Practice report

Information Management Technologies

University of Jaén

Computer Science Department

Academic year 2019-20

Web mining and analysis over collected data

Students:

Vukašin Vasiljević

vukasin.vasiljevic@gmail.com

Professor:

José María Serrano Chica

Table of Contents

Introduction

Web Mining is the process of Data Mining techniques to automatically discover and extract information from Web documents and services. The main purpose of web mining is discovering useful information from the World-Wide Web and its usage patterns.

Applications of Web Mining:

1. Web mining helps to improve the power of web search engine by classifying the web documents and identifying the web pages.
2. It is used for Web Searching e.g., Google, Yahoo etc and Vertical Searching e.g., FatLens, Become etc.
3. Web mining is used to predict user behavior.
4. Web mining is very useful of a particular Website and e-service e.g., landing page optimization.

Using web mining techniques and libraries, we are able to extract data off some website or possibly multiple websites. In this case, site that is used (www.polovniautomobili.com) to extract information from used cars that are on sale. Each add has set of information related to car. Those specification can be accessed through html content and it can be extracted using some specific tool. There are some tools in use today that are capable of extracting information and such tools are: Google Analytics, Web Miner, Similar Web, Majestic, Scrapy, etc.

Tool

Scrapy is part of Python programming language commonly used for web content extraction.

In this case, using *Beautiful Soup* as Python library, I managed to finish all necessary work required. Everything is done in Python programming language, including analysis of extracted data.

Iterating through every brand, every model of a brand, I was able to extract approximately 18000 set of data. Each car has its own specification and here is extracted only essential ones: *brand, model, age, volume, power, mileage, type and price*.

With this parameters we are able to describe most of cars with some details. More detailed version of each car is possible but with more thorough extraction methods and with Python libraries it would take more time to go over each car and each page.

Since web mining is not primary thing in this report but analysis of some given data, code that Is used for extracting data is on this github link: https://github.com/trsavi/Polovni-Automobili-Webscraper

With this dataset we are able to perform analysis of whole webpage content, because it is based on cars, and answer some of the most interesting questions regarding car specifications.

Tool that Is used for these problems are also some libraries in Python, such as *matplotlib and pandas*. I could have used some different environment for Python, for example Jypiter, Anaconda, Python R, but they are all based on Python and mostly everything that you could perform in R, you can do that too in Python, so I stayed with original version of python because we already have data set meant to be used in Python.

Data sources

For this project, data source was content of used cars on sale on mentioned website. The limit of data source is limit of website, how many cars there are, that is the final number of data in dataset. For this project, It is used first two pages on website for every brand and each model of that brand, which is enough to make 18000 sets of information for different cars. It is possible to use all the information from that website but it is time consuming because script must iterate through every page several times extracting information from each add.

After every iteration, information is stored in sql database directly from Python. On picture 1-2 is shown connection with database using python script, creating database and inserting values, respectively.

```
import requests
import urllib.request
from bs4 import BeautifulSoup as bs
import mysql.connector


# function that creates database on local server
def createDB():

    mydb = mysql.connector.connect(
        host='localhost',
        user='root',
        passwd=''
    )
    #database='testDB'

    mycursor = mydb.cursor()

    mycursor.execute("CREATE DATABASE polovniAutomobili")

    # check if database exists
    #mycursor.execute("SHOW DATABASES")
    #for db in mycursor:
    #   print(db)
```

Picture 1: Creating database

```
mycursor = mydb.cursor()

drop = 'DROP TABLE Cars'

mycursor.execute(drop)

mycursor.execute("CREATE TABLE Cars (brend VARCHAR(255), model VARCHAR(255)\
,naziv VARCHAR(255), cena INT(10), godiste INT(10), kilometraza INT(10)\
,gorivo VARCHAR(255), kubikaza INT(10), karoserija VARCHAR(255)\
,snaga VARCHAR(255))")

sqlInsert = "INSERT INTO Cars (brend, model, naziv, cena, godiste, kilometraza,\
 gorivo, kubikaza, karoserija, snaga) \
 VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

# insert cars by brand and model
brands = all_Brands()
try:
    for brand in brands:
        models = get_models(brand)

        for model in models:
            if model==None:
                pass
            else:

                cars = get_cars(brand, model)

                for car in cars:
                    if car==None:
                        pass
                    else:
                        values = tuple(car.values())
                        try:

                            mycursor.execute(sqlInsert, values)
                            mydb.commit()
                        except:
                            pass

except Exception as e:
    #print(values)
    print(e)
    pass
```

Picture 2: Inserting values into database

Development

Using this dataset of information about cars, we can raise some interesting questions. This is a list of some questions and problems that can be related to this dataset:

- How many cars from certain brand there are on mentioned website?
- How many cars from chosen model are there?
- What's the most powerful car on this website?
- What's the average mileage or power of some model or brand?
- How can we compare different model of cars or brands?
- What's the cheapest or most expensive car on website?
- Which brand on website has the newest models of cars?

etc.

