DATA1002 Written Report

Group name: DATA1002-CC-L1-G3

Members:

SID: 510067773, UniKey: eyon5512

SID: 520562286, UniKey: tdra0337

SID: 510594198, UniKey: skle4070

SID: 520604830, UniKey: asau0735

Project 1 Data 1002

Topic:

Does there exist a correlation between the mileage a car has driven and its price?

If there exists a correlation between the mileage of a car and its price, understanding the strength of this correlation will aid both the buyers and sellers of used cars by utilizing a predictive model such as a linear regression to predict the sale price of the used car based on historical data on used car prices from different countries. However, due to the different car regulations of every country, a linear regression model for each country may be required to accurately represent the price predictions of the used cars depending on which country the car will be sold in.

The model will also aid in the understanding of the depreciation rates of cars based on how often the car is driven, information that will prove to be useful in the financial and accounting industry (e.g. calculation of the future value of assets). Moreover, the modeling could be done for every car brand if more specific information on depreciation rates are needed.

Datasets:

To answer this question we are to use 5 data sets from different author's.

1. **Dataset 1: "car_price_prediction.csv", SID: 520562286**

This Dataset is called "car_price_prediction.csv" and was found on the url(https://www.kaggle.com/datasets/deepcontractor/car-price-prediction-challenge/code). This dataset has 18 columns('ID', 'Price', 'Levy', 'Manufacturer', 'Model', 'Prod. year', 'Category', 'Leather interior', 'Fuel type', 'Engine volume', 'Mileage', 'Cylinders', 'Gear box type', 'Drive wheels', 'Doors', 'Wheel', 'Color', 'Airbags') and 19238 rows. Using python this dataset was cleaned and renamed to "Car_price_prediction_clean.csv". This dataset contains 4 columns from the original dataset Manufacturer, Model, Mileage and Price and 18290 rows. In the cleaning process any null and duplicate values were removed, and any row containing values that were outside (Lower_tail,Upper_tail) as defined in the code, was removed. This creates a clean dataset containing 4 variables. Manufacturer and Model are of type strings, and Price and Mileage are of type int32.

Using this line of code, i produced a summary of the dataset **car_price_prediction.csv**

```
print(df_new[['Mileage','Price']].describe())
```

the summary looks like:

	Mileage	Price
count	18289.000000	1.828900e+04
mean	129517.340642	1.892669e+04
std	81522.645132	1.954290e+05
min	0.000000	1.000000e+00
25%	68235.000000	5.378000e+03
50%	123000.000000	1.348500e+04
75%	180000.000000	2.258000e+04
max	367053.000000	2.630750e+07

2. **Dataset 2: "USA_cars_datasets.csv", SID : 510594198**

The dataset "USA_cars_datasets.csv" was found from kaggle (https://www.kaggle.com/datasets/doaaalsenani/usa-cers-dataset), with 2499 rows and 13 columns consisting of 'price', 'brand', 'model', 'year', 'title_status', 'mileage', 'color', 'vin', 'lot', 'state', 'country', and 'condition'. However, using python it was cleaned and renamed to "USA_cars_cleaned.csv", consisting of still 2499 rows as it has no NA values that had to be removed, but reduced to only 4 columns of 'price', 'brand', 'year', and 'mileage', where the mileage in terms of miles were converted into kilometres. Below are the explanations of what I did to the file with python,

