DSCI 631-assignment3

March 14, 2021

#####

Drexel University

College of Computing and Informatics

DSCI 631: Applied Machine Learning

Assignment 3

Due Date: Sunday, March 14, 2021

This assignment counts for 18% of the final grade

DON'T FORGET TO PUT YOUR TEAM NUMBER AND MEMBERS' NAMES BELOW

0.0.1 TEAM NUMBER:

0.0.2 TEAM MEMBERS:

0.0.3 A. Assignment Overview

This assignment provides the opportunity for you to practice with various skills in unsupervised learning, supervised learning learning and feature selection.

0.0.4 B. What to Hand In

Sumbit a completed this Jupyter notebook.

0.0.5 C. How to Hand In

Submit your Jupyter notebook file through the course website in the Blackboard Learn system.

0.0.6 D. When to Hand In

1. Submit your assignment no later than 11:59 pm in the due date.
2. There will be a 10% (absolute value) deduction for each day of lateness, to a maximum of 3 days; assignments will not be accepted beyond that point. Missing work will earn a zero grade.

0.0.7 E. Written Presentation Requirements (if applicable)

Images must be clear and legible. Assignments will be judged on the basis of visual appearance, grammatical correctness, and quality of writing, as well as their contents. Please make sure that

the text of your assignments is well-structured, using paragraphs, full sentences, and other features of well-written presentation.

0.0.8 F. Academic Honesty

Each student is required to submit the Academic Honesty Form at the beginning of the term to cover all the deliverables (for example: assignments, projects, quizzes). Each piece of work must be original. That means, individual quizzes must be done individually without discussing and collaborating with anybody else. Team assignments must be written and programmed by your own team members. No team should copy any piece of work from other teams. The Drexel University Academic Honesty Rules and Procedures (as stated in the student handbook) will be adhered to strictly.

0.0.9 G. Marking Schemes:

Marking assignments will be based on several aspects: presentation, correctness and coding styles.

For programming questions, 10% of the mark will be judged on the coding style.

The following is a set of guidelines for the coding style in this course: 1. Write a good comment. 2. Use appropriate indentations to indicate control flows and blocks of code. 3. When breaking up a long line, break it before an operator, not after.

0.0.10 H. Answer the following questions:

Your answer should be combined with code and brief text answer. Please ensure that your Jupyter notebook does not have too much spurious output. If you like, you can share your notebook in progress with me on Kaggle: leiwangv (lw474@drexel.edu)

0.0.11 Data for part 1 in this assignment:

- URL: http://odds.cs.stonybrook.edu/annthyroid-dataset/
- In this assignment, you should work without the ground truth labels as much as possible. Often inspecting and visualizing the data is the only way to understand the result of clustering and outlier detection.
- For Questions 1-3 visualization, you should look at the plots before making use of the ground truth labels first, then use the ground truth labels to color the points in your plots.

[1]:
```
# You can import the data as follows after download
# Do not submit the data file to Bb Learn
#import scipy.io
#data = scipy.io.loadmat('annthyroid.mat')
```

Question 1-1: Create plots to visualize the distribution of all features (both jointly and for each class). Describe your observation.

Then visualize the data using PCA (top two principal components). Make another plot of explained variance (%) in PCA. Discuss what would be a good threshold for the number of principal components if you wanted to reduce the dimensionality of the data?

[ ]:

Question 1-2: Create plots to visualize the data using t-SNE. Would parameter tuning help to gain a better visualization? Discuss your results and findings.

[ ]:

Question 1-3: Use different clustering algorithms to cluster the data: K-Means, DBSCAN, Agglomerative clustering, Gaussian Mixture. For each algorithm, tune the parameters for a reasonable outcome, then document your tuning procedure.

Pay attention to the sizes of the clusters created. Inspect the outcome, discuss any resulting clusters are meaningful.

[ ]:

Quesetion 1-4: Evaluate your results with the ground truth label (outlier vs. inlier class) using the two scores: Normalized Mutual Information score and Adjust Rand Index. Discuss how well did you do in previous question.

