dwl_final

May 5, 2021

0.0.1 I created a scraping script using libraries requests and beautifulSoup. Also I created empty lists and wrote down all the information there. When we go to the csv file, we will see 10 columns and 600 lines in it. It contains following information about each car: Car, Price, Year, Region, Body, Engine_volume, Mileage, Transmission, Steering_wheel, Color.

[11]:
```
import requests
import re
from bs4 import BeautifulSoup
import pandas as pd


car = []
price = []
year_of_manufacture = []
region = []
body = []
engine_volume = []
mileage = []
transmission = []
steering_wheel = []
color = []

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
 ↪537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36'}
for i in range(30):
    results = requests.get(f'https://kolesa.kz/cars/avtomobili-s-probegom/?
 ↪auto-car-order=1&auto-car-volume[from]=1&auto-car-volume[to]=9&page={i}')
    soup = BeautifulSoup(results.text, 'html.parser')
    for j in soup.find_all('span',class_='a-el-info-title'):
        car.append(j.text.strip())
        results2 = requests.get('https://kolesa.kz'+ j.
 ↪find('a',class_='list-link ddl_product_link')['href'])
        soup2 = BeautifulSoup(results2.text, 'html.parser')

        p = re.sub("[^0-9]", "", soup2.find('div', class_='offer__price').text.
 ↪replace(u'\xa0', ' '))
        price.append(int(p))
```

```
        year_of_manufacture.append(int(soup2.find('span', class_='year').text.
↪strip()))
        region.append(soup2.find_all('dd',class_='value')[0].text.strip())
        body.append(soup2.find_all('dd',class_='value')[1].text.strip())

        engine_volume.append(re.compile(r'[^\d.]+').sub('', (soup2.
↪find_all('dd', class_='value')[2].text)))

        m = soup2.find_all('dd',class_='value')[3].text.strip()
        if m[-1] == ' ':
            m2 = re.sub("[^0-9]", "", m)
            mileage.append(int(m2))
            transmission.append(soup2.find_all('dd',class_='value')[4].text.
↪strip())
            steering_wheel.append(soup2.find_all('dd',class_='value')[5].text.
↪strip())
            color.append(soup2.find_all('dd',class_='value')[6].text.strip())
        else:
            mileage.append(0)
            transmission.append(soup2.find_all('dd',class_='value')[3].text.
↪strip())
            steering_wheel.append(soup2.find_all('dd',class_='value')[4].text.
↪strip())
            color.append(soup2.find_all('dd',class_='value')[5].text.strip())

myDict = {}

myDict['Car'] = car
myDict['Price'] = price
myDict['Year'] = year_of_manufacture
myDict['Region'] = region
myDict['Body'] = body
myDict['Engine_volume'] = engine_volume
myDict['Mileage'] = mileage
myDict['Transmission'] = transmission
myDict['Steering_wheel'] = steering_wheel
myDict['Color'] = color

df = pd.DataFrame.from_dict(myDict)
df.to_excel('kolesa2.xlsx', encoding='utf-8-sig')
df.to_csv('kolesa.csv', encoding='utf-8-sig')

df
```

[11]:
```
                         Car     Price  Year               Region  \
0          Chevrolet Tracker   5850000  2015
1              Opel Frontera   1650000  1994
2    Renault Sandero Stepway   3990000  2014
3        Toyota Land Cruiser  16800000  2010
4          Mitsubishi Pajero   8000000  2007
..                       …       …   …                  …
595             Daewoo Nexia    950000  2007
596                          6400000  2015
597                  BMW X5   7000000  2006    -    (   )
598   Volkswagen Transporter   1600000  1991
599      Mercedes-Benz E 230   1800000  1992    -    (   )

             Body Engine_volume  Mileage Transmission Steering_wheel  \
0                         1.8    46000
1                        2.3        0
2                          1.6   110000
3                          4   220000
4                          3   137000
..           …             …       …           …             …
595                        1.5   190000
596                       2.9        0
597                      4.8   220000
598                         2   222222
599                      2.3      222

                  Color
0
1
2
3
4
..                  …
595
596
597
598
599

[600 rows x 10 columns]
```