First, was to import and read the data to panda first, and then making a variable new_file for the cleaned dataset with only the columns that we are going to be using the data.
Then, I checked if there were any NA values on the columns that were going to be used. Since there was none, there is no need for cleaning NA values. Next, since the mileage from the other datasets were in terms of kilometres, and in this file it was in mileage, I converted it into kilometres for a fair comparison. Afterwards, the data from the year 1993 was removed as the price and mileage shows a value of 0. Finally, the new file is already clean and ready to be made into a new csv file named "USA_cars_cleaned.csv".

```
import pandas as pd

#Importing the Data
original_file = pd.read_csv("USA_cars_datasets.csv")
#print(original_file)
new_file = original_file.loc[:,["price","brand","year","mileage"]]

#Check for NA values
print("Total Number of NA Values for each")
na_values = pd.isnull(new_file).sum()
print(na_values)  #There is no NA values

#Data Cleaning
new_file = new_file.drop(labels=[545], axis=0) #Removing the data from 1993 as the price and mileage is 0
new_file["mileage"] = new_file["mileage"]*1.609
print("MAX MILEAGE:", max(new_file["mileage"]))
print("MIN MILEAGE:", min(new_file["mileage"]))

print("MAX PRICE : ", max(new_file["price"]))
print("MIN PRICE : ", min(new_file["price"]))
#New Cleaned File
new_file.to_csv("/Users/sebastianklemens/Desktop/DATA1002 /USA_cars_cleaned.csv")
```

Then, a dictionary named year_mileage was created to store the key (year) and the value (average mileage). It is done by looping through the cleaned dataset file. Below is the code given;

```
#Creating a dictionary for the year and average mileage
year_mileage = {}
is_first_line = True

for row in open("USA_cars_cleaned.csv"):
  if is_first_line:
    is_first_line = False
  else:
    values = row.split(",")
    year = values[3]
    mileage = float(values[4])
    if year in year_mileage:
        year_mileage[year].append(mileage)
    else:
        year_mileage[year] = [mileage]

for key in sorted(year_mileage) :
    mileage = year_mileage[key]
    average_mileage = sum(mileage) / len(mileage)

    print(f'{key} : {average_mileage}')
```

From the cleaned dataset, it was concluded that the mileage ranges from 0 to 1637859.024 kms, and the price ranges from $0 to $84900 USD.

Lastly, the average mileage based on the years are as shown below :

```
MAX MILEAGE: 1637859.024
MIN MILEAGE: 0.0
MAX PRICE :  84900
MIN PRICE :  0
1973 : 74377.634
1984 : 66897.393
1994 : 165857.329
1995 : 442001.95399999997
1996 : 442030.91599999997
1997 : 281743.1405
1998 : 352722.56649999996
1999 : 365197.948
2000 : 272739.57875
2001 : 293149.8242
2002 : 322009.17000000004
2003 : 323919.5893333333
2004 : 300644.59983333334
2005 : 269279.55833333335
```

```
2006 : 255717.16324999998
2007 : 266783.1948333333
2008 : 261471.9752222222
2009 : 349345.34863636363
2010 : 325711.3552307692
2011 : 191511.43486956524
2012 : 217077.3411111111
2013 : 190846.1654302326
2014 : 155540.11157692314
2015 : 113551.9487908163
2016 : 92585.9498719212
2017 : 77928.77382228125
2018 : 55153.64824050636
2019 : 38702.581155829605
2020 : 17108.664604166665
```

3. **Dataset 3: "autoscout24-germany-dataset.csv" (renamed "germany_cars.csv"), SID: 510067773**

This dataset was retrieved from kaggle and is of license CC0: Public Domain, meaning that the data may be used without prior permission, as it has been dedicated to the public and therefore, has no copyrights (https://www.kaggle.com/datasets/ander289386/cars-germany). The dataset contains the data of used cars sold in Germany from 2011 to 2021 retrieved from AutoScout24, one of Europe's largest car market for new and used cars.

The csv consisted of 9 columns, ('mileage', 'make', 'model', 'fuel', 'gear', 'offerType', 'price', 'hp' and 'year') with a total data volume of 46,405 rows. The data file was shortened into "germany_cars.csv" for ease of import and understanding. Data cleaning was done on Python using the pandas package.

