Global warming Data Science

Computing

Josh Arrowsmith
Student ID: Q13713574

January 17, 2020

Contents

1 Abstract

In this report I am going to describe a data science project related to global warming that I undertook by describing step by step by step what I did and why I did it.

2 Data Gathering

The first data-set I chose is related to “Car fuel consumption’s and emissions”. This is because it directly contained data about car emissions by model with the bonus of seeing how fuel efficient the car model is by looking at the fuel consumption numbers. This means I can see how common it is for a car model to both be fuel inefficient as well as have higher emissions. The two data fields I will be grabbing is ”CO2 emissions” which is the amount of CO2 emissions the car model makes (measured in grams per kilometre) and ”Metric combined fuel consumption” which is the fuel consumption per 100km made from the average of a test of average car use (referred to as Metric urban fuel consumption) and extreme car usage (referred to as Metric extra-urban fuel consumption) which is measured in litres per 100km. (Directgov 2013)

Now that I have selected my data-sets, I will have to load them into python so I can clean it, put it in a data store of some kind and then later pull it from the data sets for processing, analysis and visualisation so I can learn something from it. I can do this by writing a Python script using a run-time data management library called Pandas and referred to as panda. This library will allow me to load in my CSV files (as it comes and uses the CSV library) and then put the data from the file into an object called a data frame (abbreviated in my code as df).

```
import pandas as panda

alldata_df = pd.read_csv('CarFuelConsumptionEmissionsData.csv', sep=',')

inefficient_df = alldata_df[alldata_df['combined_metric'] >= 1000]

inefficient_count = inefficient_df['combined_metric'].value_counts()

highemissions_df = alldata_df[alldata_df['co_emissions'] >= 1000]

highemissions_count = highemissions_df.shape[0] - 1

inefficient_highemissions_df = alldata_df[(alldata_df['combined_metric'] >= 1000)
& (alldata_df['co_emissions'] >= 1000)]

inefficient_highemissions_count = inefficient_highemissions_df['year'].value_counts()

total_records = alldata_df.shape[0] - 3
```

3 Data Preparation

Now that we have loaded in our data-sets from their files using Python, we need to prepare or ‘clean’ the data of any anomalies (data which may not make sense or may throw the data mining way off track). We can do this by removing rows which has null in the fields we are looking at are are key to the research so in my case removing all rows which have nothing in either ”combined_metric” or ”co_emissions”. As stated by the source of my first data-set, “Note that the resulting data-set does not include all fields in the original data, only those deemed more relevant.” (Datahub 2020) Which means that the data has already been in cleaned in some ways manually by the people hosting and documenting the data-set. Despite this I think it is good practice to clean the data still due to it leading to the data mining having more accurate results. So to get rid of the anomalies in my data-set, I set upper and lower ranges on both the "combined_metric" and "co_emissions" fields.

For "co_emissions" I set the lower range to be 1 as there were a few rows (12 rows) which were really low decimal numbers as well as one negative number, both of which would throw off calculations at the end of our mining (especially the negative number as it doesn't make too much sense).

For "co_emissions" I set the upper range to be 2500 as there were a few rows (9 rows) which are in the 70 thousand range which must be industrial vehicles or really really bad cars so they need to be removed otherwise there would be a big bias towards the high emissions end of the calculations.

For "combined_metric" I set the lower range to be 1 as there were no rows bellow 1 but i wanted to add this for future iterations of the same data for the same reasons as setting the lower range of "co_emissions" to 1.

For "combined_metric" I set the upper range to be 25 as there were a few rows (4 rows) which had dramatically higher values (44 and 35.8 are much higher than the range of values of 25 to 1) and if these 4 values were to be included it would again make a bias towards the higher end of the fuel consumption numbers.