This is only part of questions that can be raised regarding this topic and using given data. Next thing we have to do is answer some of these questions and analyze solutions.

Implementation

Implementation for some of this questions is done in Python, with mentioned libraries. Here we'll discuss problems that were solved using provided tools.

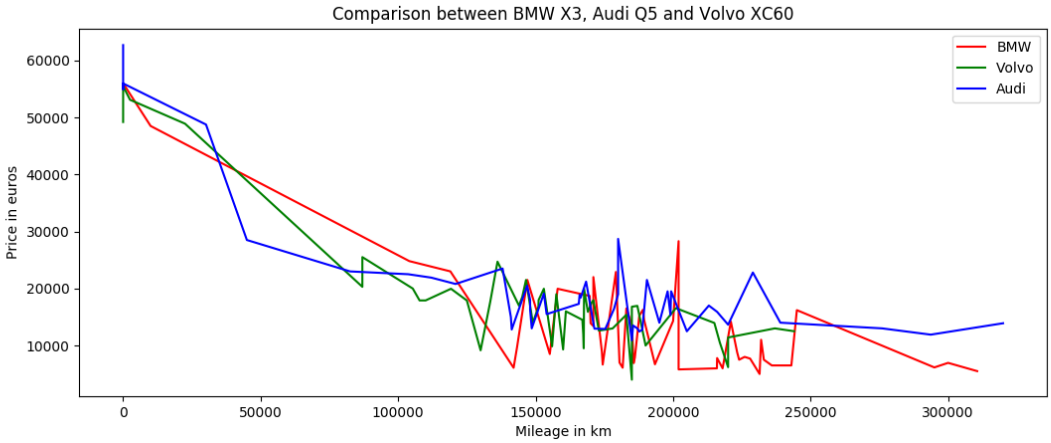
Problem 1

Comparison in mileage and price between competitive SUVs currently (BMW X3, Audi Q5, Volvo XC60). For this problem, we have to query our database 3 times for each model, specifying what we need. Part of code in which this is applied is shown in picture 3.

```
# Comparisson between bmw, audi and volvo
def comparisonBAV():
    # Making queries
    bmw = "SELECT kilometraza, cena from Cars WHERE brend='bmw' AND model='x3' ORDER BY kilometraza"
    audi = "SELECT kilometraza, cena from Cars WHERE brend='audi' AND model='q5' ORDER BY kilometraza"
    volvo = "SELECT kilometraza, cena from Cars WHERE brend='volvo' AND model='xc60' ORDER BY kilometraza"

    mycursor.execute(bmw)
    bmw_data = mycursor.fetchall()

    mycursor.execute(audi)
    audi_data = mycursor.fetchall()

    mycursor.execute(volvo)
    volvo_data = mycursor.fetchall()
```

Picture 3: Queries for comparissing X3, Q5 and XC60

After collecting the results of these queries, we can present solutions in some way. For this problem, plotting data was implemented with python libraries. Graph is presented on picture 4.