The dataset was first imported into a python dataframe from a csv file and was filtered to only contain 4 columns, as for our analysis, only these 4 columns from the dataset were required: 'make', 'year', 'mileage' and 'price':

```
import pandas as pd
# Import Data
df = pd.read_csv("germany_cars.csv")
df1 = df.loc[:,["make", "year", "mileage", "price"]]
```

The data frame was then checked to see if there were any NA values present using the pd.isnull() function, for which the result was zero NA values in all of the columns:

```
# Check for NA values
na_check = pd.isnull(df1).sum()
print(na_check)
```

```
make       0
year       0
mileage    0
price      0
dtype: int64
```

Column headings were renamed to be more specific for better accessibility. Car prices were also adjusted to USD for standardization purposes as the price data was originally in euros.

```
# Data Cleaning
df1 = df1.rename(columns = {"make":"brand", "price": "price(EUR)"})
df1["price(EUR)"] = df1["price(EUR)"]*0.99
df1 = df1.rename(columns = {"price(EUR)":"price(USD)"})
```

The cleaned data frame was then written into a new csv file ("germany_final.csv") to be used for data integration:

```
# Write clean data to new csv
df1.to_csv("/Users/ethan_yong_1/Documents/USYD_Projects/germany_final.csv")
```

The final data file contained data for used cars with the columns: 'brand', 'year', 'mileage' and 'price(USD)'. The mileage of the cars ranged from 0km to 1,111,111km, the following code was used to generate this range statistic:

```
# Summary Statistics (Mileage Range):
df2 = pd.read_csv("germany_final.csv")
mileage = df2['mileage']
max_mileage = max(mileage)
min_mileage = min(mileage)
print(f"Mileage ranges from {min_mileage} to {max_mileage}.")
```

Output:

```
Mileage ranges from 0 to 1111111.
```

Car sale prices ranged from $1089.0 to $1,187,901.0. This price range may prove to be a limitation for using this set for the linear regression analysis as it indicates that prices of the cars vary greatly depending on their brand and model which may be a confounding factor affecting the sale price of the car. The following code was used to generate this range statistic:

```
# Summary Statistics (Price Range):
prices = df2['price(USD)']
max_price = max(prices)
min_price = min(prices)
print(f"Price ranges from ${min_price} to ${max_price}.")
```

Output:

```
Price ranges from $1089.0 to $1187901.0.
```

The dataset consists of a total of 77 different car brands, the following code was used to generate this statistic:

```
# Summary Statistics (Car Brands):
brands = df2['brand']
car_brands = list(pd.unique(brands))
brand_count = 0
for values in car_brands:
    brand_count += 1
print(f"There are a total of {brand_count} car brands.")
```

Output:

```
There are a total of 77 car brands.
```

The data volume for each car brand was also generated using the following code:

```python
# Summary Statistics (Data volume per brand):
car_dic = {}
brands = df2['brand']
for brand in brands:
    if brand not in car_dic:
        car_dic[brand] = 1
    else:
        car_dic[brand] += 1
for key in car_dic:
    print(f"{key} : {car_dic[key]}")
```

Output:

BMW : 2405
Volkswagen : 6931
SEAT : 1924
Renault : 2830
Peugeot : 1232
Toyota : 1275
Opel : 4814
Mazda : 714
Ford : 4442
Mercedes-Benz : 2354
Chevrolet : 223
Audi : 2684
Fiat : 1700
Kia : 1055
Dacia : 715
MINI : 469
Hyundai : 1888
Skoda : 2889
Citroen : 957
Infiniti : 15
Suzuki : 361
SsangYong : 37
smart : 974
Cupra : 65
Volvo : 804
Jaguar : 126
Porsche : 244
Nissan : 753
Honda : 182
Lada : 33
Mitsubishi : 409
Others : 25
Lexus : 48
Jeep : 158
Maserati : 12
Bentley : 15
Land : 164
Alfa : 132
Subaru : 57
Dodge : 23
Microcar : 12
Lamborghini : 9
Baic : 3
Tesla : 24
Chrysler : 5
9ff : 1
McLaren : 11
Aston : 30
Rolls-Royce : 3
Alpine : 5
Lancia : 18
Abarth : 43
DS : 16
Daihatsu : 10
Ligier : 5
Ferrari : 11
Caravans-Wohnm : 3
Aixam : 3
Piaggio : 5
Zhidou : 1
Morgan : 2
Maybach : 3
Tazzari : 1
Trucks-Lkw : 1
RAM : 2
Iveco : 4
DAF : 1
Alpina : 10
Polestar : 4
Brilliance : 1
FISKER : 1
Cadillac : 7
Trailer-Anhänger : 4
Isuzu : 1
Corvette : 3
DFSK : 2
Estrima : 2