0.0.2 Return DataFrame with duplicate rows removed and remove null properties.

[52]:
```
df = pd.read_csv('kolesa.csv')
df = df.drop('Unnamed: 0', axis='columns')
df= df.dropna()
df= df.drop_duplicates()
```

```
df.head()
```

[52]:
```
                       Car     Price  Year         Region         Body  \
0        Chevrolet Tracker   5850000  2015
1            Opel Frontera   1650000  1994
2  Renault Sandero Stepway   3990000  2014
3      Toyota Land Cruiser  16800000  2010
4        Mitsubishi Pajero   8000000  2007

   Engine_volume  Mileage Transmission Steering_wheel                Color
0            1.8    46000
1            2.3        0
2            1.6   110000
3            4.0   220000
4            3.0   137000
```

[53]:
```
df.sort_values(by=['Price'])
```

[53]:
```
                          Car     Price  Year               Region  \
192              (Lada) 2107    350000  2006
79     (Lada) 2114 (    )    380000  2003
238              (Lada) 2107    400000  2008
564                     968    500000  1989
118                Opel Omega    550000  1987
..                        …       …   …                  …
412        Toyota Land Cruiser  31600000  2017    -    (   )
257              Lexus RX 350  33500000  2019
228  Toyota Land Cruiser Prado  36200000  2021
55                     BMW X5  44300000  2019
578    Land Rover Range Rover  55000000  2018    -    (   )

            Body  Engine_volume  Mileage Transmission Steering_wheel  \
192                        1.5        0
79                        1.6   280000
238                        1.7        0
564                        1.2    80000
118                        2.0        0
..           …             …      …           …             …
412                      4.6    53000
257                       3.5     9000
228                      4.0        0
55                        3.0       40
578                      3.0    25500

            Color
192
79
```

```
238
564
118
..              …
412
257
228
55
578

[517 rows x 10 columns]
```

[54]:
```
df.groupby(['Year']).first()
```

[54]:

Year	Car	Price	Region	Body
1987	Opel Omega	550000		
1989	Mazda 626	1450000		
1990	Mazda 626	850000		
1991	Opel Vectra	950000		
1992	Toyota Land Cruiser Prado	6800000		
1993	Volkswagen Passat	1550000		
1994	Opel Frontera	1650000		
1995	Mitsubishi RVR	1600000		
1996	Toyota Caldina	2250000	-	
1997	Toyota Mark II	2550000		
1998	Mercedes-Benz E 280	2870000		
1999	Mazda MPV	3000000		
2000	Daewoo Matiz	900000		
2001	Opel Zafira	2300000		
2002	Lexus ES 300	5950000		
2003	Toyota Camry	3800000		
2004	Land Rover Range Rover	3000000		
2005	Land Rover Range Rover	5800000		
2006	Toyota Land Cruiser	7100000		
2007	Mitsubishi Pajero	8000000		
2008	Mitsubishi Pajero	7500000		
2009	Daewoo Nexia	1350000		
2010	Toyota Land Cruiser	16800000		
2011	Hyundai Sonata	3700000		
2012	(Lada) 2190 ( )	2190000	-	( )
2013	Nissan Juke	5950000		
2014	Renault Sandero Stepway	3990000		
2015	Chevrolet Tracker	5850000		
2016	Chevrolet Spark	3500000		
2017	Toyota Tundra	23000000		
2018	Hyundai Tucson	11350000		

```
2019                Toyota Camry  13900000
2020                Toyota Camry  15800000
2021  Toyota Land Cruiser Prado  36200000

      Engine_volume  Mileage Transmission Steering_wheel                Color
Year
1987            2.0        0
1989            2.0   310915
1990            2.0        0
1991            2.0   145000
1992            4.0   100000
1993            1.8   250000
1994            2.3        0
1995            2.0   444444
1996            1.8        0
1997            2.0        0
1998            2.8        0
1999            2.5        0
2000            8.0        0
2001            2.2   160000
2002            3.0   154000
2003            2.4        0
2004            4.4        0
2005            4.4   165000
2006            4.7   182000
2007            3.0   137000
2008            3.0   219000
2009            1.6   126000
2010            4.0   220000
2011            2.4   197102
2012            1.6   146000
2013            1.6    39000
2014            1.6   110000
2015            1.8    46000
2016            1.0   135000
2017            4.6        0
2018            2.0    53000
2019            2.5    18000
2020            2.5     1955
2021            4.0        0
```