[ ]:

Question 1-5: Supervised learning with imbalanced data: We are using the target variable here. Split the data in train and test set first.

Utilize Random Forest Classifier and another self-selected classification methods, compare the performance on test set, interpret the results in terms of AUC and average precision.

Tune the parameters for Random Forest Classifier, does changing the `class-weight` to `balanced` help? Discuss your results and findings.

[ ]:

0.0.12 Data for part 2 in this assignment (price prediction):

- URL: https://www.kaggle.com/austinreese/craigslist-carstrucks-data
- On Kaggle Notebook, you can add the data set by searching the above URL
- You do not have to use the whole dataset, it is strongly recommended that you subsample the data while developing your solution.
- The goal of this part is to provide a realistic setting for a machine learning task. Therefore instructions will not specify the exact steps to carry out. Instead, it is part of the assignment to identify promising features, models and preprocessing methods and apply them as appropriate.

[2]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_openml
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import scale, StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib as mpl

# Ignore useless warnings
import warnings
warnings.filterwarnings(action="ignore", message="^internal gelsd")
```

[3]:
```
# On Kaggle Notebook, after adding the data, you can import the data as Pandas␣
 ↪DataFrame as follows
import pandas as pd
df = pd.read_csv("/Users/dustin.ellis/Desktop/vehicles.csv")
```

[4]:
```
df.head()
```

[4]:

	Unnamed: 0	id	url	region	region_url	price	year	manufacturer	model	condition	…	drive	size	type	paint_color
0	0	7240372487	https://auburn.craigslist.org/ctd/d/auburn-uni…	auburn	https://auburn.craigslist.org	35990	2010.0	chevrolet	corvette grand sport	good	…	rwd	NaN	other	NaN
1	1	7240309422	https://auburn.craigslist.org/cto/d/auburn-201…	auburn	https://auburn.craigslist.org	7500	2014.0	hyundai	sonata	excellent	…	fwd	NaN	sedan	NaN
2	2	7240224296	https://auburn.craigslist.org/cto/d/auburn-200…	auburn	https://auburn.craigslist.org	4900	2006.0	bmw	x3 3.0i	good	…	NaN	NaN	SUV	blue
3	3	7240103965	https://auburn.craigslist.org/cto/d/lanett-tru…	auburn	https://auburn.craigslist.org	2000	1974.0	chevrolet	c-10	good	…	rwd	full-size	pickup	blue
4	4	7239983776	https://auburn.craigslist.org/cto/d/auburn-200…	auburn	https://auburn.craigslist.org	19500	2005.0	ford	f350 lariat	excellent	…	4wd	full-size	pickup	blue

```
                                              image_url  \
0  https://images.craigslist.org/00N0N_ipkbHVZYf4…
1  https://images.craigslist.org/00s0s_gBHYmJ5o7y…
2  https://images.craigslist.org/00B0B_5zgEGWPOrt…
3  https://images.craigslist.org/00M0M_6o7KcDpArw…
4  https://images.craigslist.org/00p0p_b95l1EgUfl…

                                         description state        lat  \
0  Carvana is the safer way to buy a car During t…    al  32.590000
1  I'll move to another city and try to sell my c…    al  32.547500
2  Clean 2006 BMW X3 3.0I.  Beautiful and rare Bl…    al  32.616807
3  1974 chev. truck (LONG BED) NEW starter front …    al  32.861600
4  2005 Ford F350 Lariat (Bullet Proofed). This t…    al  32.547500

         long              posting_date
0 -85.480000  2020-12-02T08:11:30-0600
1 -85.468200  2020-12-02T02:11:50-0600
2 -85.464149  2020-12-01T19:50:41-0600
3 -85.216100  2020-12-01T15:54:45-0600
4 -85.468200  2020-12-01T12:53:56-0600

