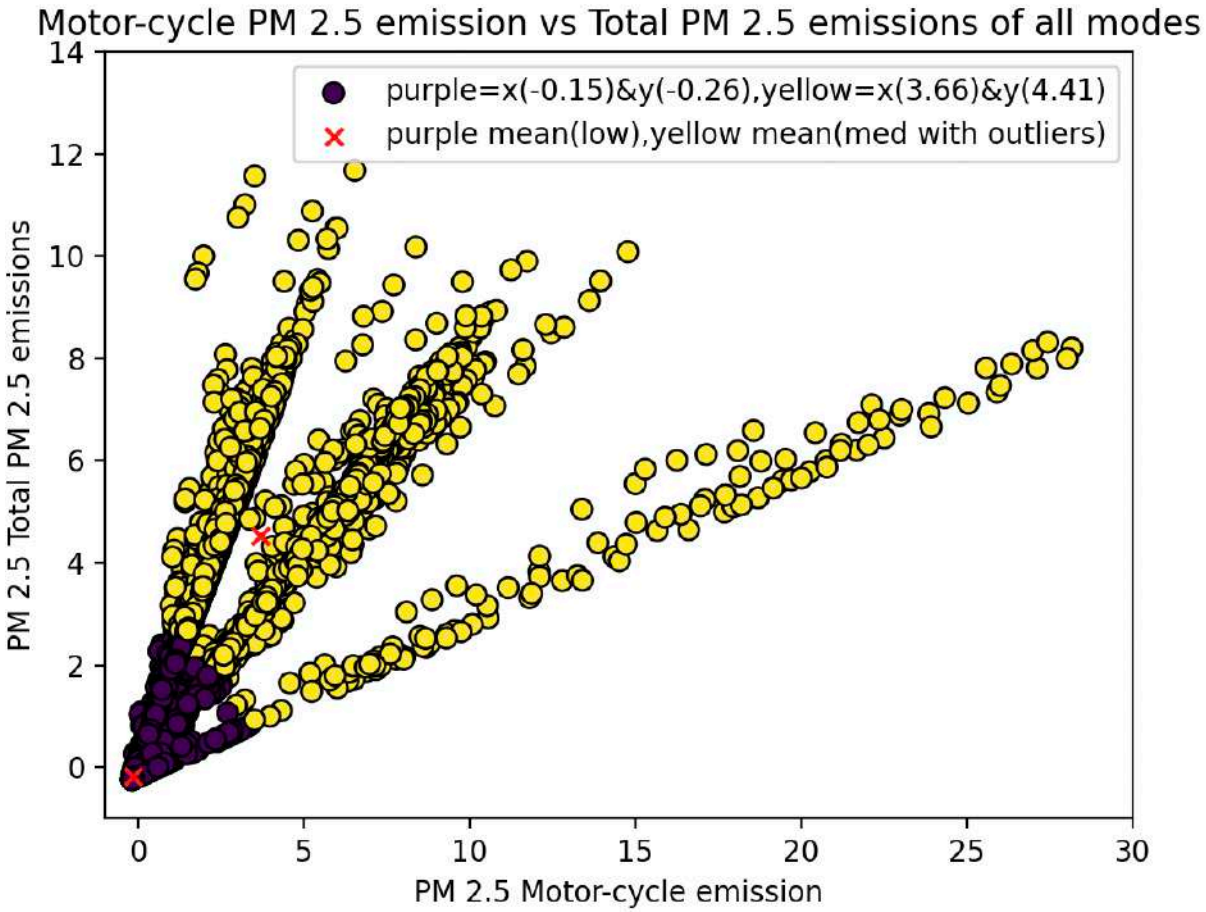
K-means clustering models 13

```
#Diesel LGV vs petrol LGV.
optimise_k_means(df[['Motor-cycle','Total PM 2.5 emissions']],10)
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 14

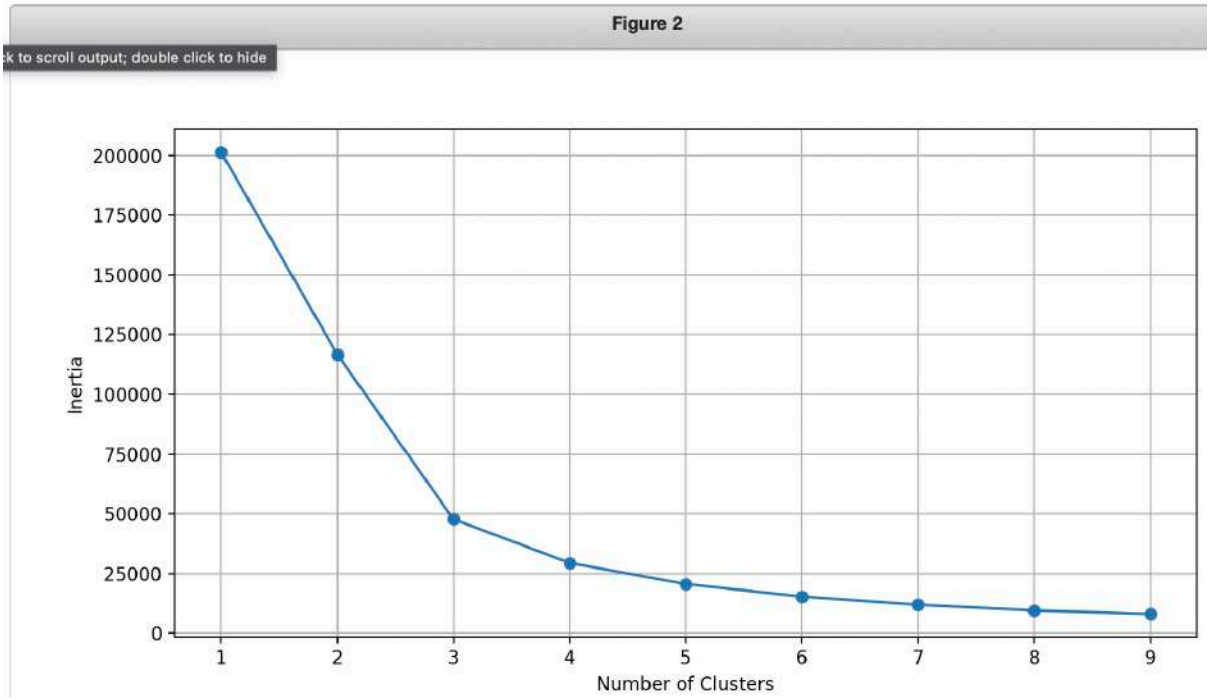


K-means clustering models 15

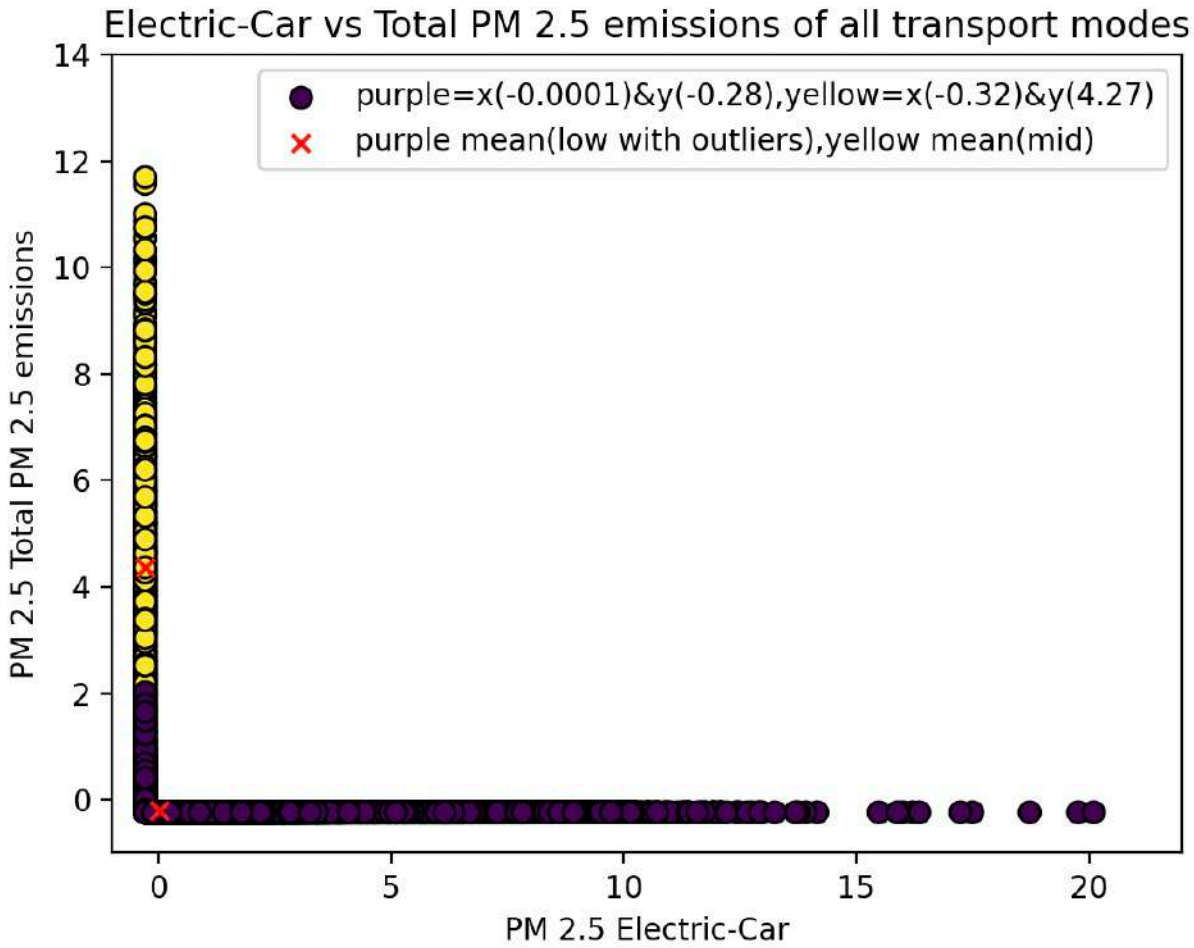
```
%matplotlib notebook
x=df['Electric-Car']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='purple=x(-0.0001)&y(-0.28),yellow=x(-0.32)&y(4.27)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(low with outliers),yellow mean(mid)')
plt.xlim(-1, 22)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Electric-Car')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Electric-Car vs Total PM 2.5 emissions of all transport modes')
plt.legend()
plt.show()
```

K-means clustering models 16

```
optimise_k_means(df[['Electric-Car','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Electric-Car','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```