[55]:
```
df.groupby(['Year']).get_group(2008)
```

[55]:
```
                          Car     Price  Year              Region  \
7           Mitsubishi Pajero   7500000  2008
14          Toyota Highlander   8800000  2008
41              Chrysler 300C   5500000  2008
```

```
78                          3199000  2008
151    (Lada) 2114 (    )   1000000  2008
182              Toyota Camry   4000000  2008
185              Toyota Camry   4000000  2008
206    (Lada) 2113 (    )    550000  2008
217             Hyundai Accent   2100000  2008
222                    BMW X6   7500000  2008
225              Daewoo Nexia    550000  2008    -    (   )
229    (Lada) 2114 (    )    650000  2008
238              (Lada) 2107    400000  2008
250            Hyundai Tuscani   3100000  2008
253    (Lada) 2114 (    )    900000  2008
356                          4400000  2008    -    (   )
361       Volkswagen Golf Plus   3300000  2008
375              Hyundai Getz   2350000  2008
407    (Lada) 2114 (    )    950000  2008
421          Chevrolet Captiva   4800000  2008
424    Land Rover Range Rover  11200000  2008    -    (   )
439            SsangYong Musso   2400000  2008
457          Volkswagen Tiguan   6000000  2008
467    (Lada) 2172 (    )    900000  2008
478              Nissan Teana   5000000  2008
510    (Lada) 2114 (    )    890000  2008
513      (Lada) 2170 (   )   1250000  2008
527  Toyota Land Cruiser Prado  11800000  2008
548              Toyota Camry   3999000  2008
561       Toyota Land Cruiser  15700000  2008
568              Toyota Camry   5500000  2008
575         Toyota Highlander  10000000  2008
576                   Audi Q7   5800000  2008    -    (   )
582       Toyota Land Cruiser  12000000  2008

           Body  Engine_volume  Mileage Transmission Steering_wheel  \
7                      3.0   219000
14                      3.5   210000
41                        2.7   205000
78                        2.9   395700
151                      1.6        0
182                       3.5        0
185                       3.5        0
206                      1.5      320
217                       1.6   180000
222                      3.0   139000
225                       1.5   150000
229                      1.5   190000
238                       1.7        0
250                        2.0   117000
```

253		1.5	0
356		2.9	0
361		1.4	215000
375		1.4	160000
407		1.5	165000
421		2.4	141000
424		4.2	185000
439		3.2	0
457		2.0	167000
467		1.6	203630
478		3.5	199000
510		1.5	0
513		1.6	0
527		4.0	0
548		2.4	0
561		4.7	166000
568		2.4	0
575		3.5	0
576		4.2	110
582		4.7	212000

	Color
7	
14	
41	
78	
151	
182	
185	
206	
217	
222	
225	
229	
238	
250	
253	
356	
361	
375	
407	
421	
424	
439	
457	
467	
478	

510
513
527
548
561
568
575
576
582

[57]:
```
import sklearn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score,␣
↪GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score, precision_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
```

0.0.3 Data preprocessing. Decision trees. We encode categorical features using an ordinal encoding scheme. Encode categorical features as a one-hot numeric array. LabelEncoder can be used to normalize labels. It can also be used to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels.