To automate the process of data cleaning/preparation, we can use the following Python script I have written:

```
cleaned_df = alldata_df.dropna(subset=['combined_metric', 'co_emissions'])

cleaned_df = cleaned_df.loc[cleaned_df['co_emissions'] >= 1]

cleaned_df = cleaned_df.loc[cleaned_df['co_emissions'] <= 2500]

cleaned_df = cleaned_df.loc[cleaned_df['combined_metric'] >= 1]

cleaned_df = cleaned_df.loc[cleaned_df['combined_metric'] <= 25]
```

Now that we have cleaned our data, we can put it our data store and for this project I will be using mongoDB as NoSQL databases are more useful for data science due there being no complicated and semi pointless (for this purpose anyway) relational model (Sugam Sharma 2015). For data science to be fast from data selection to visualisation, we need putting in the data to a database to be fast and easy to set up and mongoDB supports this well due to its file data storage structure and easy to set up database structure (Sugam Sharma 2016). Once we have that set up, we can automate the process of putting our data into a mongoDB database by using the python library "PyMongo". This library can be used to get mongoDB database then take collections (mongoDB's name for their equivalent of a "table") and from there we can pull and push data to the collections we need. So normally we would take all of the cleaned data and then make a new collection for it then push it to the new collection and when we need it for mining it can easily be accessed by connecting to the mongoDB client and grabbing fields we need.

Before we begin data mining, we need to work out a few values like the total number of rows in the cleaned data-set, the number of fuel inefficient cars, the number of high emission cars and the number of fuel inefficient and high emission cars. I worked out by manually looking at the ranges of fuel efficiency as well as doing research, a car that produces more than 250 grams per km is considered a high emissions car. (BuyaCar 2018). This article states 108g/km is bad but i wanted to have a value a fair amount higher as we are looking for the bad end of the emissions spectrum.

I also found out by manually looking at the ranges of fuel consumption as well as doing research, a car that uses more than about 12 litres of fuel per 100km is a fuel inefficient car which is about 8.3 km per litre. I chose to look for 12 l/100km or more because the article suggests that "less than 6.0 litres/100km are considered to have 'good' MPG" so I wanted to double a good MPG to get a round-about "bad mileage" for a car (May 2016)

```
inefficient_df = cleaned_df[cleaned_df['combined_metric'] >= 12]

inefficient_count= len(inefficient_df) - 1

highemissions_df = cleaned_df[cleaned_df['co_emissions'] >= 250]

highemissions_count = len(highemissions_df) - 1

inefficient_highemissions_df = cleaned_df[(cleaned_df['combined_metric'] >= 12)
 & (cleaned_df['co_emissions'] >= 250)]

inefficient_highemissions_count = len(inefficient_highemissions_df) - 1

total_records = len(cleaned_df) - 1
```

4 Data Exploration

Now that we have prepared and cleaned our data of most visible anomalies, we can move onto mining the data for any patterns and trends so we can pull some meaning, connections and understanding from the data and how it links to the bigger picture of global warming. To do this I will be using several techniques.

The first we will be doing some Association Rule Mining using Apriori Algorithm. This is where we use some algebraic algorithms to work out values which can tell us things like how often one value comes up in the data set with another (referred to as Support) calculated using the bellow algorithm:

Support = Frequency of A and B together / Total rows in the data-set (Hipp, Güntzer & Nakhaeizadeh 2000)

The next value is called Confidence which will tell us how often A and B occur together compared to how often A comes up overall:

Confidence = Support of a, b / Support of a (Michael Hahsler 2005)

We then calculate Lift which shows us how strong the rule is as the higher the lift the more likely these is a link between A and B rather than it being random chance:

Lift = Support of a,b / Support of a * Support of b (Maitra 2019)

There is another way to calculate Lift:

Lift2 = Confidence of a,b / Support of b (Provost & Fawcett 2013)

The last value is called Conviction which represents how much more often the rule is found to be incorrect if the connection between values A and B was just random chance. For example, if we got a conviction value of 1.6 the rule would be wrong 1.6 times more (60% more) if the rule was chosen randomly. This can be used almost as a frequency that the rule makes an incorrect prediction.