[5 rows x 26 columns]
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 458213 entries, 0 to 458212
Data columns (total 26 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Unnamed: 0    458213 non-null  int64
 1   id            458213 non-null  int64
 2   url           458213 non-null  object
 3   region        458213 non-null  object
 4   region_url    458213 non-null  object
 5   price         458213 non-null  int64
 6   year          457163 non-null  float64
 7   manufacturer  439993 non-null  object
 8   model         453367 non-null  object
 9   condition     265273 non-null  object
 10  cylinders     287073 non-null  object
 11  fuel          454976 non-null  object
 12  odometer      402910 non-null  float64
 13  title_status  455636 non-null  object
 14  transmission  455771 non-null  object
 15  VIN           270664 non-null  object
```

```
 16  drive         324025 non-null  object
 17  size          136865 non-null  object
 18  type          345475 non-null  object
 19  paint_color   317370 non-null  object
 20  image_url     458185 non-null  object
 21  description   458143 non-null  object
 22  state         458213 non-null  object
 23  lat           450765 non-null  float64
 24  long          450765 non-null  float64
 25  posting_date  458185 non-null  object
dtypes: float64(4), int64(3), object(19)
memory usage: 90.9+ MB
```

[6]:
```
df.columns
```

[6]:
```
Index(['Unnamed: 0', 'id', 'url', 'region', 'region_url', 'price', 'year',
       'manufacturer', 'model', 'condition', 'cylinders', 'fuel', 'odometer',
       'title_status', 'transmission', 'VIN', 'drive', 'size', 'type',
       'paint_color', 'image_url', 'description', 'state', 'lat', 'long',
       'posting_date'],
      dtype='object')
```

[7]:
```
#examine pricing amounts. Many listing have $0 pricing, which could be due to␣
↪laziness of the poster or it could not be a legitimate post.

df['price'].value_counts().sort_values(ascending = True)
```

[7]:
```
12422        1
3782         1
14273        1
22080        1
6860         1
          ...
3500      3680
7995      3701
5995      3760
6995      4003
0        33753
Name: price, Length: 16924, dtype: int64
```

[8]:
```
#check for null values

null_counts = df.isnull().sum()
null_counts[null_counts > 0].sort_values(ascending=False)
```

[8]:
```
size            321348
condition       192940
```

```
VIN             187549
cylinders       171140
paint_color     140843
drive           134188
type            112738
odometer         55303
manufacturer     18220
lat               7448
long              7448
model             4846
fuel              3237
title_status      2577
transmission      2442
year              1050
description         70
image_url           28
posting_date        28
dtype: int64
```

[9]:
```
#drop null values
df.dropna(axis=0, how='any', inplace=True)
```

[10]:
```
#preview and confirm nulls were dropped
df
```

[10]:

	Unnamed: 0	id	url	region
19	19	7235942858	https://auburn.craigslist.org/cto/d/auburn-202…	auburn
91	91	7240569685	https://bham.craigslist.org/ctd/d/cartersville…	birmingham
92	92	7240567296	https://bham.craigslist.org/ctd/d/summerville-…	birmingham
93	93	7240566811	https://bham.craigslist.org/ctd/d/summerville-…	birmingham
95	95	7240566722	https://bham.craigslist.org/ctd/d/summerville-…	birmingham
…	…	…	…	…
458154	458154	7240979817	https://milwaukee.craigslist.org/ctd/d/mukwona…	milwaukee
458195	458195	7240981040	https://sheboygan.craigslist.org/ctd/d/manitow…	sheboygan
458202	458202	7240989873		
458204	458204	7240975107		
458211	458211	7240600465		

```
458202  https://wausau.craigslist.org/ctd/d/auburndale…      wausau
458204  https://wausau.craigslist.org/ctd/d/auburndale…      wausau
458211  https://wyoming.craigslist.org/cto/d/sheridan-…     wyoming