4. Dataset 4: “Belarus Used Cars Prices.csv”, SID: 520604830

This dataset was obtained from the Kaggle platform from the user Slava Pasedko (https://www.kaggle.com/datasets/slavapasedko/belarus-used-cars-prices?resource=download). The dataset originally called “cars.csv” contains information from the price of different used cars in the Belarus market in 2019. The dataset includes 12 categories from the sold cars. The most important categories are the car’s manufacturer, model, price in USD, year of the model, and mileage in kilometers. There is no specific information about the metadata of the dataset but the author confirms it was retrieved from the Belarus car market automatically except for the Segment section.

The dataset was first analyzed in python to identify which columns contained NaNs. If the column containing NaNs was not fundamental for future analysis then that same column would be removed from the dataset. If a column containing NaNs was in fact important for future analysis, then the row would be removed completely. After having the relevant data, a cleaning of outliers was implemented from ground without using existing functions to clean outliers automatically.

The first step was to import the pandas library to facilitate the manipulation of csv files in python as Dataframes. The numpy library was imported in case there was a need to do simple arithmetic operations. After importing the data using Pandas’ function “pd.read_csv” and saving it into a variable called “data”. After reviewing the result we identify it as a Pandas Dataframe with 56,244 and 12 columns.

```
[128] import pandas as pd  # import panda functions
      import numpy as np  # import statistic and math functions
      data = pd.read_csv('Belarus Used Cars Prices.csv')  # import csv data into a Panda Data Frame
      data
```

	make	model	priceUSD	year	condition	mileage(kilometers)	fuel_type	volume(cm3)	color	transmission	drive_unit	segment
0	acura	mdx	31900	2014	with mileage	125000.00	petrol	3500.0	white	auto	all-wheel drive	J
1	acura	mdx	31500	2014	with mileage	89500.00	petrol	3500.0	black	auto	all-wheel drive	J
2	acura	mdx	31000	2014	with mileage	118000.00	petrol	3500.0	white	auto	part-time four-wheel drive	J
3	acura	mdx	29900	2014	with mileage	72000.00	petrol	3500.0	black	auto	all-wheel drive	J
4	acura	mdx	28400	2008	with mileage	299337.24	petrol	3700.0	burgundy	auto	all-wheel drive	J
...	...	...	...	...	...	...	...	...	...	...	...	...
56239	zaz	968	48	1910	with damage	25000.00	petrol	1100.0	green	mechanics	rear drive	B
56240	zotye	t600	15000	2018	with mileage	19627.00	petrol	1500.0	white	mechanics	front-wheel drive	NaN
56241	zotye	z300	5750	2014	with mileage	65600.00	petrol	1500.0	gray	mechanics	front-wheel drive	NaN
56242	zotye	z300	3999	2013	with mileage	275000.00	petrol	1500.0	black	mechanics	front-wheel drive	NaN
56243	zotye	z300	3000	2013	with mileage	250000.00	petrol	1500.0	white	mechanics	front-wheel drive	NaN

56244 rows × 12 columns

Now we check the number of NaNs or empty spaces in each column to review which columns would be removed and if there was any necessity to remove rows. The functions used are integrated in the Pandas library and return the sumatory of NaNs and blank spaces in each column. After observing the results we are able to remove the columns along with the rest of the columns that are not important for the analysis.

```
[129] data.isna().sum()  # Check if there are any Nans in each column
      data.isnull().sum()  # Check if there are any blank spaces in each column

make                    0
model                   0
priceUSD                0
year                    0
condition               0
mileage(kilometers)     0
fuel_type               0
volume(cm3)            47
color                   0
transmission            0
drive_unit           1905
segment              5291
dtype: int64
```