Conviction = 1 minus Support of b / 1 minus Confidence of a,b (Brin, Motwani, Ullman & Tsur 1997)

I have made pseudo logic of what I am trying to prove with my first data-set: Let A be the fact that a car model is fuel inefficient and let B be a car model which has high emissions; support shows how common it is to have a car model which is both fuel inefficient and high in emissions out of the total number of car models in our data-set. Confidence compares frequency of a car which is both fuel inefficient and has high emissions to the frequency of a fuel inefficient car. Lift shows the frequency of a high emissions, fuel inefficient car is verses how common a high emissions car is. Conviction shows me how good or bad my assumption is.

Now this can be a very time consuming manually so I have made a python script to do this automatically:

```
# Support {a, b} = transactions with {a, b} / total transactions (fraction of a,b
 transactions)
support_rule = inefficient_highemissions_count/total_records

support_inefficient = inefficient_count/total_records

support_highemissions = highemissions_count/total_records

# Confidence = transactions with {a, b} / transactions with {a}
confidence = inefficient_highemissions_count/inefficient_count

# Lift method 1 = support of {a,b} / ( support of {a} * support  {b} )
lift = support_rule/(support_inefficient*support_highemissions)

# Lift method 2 = confidence / support of {b}
lift2 = confidence / support_highemissions
print("Lift of the rule (Method 2): " + str(lift2))

# conviction = ( 1 - support of {b} ) / ( 1 - confidence )
conviction = (1 - support_highemissions) / (1 - confidence)
```

So when i ran my code I found the values bellow:
Support of my rule: 0.12790594695124652
Support of fuel inefficient cars: 0.14836381348802197
Support of high emission cars: 0.7285568790683258
Confidence: 0.8621101328159976
Lift method 1: 1.1833120482212165
Lift method 2: 1.1833120482212165
Conviction: 1.9685501659774334

5 Data Modelling and Visualisation

Now that we have a way of analysing the data, it is a good idea to make a model based upon it so we can see what would happen in the future if we don't do anything about the problem or the link I have found. This will allow to see the consequences if we don't do something about it and will allow us in the future to see if things get better once we start helping solve the issue or at least take action on the link I found. To do this I will make a visualisation in which we can make a prediction for potential

future values using its trend visually.
Now the method I will use to visualise what I have found from the data will be using a graph drawing library called Matplotlib. This will allow me to turn the mined data from raw numbers which are harder to understand into a visual graph which can be much more easily understood as well as easier to compare data within. I will be making a bar chart to show of what I found from the first method of data mining used (Apriori) which will show the supports, confidence, lift (both methods) and conviction of the rule I have made:

```
import matplotlib.pyplot as graphs

#graph 1 bar chart
fig = graphs.figure()
bargraph1 = fig.add_subplot()

bargraph1.bar(["Rule\nSupport", "Inefficient\nSupport", "High\nEmissions\nSupport",
 "Confidence","Lift 1","Lift 2", "conviction"],[support_rule,support_inefficient
  ,support_highemissions,confidence,lift,lift2,conviction], width=0.5,
  bottom=None,align='center', data=None,)

fig.savefig("Graph1.png")
```

And here is the Graph which the code produces:

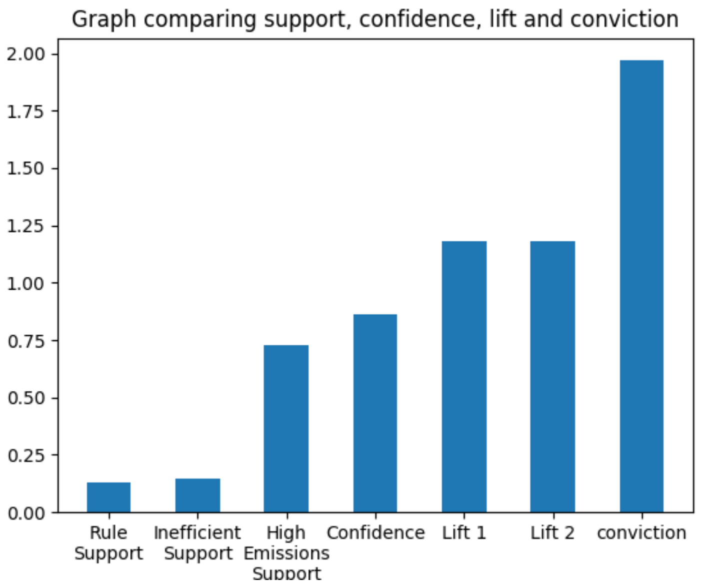

Figure 1: Graph comparing support, confidence, lift and conviction

This shows us by eye that our confidence is very high in comparison to our support which means that it is uncommon to see cars like the one my rule proves but it is very common to have both traits on the same car. This also shows me using lift and conviction that in general my rule is very common but cars which are both are rare in comparison to the amount of total cars.
So the modelling show us that while not a big problem in comparison to the amount of car models

that exist, cars being fuel inefficient still contributes to a car being higher in emissions which results in an increase in overall CO2 emissions from cars. If we ran this model in the future and we did nothing about the problem then the model would show that CO2 emissions from fuel inefficient cars would get worse. While it would show that if we did do something about it then overall the amount of high emission cars would reduce and that would in turn reduce car CO2 emissions overall.

I also wanted to make a visualisation of all of the found totals from my data-set so we can see the comparison of the cars which meet my rule verses the total cars as well as the the cars which meet my rule verses the total amount of fuel inefficient cars:

```
#graph 2 horizontal bar chart

fig = graphs.figure()
hbargraph1 = fig.add_subplot()

hbargraph1.barh(["Total Cars", "Total\nHigh\nEmissions\nCars", "Total\nInefficient\n
Cars","Total\nInefficient\nHigh\nEmissions\nCars"],[total_records,highemissions_count,
inefficient_count,inefficient_highemissions_count], height=0.5, left=None,
align='center', data=None)
graphs.xlabel("Number of cars")

fig.savefig("Graph2.png")
```

And here is the Graph which the code produces:

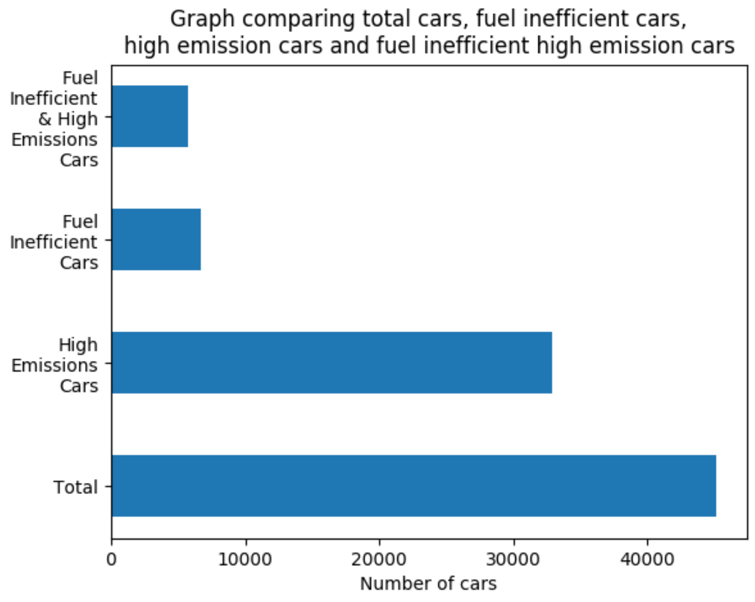

Figure 2: Graph comparing total cars, fuel inefficient cars ,high emission cars and fuel inefficient high emission cars

This shows the big difference between the number of cars in my rule to the total as well as compared to the number of fuel inefficient cars

