Звіт

Виконала студентка групи ПП-41/1

Терещук Анна

Перед початком виконання основних завдань проводимо дослідницький аналіз даних:

In [37]:
```
data_frame = pd.read_csv("Automobile_data.csv")
display(data_frame)
data_frame[data_frame["normalized-losses"] == "?"]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepow
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	1
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	1
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	1
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4	10.0	1
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	8.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	1
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	1
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	1
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.4	23.0	1
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	1

205 rows × 26 columns

In [13]:
```
data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       205 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    object
 19  stroke             205 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object
 22  peak-rpm           205 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

In [14]:
```
data_frame.isnull().sum()
```

Out[14]:
```
symboling            0
normalized-losses    0
make                 0
fuel-type            0
aspiration           0
num-of-doors         0
body-style           0
drive-wheels         0
engine-location      0
wheel-base           0
length               0
width                0
height               0
curb-weight          0
engine-type          0
num-of-cylinders     0
engine-size          0
fuel-system          0
bore                 0
stroke               0
compression-ratio    0
horsepower           0
peak-rpm             0
city-mpg             0
highway-mpg          0
price                0
dtype: int64
```

In [15]:
```
data_frame.duplicated().sum()
```

Out[15]: 0

In [17]:
```
data_frame
```

Out[17]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepow
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	1
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	1
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	1
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4	10.0	1
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	8.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	1
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	1
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	1
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.4	23.0	1
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	1

205 rows × 26 columns

Використовуємо методи для оцінки значущості факторів.

```
In [23]: num_val = transformed_data.select_dtypes(exclude='object')
         obj_val = transformed_data.select_dtypes(exclude='int64')
         obj_val = transformed_data.select_dtypes(exclude='float64')

         varianceThreshold = VarianceThreshold(threshold=(0.5))
         varianceThreshold.fit(num_val)
         selected_data = varianceThreshold.transform(num_val)
         selected_data
```

```
Out[23]: array([[1.6400e+02, 0.0000e+00, 3.0000e+00, ..., 2.4000e+01, 3.0000e+01,
                 1.3950e+04],
                [1.6400e+02, 0.0000e+00, 3.0000e+00, ..., 1.8000e+01, 2.2000e+01,
                 1.7450e+04],
                [1.5800e+02, 0.0000e+00, 3.0000e+00, ..., 1.9000e+01, 2.5000e+01,
                 1.7710e+04],
                ...,
                [9.5000e+01, 1.7000e+01, 3.0000e+00, ..., 1.8000e+01, 2.3000e+01,
                 2.1485e+04],
                [9.5000e+01, 1.7000e+01, 3.0000e+00, ..., 2.6000e+01, 2.7000e+01,
                 2.2470e+04],
                [9.5000e+01, 1.7000e+01, 3.0000e+00, ..., 1.9000e+01, 2.5000e+01,
                 2.2625e+04]])
```

```
In [24]: num_val
```

Out[24]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	pe r
3	0.8	164.0	0	1	0	0	3	1	0	99.8	...	109	4	3.19	3.40	10.0	102	55
4	0.8	164.0	0	1	0	0	3	0	0	99.4	...	136	4	3.19	3.40	8.0	115	55
6	0.6	158.0	0	1	0	0	3	1	0	105.8	...	136	4	3.19	3.40	8.5	110	55
8	0.6	158.0	0	1	1	0	3	1	0	105.8	...	131	4	3.13	3.40	8.3	140	55
10	0.8	192.0	1	1	0	1	3	2	0	101.2	...	108	4	3.50	2.80	8.8	101	58
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
200	0.2	95.0	17	1	0	0	3	2	0	109.1	...	141	4	3.78	3.15	9.5	114	54
201	0.2	95.0	17	1	1	0	3	2	0	109.1	...	141	4	3.78	3.15	8.7	160	53
202	0.2	95.0	17	1	0	0	3	2	0	109.1	...	173	4	3.58	2.87	8.8	134	55
203	0.2	95.0	17	0	1	0	3	2	0	109.1	...	145	2	3.01	3.40	23.0	106	48
204	0.2	95.0	17	1	1	0	3	2	0	109.1	...	141	4	3.78	3.15	9.5	114	54

159 rows × 26 columns

```
In [25]: obj_val
```

Out[25]:

	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	horsepower	peak-rpm	city-mpg	highway-mpg
3	0	1	0	0	3	1	0	2337	2	2	109	4	102	5500	24	30
4	0	1	0	0	3	0	0	2824	2	1	136	4	115	5500	18	22
6	0	1	0	0	3	1	0	2844	2	1	136	4	110	5500	19	25
8	0	1	1	0	3	1	0	3086	2	1	131	4	140	5500	17	20
10	1	1	0	1	3	2	0	2395	2	2	108	4	101	5800	23	29
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	17	1	0	0	3	2	0	2952	2	2	141	4	114	5400	23	28
201	17	1	1	0	3	2	0	3049	2	2	141	4	160	5300	19	25
202	17	1	0	0	3	2	0	3012	4	3	173	4	134	5500	18	23
203	17	0	1	0	3	2	0	3217	2	3	145	2	106	4800	26	27
204	17	1	1	0	3	2	0	3062	2	2	141	4	114	5400	19	25

Наступний метод обирає k найвпливовіших факторів:

```
In [28]: columns_array = transformed_data.columns
         kbest = sklearn.feature_selection.SelectKBest(sklearn.feature_selection.chi2, k=5)
         transformed_data_kbest = kbest.fit_transform(num_val.drop(["price"], axis=1),
                                                      num_val["price"])
         selected_features = kbest.get_feature_names_out(columns_array[:len(columns_array) - 1])
         transformed_data_kbest = pd.DataFrame(transformed_data_kbest, columns=selected_features)
         transformed_data_kbest
```

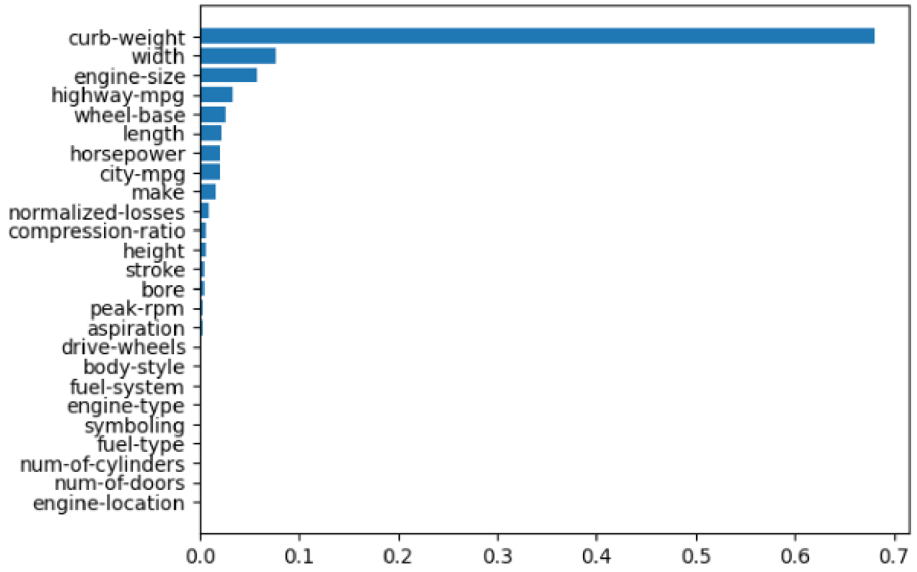
Out[28]:

	normalized-losses	curb-weight	engine-size	horsepower	peak-rpm
0	164.0	2337.0	109.0	102.0	5500.0
1	164.0	2824.0	136.0	115.0	5500.0
2	158.0	2844.0	136.0	110.0	5500.0
3	158.0	3086.0	131.0	140.0	5500.0
4	192.0	2395.0	108.0	101.0	5800.0
...	...	...	...	...	...
154	95.0	2952.0	141.0	114.0	5400.0
155	95.0	3049.0	141.0	160.0	5300.0
156	95.0	3012.0	173.0	134.0	5500.0
157	95.0	3217.0	145.0	106.0	4800.0
158	95.0	3062.0	141.0	114.0	5400.0

159 rows × 5 columns

Зображуємо важливість кожної метрики за допомогою RandomForestRegressor:

```
In [33]: feature_importances = tree_classifier.feature_importances_
         indices = np.argsort(feature_importances)
         plt.yticks(range(len(indices)), np.array(num_val.drop(["price"], axis=1).columns)[indices])
         plt.barh(range(len(indices)), feature_importances[indices])
         plt.show()
```



На наступному графіку можна побачити загальну кореляцію метрик, що відображає значення із найбільшою кореляцією:

```
Out[36]: <AxesSubplot: >
```