                            region_url  price    year   manufacturer  \
19         https://auburn.craigslist.org  47000  2020.0           jeep
91           https://bham.craigslist.org  24999  2016.0  mercedes-benz
92           https://bham.craigslist.org  41900  2016.0           jeep
93           https://bham.craigslist.org  23900  2005.0            gmc
95           https://bham.craigslist.org  18900  2012.0      chevrolet
…                                      …    …      …              …
458154  https://milwaukee.craigslist.org      0  2015.0           jeep
458195  https://sheboygan.craigslist.org  20488  2010.0            gmc
458202     https://wausau.craigslist.org   4995  2005.0          buick
458204     https://wausau.craigslist.org   4495  2006.0          buick
458211    https://wyoming.craigslist.org   1300  2008.0           jeep

                           model  condition  … drive       size    type  \
19                      gladiator   like new  …   4wd  full-size  pickup
91                benz c300 4matic   like new  …   rwd  full-size   sedan
92                        wrangler       good  …   4wd  full-size     SUV
93                     sierra 3500       good  …   4wd  full-size   truck
95                  silverado 3500hd       good  …   rwd  full-size   truck
…                                 …        …  …   …        …     …
458154  wrangler unlimited sahara  excellent  …   4wd  full-size     SUV
458195                  sierra 1500  excellent  …   4wd   mid-size   truck
458202               rendezvous cx       good  …   fwd    compact     SUV
458204                 lacrosse cx       good  …   fwd   mid-size   sedan
458211              grand cherokee       good  …   4wd   mid-size     SUV

       paint_color                                          image_url  \
19            grey  https://images.craigslist.org/00909_kPkElEcTZ5…
91           white  https://images.craigslist.org/00M0M_jvdDIzsekt…
92           white  https://images.craigslist.org/00M0M_cquuYs5OeK…
93           white  https://images.craigslist.org/00202_jIsmWQ0vhC…
95           white  https://images.craigslist.org/00y0y_NL74aBjE1B…
…                …                                                …
458154         red  https://images.craigslist.org/00e0e_bk64cPmhJ4…
458195       black  https://images.craigslist.org/00P0P_bnajhQgGn3…
458202        grey  https://images.craigslist.org/00j0j_fzzhvpvful…
458204       black  https://images.craigslist.org/01010_cdS54li18Y…
458211       white  https://images.craigslist.org/00C0C_flONW1IeJw…

                                              description state        lat  \
19      I'm putting up for sale my Jeep Gladiator. I j…    al  32.611442
91      2016 Mercedes BENZ C300-4MATIC-AWD YES ONLY 18…    al  34.206619
92      2016 Jeep Wrangler Unlimited Sahara 4WD - $41,…    al  34.466560
```

```
93      2005 GMC Sierra 3500 SLT Crew Cab 4WD - $23,90…    al  34.466560
95      2012 Chevrolet Silverado 3500HD Work Truck Lon…    al  34.466560
…                                                     …   …        …
458154    WE ARE A DEALERSHIP AND WE USE LIVE MARKET…    wi  42.857878
458195  big> 2010 GMC Sierra 1500 SLT - Carbon Black M…    wi  44.078180
458202  2005 Buick Rendezvous CX.  3.4 V6.  Need that …    wi  44.631225
458204  2006 Buick Lacrosse, CX 3.8 V6, 1 owner, clean…    wi  44.631225
458211  PRICE REDUCTION  Turns out the engine is toast…    wy  44.773500

              long              posting_date
19      -85.481615  2020-11-23T15:02:02-0600
91      -84.777696  2020-12-02T13:14:57-0600
92      -85.358940  2020-12-02T13:11:19-0600
93      -85.358940  2020-12-02T13:10:38-0600
95      -85.358940  2020-12-02T13:10:29-0600
…              …                       …
458154  -88.309457  2020-12-03T09:32:43-0600
458195  -87.696800  2020-12-03T09:34:37-0600
458202  -90.022076  2020-12-03T09:48:38-0600
458204  -90.022076  2020-12-03T09:24:54-0600
458211 -106.939600  2020-12-02T13:01:04-0700