Picture 4: Comparison of BMW, Audi and Volvo model

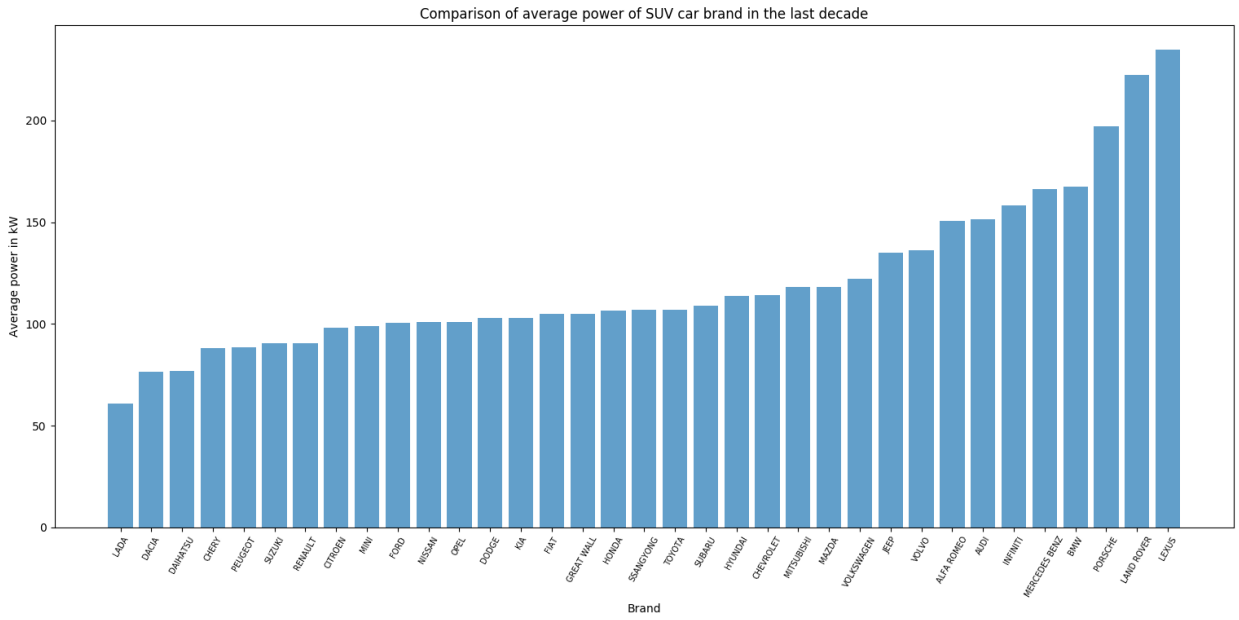
From graphical representation we can see that these models are very similar regarding mileage and price for given mileage. We can also see how prices are different in some cases.

Problem 2

If we want to see what's the most powerful SUV (Sport Utility Vehicle) car in the last decade, query would go like it is shown in picture 5 and graphical representation of data is shown on picture 6. We can modify part of the code in order to change years and we can specify and obtain other results based on just one query.

```
# visualizing comparisson of strongest SUVs currently on website by average value in HP
def strongestSUV():

    query = "SELECT DISTINCT brend, model, godiste, snaga FROM Cars WHERE (godiste>=2010 AND godiste<=2019 AND karoserija='Dzip/SUV') "
    mycursor.execute(query)
    data = mycursor.fetchall()
    np.set_printoptions(threshold= np.inf)
    data = np.asarray(data)
```

Picture 5: Query for the most powerful SUV car in the last decade



Picture 6: Graphical representation for previous query

This is just one option of comparing different car types based on some factor, we can use very similar approach and change parameters in order for other similar problems.