K-means clustering models 17

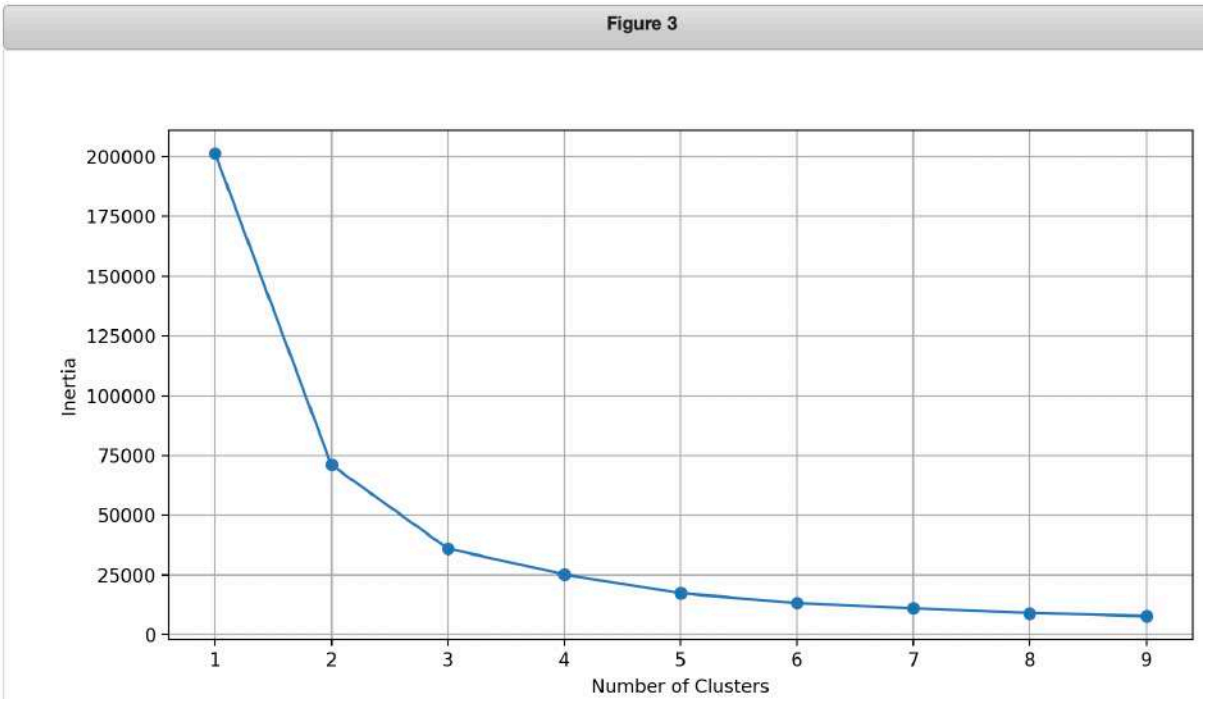


K-means clustering models 18

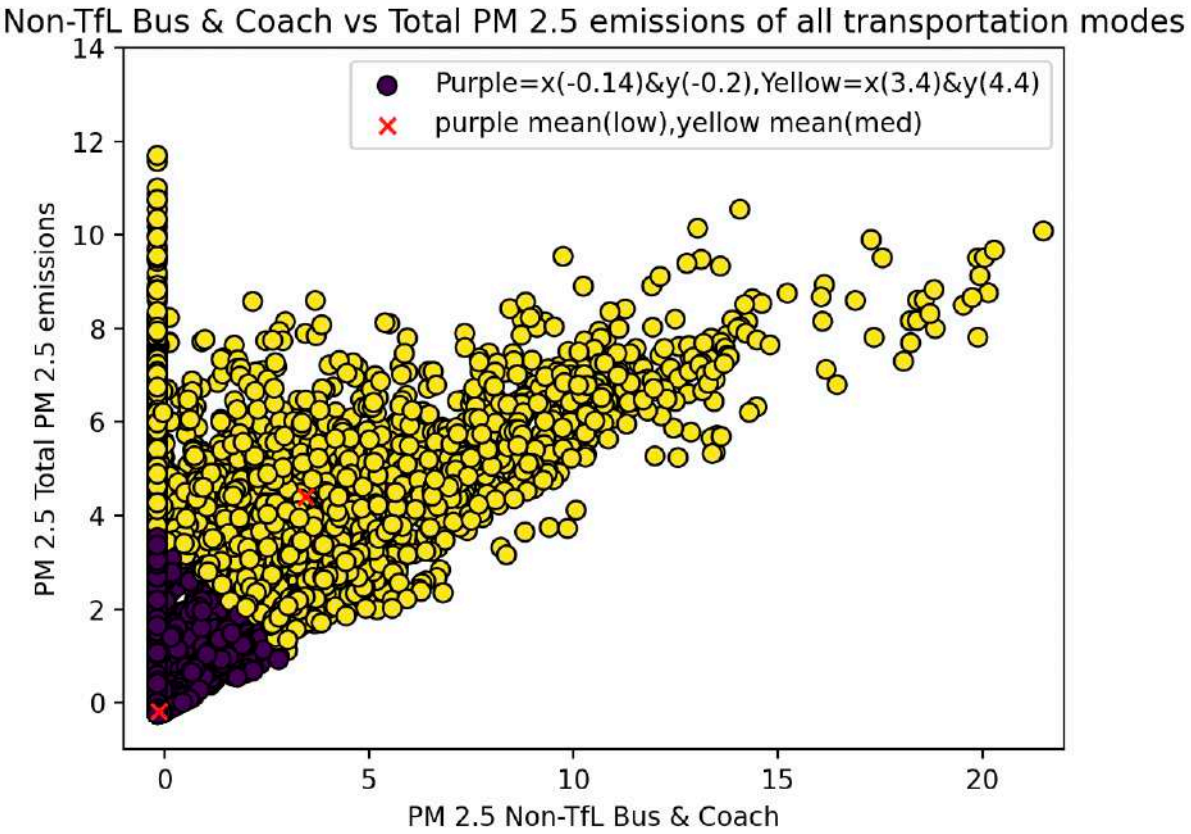
```
%matplotlib notebook
x=df['Non-TfL Bus & Coach']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='Purple=x(-0.14)&y(-0.2),Yellow=x(3.4)&y(4.4)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(low),yellow mean(med)')
plt.xlim(-1, 22)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Non-TfL Bus & Coach')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Non-TfL Bus & Coach vs Total PM 2.5 emissions of all transportation modes')
plt.legend()
plt.show()
```

K-means clustering models 19

```
optimise_k_means(df[['Non-TfL Bus & Coach','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Non-TfL Bus & Coach','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```



K-means clustering models 20



K-means clustering models 21

```
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Petrol-Car','Total PM 2.5 emissions']])
```

```
KMeans(n_clusters=2)
```

```
df['kmeans_3']=kmeans.labels_
```

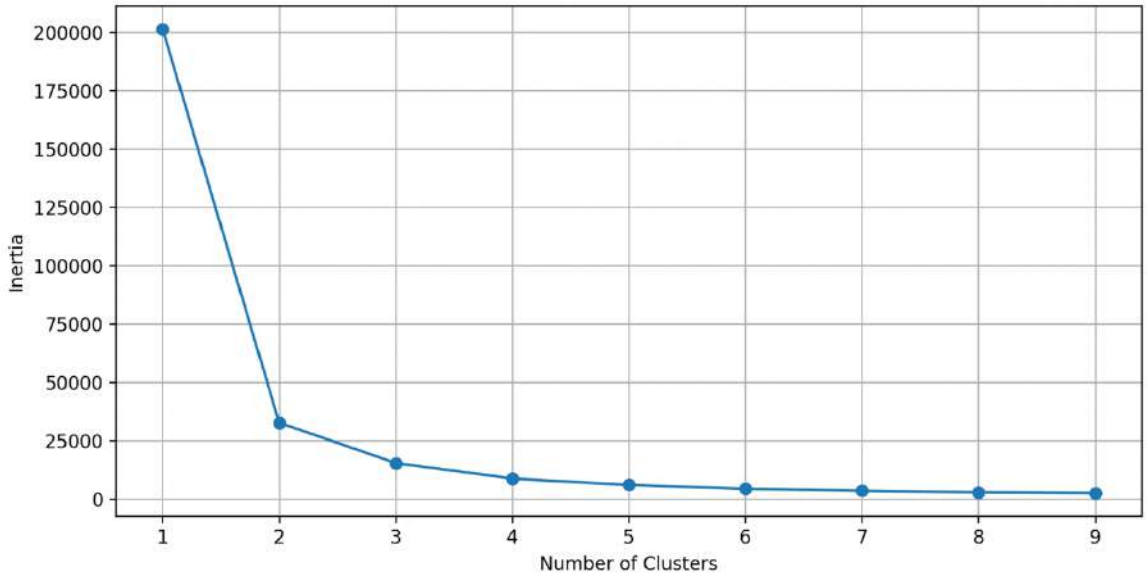
```
centroids=kmeans.cluster_centers_
```

```
#plotting the results
%matplotlib notebook
x=df['Petrol-Car']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='purple=x(-0.2169)&y(-0.2164),yellow=x(4.346)&y(4.347)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(low),yellow mean(med and upper quartil
plt.xlim(-1, 14)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Petrol-Car emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Petrol-Car PM 2.5 emission vs Total PM 2.5 emissions of all modes')
plt.legend()
plt.show()
```

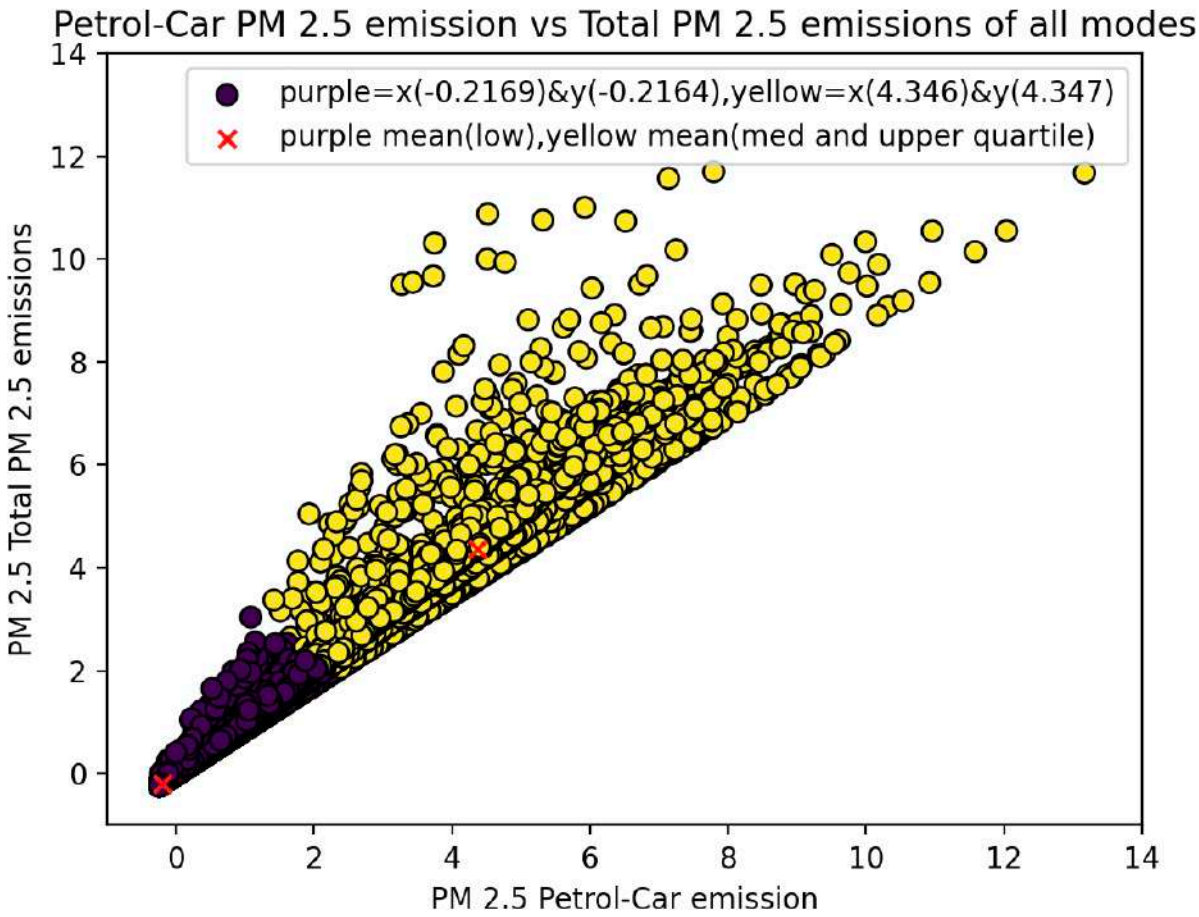
K-means clustering models 22

```
#Petrol-Car vs Total PM 2.5 emissions.
optimise_k_means(df[['Petrol-Car','Total PM 2.5 emissions']],10)
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 23



K-means clustering models 24

```
kmeans=KMeans(n_clusters=2)
kmeans.fit(df[['Petrol-LGV','Total PM 2.5 emissions']])
```

```
KMeans(n_clusters=2)
```

```
df['kmeans_3']=kmeans.labels_
```

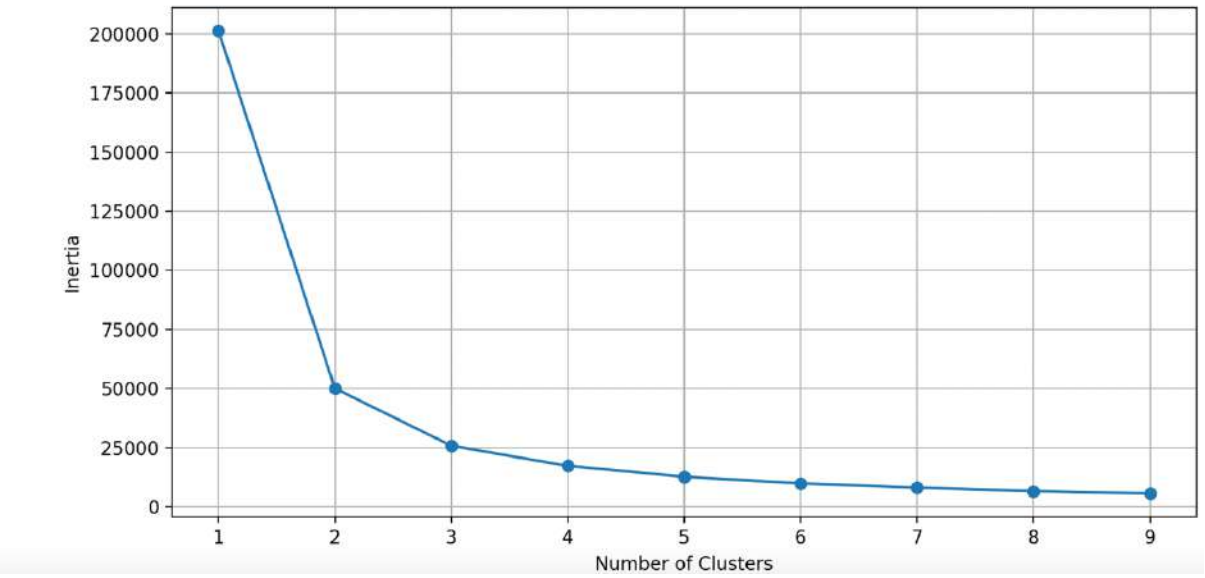
```
centroids=kmeans.cluster_centers_
```

```
#plotting the results
%matplotlib notebook
x=df['Petrol-LGV']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='Purple=x(-0.16)&y(-0.19),Yellow=x(4.05)&y(4.4)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(lower),yellow mean(mid)')
plt.xlim(-1, 22)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Petrol-LGV emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Petrol-LGV PM 2.5 emission vs Total PM 2.5 emissions of all modes')
plt.legend()
plt.show()
```