[42384 rows x 26 columns]
```

[11]:
```
#drop rows for which price = 0, as these could be errors/forgotten input

# Get names of indexes for which column Age has value 30
indexNames = df[df['price'] == 0 ].index
# Delete these row indexes from dataFrame
df.drop(indexNames , inplace=True)
```

[12]:
```
#examining potential variables that influence price

price_data = df.loc[:,['odometer', 'year', 'manufacturer', 'model', 'condition',
       'drive', 'size', 'paint_color', 'price']]
price_data
```

[12]:
```
        odometer    year   manufacturer             model  condition drive  \
19       10500.0  2020.0           jeep         gladiator   like new   4wd
91       18823.0  2016.0  mercedes-benz  benz c300 4matic   like new   rwd
92       13036.0  2016.0           jeep          wrangler       good   4wd
93      145970.0  2005.0            gmc       sierra 3500       good   4wd
95      177450.0  2012.0      chevrolet  silverado 3500hd       good   rwd
…            …       …              …                 …          …   …
458141  124900.0  2012.0           ford   transit connect       good   fwd
458195   63812.0  2010.0            gmc       sierra 1500  excellent   4wd
458202  137962.0  2005.0          buick     rendezvous cx       good   fwd
```

```
458204  121488.0  2006.0          buick       lacrosse cx       good   fwd
458211  164000.0  2008.0           jeep    grand cherokee       good   4wd

             size paint_color  price
19      full-size        grey  47000
91      full-size       white  24999
92      full-size       white  41900
93      full-size       white  23900
95      full-size       white  18900
…             …           …      …
458141   mid-size      silver   5250
458195   mid-size       black  20488
458202    compact        grey   4995
458204   mid-size       black   4495
458211   mid-size       white   1300

[39712 rows x 9 columns]
```

[13]:
```
#get descriptive statistics
df.describe()
```

[13]:
```
          Unnamed: 0            id         price          year      odometer  \
count   39712.000000  3.971200e+04  3.971200e+04  39712.000000  3.971200e+04
mean   226476.337127  7.235415e+09  1.550879e+04   2010.829573  1.594026e+05
std    129114.411238  4.447097e+06  1.109585e+05      6.867688  1.025568e+07
min        19.000000  7.224683e+09  1.000000e+00   1927.000000  0.000000e+00
25%    114480.750000  7.232452e+09  6.900000e+03   2008.000000  6.525800e+04
50%    220783.500000  7.236731e+09  1.195000e+04   2012.000000  1.031040e+05
75%    338331.500000  7.239389e+09  1.989575e+04   2015.000000  1.410450e+05
max    458211.000000  7.241017e+09  2.200000e+07   2021.000000  2.043756e+09

                lat          long
count  39712.000000  39712.000000
mean      38.751363    -92.105025
std        5.689030     17.517312
min       -1.121187   -159.365637
25%       35.110400   -103.191663
50%       39.654965    -86.577305
75%       42.610220    -79.441100
max       64.823942     94.163200
```

[14]:
```
#run preliminary correlation matrix, nothing is really significant