I wanted to make one last graph which showed the CO2 emissions against the fuel efficiency of cars that fit into my rule. I can do this with the code bellow:

```
#graph 3 scatter chart

fig = graphs.figure()
scattergraph1 = fig.add_subplot()

graphs.title("Graph comparing  and fuel efficiency of cars\nthat fit into my rule")
scattergraph1.scatter(inefficient_highemissions_df.co_emissions,
inefficient_highemissions_df.combined_metric)
graphs.xlabel("CO2 emissions (Grams per Kilometre)")
graphs.ylabel("Fuel efficiency (Litres per 100 Kilometre)")

fig.savefig("Graph3.png")
```

And here is the Graph which the code produces:

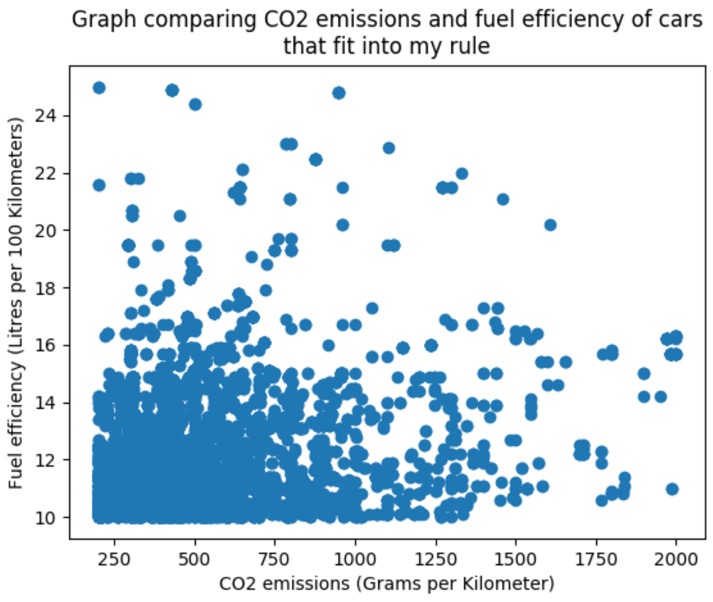

Figure 3: Graph comparing CO2 emissions and fuel efficiency of cars that fit into my rule

This shows that while pretty spread out, if the line of best fit to show a trend is used and averaged to keep an even number on each side, it has a positive correlation between the two, supporting my theory

6 Evaluation

So overall, from my data mining and visualisation as well running my data model I can infer that while it is not common to have a car model which is both fuel inefficient and have high emissions when compared to the overall number in the data-set, there is a clear link between fuel inefficiency and high emissions. the confidence value shows that in 80% of cases where a car was fuel inefficient, it also had high emission values. Though my conviction shows that trying to say that every high emissions car is also fuel inefficient is more than 300% wrong, my value for lift (method 2) as it shows that my rule is right more than it is wrong.

So based on my findings, I think to help tackle the issue of global warming, car manufactures need to reduce the car models they make which are fuel inefficient as this would directly reduce the number of high emission cars.

References

Arrowsmith, J. (2020), 'Global warming data science github repo'. (Accessed on 01/17/2020).
URL: *https://github.com/Josh-Arrowsmith/Global_Warming_Data_Science*

Brin, S., Motwani, R., Ullman, J. D. & Tsur, S. (1997), 'Dynamic itemset counting and implication rules for market basket data', *SIGMOD Rec.* **26**(2), 255–264.
URL: *https://doi.org/10.1145/253262.253325*

BuyaCar (2018), 'Co2 emissions and g/km meaning — buyacar'. (Accessed on 01/17/2020).
URL: *https://www.buyacar.co.uk/cars/economical-cars/low-emission-cars/337/co2-emissions-and-gkm-meaning*