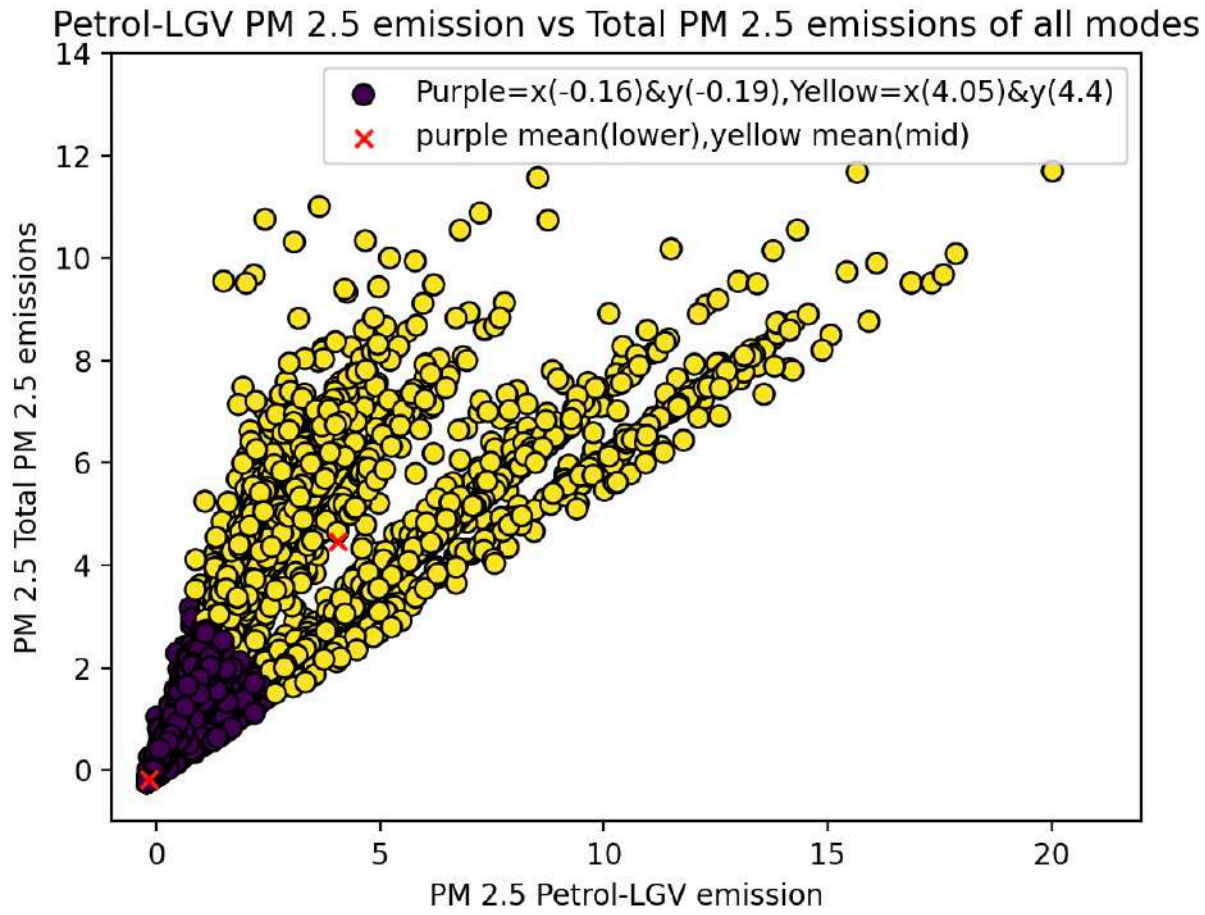
K-means clustering models 25

```
optimise_k_means(df[['Petrol-LGV','Total PM 2.5 emissions']],10)
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 26



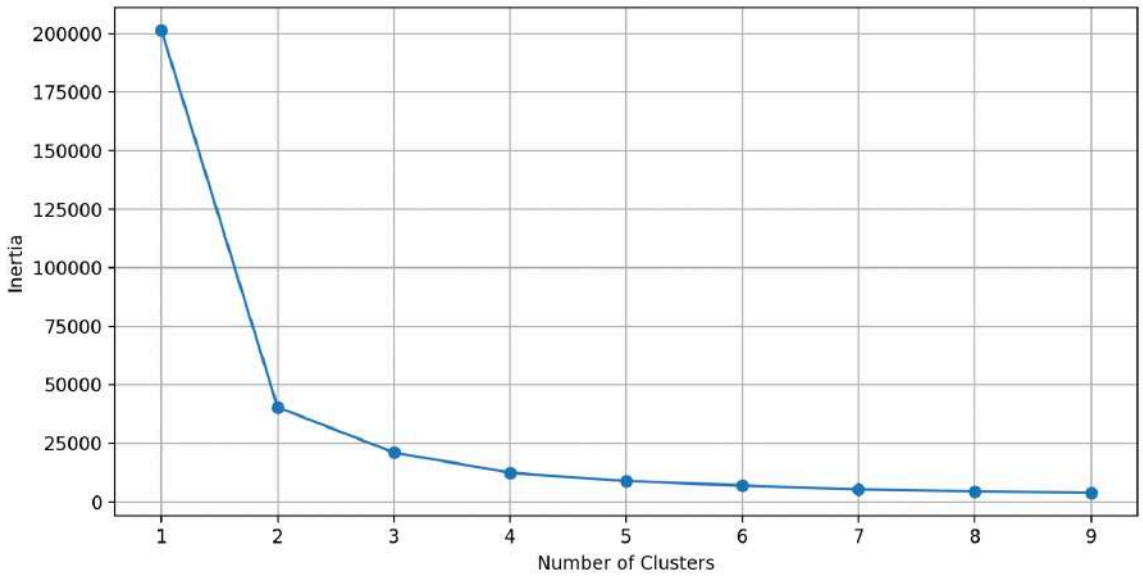
K-means clustering models 27

```
%matplotlib notebook
x=df['Rigid-HGV']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='Purple=x(-0.19)&y(-0.2),Yellow=x(4.2)&y(4.3)',edgecolor='k')
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='purple mean(low),yellow mean(mid with outliers)')
plt.xlim(-1, 22)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 Rigid-HGV')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('Rigid-HGV vs Total PM 2.5 emissions of all transportation modes')
plt.legend()
plt.show()
```

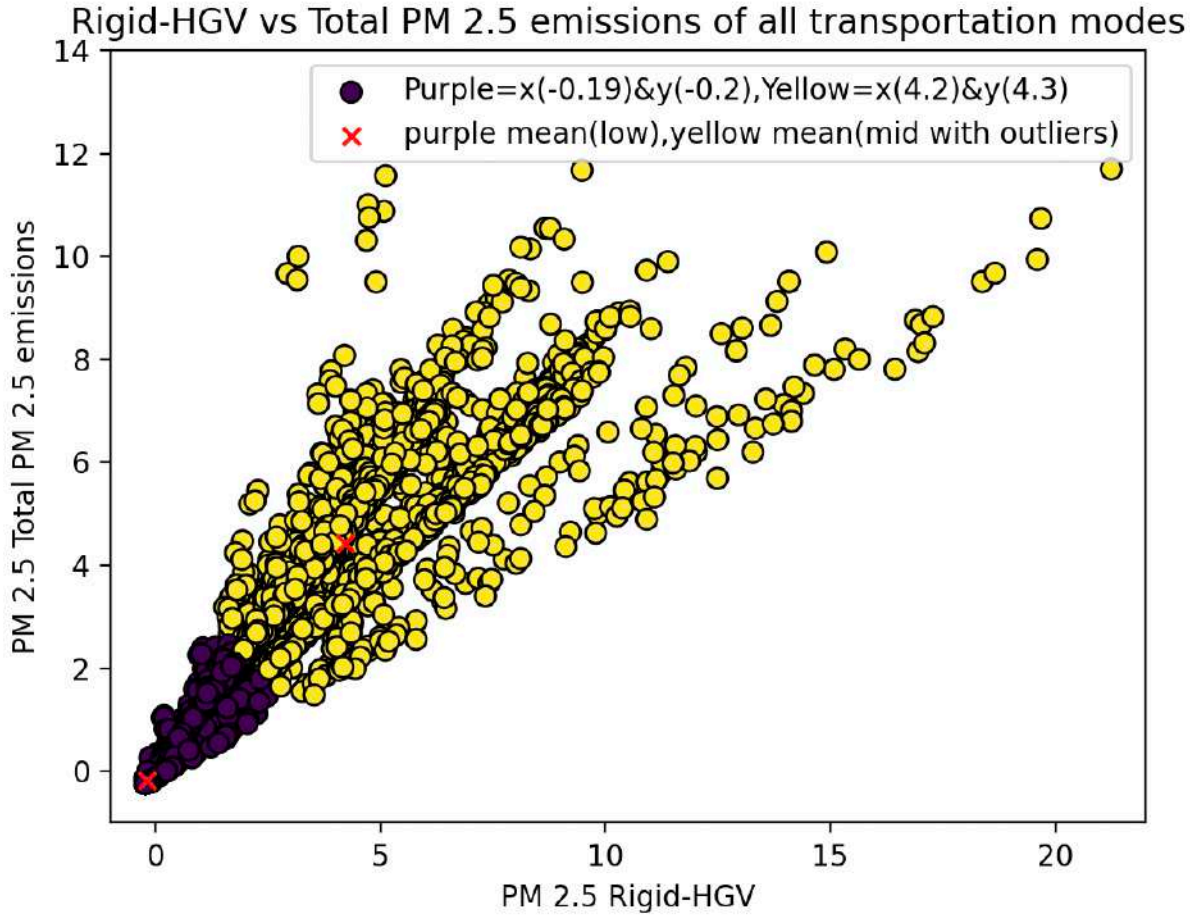
K-means clustering models 28

```
optimise_k_means(df[['Rigid-HGV','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=3)
kmeans.fit(df[['Rigid-HGV','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```

```
<IPython.core.display.Javascript object>
```



K-means clustering models 29



K-means clustering models 30

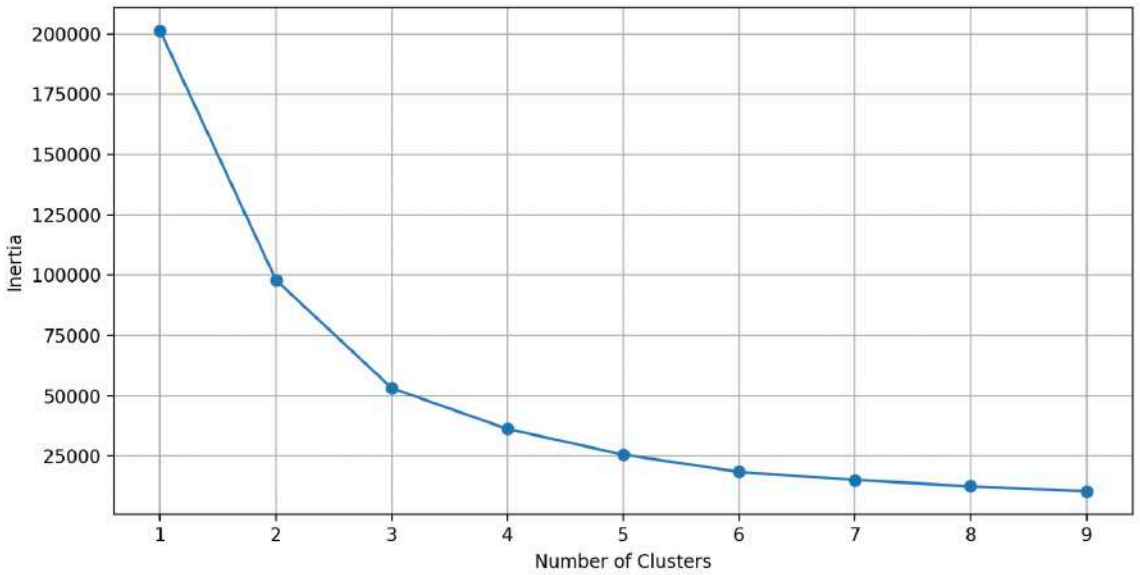
```
optimise_k_means(df[['TfL-Bus','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=3)
kmeans.fit(df[['TfL-Bus','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```

K-means clustering models 31

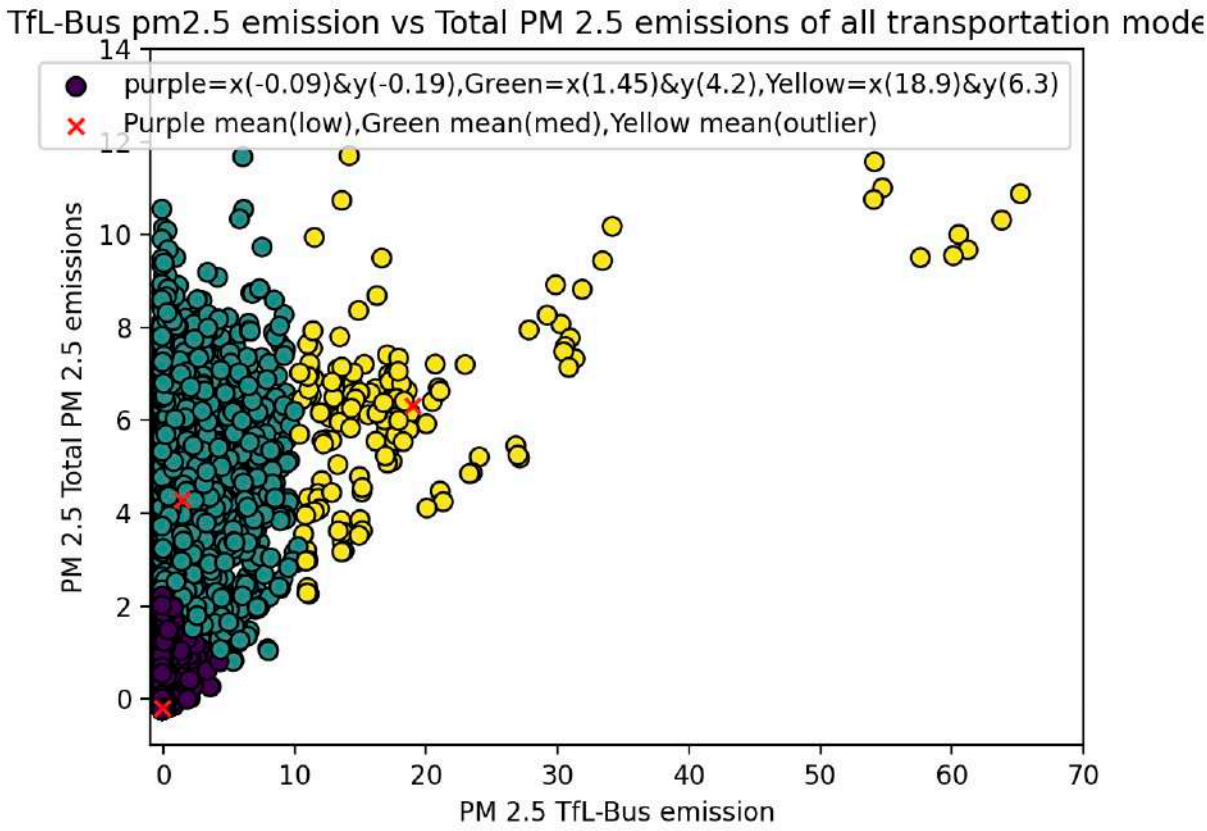
```
%matplotlib notebook
x=df['TfL-Bus']
y=df['Total PM 2.5 emissions']
plt.scatter(x,y,c=df['kmeans_3'],s=50,label='purple=x(-0.09)&y(-0.19),Green=x(1.45)&y(4.2),Yellow=x(18.9)&y(6.3)',ed
plt.scatter(centroids[:,0],centroids[:,1],marker='x',c='r',label='Purple mean(low),Green mean(med),Yellow mean(outli
plt.xlim(-1, 70)
plt.ylim(-1, 14)
plt.xlabel('PM 2.5 TfL-Bus emission')
plt.ylabel('PM 2.5 Total PM 2.5 emissions')
plt.title('TfL-Bus pm2.5 emission vs Total PM 2.5 emissions of all transportation modes')
plt.legend()
plt.show()
```

K-means clustering models 32