If there were some relevant columns with NaNs or empty spaces then the next function would've been used.

```
[46] #data = data.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False) # remove all the blank rows by using the data.dropna() function
```

The next step was to filter the dataframe by only keeping the relevant columns. After that, we transformed the filtered dataframe into a dictionary with 56244 values (each row), each of them containing 5 keys corresponding to each column. To facilitate the identification of outliers, 5 lists were created to represent each column. Then we verify the maximum and minimum values for year and price to give ourselves and idea of the information contained in the dataset.

```
[130] UD = data[['make','model','priceUSD','year','mileage(kilometers)']]  # Filter the data by only keeping the columns we need
      UD = UD.to_dict('records')  # Transform the Pandas Data Frame into a list where each entry is a dictionary for each row. The keys are the columns.

      c = 0  # counter
      tf = True
      for row in UD:  # We then create a list for each column to facilitate data manipulation
        if tf:
          tf = False
          Brand = [row['make']]
          Model = [row['model']]
          Year = [row['year']]
          Price = [row['priceUSD']]
          Mileage = [row['mileage(kilometers)']]
        else:
          Brand.append(row['make'])
          Model.append(row['model'])
          Year.append(row['year'])
          Price.append(row['priceUSD'])
          Mileage.append(row['mileage(kilometers)'])
        c += 1
      print('Biggest year:',max(Year))  # We then review the maxs and mins
      print('Shortest year:',min(Year))
      print('Most expensive:',max(Price))
      print('Cheapest:',min(Price))

      Biggest year: 2019
      Shortest year: 1910
      Most expensive: 235235
      Cheapest: 48
```

To find outliers in the dataset a function was created to identify the position and values of the quartiles. Then the lower and upper bound were set using the interquartile range. Finally a list containing all values below and above the bounds was created to store the outliers.

```python
[124] # Outlier identification function
def Outlier_IQR(info):
  info = sorted(info)
  outlier = []
  Q1 = len(info)//4-1
  Q2 = int(len(info)/2-1)
  Q3 = 3*len(info)//4-1
  Q4 = len(info)-1
  iQ1 = info[Q1]
  iQ2 = np.mean([info[Q2],info[Q2+1]])
  iQ3 = info[Q3]
  iQ4 = info[Q4]
  IQR = iQ3-iQ1
  #print(Q1,Q3,Q3-Q1-2,Q2)
  #print(info[0],iQ1,iQ2,iQ3,iQ4)
  lb = iQ1 - (1.5 * IQR)
  ub = iQ3 + (1.5 * IQR)

  print("Lower bound:",lb)
  print("Upper bound:", ub)

  for item in info:
    if item < lb or item > ub:
      outlier.append(item)
  print("There are",len(outlier),"outliers")
  return outlier
```

The function was applied to the “Year”, “Price”, and “Mileage” lists. We can see by the output that there were prices over the limit price, model years below the limit and mileage above the limit. This doesn’t necessarily mean that these values are wrong because some of the expensive cars are from luxurious brands like Bently and some models were just really old. For the sake of this exercise, these outstanding values were removed.

```
Price_Outliers = Outlier_IQR(Price)  #We save the price outliers in a list
Year_Outliers = Outlier_IQR(Year)  #We save the year outliers in a list
Mileage_Outliers = Outlier_IQR(Mileage)  #We save the Mileage outliers in a list
```

```
Q0: 48
Q1: 2350
Q2: 5350.0
Q3: 9800
Q4: 235235
Lower bound: -8825.0
Upper bound: 20975.0
There are 2555 outliers

Q0: 1910
Q1: 1998
Q2: 2004.0
Q3: 2010
Q4: 2019
Lower bound: 1980.0
Upper bound: 2028.0
There are 258 outliers

Q0: 0.0
Q1: 137000.0
Q2: 228500.0
Q3: 310000.0
Q4: 9999999.0
Lower bound: -122500.0
Upper bound: 569500.0
There are 689 outliers
```