Datahub (2020), 'Car fuel consumptions and emissions 2000-2013 - the Datahub'. [Online; accessed 15. Jan. 2020].
URL: *https://old.datahub.io/dataset/car-fuel-consumptions-and-emissions*

Directgov (2013), 'Car fuel data, co2 and vehicle tax tools'. (Accessed on 01/15/2020), "Note: to get to the glossary where i got the descriptions from, you have to fill in the from prompted in the URL with any data then click the 'Show Glossary' button".
URL: *https://carfueldata.vehicle-certification-agency.gov.uk/search-new-or-used-cars.aspx*

Hipp, J., Güntzer, U. & Nakhaeizadeh, G. (2000), 'Algorithms for association rule mining — a general survey and comparison', *SIGKDD Explor. Newsl.* **2**(1), 58–64.
URL: *https://doi.org/10.1145/360402.360421*

Maitra, S. (2019), 'Association rule mining using market basket analysis: Knowledge discovery in database using python'. [Online; accessed 15. Jan. 2020].
URL: *https://towardsdatascience.com/market-basket-analysis-knowledge-discovery-in-database-simplistic-approach-dc41659e1558*

May, M. (2016), 'What counts as 'good' mpg nowadays? · thejournal.ie'. (Accessed on 01/17/2020).
URL: *https://www.thejournal.ie/dear-driver-what-is-good-mpg-3150884-Dec2016/*

Michael Hahsler, Bettina Grün, K. H. (2005), 'Algorithms for association rule mining — a general survey and comparison'.
URL: *https://dl.acm.org/doi/pdf/10.1145/360402.360421?download=true*

Provost, F. & Fawcett, T. (2013), *Data Science for Business: What you need to know about data mining and data-analytic thinking*, " O'Reilly Media, Inc.".

Sugam Sharma, Ritu Shandilya, S. P. A. M. (2016), 'Leading nosql models for handling big data: a brief review: International journal of business information systems: Vol 22, no 1'. (Accessed on 01/17/2020).

Sugam Sharma, Udoyara Sunday Tim, S. G. J. W. R. S. S. K. P. (2015), ‘Classification and comparison of nosql big data models: International journal of big data intelligence: Vol 2, no 3’. (Accessed on 01/17/2020).
URL: *https://www.inderscienceonline.com/doi/abs/10.1504/IJBDI.2015.070602*

My personal contributions to this paper include the Python scripts made to load in the data-set; clean, mine, explore, model and visualise the data. See figures 1 & 2. For the code, see the GitHub Repo (Arrowsmith 2020) or check bellow in the Appendix