```
optimise_k_means(df[['TfL-Bus','Total PM 2.5 emissions']],10)
kmeans=KMeans(n_clusters=3)
kmeans.fit(df[['TfL-Bus','Total PM 2.5 emissions']])
df['kmeans_3']=kmeans.labels_
centroids=kmeans.cluster_centers_
```



K-means clustering models 33

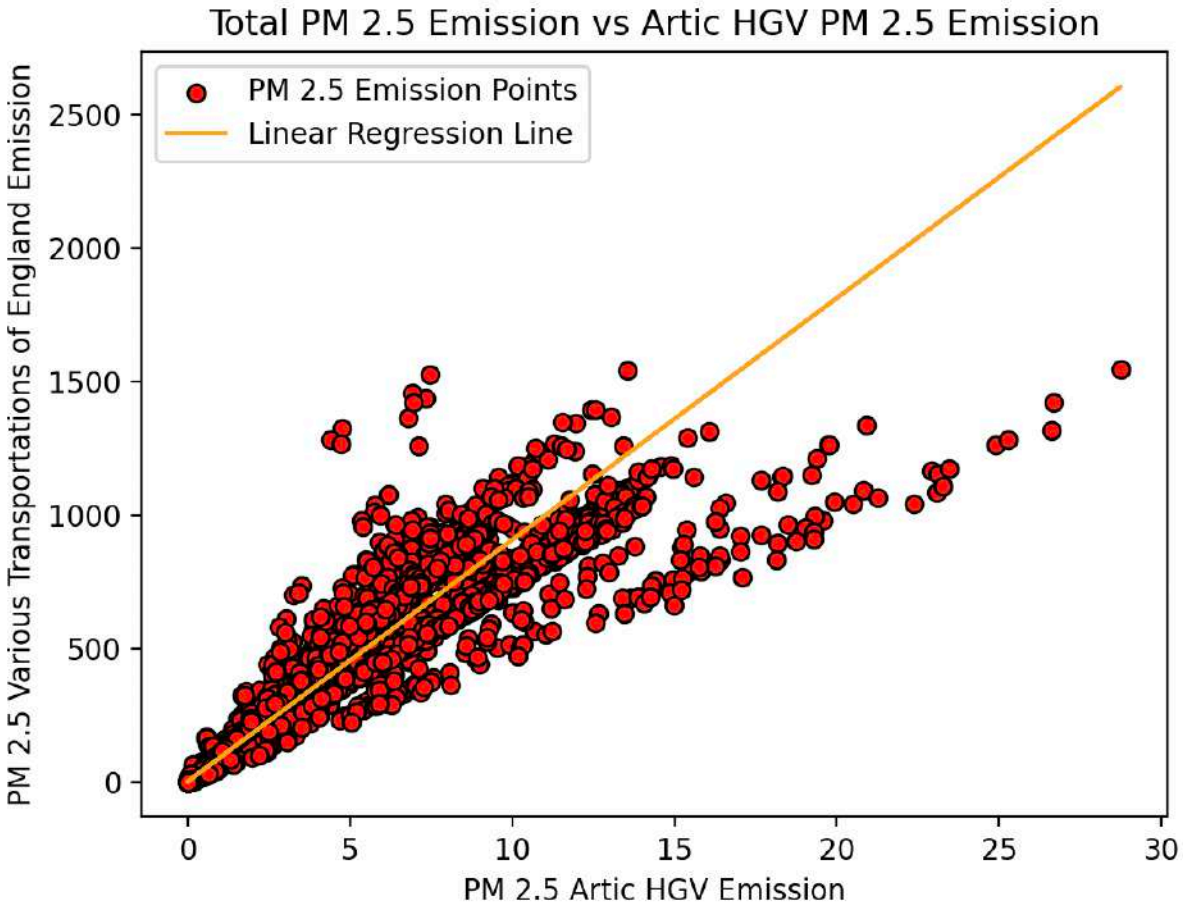


K-means clustering models 34

Appendix for linear regression:

In [21]:
```
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from sklearn.linear_model import LinearRegression


%matplotlib notebook

df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
x=df['Artic HGV']
y=df['Total']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
plt.xlabel('PM 2.5 Artic HGV Emission')
plt.ylabel('PM 2.5 Various Transportations of England Emission')
plt.title('Total PM 2.5 Emission vs Artic HGV PM 2.5 Emission')
plt.legend()
plt.show()
```

linear regression 1

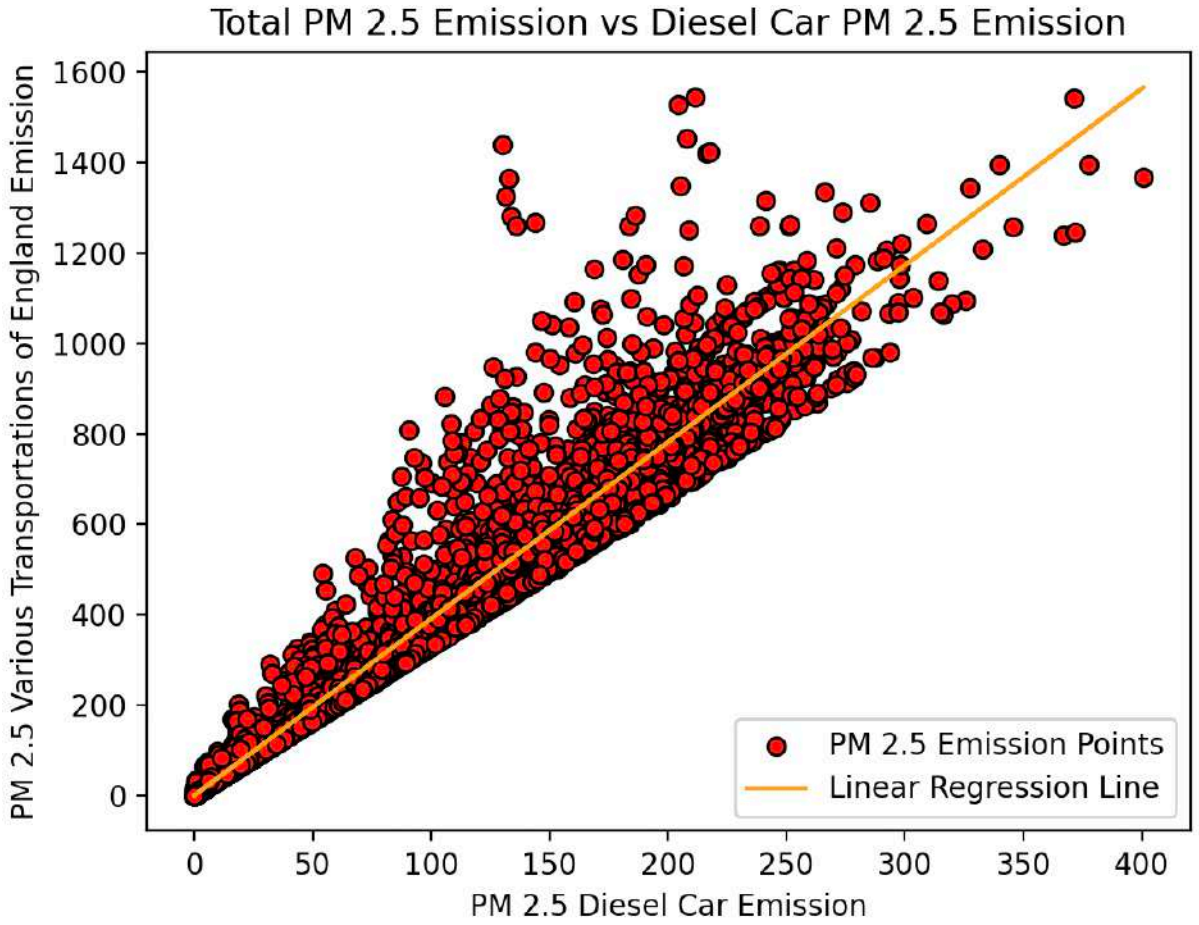


linear regression 2

In [14]:
```
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

%matplotlib notebook

df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
x=df['Diesel Car']
y=df['Total']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
plt.xlabel('PM 2.5 Diesel Car Emission')
plt.ylabel('PM 2.5 Various Transportations of England Emission')
plt.title('Total PM 2.5 Emission vs Diesel Car PM 2.5 Emission')
plt.legend()
plt.show()
```

```
<IPython.core.display.Javascript object>
```

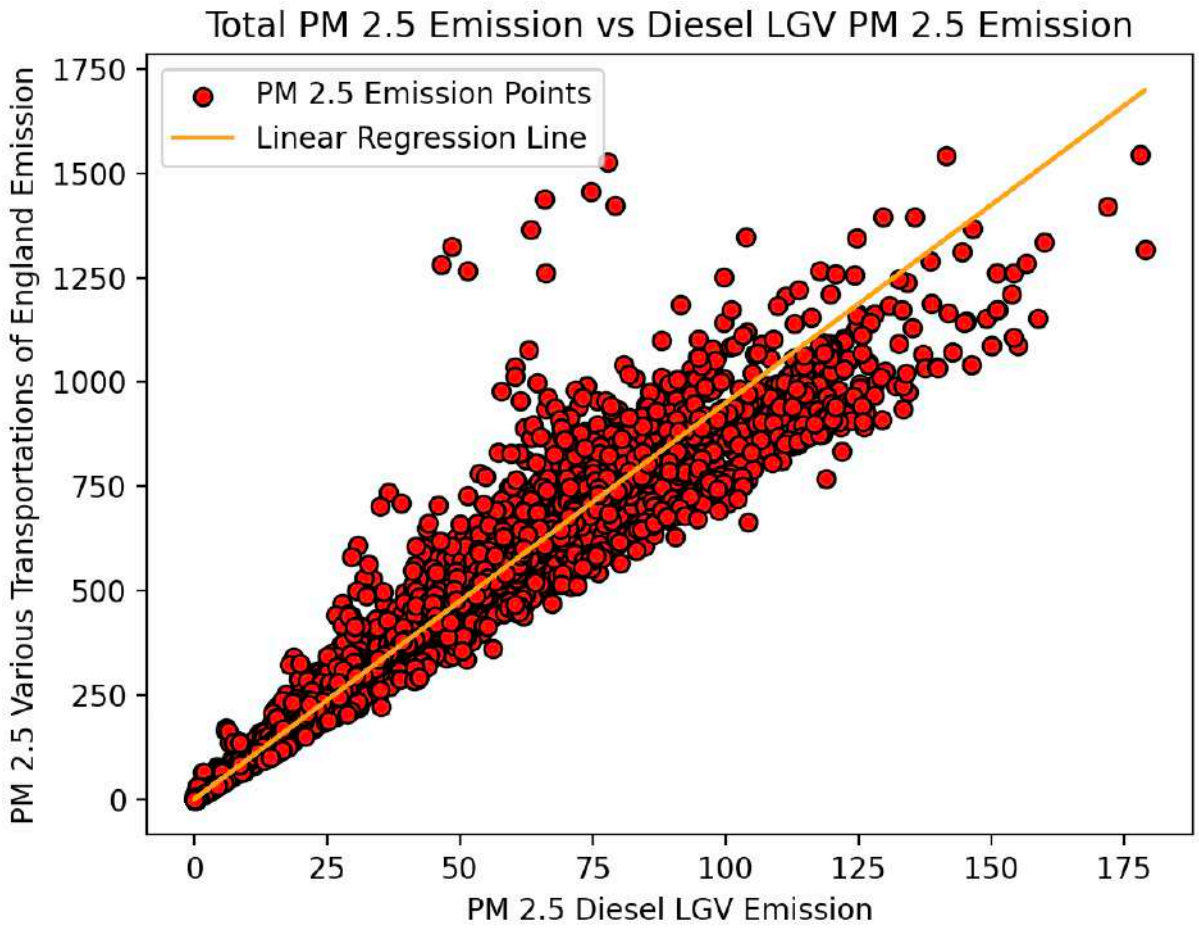
linear regression 3



linear regression 4