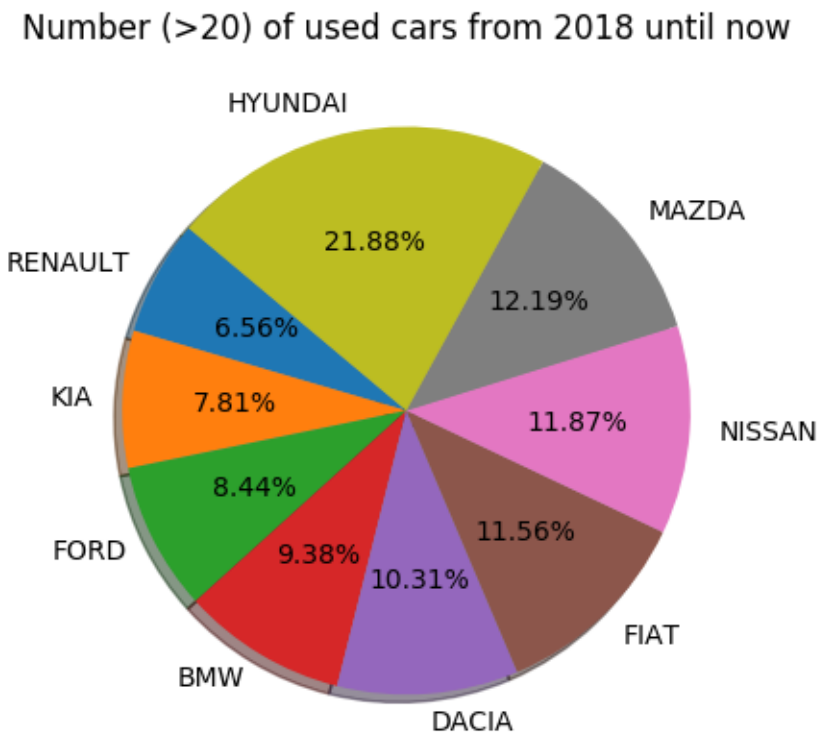
Problem 3

If we want to know, for example, how many used cars there are for every brand on this website, we can use query on our database like in picture 7. In this, we can specify year of car production or some other parameter and focus our search in more detail. In this example, year is 2018> which means that result will only be most recent models of every brand and ordering them by number of used cars.

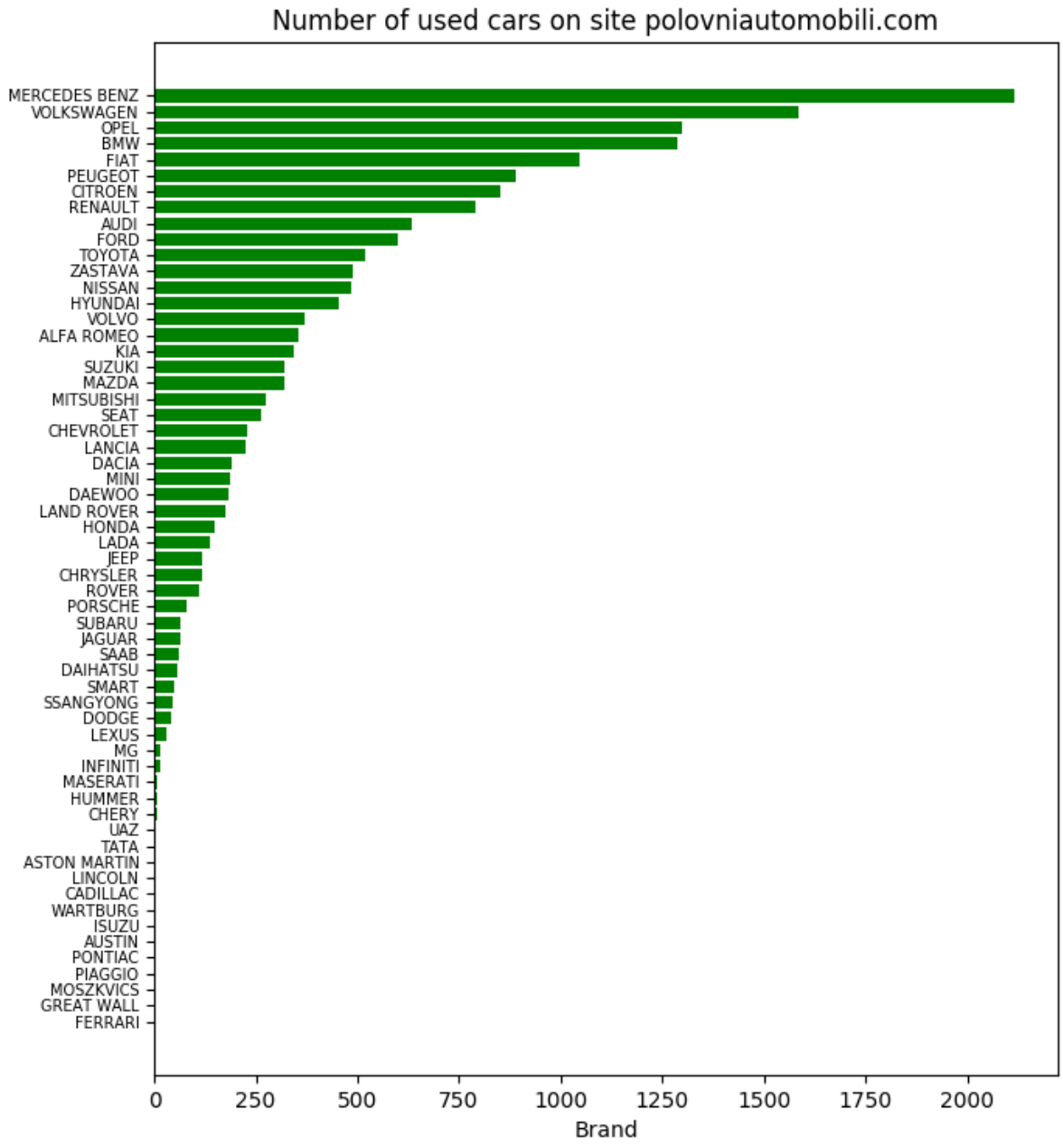
```
    plt.show()


def countCars():

    query = "SELECT brend, COUNT(*) FROM Cars WHERE godiste>2018 GROUP BY brend HAVING COUNT(*)>20"
    mycursor.execute(query)
    data = mycursor.fetchall()

    dictionary = dict(data)

    sorted_x = sorted(dictionary.items(), key=operator.itemgetter(1))

    x, y = zip(*sorted_x)
    #plt.figure(figsize=(7,12))

    # plt.barh(x,y, align='center', color= 'g')
    plt.pie(y, labels=x,autopct='%1.2f%%', shadow=True, startangle=140)
    plt.title("Number (>20) of used cars from 2018 until now")
    plt.tick_params(axis='y', which='major', labelsize=7)
    plt.show()
```

