Import modules

In [1]:
```python
import numpy
import scipy
import pandas
import matplotlib.pyplot as plt
import sklearn
```

Dataset we have for this week contains information related to used car listings in Canada:

In [2]:
```python
df = pandas.read_csv('ca-dealers-used.csv')
df.head()
```

```
c:\users\turha\pycharmprojects\msci433w22\venv\lib\site-packages\IPython\core
\interactiveshell.py:3251: DtypeWarning: Columns (13,15) have mixed types.Spe
cify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[2]:

	id	vin	price	miles	stock_no	year	make	model	trim
0	b39ea795-eca9	19UNC1B01HY800062	179999.0	9966.0	V-P4139	2017.0	Acura	NSX	Base
1	026cb5b1-6e3e	19UNC1B02HY800023	179995.0	5988.0	PPAP70374	2017.0	Acura	NSX	Base
2	5cd5d5b2-5cc2	19UNC1B02HY800071	168528.0	24242.0	B21085	2017.0	Acura	NSX	Base
3	b32473ed-5922	19UNC1B02LY800001	220000.0	6637.0	AP5333	2020.0	Acura	NSX	Base
4	ac40c9fc-0676	19UNC1B02LY800001	220000.0	6637.0	AP5333	2020.0	Acura	NSX	Base

5 rows × 21 columns

Let's filter-out the unnecessary columns such as id, vin, etc.

In [3]:
```
# in case we need to display multiple data in one chunk:
from IPython.display import display

to_drop = ['id', 'vin', 'stock_no', 'seller_name', 'street', 'city', 'state',
'zip']
df_filtered = df.drop(columns=to_drop)
display(df_filtered.info())
display(df_filtered.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 393603 entries, 0 to 393602
Data columns (total 13 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   price         358486 non-null  float64
 1   miles         366590 non-null  float64
 2   year          393586 non-null  float64
 3   make          393603 non-null  object 
 4   model         388809 non-null  object 
 5   trim          354824 non-null  object 
 6   body_type     359578 non-null  object 
 7   vehicle_type  355365 non-null  object 
 8   drivetrain    354608 non-null  object 
 9   transmission  357922 non-null  object 
 10  fuel_type     322790 non-null  object 
 11  engine_size   320950 non-null  float64
 12  engine_block  320439 non-null  object 
dtypes: float64(4), object(9)
memory usage: 39.0+ MB
```

None

	price	miles	year	make	model	trim	body_type	vehicle_type	drivetrain	transmiss
0	179999.0	9966.0	2017.0	Acura	NSX	Base	Coupe	Car	4WD	Autom
1	179995.0	5988.0	2017.0	Acura	NSX	Base	Coupe	Car	4WD	Autom
2	168528.0	24242.0	2017.0	Acura	NSX	Base	Coupe	Car	4WD	Autom
3	220000.0	6637.0	2020.0	Acura	NSX	Base	Coupe	Car	4WD	Autom
4	220000.0	6637.0	2020.0	Acura	NSX	Base	Coupe	Car	4WD	Autom

The data types of columns are important while building a model, especially the columns that contain continuous values that act as discrete / category variables, i.e. year and engine_size:

In [4]:
```
# filtering out rows that contain a null value
df_filtered.dropna(inplace=True)
# since years don't have fractional part, we only keep the integer part:
df_filtered = df_filtered.astype({'year': int}).astype({'year': str})
# engine sizes may have fractional part, so we keep them as they are:
df_filtered = df_filtered.astype({'engine_size': str})
display(df_filtered.info())
display(df_filtered.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 274852 entries, 0 to 393602
Data columns (total 13 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   price         274852 non-null  float64
 1   miles         274852 non-null  float64
 2   year          274852 non-null  object 
 3   make          274852 non-null  object 
 4   model         274852 non-null  object 
 5   trim          274852 non-null  object 
 6   body_type     274852 non-null  object 
 7   vehicle_type  274852 non-null  object 
 8   drivetrain    274852 non-null  object 
 9   transmission  274852 non-null  object 
 10  fuel_type     274852 non-null  object 
 11  engine_size   274852 non-null  object 
 12  engine_block  274852 non-null  object 
dtypes: float64(2), object(11)
memory usage: 29.4+ MB
```

None

	price	miles	year	make	model	trim	body_type	vehicle_type	drivetrain	transmissic
0	179999.0	9966.0	2017	Acura	NSX	Base	Coupe	Car	4WD	Automat
1	179995.0	5988.0	2017	Acura	NSX	Base	Coupe	Car	4WD	Automat
2	168528.0	24242.0	2017	Acura	NSX	Base	Coupe	Car	4WD	Automat
3	220000.0	6637.0	2020	Acura	NSX	Base	Coupe	Car	4WD	Automat
4	220000.0	6637.0	2020	Acura	NSX	Base	Coupe	Car	4WD	Automat

Dataset is now ready, let's build a linear regression model for the price of the car using the other explanatory variables.

Since we have (a lot of) categorical variables, we'll need to work with indicator variables, while this process is done automatically in R, we have to do the following in Python:

In [5]:
```
# importing the linear regression model:
from sklearn.linear_model import LinearRegression