```
In [10]: import matplotlib.pyplot as plt
         from scipy import stats
         import pandas as pd
         from sklearn.linear_model import LinearRegression
         %matplotlib notebook

         df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
         x=df['Diesel LGV']
         y=df['Total']

         slope, intercept, r, p, std_err = stats.linregress(x, y)

         def myfunc(x):
           return slope * x + intercept

         mymodel = list(map(myfunc, x))

         plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
         plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
         plt.xlabel('PM 2.5 Diesel LGV Emission')
         plt.ylabel('PM 2.5 Various Transportations of England Emission')
         plt.title('Total PM 2.5 Emission vs Diesel LGV PM 2.5 Emission')
         plt.legend()
         plt.show()
```

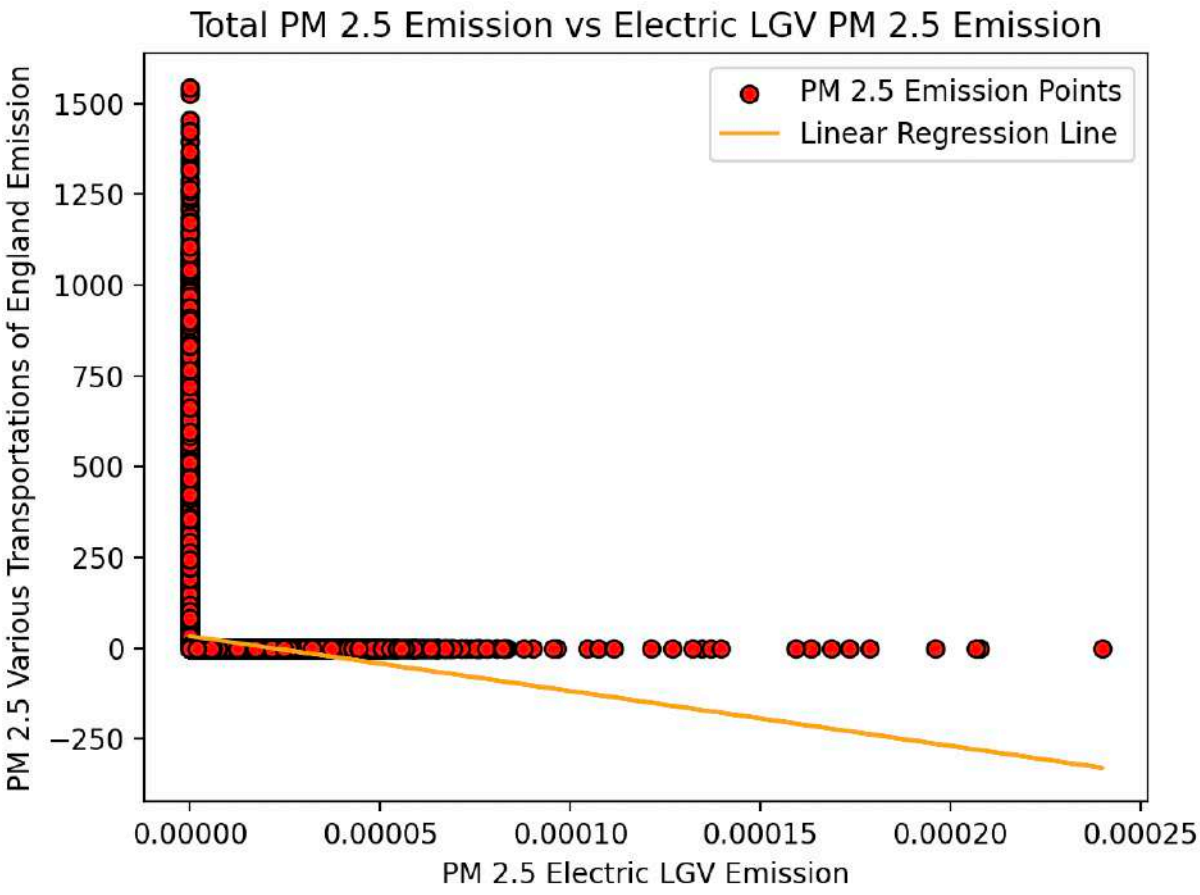
linear regression 5



linear regression 6

```
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from sklearn.linear_model import LinearRegression
%matplotlib notebook

df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
x=df['Electric LGV']
y=df['Total']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
plt.xlabel('PM 2.5 Electric LGV Emission')
plt.ylabel('PM 2.5 Various Transportations of England Emission')
plt.title('Total PM 2.5 Emission vs Electric LGV PM 2.5 Emission')
plt.legend()
plt.show()
```

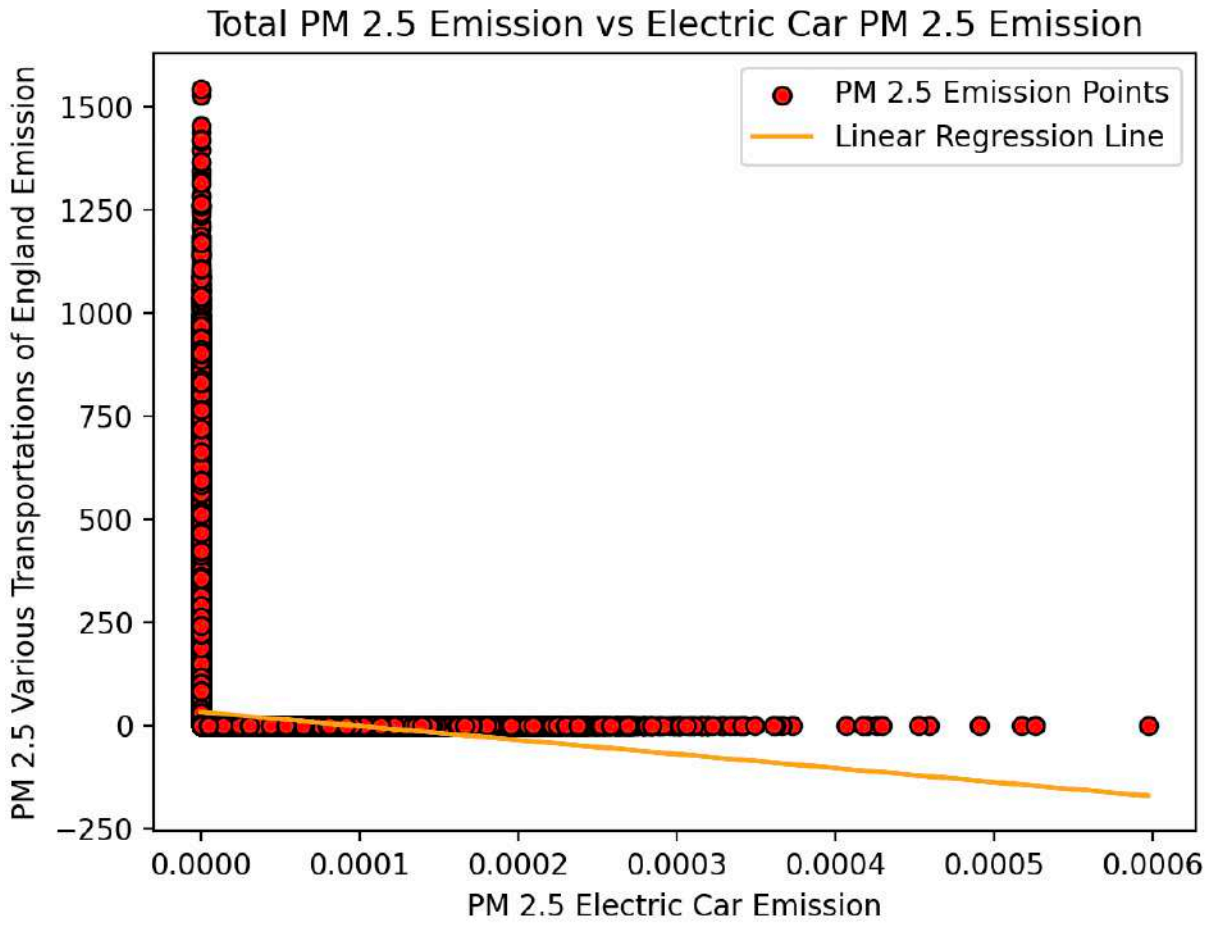
linear regression 7



linear regression 8

```
In [6]: import matplotlib.pyplot as plt
        from scipy import stats
        import pandas as pd
        import numpy as np
        from sklearn.linear_model import LinearRegression
        from sklearn.model_selection import train_test_split

        %matplotlib notebook

        df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
        x=df['Electric Car']
        y=df['Total']

        slope, intercept, r, p, std_err = stats.linregress(x, y)

        def myfunc(x):
          return slope * x + intercept

        mymodel = list(map(myfunc, x))

        plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
        plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
        plt.xlabel('PM 2.5 Electric Car Emission')
        plt.ylabel('PM 2.5 Various Transportations of England Emission')
        plt.title('Total PM 2.5 Emission vs Electric Car PM 2.5 Emission')
        plt.legend()
        plt.show()
```

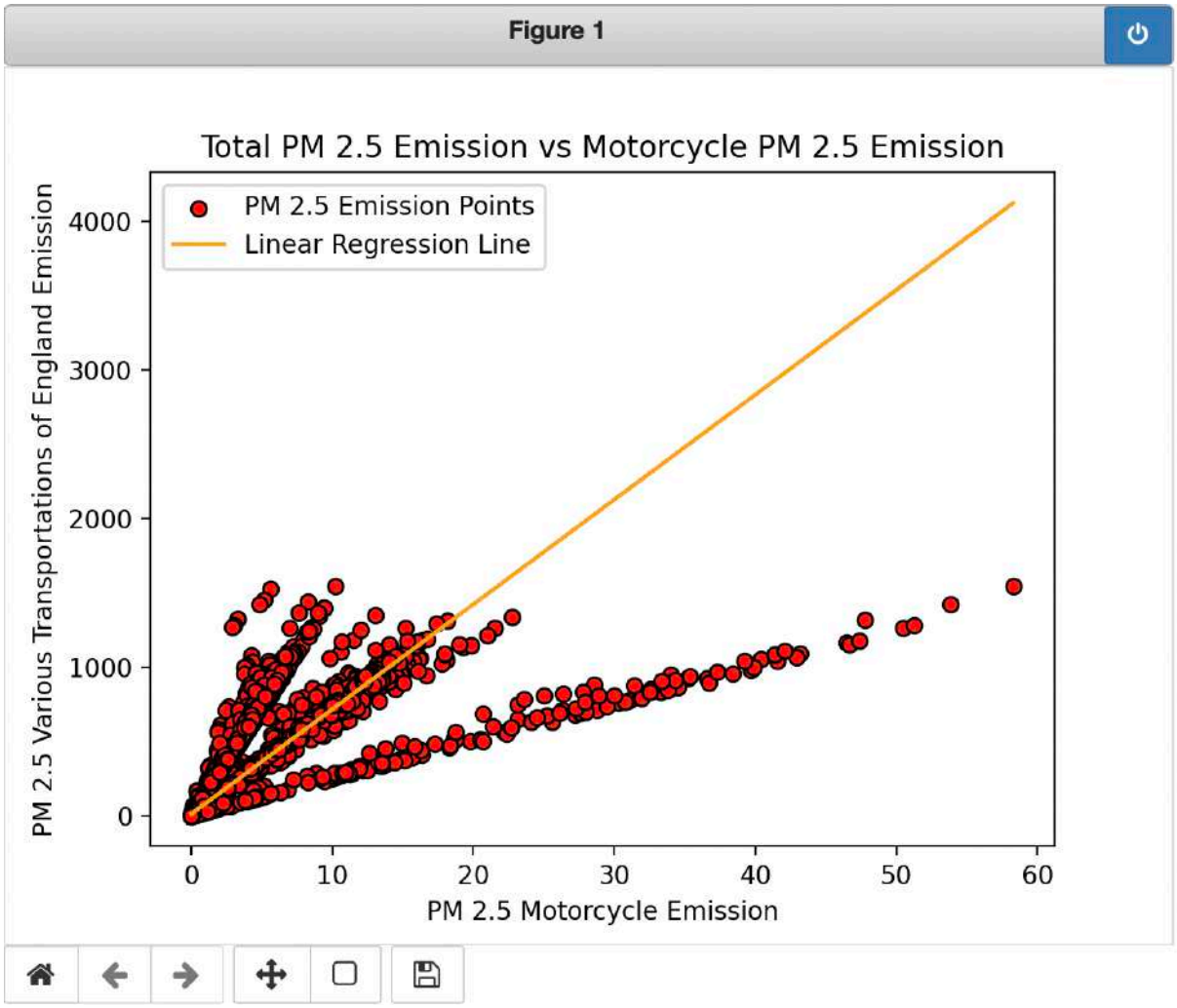
linear regression 9



linear regression 10