Picture 7: Code for counting number of used cars for every model

We can take it a step further and define parameters to only show brands that has amount of cars that is higher than 20. Result of this search is shown on picture 8 and it is presented in percentage of all the cars that satisfy this query. Result of counting used cars without limitations is shown on picture 9 and we can see the difference, beside the different graph types.



Picture 8: Number of used cars > 20 for the past 2 years



Picture 9: Number of all cars per brand without limitations

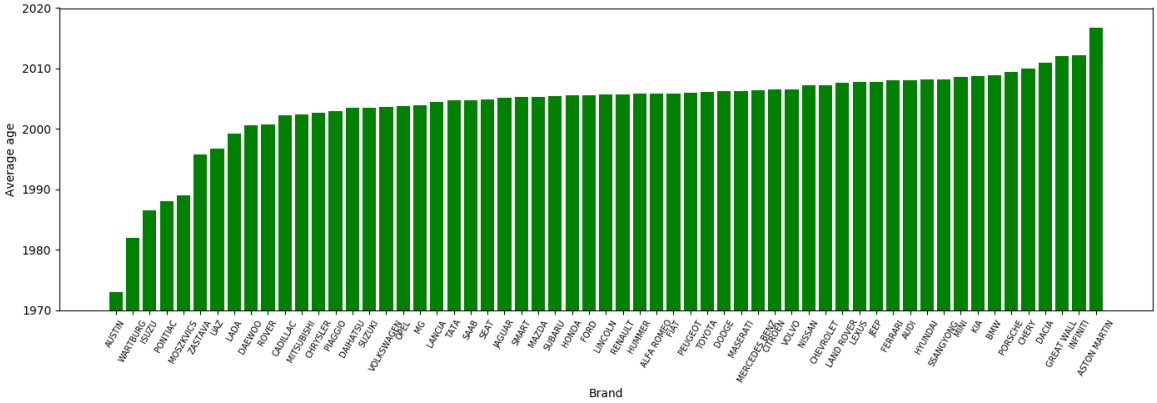
Problem 4

Next question that we can answer is: What is the "oldest" brand on this website? Not just one but ordering brands by average age of cars. Query for this part is shown on picture 10.

```
def oldestCars():

    averageYears = "SELECT brend, AVG(godiste) FROM Cars GROUP BY brend"
    mycursor.execute(averageYears)
    data = mycursor.fetchall()
    data = np.asarray(data)

    dictionary = dict(data)

    sorted_x = sorted(dictionary.items(), key=operator.itemgetter(1))

    x, y = zip(*sorted_x)
    plt.figure(figsize=(15,5))
    plt.bar(x,y,  color='g')
    plt.xlabel('Brand')
    plt.xticks(rotation=60)
    plt.tick_params(axis='x', which='major', labelsize=7)
    plt.title("Average age of used cars")
    plt.ylabel('Average age')
    plt.ylim((1970,2020))
    plt.show()
```

Picture 10: Code for function oldestCars()

Result of calling this function is shown on picture 11.



Picture 11: Graph representation for average age of brands

As we can see, majority of brands have similar average age for models but few of them stands out.