[58]:
```
data = df.drop('Price',axis='columns')
target = df['Price']
car_n = LabelEncoder()
body_n = LabelEncoder()
tr_n = LabelEncoder()
data['Car_2'] = car_n.fit_transform(data['Car'])
data['Body_2'] = body_n.fit_transform(data['Body'])
data['Transmission_2'] = tr_n.fit_transform(data['Transmission'])
data
```

[58]:

	Car	Year	Region	Body	\
0	Chevrolet Tracker	2015			
1	Opel Frontera	1994			
2	Renault Sandero Stepway	2014			
3	Toyota Land Cruiser	2010			
4	Mitsubishi Pajero	2007			
..	…	…	…	…	

```
595             Daewoo Nexia  2007
596                          2015
597                   BMW X5  2006    -    (   )
598   Volkswagen Transporter  1991
599      Mercedes-Benz E 230  1992    -    (   )

     Engine_volume  Mileage Transmission Steering_wheel                Color  \
0              1.8    46000
1              2.3        0
2              1.6   110000
3              4.0   220000
4              3.0   137000
..             …      …          …             …                     …
595            1.5   190000
596            2.9        0
597            4.8   220000
598            2.0   222222
599            2.3      222

     Car_2  Body_2  Transmission_2
0       23       2               0
1      123       0               2
2      134      12               0
3      156       0               0
4      109       0               0
..     …     …              …
595     30       8               2
596    192      11               2
597     12       2               0
598    174       6               2
599     86      10               2

[517 rows x 12 columns]
```

[59]:
```
len(df.Car.unique())
```

[59]: 197

[60]:
```
data = data[['Year','Engine_volume', 'Body_2']]
```

[61]:
```
X, y = data, target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3)
clf = LinearRegression()
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

[61]: 0.6259556263293345

[62]:
```
X_test
```

[62]:
```
     Year  Engine_volume  Body_2
144  2005            2.4       8
62   2018            2.0       2
354  2019            5.6       0
248  2009            4.7       0
505  2005            1.6      12
..    …              …         …
536  1989            2.3       8
179  2007            1.6      12
416  2017            2.0       2
485  1999            4.0       0
471  2010            1.6       8

[173 rows x 3 columns]
```

[63]:
```
clf.predict(X_test)[:3]
```

[63]:
```
array([ 4172022.42105854, 11208486.00488257, 19079302.39893699])
```

[64]:
```
y_test[:3]
```

[64]:
```
144     4600000
62     11350000
354    29000000
Name: Price, dtype: int64
```

[65]:
```
model = DecisionTreeClassifier()
model.fit(data, target)
model.score(data,target)
```

[65]:
```
0.7156673114119922
```

[66]:
```
df.tail()
```

[66]:
```
                        Car    Price  Year               Region       Body  \
595           Daewoo Nexia   950000  2007
596                        6400000  2015
597                 BMW X5  7000000  2006    -    (    )
598  Volkswagen Transporter  1600000  1991
599     Mercedes-Benz E 230  1800000  1992    -    (    )

     Engine_volume  Mileage Transmission Steering_wheel            Color
595            1.5   190000
596            2.9        0
597            4.8   220000
598            2.0   222222
```

```
599            2.3      222
```

[67]:
```
model.predict([[2007,1.5,8]])
```

[67]:
```
array([950000], dtype=int64)
```

[68]:
```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},]
svc = svm.SVC()
grid_svc = GridSearchCV(svc, param_grid, cv=3)
grid_svc.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\model_selection\_split.py:657: Warning: The least populated
class in y has only 1 members, which is too few. The minimum number of members
in any class cannot be less than n_splits=3.
  % (min_groups, self.n_splits)), Warning)
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\model_selection\_search.py:814: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will
be removed in 0.24. This will change numeric results when test-set sizes are
unequal.
  DeprecationWarning)
```

[68]:
```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='auto_deprecated', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='warn', n_jobs=None,
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                         {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                          'kernel': ['rbf']}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[70]:
```
clf_svc = grid_svc.best_estimator_
print(clf_svc)
```

```
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[71]:
```
param_grid = [{'n_neighbors':[2,3,4,5,6]}]
knn = KNeighborsClassifier()
```