```
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from sklearn.linear_model import LinearRegression


%matplotlib notebook

df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
x=df['Motorcycle']
y=df['Total']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
plt.xlabel('PM 2.5 Motorcycle Emission')
plt.ylabel('PM 2.5 Various Transportations of England Emission')
plt.title('Total PM 2.5 Emission vs Motorcycle PM 2.5 Emission')
plt.legend()
plt.show()
```

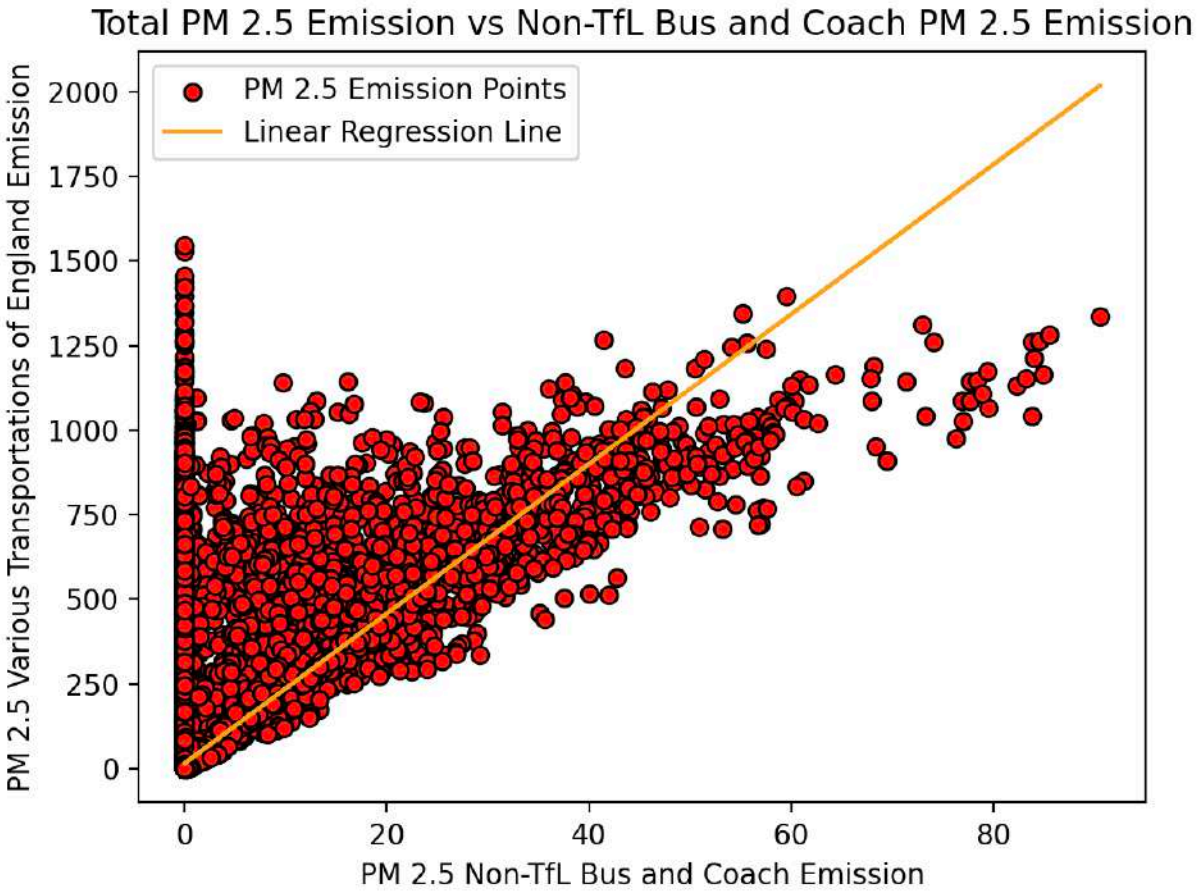
linear regression 11



linear regression 12

In [19]:
```
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from sklearn.linear_model import LinearRegression


%matplotlib notebook

df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
x=df['Non-TfL Bus and Coach']
y=df['Total']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
plt.xlabel('PM 2.5 Non-TfL Bus and Coach Emission')
plt.ylabel('PM 2.5 Various Transportations of England Emission')
plt.title('Total PM 2.5 Emission vs Non-TfL Bus and Coach PM 2.5 Emission')
plt.legend()
plt.show()
```

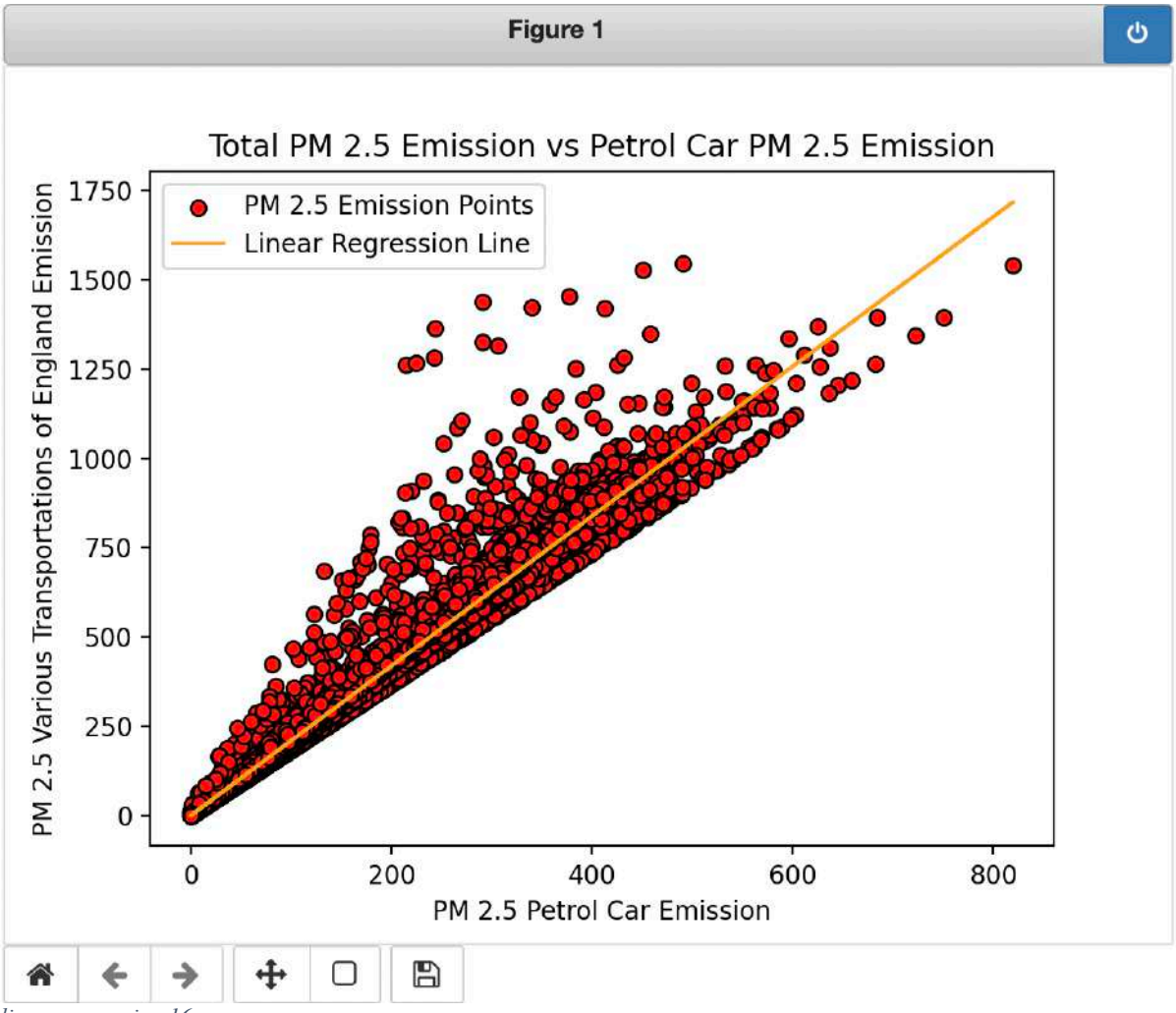
linear regression 13



linear regression 14

```
In [3]: import matplotlib.pyplot as plt
        from scipy import stats
        import pandas as pd
        from sklearn.linear_model import LinearRegression
        %matplotlib notebook

        df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
        x=df['Petrol Car']
        y=df['Total']

        slope, intercept, r, p, std_err = stats.linregress(x, y)

        def myfunc(x):
          return slope * x + intercept

        mymodel = list(map(myfunc, x))

        plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
        plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
        plt.xlabel('PM 2.5 Petrol Car Emission')
        plt.ylabel('PM 2.5 Various Transportations of England Emission')
        plt.title('Total PM 2.5 Emission vs Petrol Car PM 2.5 Emission')
        plt.legend()
        plt.show()

        <IPython.core.display.Javascript object>
```

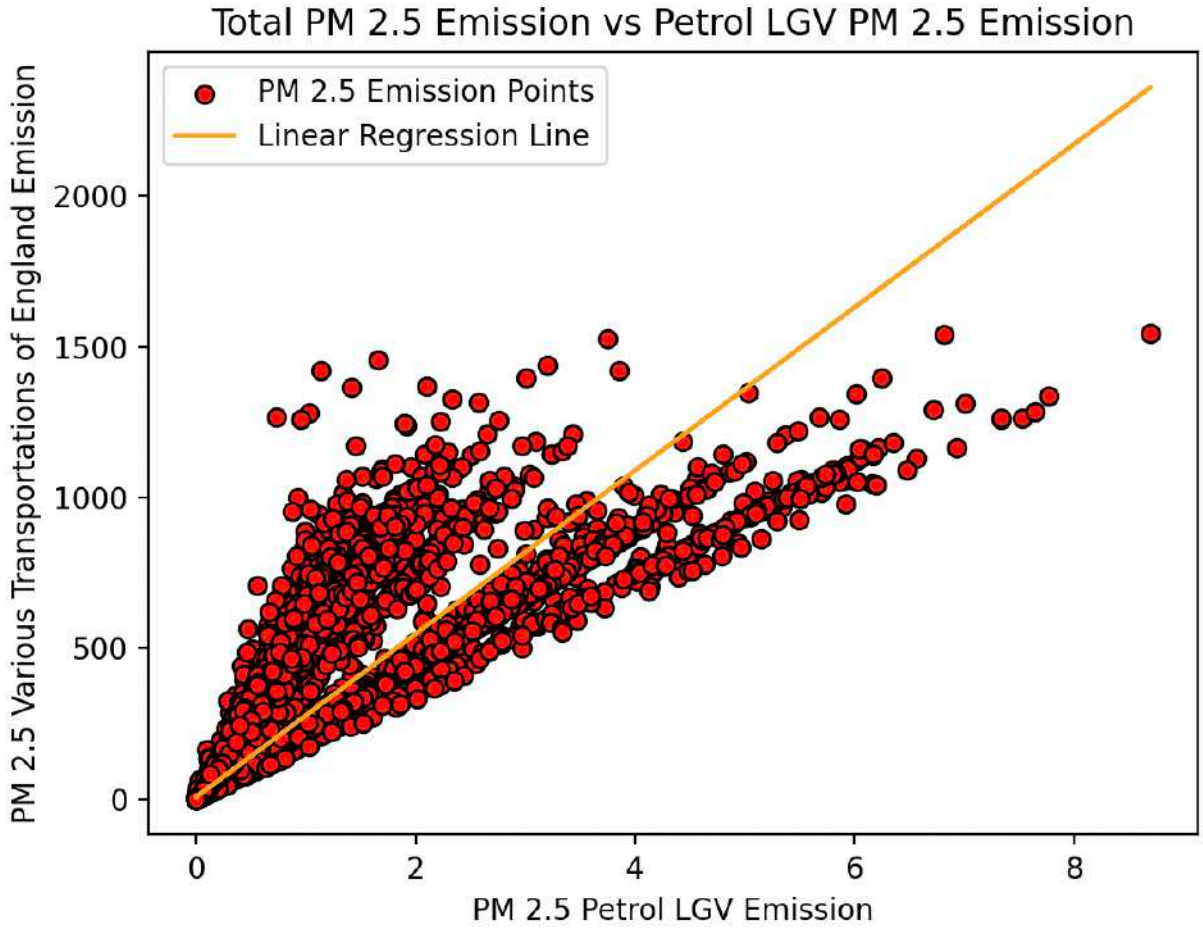
linear regression 15



linear regression 16