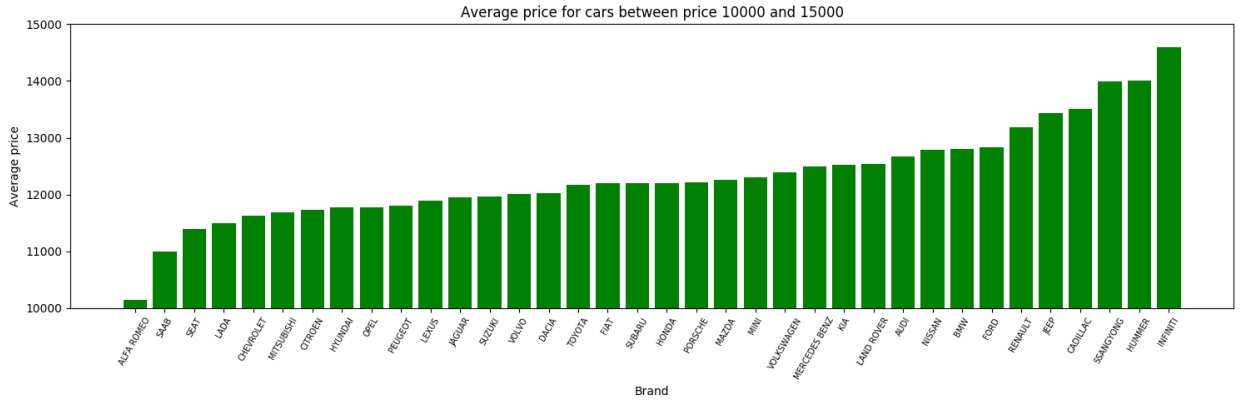
Problem 5

If we want to search for brand that has most expensive models in some price range we can define query similar like it is shown on picture 12. For this function we have to define price1 and pricw2 in order to obtain average price of models in between that prices.

Graph representing result of query is shown on picture 13. For every problem mentioned previously, we have to format data to fit the standards for plotting in Python.

```
def mostExpensive(price1,price2):


    mycursor.execute("SELECT brend, AVG(cena) FROM Cars WHERE cena<%s and cena>%s GROUP BY brend", (price2,price1))
    data = mycursor.fetchall()
    data = np.asarray(data)

    dictionary = dict(data)

    sorted_x = sorted(dictionary.items(), key=operator.itemgetter(1))

    x, y = zip(*sorted_x)
    plt.figure(figsize=(15,5))
    plt.bar(x,y,  color='g')
    plt.xlabel('Brand')
    plt.xticks(rotation=60)
    plt.tick_params(axis='x', which='major', labelsize=7)
    plt.title("Average price for cars between price %s and %s" %(price1,price2))
    plt.ylabel('Average price')
    plt.ylim((price1,price2))
    plt.show()
```

Picture 12: Code for function mostExpensive(price1,price2)



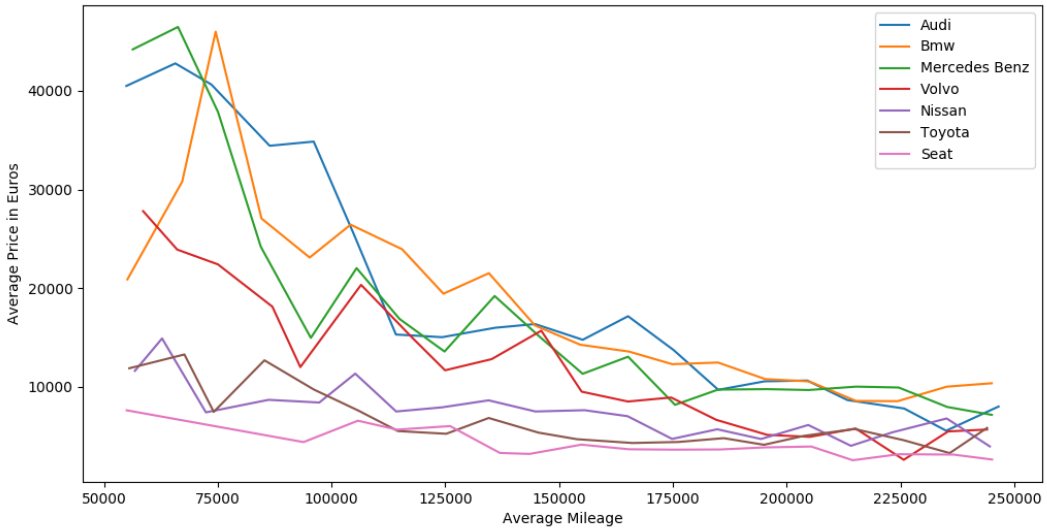
Picture 13: Average price of brands (models) represented on a bar graph