corr_matrix = df.corr()
corr_matrix["price"].sort_values(ascending=False)
```

[14]:
```
price          1.000000
year           0.041708
lat            0.000525
odometer      -0.000225
long          -0.001786
Unnamed: 0    -0.002245
id            -0.007315
Name: price, dtype: float64
```

[15]:
```
df.columns
```

[15]:
```
Index(['Unnamed: 0', 'id', 'url', 'region', 'region_url', 'price', 'year',
       'manufacturer', 'model', 'condition', 'cylinders', 'fuel', 'odometer',
       'title_status', 'transmission', 'VIN', 'drive', 'size', 'type',
       'paint_color', 'image_url', 'description', 'state', 'lat', 'long',
       'posting_date'],
      dtype='object')
```

[16]:
```
# Remove unnecessary columns
df = df.drop(['Unnamed: 0', 'id', 'url','region_url','VIN',
        'image_url', 'description', 'state', 'lat', 'long','posting_date'],␣
 ↪axis = 1)
```

Question 2-1: Assemble a dataset consisting of predictors/features and target variable from subsampled data.

What features are relevant for the prediction task? Are there any features that should be excluded because they leak the target information? Show visualizations or statistics to support your selection.

[17]:
```
#took 3000 data elements
df = df.iloc[0:3000,:]
```

[18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3000 entries, 19 to 37483
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   region        3000 non-null   object
 1   price         3000 non-null   int64
 2   year          3000 non-null   float64
 3   manufacturer  3000 non-null   object
 4   model         3000 non-null   object
 5   condition     3000 non-null   object
 6   cylinders     3000 non-null   object
 7   fuel          3000 non-null   object
 8   odometer      3000 non-null   float64
```

```
 9   title_status  3000 non-null   object
 10  transmission  3000 non-null   object
 11  drive         3000 non-null   object
 12  size          3000 non-null   object
 13  type          3000 non-null   object
 14  paint_color   3000 non-null   object
dtypes: float64(2), int64(1), object(12)
memory usage: 375.0+ KB
```

[19]:
```
#convert float columns to int, was giving issues when running code
cols = ['year', 'odometer']
df[cols] = df[cols].applymap(np.int64)
```

[20]:
```
#separate target variable from features

target = df[['price']]

features = df[["region", "year","manufacturer", "model", "condition",␣
 ↪"cylinders", "odometer","fuel","title_status",
          "transmission", "drive", "size", "type", "paint_color"]]

print(features.shape)
```

```
(3000, 14)
```

[21]:
```
features.columns
```

[21]:
```
Index(['region', 'year', 'manufacturer', 'model', 'condition', 'cylinders',
       'odometer', 'fuel', 'title_status', 'transmission', 'drive', 'size',
       'type', 'paint_color'],
      dtype='object')
```

[22]:
```
#split training data from testing data


X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪random_state = 42, test_size = 0.2)
```

[23]:
```
#Copying the dataset to use the dummy encoder to test out feature selection␣
 ↪model
copy_df = df.copy()
copy_df_dum = pd.get_dummies(data = copy_df)
copy_df_dum.head()
```

[23]:

	price	year	odometer	region_anchorage / mat-su	region_auburn	\
19	47000	2020	10500	0	1	
91	24999	2016	18823	0	0	

```
92  41900  2016     13036                          0              0
93  23900  2005    145970                          0              0
95  18900  2012    177450                          0              0

    region_bakersfield  region_birmingham  region_chico  region_dothan  \
19                   0                  0             0              0
91                   0                  1             0              0
92                   0                  1             0              0
93                   0                  1             0              0
95                   0                  1             0              0

    region_fairbanks  …  paint_color_brown  paint_color_custom  \
19                 0  …                  0                   0
91                 0  …                  0                   0
92                 0  …                  0                   0
93                 0  …                  0                   0
95                 0  …                  0                   0

    paint_color_green  paint_color_grey  paint_color_orange  \
19                  0                 1                   0
91                  0                 0                   0
92                  0                 0                   0
93                  0                 0                   0
95                  0                 0                   0

    paint_color_purple  paint_color_red  paint_color_silver  \
19                   0                0                   0
91                   0                0                   0
92                   0                0                   0
93                   0                0                   0
95                   0                0                   0

    paint_color_white  paint_color_yellow
19                  0                   0
91                  1                   0
92                  1                   0
93                  1                   0
95                  1                   0

[5 rows x 1189 columns]
```

[24]:
```
#examine columns from copied dummy dataset
copy_df_dum.columns
```

[24]:
```
Index(['price', 'year', 'odometer', 'region_anchorage / mat-su',
       'region_auburn', 'region_bakersfield', 'region_birmingham',
       'region_chico', 'region_dothan', 'region_fairbanks',
```

```
       …
       'paint_color_brown', 'paint_color_custom', 'paint_color_green',
       'paint_color_grey', 'paint_color_orange', 'paint_color_purple',
       'paint_color_red', 'paint_color_silver', 'paint_color_white',
       'paint_color_yellow'],
      dtype='object', length=1189)
```