The final step was to remove the rows containing outliers in price and year from our dataframe and import our new dataset into a new clean csv file.

```
[143] df = data[~data['priceUSD'].isin(Price_Outliers)]  #We now remove the rows with the values containing the outliers
      df1 = df[~df['year'].isin(Year_Outliers)]
      df2 = df1[~df1['mileage(kilometers)'].isin(Mileage_Outliers)]
      df3 = df2[['make','model','priceUSD','year','mileage(kilometers)']]
      df3.to_csv('Belarus Used Cars Prices_clean.csv')  #We now import our dataframe as a new csv file
```

After obtaining the final cleaned dataset we can now analyze the information. We can observe from the results of the outliers function shown above the quartiles for the price, year and mileage variables. We can also render another summary of the data with the following code.

```
print("Maximum Price:",max(df3['priceUSD']),"$","     Minimum Price:",min(df3['priceUSD']),"$")
print("Maximum Year:",max(df3['year']),"     Minimum Year:",min(df3['year']))
print("Maximum Mileage:",max(df3['mileage(kilometers)']),"km","     Minimum Mileage:",min(df3['mileage(kilometers)']),"km")
manu = {}
maker = df3['make']
for brand in maker:
  if brand not in manu:
    manu[brand] = 1
  else:
    manu[brand] += 1
for key in manu:
  print(key+":",manu[key])
```

```
Maximum Price: 20970 $      Minimum Price: 95 $
Maximum Year: 2019      Minimum Year: 1980
Maximum Mileage: 568956.0 km      Minimum Mileage: 0.0 km
```

```
acura: 88
alfa-romeo: 225
aro: 2
asia: 1
audi: 3724
bmw: 3498
bogdan: 1
brilliance: 2
buick: 82
byd: 3
cadillac: 55
changan: 4
chery: 83
chevrolet: 577
chrysler: 507
citroen: 1971
dacia: 74
daewoo: 314
daihatsu: 33
datsun: 17
dong-feng: 1
eksklyuziv: 41
faw: 3
fiat: 922
ford: 2970
fso: 3
gaz: 113
geely: 83
gmc: 17
great-wall: 66
hafei: 6
haval: 4
honda: 919
```

```
hyundai: 1433
infiniti: 217
iran-khodro: 13
isuzu: 25
izh: 13
jac: 1
jaguar: 44
jeep: 179
kia: 1141
lada-vaz: 932
lancia: 114
land-rover: 242
lexus: 249
lifan: 82
lincoln: 35
luaz: 29
maserati: 2
mazda: 1942
mercedes-benz: 2978
mercury: 7
mg: 7
mini: 85
mitsubishi: 1298
moskvich: 30
nissan: 2163
oldsmobile: 2
opel: 3737
peugeot: 2850
plymouth: 14
pontiac: 31
porsche: 86
proton: 21
raf: 1
```

```
ravon: 12
renault: 3662
roewe: 1
rover: 299
saab: 125
saipa: 1
saturn: 17
scion: 9
seat: 464
shanghai-maple: 1
skoda: 1112
smart: 29
ssangyong: 139
subaru: 360
suzuki: 330
tagaz: 1
tata: 2
toyota: 1894
uaz: 129
volkswagen: 6548
volvo: 1138
vortex: 6
wartburg: 5
zaz: 58
zotye: 4
```

Combined data set:

To combine the data set the following code was used:

```
def combine_data(dataset1,dataset2,atributes):
    return pd.concat([dataset1[liste],dataset2[liste]],keys = atributes,ignore_index=True)
```

The function takes 3 inputs, dataset1 and dataset 2 are the two datasets that's being combined and atributes is a list containing the keys to the columns of interest. In this project atributes looks like:

```
liste = ['Manufacturer','Price','Mileage']
```

.

The function returns a single combined dataset containing only the variables associated with the keys in liste from the two input datasets.

For the function to work the two datasets need similar keys. As an example the keys from the **germany_cars.csv** were changed using the code below:

```
df_german = pd.read_csv('germany_final.csv')
df_german['Price'] = df_german['price(USD)']
df_german['Manufacturer'] = df_german['brand']
df_german['Mileage'] = df_german['mileage']
```