# adding the indicator variables:
X = pandas.get_dummies(data=df_filtered.drop(columns='price'))

# building the model, takes some time!
model = LinearRegression().fit(X=X, y=df_filtered.price)
```

Once we have the model ready, we can check the coefficients for each variable:

In [6]:
```
coefficients = pandas.DataFrame(model.coef_, X.columns, columns=['Coefficient'
])
print(f'Intercept: {model.intercept_}\tR2: {model.score(X, df_filtered.pric
e)}')
coefficients
```

```
Intercept: 56176.960533302685	R2: 0.9384244762475383
```

Out[6]:

	Coefficient
miles	-0.050190
year_1990	-2675.448052
year_1991	5081.391780
year_1992	-37200.829686
year_1994	-18517.661545
...	...
engine_size_8.3	-19213.133584
engine_size_8.4	8905.846595
engine_block_H	-4471.128981
engine_block_I	598.121389
engine_block_V	3873.008349

2239 rows × 1 columns

From the coefficients, we can see that each mile reduces the car's predicted sale price by 5 cents and the sale price decreases as car gets older.

Since data includes a lot of categorical variables, it's hard to make analysis on every variable, let's simplify:

In [7]:
```
df_small = df_filtered.loc[:, ['price', 'miles', 'year']]
X_small = pandas.get_dummies(data=df_small.drop(columns='price'))
model_small = LinearRegression().fit(X=X_small, y=df_small.price)
coefficients_small = pandas.DataFrame(model_small.coef_, X_small.columns, colu
mns=['Coefficient'])
print(f'Intercept: {model_small.intercept_}\tR2: {model_small.score(X_small, d
f_small.price)}')
coefficients_small
```

```
Intercept: 26863.533537565752   R2: 0.22806182091738292
```

Out[7]:

	Coefficient
miles	-0.055160
year_1990	-3955.127728
year_1991	-1572.213176
year_1992	-8792.098292
year_1994	-9593.653404
year_1996	1694.597681
year_1997	15079.145221
year_1998	3867.561926
year_1999	2504.188068
year_2000	-3948.930136
year_2001	-6616.524273
year_2002	-3119.635141
year_2003	-3575.850031
year_2004	-5350.153193
year_2005	-7289.601251
year_2006	-6084.083171
year_2007	-6652.659471
year_2008	-5704.476430
year_2009	-6949.650472
year_2010	-7307.155906
year_2011	-6472.590308
year_2012	-6195.134500
year_2013	-5398.988943
year_2014	-3600.754548
year_2015	-1703.459302
year_2016	-972.591549
year_2017	2113.898574
year_2018	6496.371893
year_2019	8889.293354
year_2020	13943.938185
year_2021	26410.769328
year_2022	29855.566994

As expected, smaller model has a lower R^2 value, let's calculate the p-values of variables, manually:

In [8]:

```
from scipy import stats
sse = numpy.sum((model_small.predict(X_small) - df_small.price) ** 2, axis=0)
/ float(X_small.shape[0] - X_small.shape[1])
se = numpy.array([
    numpy.sqrt(numpy.diagonal(sse * numpy.linalg.inv(numpy.dot(X_small.T, X_sm
all))))
            ])

t_values = model_small.coef_ / se
p_values = 2 * (1 - stats.t.cdf(numpy.abs(t_values), df_small.price.shape[0] -
X_small.shape[1]))
p_values_df = pandas.DataFrame(p_values.T, X_small.columns, columns=['Coeffici
ent'])
p_values_df
```

Out[8]:

	Coefficient
miles	0.000000e+00
year_1990	8.197154e-01
year_1991	8.980567e-01
year_1992	6.124121e-01
year_1994	5.803959e-01
year_1996	8.901633e-01
year_1997	2.624632e-03
year_1998	2.656495e-01
year_1999	4.295245e-01
year_2000	3.760851e-02
year_2001	9.240521e-05
year_2002	1.524093e-02
year_2003	5.008985e-04
year_2004	1.232303e-11
year_2005	0.000000e+00
year_2006	0.000000e+00
year_2007	0.000000e+00
year_2008	0.000000e+00
year_2009	0.000000e+00
year_2010	0.000000e+00
year_2011	0.000000e+00
year_2012	0.000000e+00
year_2013	0.000000e+00
year_2014	0.000000e+00
year_2015	0.000000e+00
year_2016	6.838974e-14
year_2017	0.000000e+00
year_2018	0.000000e+00
year_2019	0.000000e+00
year_2020	0.000000e+00
year_2021	0.000000e+00
year_2022	0.000000e+00

At 95% confidence level, we can conclude that the following variables could have a zero valued coefficient:

In [9]:
```
p_values_df.loc[p_values_df.Coefficient > 1 - 0.95]
```

Out[9]:

	Coefficient
year_1990	0.819715
year_1991	0.898057
year_1992	0.612412
year_1994	0.580396
year_1996	0.890163
year_1998	0.265649
year_1999	0.429524