```
grid_knn = GridSearchCV(knn, param_grid, cv=3)
grid_knn.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\model_selection\_split.py:657: Warning: The least populated
class in y has only 1 members, which is too few. The minimum number of members
in any class cannot be less than n_splits=3.
  % (min_groups, self.n_splits)), Warning)
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\model_selection\_search.py:814: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will
be removed in 0.24. This will change numeric results when test-set sizes are
unequal.
  DeprecationWarning)
```

[71]:
```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='warn', n_jobs=None,
             param_grid=[{'n_neighbors': [2, 3, 4, 5, 6]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[72]:
```
clf_knn = grid_knn.best_estimator_
print(clf_knn)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform')
```

[73]:
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
for model in [clf_knn, clf_svc, decision_tree]:
    y_pred = model.predict(X_test)
    score = precision_score(y_test.to_numpy().T, y_pred, average='weighted')
    print(model, 'score:', score)
```

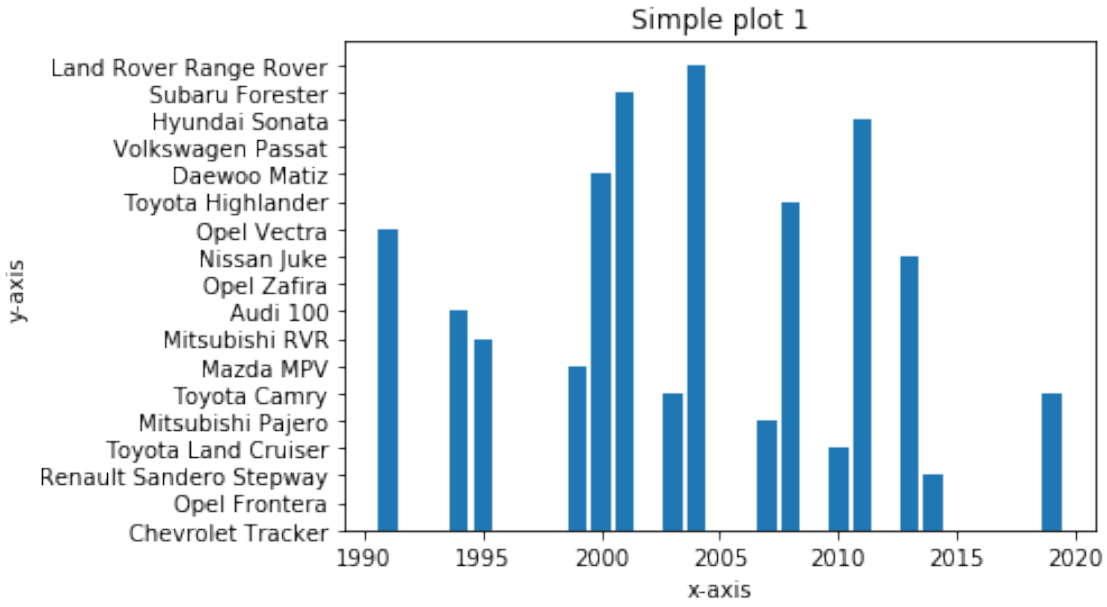
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform') score: 0.03177814478392513
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False) score: 0.001707238866116075
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
```

```
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best') score:
0.024116891457931923
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
```

[74]:

```
df = pd.read_csv('kolesa.csv')
df = df[:40]
x = df['Year']
y = df['Car']
plt.bar(x, y)

plt.title("Simple plot 1")
plt.ylabel("y-axis")
plt.xlabel("x-axis")

plt.show()
```