```
In [6]: import matplotlib.pyplot as plt
        from scipy import stats
        import numpy as np
        from sklearn.linear_model import LinearRegression
        %matplotlib notebook

        df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
        x=df['Petrol LGV']
        y=df['Total']

        slope, intercept, r, p, std_err = stats.linregress(x, y)

        def myfunc(x):
          return slope * x + intercept

        mymodel = list(map(myfunc, x))

        plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
        plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
        plt.xlabel('PM 2.5 Petrol LGV Emission')
        plt.ylabel('PM 2.5 Various Transportations of England Emission')
        plt.title('Total PM 2.5 Emission vs Petrol LGV PM 2.5 Emission')
        plt.legend()
        plt.show()
```

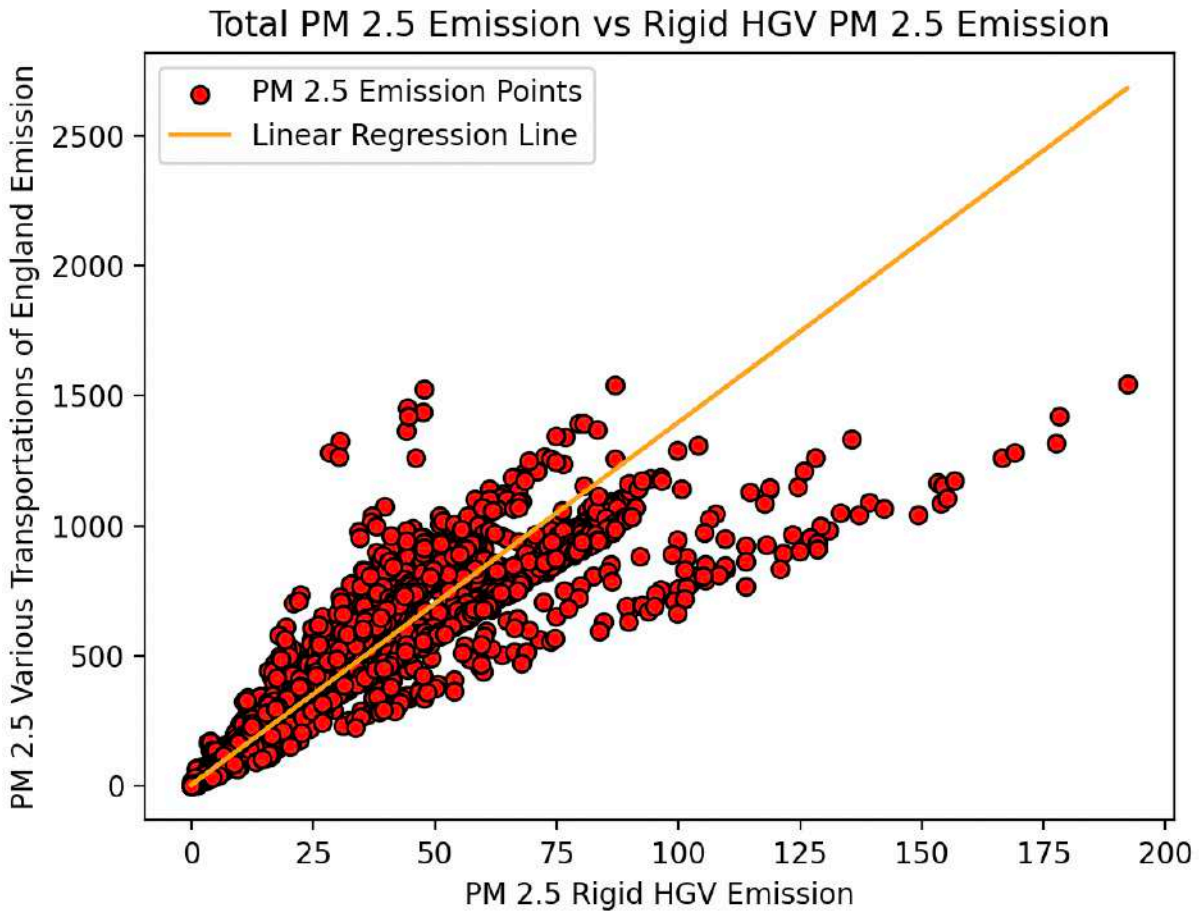
linear regression 17



linear regression 18

```
In [18]: import matplotlib.pyplot as plt
         from scipy import stats
         import pandas as pd
         from sklearn.linear_model import LinearRegression


         %matplotlib notebook

         df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
         x=df['Rigid HGV']
         y=df['Total']

         slope, intercept, r, p, std_err = stats.linregress(x, y)

         def myfunc(x):
           return slope * x + intercept

         mymodel = list(map(myfunc, x))

         plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
         plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
         plt.xlabel('PM 2.5 Rigid HGV Emission')
         plt.ylabel('PM 2.5 Various Transportations of England Emission')
         plt.title('Total PM 2.5 Emission vs Rigid HGV PM 2.5 Emission')
         plt.legend()
         plt.show()
```

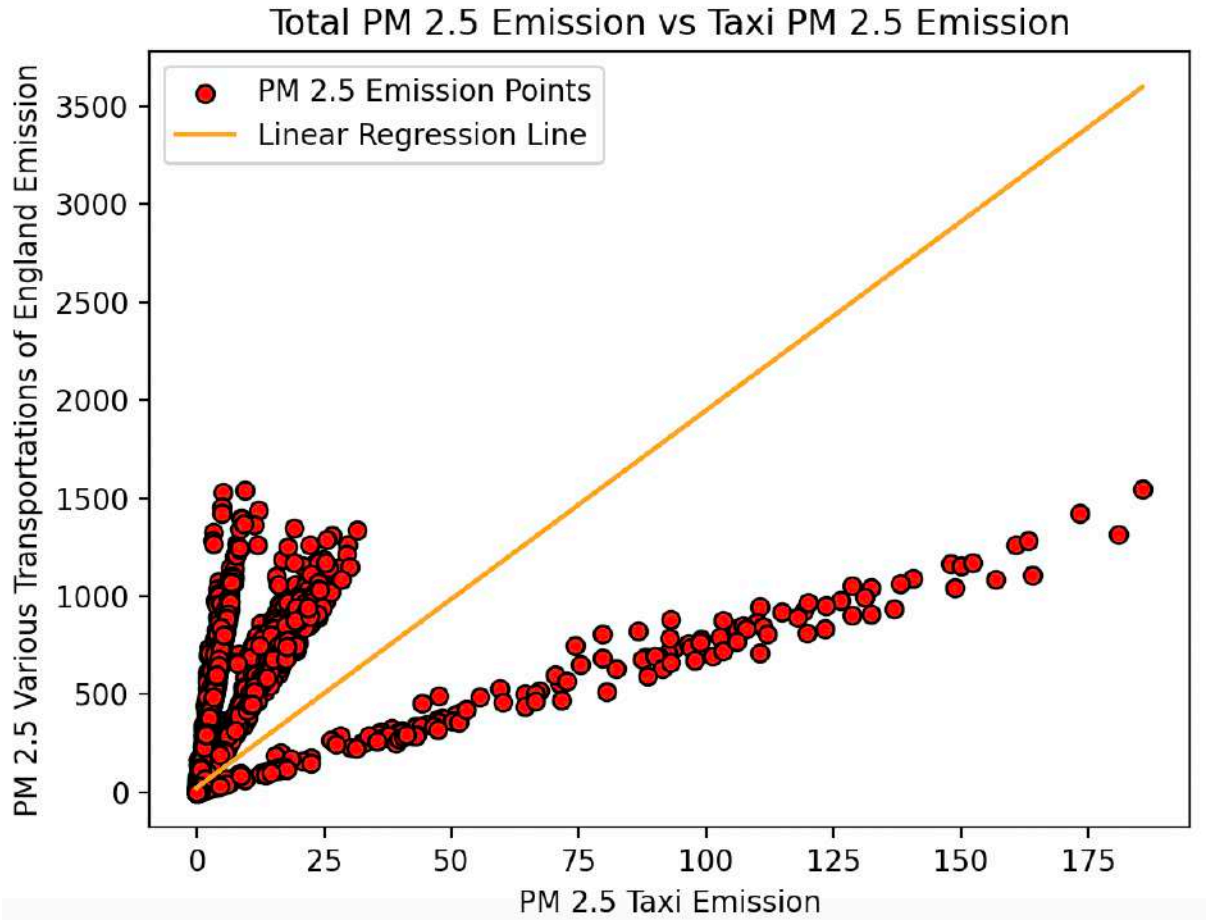
linear regression 19



linear regression 20

```
In [5]: import matplotlib.pyplot as plt
        from scipy import stats
        import pandas as pd
        from sklearn.linear_model import LinearRegression
        %matplotlib notebook

        df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
        x=df['Taxi']
        y=df['Total']

        slope, intercept, r, p, std_err = stats.linregress(x, y)

        def myfunc(x):
          return slope * x + intercept

        mymodel = list(map(myfunc, x))

        plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
        plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
        plt.xlabel('PM 2.5 Taxi Emission')
        plt.ylabel('PM 2.5 Various Transportations of England Emission')
        plt.title('Total PM 2.5 Emission vs Taxi PM 2.5 Emission')
        plt.legend()
        plt.show()
```

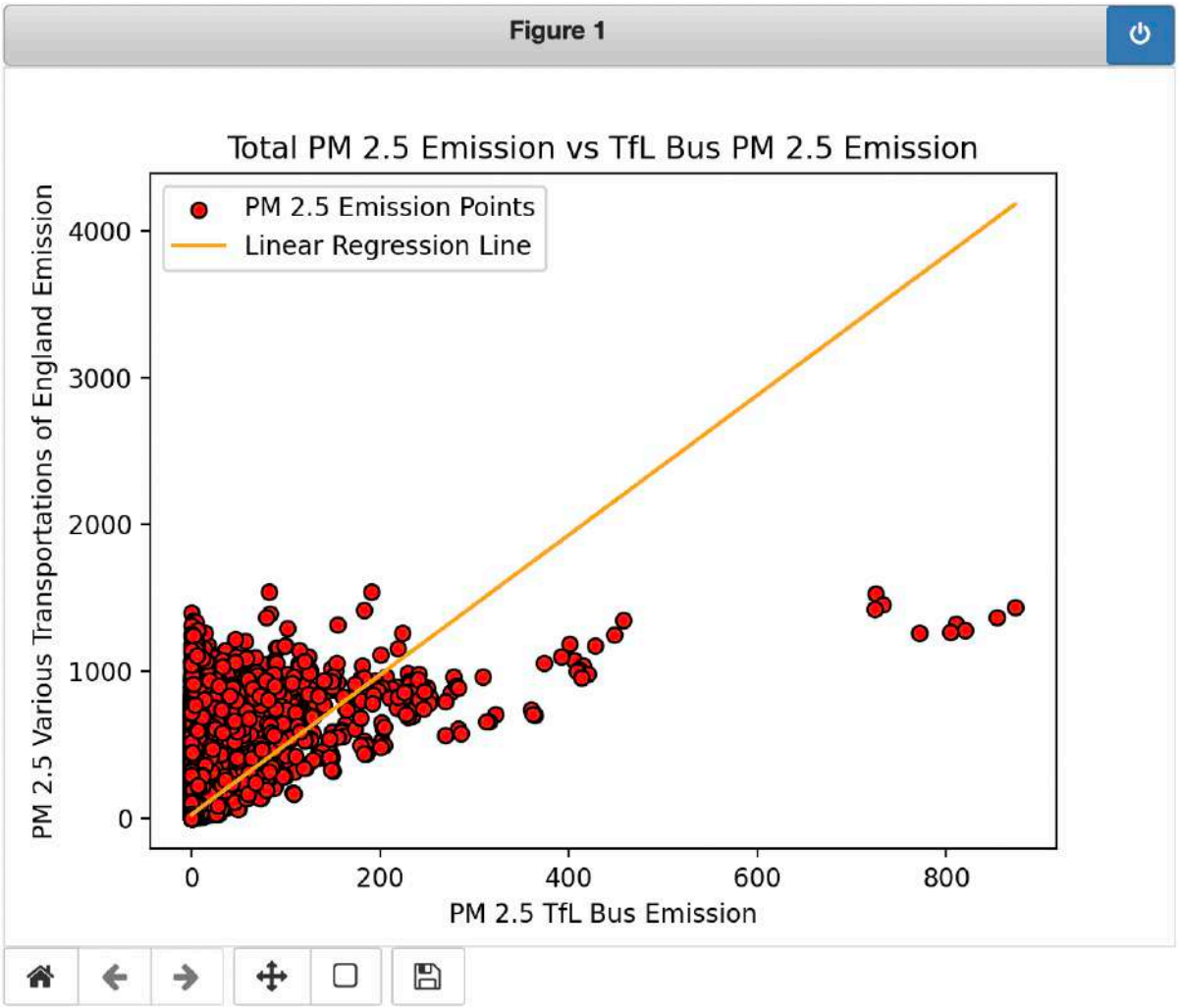
linear regression 21



linear regression 22

In [20]:
```
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from sklearn.linear_model import LinearRegression