Problem 6

Comparing more than two brand cars is always helpful to see how much are they different from each other. This problem is solved by averaging mileage wiht combination of average price for every iteration based on mileage. Average mileage is calculated in between range of low and high value and a step that increases every iteration. Everything between steps is calculated and it's taken into account, because most of used cars doesn't have round mileage, for example 135000km but something like 135928km. That's why we need to take into account values between two steps. Code that is representing this logic is displayed on picture 14. Low and high value can be modified in function also with different value for step. For simplicity reason, few car brands have been analyzed, not all of them. Looking at the result, we can clearly see difference in brands and how it affects cost of car ranging from cheaper companies to more expensive ones.

```
def compareSpecs():


    plt.figure(figsize=(12,6))
    result = ['Audi','Bmw','Mercedes Benz', 'Volvo','Nissan','Toyota','Seat','Skoda']
    for brend in result:
        array = dict()

        for step in range(50000,250000,10000):

            mycursor.execute("SELECT AVG(kilometraza), AVG(cena) FROM Cars WHERE (brend=%s and kilometraza>%s and kilometraza<%s+10000)", (str(brend),step,step))

            data = mycursor.fetchall()
            data = np.asarray(data)
            #print(data[0][0])


            try:
                array[data[0][0]] = data[0][1]
                #print(array)
            except:
                pass


        for i in list(array):
            if i is None:
                del array[i]


        try:
            sorted_x = sorted(array.items(), key=operator.itemgetter(0))
            x, y = zip(*sorted_x)
            plt.plot(x,y,label='%s' %brend)

        except:
            continue

    plt.legend()
    plt.title("Average price for average mileage in range 50000km - 200000km")
    plt.xlabel('Average Mileage')
    plt.ylabel('Average Price in Euros')
    plt.show()
```

Picture 14: Code for function compareSpecs()



Picture 15: Graphical representation of correlation between average mileage and average price for some of the car brands