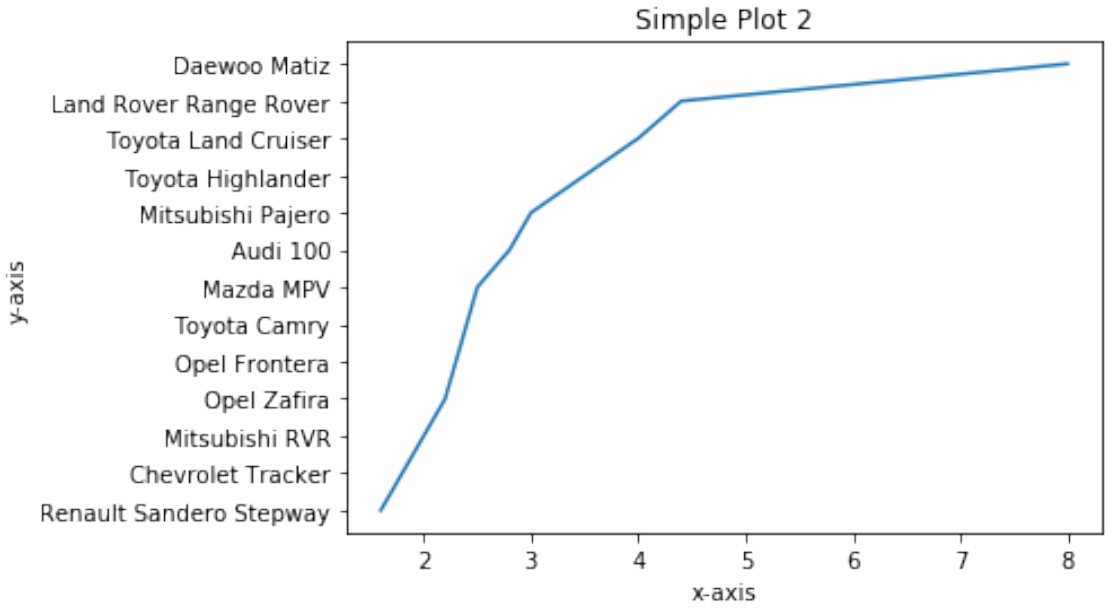
[75]:
```
df = df.groupby(['Engine_volume']).first()
df
```

[75]:

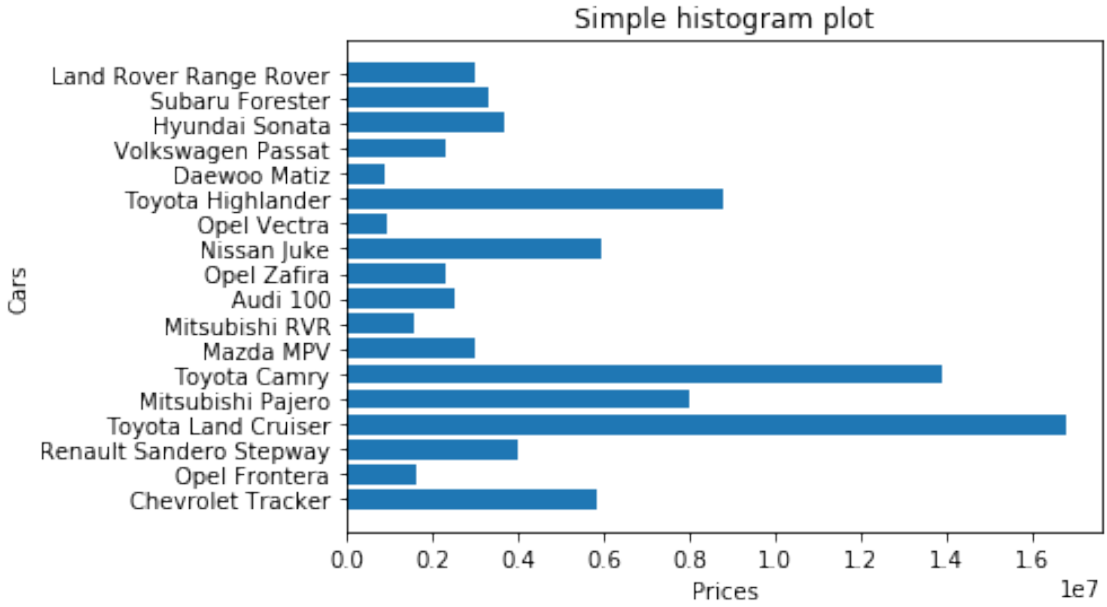
Engine_volume	Unnamed: 0	Car	Price	Year	Region	Body	Mileage	Transmission
1.6	2	Renault Sandero Stepway	3990000	2014			110000	
1.8	0	Chevrolet Tracker	5850000	2015			46000	
2.0	8	Mitsubishi RVR	1600000	1995			444444	
2.2	10	Opel Zafira	2300000	2001			160000	
2.3	1	Opel Frontera	1650000	1994			0	
2.4	5	Toyota Camry	3800000	2003			0	
2.5	6	Mazda MPV	3000000	1999				
2.8	9	Audi 100	2500000	1994				
3.0	4	Mitsubishi Pajero	8000000	2007				
3.5	14	Toyota Highlander	8800000	2008				
4.0	3	Toyota Land Cruiser	16800000	2010				
4.4	19	Land Rover Range Rover	3000000	2004				
8.0	15	Daewoo Matiz	900000	2000				

```
2.5                                      0
2.8                                 300000
3.0                                 137000
3.5                                  210000
4.0                                 220000
4.4                                    0
8.0                                        0

               Steering_wheel                 Color
Engine_volume
1.6
1.8
2.0
2.2
2.3
2.4
2.5
2.8
3.0
3.5
4.0
4.4
8.0
```

[76]:
```
x = df['Car']
y = df.index
plt.plot(y,x)
plt.title("Simple Plot 2")
plt.ylabel("y-axis")
plt.xlabel("x-axis")

plt.show()
```



[77]:
```
df = pd.read_csv('kolesa.csv')
df = df[:20]
plt.barh(df['Car'], df['Price'])
plt.title("Simple histogram plot")
plt.xlabel('Prices')
plt.ylabel('Cars')
plt.show()
```



[78]:
```
df = pd.read_csv('kolesa.csv')
df = df.groupby(['Body']).first()
df[2:13]
```

[78]:

Body	Unnamed: 0	Car	Price	Year	Region	Engine_volume	Mileage	Transmission	Steering_wheel	Color
	1	Opel Frontera	1650000	1994		2.3	0			
	244	BMW 650	10000000	2007		4.8	137000			
	0	Chevrolet Tracker	5850000	2015		1.8	46000			
	250	Hyundai Tuscani	3100000	2008		2.0	117000			
	176	Toyota Carina E	1900000	1992		2.0	0			
	86	Volkswagen Transporter	1800000	1997		1.6	0			
	6	Mazda MPV	3000000	1999		2.5	0			
	51	Toyota Tundra	23000000	2017		4.6	0			
	5	Toyota Camry	3800000	2003		2.4	0			
	587	Toyota Supra	3000000	1991						
	9	Audi 100	2500000	1994						
-	( )					2.5	0			
						2.8	300000			

[25]:
```
import numpy as np
x = df.index[2:13]
y = df['Region'][2:13]

plt.ylabel("Body")
plt.xlabel("Region")
plt.title("Simple Scatter plot")
rng = np.random.RandomState(0)
x = df.index[2:10]
y = df['Region'][2:10]
colors = rng.rand(8)
sizes = 1000 * rng.rand(8)
plt.scatter(y, x, c=colors, s=sizes)
plt.colorbar()
```

[25]:
```
<matplotlib.colorbar.Colorbar at 0x259e984b548>
```