[25]:
```
#run correlation matrix for copied dummy dataset
corr = copy_df_dum.corr()
corr['price'].sort_values(ascending = False).head(10)
```

[25]:
```
price                   1.000000
drive_4wd               0.429132
type_truck              0.371729
year                    0.349340
fuel_diesel             0.342078
manufacturer_ferrari    0.339496
model_488 gtb           0.339496
cylinders_8 cylinders   0.325477
size_full-size          0.278821
model_benz g550         0.273668
Name: price, dtype: float64
```

[37]:
```
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

#Separate categorical and numerical variables for pipeline
cat_attr = ['drive','type', 'fuel', 'manufacturer', 'cylinders', 'size',␣
 ↪'model']
num_attr = ["year", "odometer"]

#transformation pipeline
combine_pipeline = ColumnTransformer([
    ("num", StandardScaler(), num_attr),
    ("cat", OneHotEncoder(handle_unknown = "ignore"), cat_attr),
    ])

#fit/transform data
X_train_prepared = combine_pipeline.fit_transform(X_train)
X_test_prepared = combine_pipeline.transform(X_test)
```

[38]:
```
#Linear Regression

from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
lm = lm.fit(X_train_prepared, y_train)
```

[40]:
```
#Linear Regression Metrics

from sklearn import metrics

y_train_pred = lm.predict(X_test_prepared)
train_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_train_pred))
print('Training RMSE: ' , train_rmse)
```

```
Training RMSE:  13286.580449897647
```

[41]:
```
#Linear regression MAE
from sklearn.metrics import mean_absolute_error

lin_mae = mean_absolute_error(y_test, y_train_pred)
lin_mae
```

[41]:
```
5158.257750123269
```

Question 2-2: Perform feature selection with a linear model, with appropriate preprocessing and cross-validation, evaluate the generalization performance.

[42]:
```
#k-folds
#Cross Validation
from sklearn.model_selection import cross_val_score

val_score = cross_val_score(lm, X_train_prepared, y_train, scoring =␣
↪"neg_mean_squared_error", cv = 10)
val_mse = np.sqrt(-val_score)
print(val_mse.mean())
```

```
7851.640342183135
```

Question 2-3: Use any non-linear regression model we introduced (tree, forest, gradient boosting) to improve your result. You can (and probably should) change your preprocessing and feature engineering to be suitable for the model, tune hyperparameters as appropriate. What is the best prediction you can get? Discuss your work here.

You are not required to try all of these models.

[44]:
```
#Tree
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor()
dt.fit(X_train_prepared,y_train)
```

```
y_train_pred = dt.predict(X_train_prepared)
y_pred = dt.predict(X_test_prepared)
```

[49]:
```
train_rmse = np.sqrt(metrics.mean_squared_error(y_train, y_train_pred))
test_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('Training Error: '+ str(train_rmse) )
print('Testing Error: '+ str(test_rmse) )
```

```
Training Error: 1685.1873308731938
Testing Error: 11082.576107968862
```

[50]:
```
#Random Forest
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators = 200, n_jobs = -1, verbose = 1)
rf.fit(X_train_prepared,y_train)

y_train_pred = rf.predict(X_train_prepared)
y_pred = rf.predict(X_test_prepared)
```

```
<ipython-input-50-65ad2c6545c9>:5: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  rf.fit(X_train_prepared,y_train)
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    1.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:    5.6s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:    5.8s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.1s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:    0.0s finished
```

[51]:
```
train_rmse = np.sqrt(metrics.mean_squared_error(y_train, y_train_pred))
test_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('Training Error: '+ str(train_rmse) )
print('Testing Error: '+ str(test_rmse) )
```

```
Training Error: 1618.0175384940198
Testing Error: 11054.344910857879
```

[ ]:
```

```