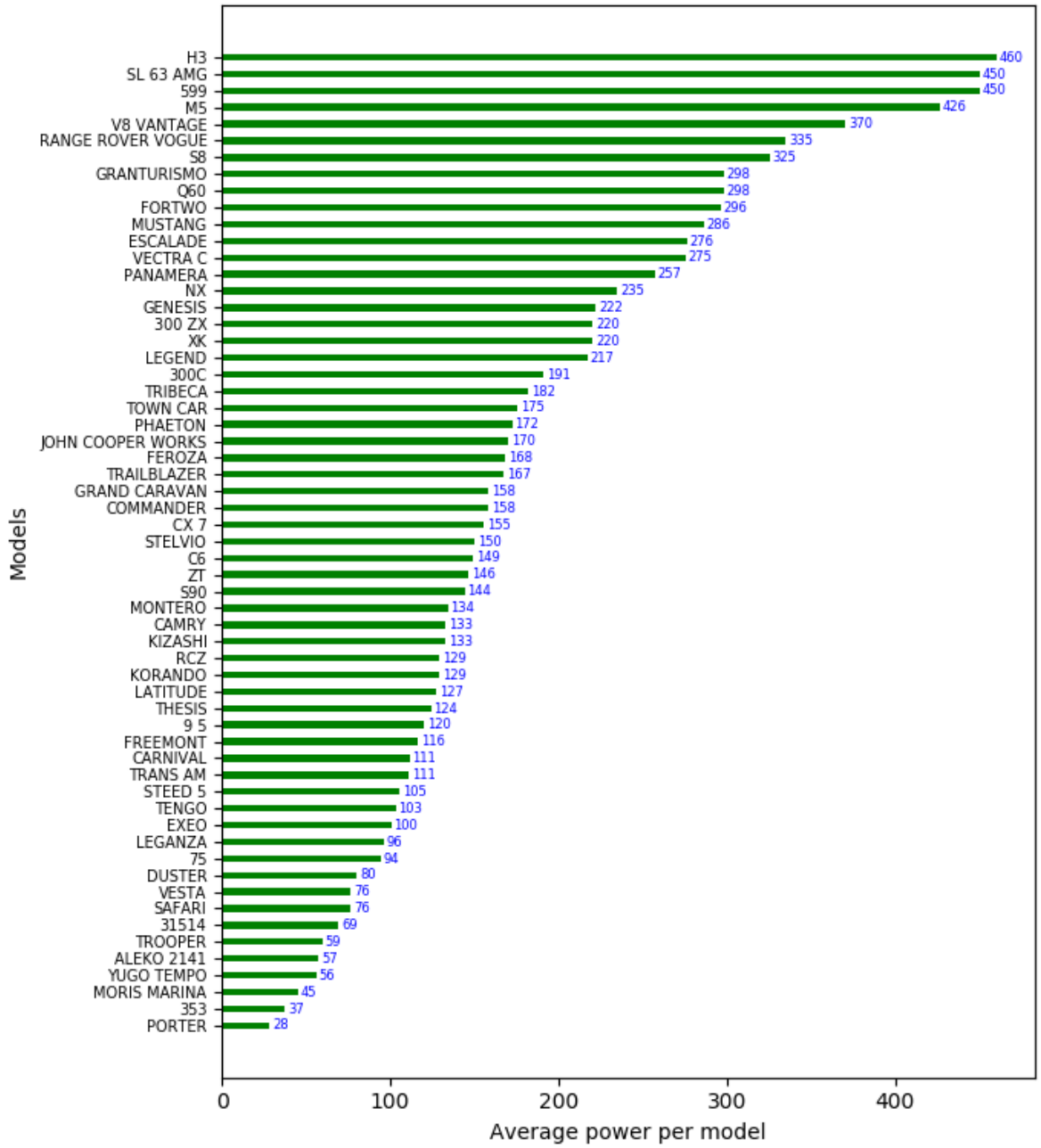
Problem 7

We already examined which brand has the highest average power of SUV cars but here we have problem of finding out which models have the most powerful engine on this website, including other types of vehicle, not just SUV. We can approach problem wiht similar logic and that logic is displayed in picture 16, where is represented part of a code.

```
def strongestCars():

    mycursor.execute("SELECT brend FROM Cars GROUP BY brend")
    brendovi = mycursor.fetchall()
    brendovi = list(brendovi)

    array =  dict()
    for x in brendovi:

        mycursor.execute("SELECT model, AVG(snaga) as asnaga from Cars WHERE brend=%s GROUP BY model ORDER BY asnaga DESC LIMIT 1", (x[0],))
        data = mycursor.fetchall()
        data = np.asarray(data)
        array[data[0][0]] = float(data[0][1])

    sorted_x = sorted(array.items(), key=operator.itemgetter(1))
    width = 0.4
    x, y = zip(*sorted_x)


    plt.figure(figsize=(7,10))
    plt.xlabel('Average power per model')
    plt.ylabel('Models')
    #plt.xticks(rotation=50)
    plt.tick_params(axis='y', which='major', labelsize=7)
    plt.barh(x,y,width,color='green')
    tup = []
    for i in range(0,len(y)):
        # print(y[i])
        tup.append(int(y[i]))
    for i, v in enumerate(tup):
        plt.text(v + 2, i -0.25, str(v), color='blue', fontsize=6)
    plt.show()
```

Picture 16: Part of a code responsible for finding the most powerful cars

We can calculate average power for each model of each brand and represent result on graph. That graph is displayed on picture 17, where we can clearly see the most powerful car for every brand ordered from most powerful to least powerful.



Picture 17: For every brand is represented most powerful car based on average power in kW

Conclusion

Data analysis can be very useful in order to understand better some problems and it is very helpful with decision making problems. As we can see in this project, we have various kind of ways for exploiting and searching for different solutions to given problems. We are able to compare different features and specifications that we have, combine some solutions and make assumptions based on the results.

With analyzed data it is much easier to find errors and to improve overall efficiency for some jobs or simply to make decision which car to buy or not based on some corresponding specifications.

Reference

1. https://realpython.com/beautiful-soup-web-scraper-python/ - Python web scraping
2. https://realpython.com/python-web-scraping-practical-introduction/ - Web scraping
3. https://matplotlib.org/ - Plotting in Python
4. https://pandas.pydata.org/ - Pandas in Python