7 Appendix

7.1 Full Python script code

```
import pandas as panda
import matplotlib.pyplot as graphs

mode = input("1 for Normal mode or 2 for debug mode \n")

alldata_df = panda.read_csv('CarFuelConsumptionEmissionsData.csv', sep=',', dtype={'
 file' : str, 'year': int, 'manufacturer' : str, 'model' : str, 'description' : str,
 'manufacturer' : str, 'manufacturer' : str})

alldata_df = alldata_df[['co_emissions','combined_metric', 'manufacturer', 'model',
 'year']]

cleaned_df = alldata_df.dropna(subset=['combined_metric', 'co_emissions'])

cleaned_df = cleaned_df.loc[cleaned_df['co_emissions'] >= 1]

cleaned_df = cleaned_df.loc[cleaned_df['co_emissions'] <= 2500]

cleaned_df = cleaned_df.loc[cleaned_df['combined_metric'] >= 1]

cleaned_df = cleaned_df.loc[cleaned_df['combined_metric'] <= 25]

inefficient_df = cleaned_df[cleaned_df['combined_metric'] >= 10]

inefficient_count= len(inefficient_df) - 1

highemissions_df = cleaned_df[cleaned_df['co_emissions'] >= 200]

highemissions_count = len(highemissions_df) - 1

inefficient_highemissions_df = cleaned_df[(cleaned_df['combined_metric'] >= 10) &
 (cleaned_df['co_emissions'] >= 200)]

inefficient_highemissions_count = len(inefficient_highemissions_df) - 1

total_records = len(cleaned_df) - 1
```

```
#print("Full dataset:")
#print(alldata_df)
#print("inefficient_highemissions_df:")
#print(inefficient_highemissions_df)
print("Total records: " + str(total_records))
print("Total fuel inefficient: " + str(inefficient_count))
print("Total high emission: " + str(highemissions_count))
print("Total fuel inefficient, high emission cars: " +
 str(inefficient_highemissions_count))

# Support {a, b} = transactions with {a, b} / total transactions (fraction of a,b
 transactions)
support_rule = inefficient_highemissions_count/total_records
print("Support of rule: " + str(support_rule))

support_inefficient = inefficient_count/total_records
print("Support of fuel inefficient cars: " + str(support_inefficient))

support_highemissions = highemissions_count/total_records
print("Support of high emission cars: " + str(support_highemissions))

# Confidence = transactions with {a, b} / transactions with {a}
confidence = inefficient_highemissions_count/inefficient_count
print("Confidence of the rule: " + str(confidence))

# Lift method 1 = support of {a,b} / ( support of {a} * support  {b} )
lift = support_rule/(support_inefficient*support_highemissions)
print("Lift of the rule (Method 1): " + str(lift))

# Lift method 2 = confidence / support of {b}
lift2 = confidence / support_highemissions
print("Lift of the rule (Method 2): " + str(lift2))

# conviction = ( 1 - support of {b} ) / ( 1 - confidence )
conviction = (1 - support_highemissions) / (1 - confidence)
print("Conviction of the rule: " + str(conviction))

if(mode == "2"):
alldata_df.info()
cleaned_df.info()
inefficient_df.info()
highemissions_df.info()
inefficient_highemissions_df.info()
print(alldata_df.head(5))
print(cleaned_df.head(5))
print(inefficient_df.head(5))
print(highemissions_df.head(5))
print(inefficient_highemissions_df.head(5))

#graph 1 bar(x, height, width=0.8, bottom=None, *, align='center', data=None,
 **kwargs)
```

```
fig = graphs.figure()
bargraph1 = fig.add_subplot()
graphs.title("Graph comparing support, confidence, lift and conviction")
bargraph1.bar(["Rule\nSupport", "Inefficient\nSupport", "High\nEmissions\nSupport",
 "Confidence", "Lift 1", "Lift 2", "conviction"],[support_rule,support_inefficient,
 support_highemissions,confidence,lift,lift2,conviction], width=0.5, bottom=None,
 align='center', data=None,)

fig.savefig("Graph1.png")

#graph 2 barh(y, width, height=0.8, left=None, *, align='center', **kwargs)

fig = graphs.figure()
hbargraph1 = fig.add_subplot()
graphs.title("Graph comparing total cars, fuel inefficient cars,\nhigh emission cars
 and fuel inefficient high emission cars")
hbargraph1.barh(["Total", "High\nEmissions\nCars", "Fuel\nInefficient\nCars",
 "Fuel\nInefficient\n& High\nEmissions\nCars"],[total_records,highemissions_count,
 inefficient_count,inefficient_highemissions_count], height=0.5, left=None,
 align='center', data=None,)
graphs.xlabel("Number of cars")

fig.savefig("Graph2.png")

#scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None,
 vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, *,
 plotnonfinite=False, data=None, **kwargs)[source]

fig = graphs.figure()
scattergraph1 = fig.add_subplot()

graphs.title("Graph comparing CO2 emissions and fuel efficiency of cars\nthat fit
 into my rule")
scattergraph1.scatter(inefficient_highemissions_df.co_emissions,
 inefficient_highemissions_df.combined_metric)
graphs.xlabel("CO2 emissions (Grams per Kilometre)")
graphs.ylabel("Fuel efficiency (Litres per 100 Kilometres)")

fig.savefig("Graph3.png")
```