%matplotlib notebook

df=pd.read_excel('minor roads emission 2010,13&16..xlsx')
x=df['TfL Bus']
y=df['Total']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y,label='PM 2.5 Emission Points',color='red',edgecolor='k',)
plt.plot(x, mymodel,color='orange',label='Linear Regression Line')
plt.xlabel('PM 2.5 TfL Bus Emission')
plt.ylabel('PM 2.5 Various Transportations of England Emission')
plt.title('Total PM 2.5 Emission vs TfL Bus PM 2.5 Emission')
plt.legend()
plt.show()
```

linear regression 23



linear regression 24

Appendix for model assessment:

k-means clustering results of Lowest to highest PM 2.5 emission combined (2010,13&16):

1. For electric light goods vehicle, the mean of purple cluster is x (-0.0001) & y (-0.25). The mean of yellow cluster is x (-0.0004) & y (4.29).
2. For electric car and total PM 2.5 emission, the mean of purple cluster is x (-0.0001) &y (-0.28). the mean of yellow cluster is x (-0.32) &y (4.27).
3. For TfL bus emission and total PM 2.5 emission, the mean of purple cluster is x (-0.09) &y (-0.19), the mean of green cluster is x (1.45) &y (4.2), the mean of yellow cluster is x (18.9) &y (6.3).
4. For cab emission and total PM 2.5 emission the mean of purple cluster is x (-0.13) & y (-0.23). the mean of yellow cluster is x (2.44) & y (4.44).
5. For non-TfL bus and coach, the mean of purple cluster is x (-0.14) & y (-0.2). the mean of yellow cluster is x (3.4) & y (4.4).
6. For motorcycle and total PM 2.5 emissions, the mean of purple cluster is x (-0.15) &y (-0.26), yellow cluster is x (3.66) & y (4.41).
7. For petrol light goods vehicle, the mean of purple cluster is x (-0.16) & y (-0.19). The mean of yellow cluster is x (4.05) & y (4.4).
8. For rigid heavy goods vehicle, the mean of purple cluster is x (-0.19) & y (-0.2). The mean of yellow cluster is x (4.2) & y (4.3).
9. For artic heavy goods vehicle, the mean of purple cluster is x (-0.19) & y (-0.2). the mean of yellow cluster is x (4.2) & y (4.4).
10. For diesel light goods vehicle, the mean of purple cluster is x (-0.19) & y (-0.30). the mean of yellow cluster is x (4.31) & y (4.33).
11. For diesel car and total PM 2.5 emissions, the purple clusters have a mean of x (-0.21) & y (-0.21). The mean of yellow clusters is x (4.3) & y (4.3). For petrol car and total PM 2.5 emissions, the mean of purple cluster is x (-0.2169) &y (-0.2164). The mean of yellow cluster is x (4.346) &y (4.347).
12. Y-axis is total emissions of transport (dependent variable) & X is different vehicles (independent variable).

k-means clustering model assessment 1

Linear regression results of Total PM 2.5 Emission Ranges combined (2010,13&16).

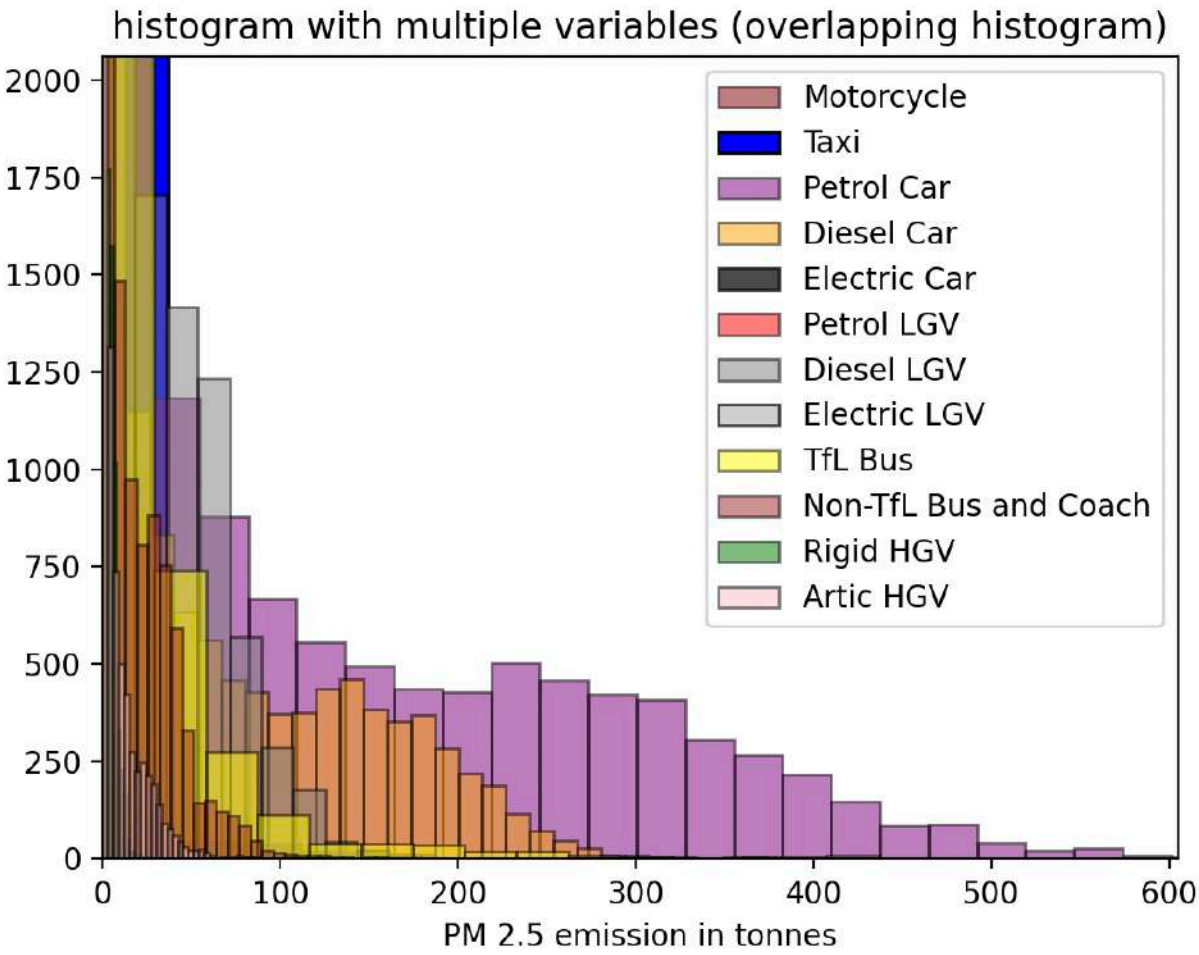
1. Electric car Emission ranges thin from 0 to -0.0003 tonnes in 3 years with -0.0004 to -0.0006 being datapoints which are outliers.
2. Electric light goods vehicles emission ranges dense from 0 to 0.0001 tonnes in 3 years. Where emission range from 0.0001 to 0.0023 is outlier emission datapoints.
3. Petrol light goods vehicles emission ranges dense from 0 to 6.4 tonnes in 3 years with 6.4 to 9 containing outlier data point emission of PM 2.5.
4. Artic heavy goods vehicle emission ranges dense from 0 to 15 tonnes in 3 years. Outlier datapoint emission ranges from 15 to 29 tonnes in 3 years.
5. Motorcycles emission Ranges thin 0 to 40 tonnes in a mix of 3 years of time, with some data points being outliers ranging from 40 to 60.
6. Non-TFL bus and coach emission ranges densely from 0 to 60 tonnes in 3 years, with 60 to 80 contain outlier datapoints.
7. Rigid heavy goods vehicle emission ranges dense from 0 to 100 tonnes in 3 years. Outlier datapoints of emission ranges from 100 to 190 tonnes in 3 years.
8. Taxi emission Ranges thin from 0 to 125 tonnes in 3 years.
9. Diesel light goods vehicle emission ranges densely from 0 to 130 tonnes in 3 years. Where emission ranging 130 to 180 are outlier datapoints.
10. TFL bus Emissions ranges dense from 0 to 250 tonnes in 3 years, where datapoints ranging from 300 to 900 contain outlier datapoints.
11. Diesel car Emissions ranges densely from 0 to 300 tonnes in 3 years, with 300 to 400 having data points with outliers for x axis.
12. Petrol car emission ranges densely from 0 to 600 tonnes in 3 years.600 to 800 data point being outliers.
13. Y axis will remain the same for all the plots that is the Total Transport Emissions, which ranges from 0 to 1600 tonnes in 3 years. X- axis is different modes of Transport like petrol, electric car etc.

linear regression model assessment 1

Appendix for histogram analysis:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib notebook


df=pd.read_excel('minor roads emission 2010,13&16..xlsx')

# plotting three histograms on the same axis
plt.hist(df['Motorcycle'],bins = 5, alpha = 0.5,
         color = 'maroon',edgecolor='k')
plt.hist(df['Taxi'],bins = 5, alpha = 1,
         color = 'blue',edgecolor='k')
plt.hist(df['Petrol Car'],bins = 30, alpha = 0.5,
         color = 'purple',edgecolor='k')
plt.hist(df['Diesel Car'],bins = 30, alpha = 0.5,
         color = 'orange',edgecolor='k')
plt.hist(df['Electric Car'],bins = 1, alpha = 0.7,
         color = 'black',edgecolor='k')
plt.hist(df['Petrol LGV'],bins = 10, alpha = 0.5,
         color = 'red',edgecolor='k')
plt.hist(df['Diesel LGV'],bins = 10, alpha = 0.5,
         color = 'grey',edgecolor='k')
plt.hist(df['Electric LGV'],bins = 1, alpha = 0.7,
         color = 'silver',edgecolor='k')
plt.hist(df['TfL Bus'],bins = 30, alpha = 0.5,
         color = 'yellow',edgecolor='k')
plt.hist(df['Rigid HGV'],bins = 30, alpha = 0.5,
         color = 'brown',edgecolor='k')
plt.hist(df['Artic HGV'],bins = 15, alpha = 0.5,
         color = 'green',edgecolor='k')
plt.hist(df['Non-TfL Bus and Coach'],bins = 30, alpha = 0.5,
         color = 'pink',edgecolor='k')

plt.title("histogram with multiple \
variables (overlapping histogram)")
plt.legend(['Motorcycle','Taxi','Petrol Car','Diesel Car','Electric Car','Petrol LGV','Diesel LGV','Electric LGV','T
plt.xlabel('PM 2.5 emission in tonnes')
plt.show()
```

histogram 3



histogram 4

Reference

D. Massey, A. Kulshrestha, J. Masih, and A. Taneja, "Seasonal trends of PM10, PM5.0, PM2.5 & PM1.0 in indoor and outdoor environments of residential homes located in north-central india," *Building and Environment*, vol. 47, pp. 223–231, 2012.

F. Huang, B. Pan, J. Wu, E. Chen, and L. Chen, "Relationship between exposure to PM2.5 and lung cancer incidence and mortality: A meta-analysis," *Oncotarget*, vol. 8, no. 26, pp. 43322–43331, 2017.

J. Meng, J. Liu, Y. Xu, D. Guan, Z. Liu, Y. Huang, and S. Tao, "Globalization and pollution: Tele-connecting local primary PM 2.5 emissions to global consumption," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 472, no. 2195, p. 20160380, 2016.

M. Tessum, S. Anenberg, Z. Chafe, D. Henze, G. Kleiman, I. Kheirbek, J. Marshall, and C. Tessum, "Sources of ambient PM2.5 exposure in 96 global cities," 2022.

M. Yang, Y.-M. Guo, M. S. Bloom, S. C. Dharmagee, L. Morawska, J. Heinrich, B. Jalaludin, I. Markevychd, L. D. Knibbsf, S. Lin, S. Hung Lan, P. Jalava, M. Komppula, M. Roponen, M.-R. Hirvonen, Q.-H. Guan, Z.-M. Liang, H.-Y. Yu, L.-W. Hu, B.-Y. Yang, X.-W. Zeng, and G.-H. Dong, "Is PM1 similar to PM2.5? A new insight into the Association of PM1 and PM2.5 with children's lung function," *Environment International*, vol. 145, p. 106092, 2020.

S. E. Alexeeff, N. S. Liao, X. Liu, S. K. Van Den Eeden, and S. Sidney, "Long-term PM 2.5 exposure and risks of ischemic heart disease and stroke events: Review and meta-analysis," *Journal of the American Heart Association*, vol. 10, no. 1, 2021.

Sahu, S.K. and Kota, S.H., 2017. Significance of PM2. 5 air quality at the Indian capital. *Aerosol and air quality research*, *17*(2), pp.588-597.

Baker, K.R. and Foley, K.M., 2011. A nonlinear regression model estimating single source concentrations of primary and secondarily formed PM2. 5. *Atmospheric Environment*, *45*(22), pp.3758-3767.

Zhao, R., Gu, X., Xue, B., Zhang, J. and Ren, W., 2018. Short period PM2. 5 predictions based on the multivariate linear regression model. *PloS one*, *13*(7), p.e0201011.

Karimian, H., Li, Q., Wu, C., Qi, Y., Mo, Y., Chen, G., Zhang, X. and Sachdeva, S., 2019. Evaluation of different machine learning approaches to forecasting PM2. 5 mass concentrations. *Aerosol and Air Quality Research*, *19*(6), pp.1400-1410.

Liu, H. and Chen, C., 2020. Prediction of outdoor PM2. 5 concentrations based on a three-stage hybrid neural network model. *Atmospheric Pollution Research*, *11*(3), pp.469-481.