In [462]:
```
# -*- coding: utf-8 -*-
"""
Created on Sat Aug  8 22:47:21 2020

@author: dhruv
"""

import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import precision_recall_curve,accuracy_score,recall_score,co
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

In [390]:
```
df = pd.read_csv("datasets_1291_2355_Automobile_data.csv")  ##importing dataset
```

In [391]:
```
df.head()
```

Out[391]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	.
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	.
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	.
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	.

5 rows × 26 columns

In [463]:
```
df.info() # dataset info
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 193 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          193 non-null    int64  
 1   normalized-losses  193 non-null    object 
 2   make               193 non-null    object 
 3   fuel-type          193 non-null    object 
 4   aspiration         193 non-null    object 
 5   num-of-doors       193 non-null    object 
 6   body-style         193 non-null    object 
 7   drive-wheels       193 non-null    object 
 8   engine-location    193 non-null    object 
 9   wheel-base         193 non-null    float64
 10  length             193 non-null    float64
 11  width              193 non-null    float64
 12  height             193 non-null    float64
 13  curb-weight        193 non-null    int64  
 14  engine-type        193 non-null    object 
 15  num-of-cylinders   193 non-null    object 
 16  engine-size        193 non-null    int64  
 17  fuel-system        193 non-null    object 
 18  bore               193 non-null    object 
 19  stroke             193 non-null    object 
 20  compression-ratio  193 non-null    float64
 21  horsepower         193 non-null    object 
 22  peak-rpm           193 non-null    object 
 23  city-mpg           193 non-null    int64  
 24  highway-mpg        193 non-null    int64  
 25  price              193 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 40.7+ KB
```

In [393]:
```
df.describe() # dataset stats
```

Out[393]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	comp
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	20
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	1
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	2

In [394]:
```
## EDA
```

In [464]:
```
df.isnull().sum()  # checking for missing values
```

Out[464]:
```
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 [396]:
```
# so no missing values
```

In [397]:
```
df.shape
```

Out[397]:
```
(205, 26)
```

In [398]:
```
df.columns
```

Out[398]:
```
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
```

In [465]:
```
df['symboling'].nunique()  ## checking for our target class
```

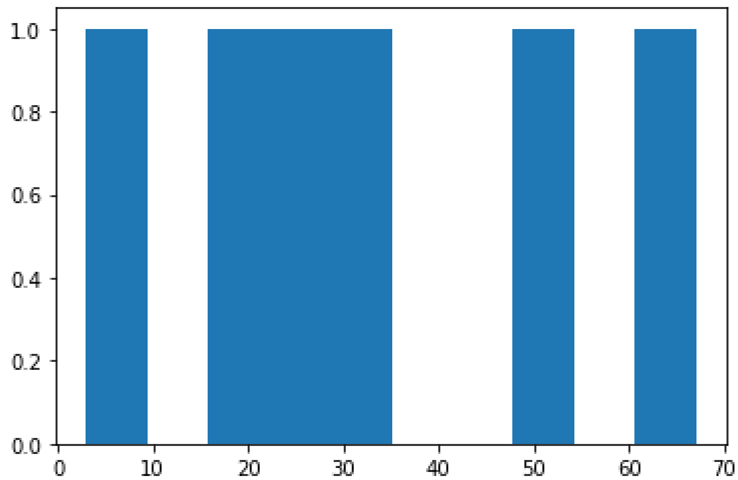
Out[465]:
```
6
```

In [400]:
```
df['symboling'].value_counts()
```

Out[400]:
```
 0    67
 1    54
 2    32
 3    27
-1    22
-2     3
Name: symboling, dtype: int64
```

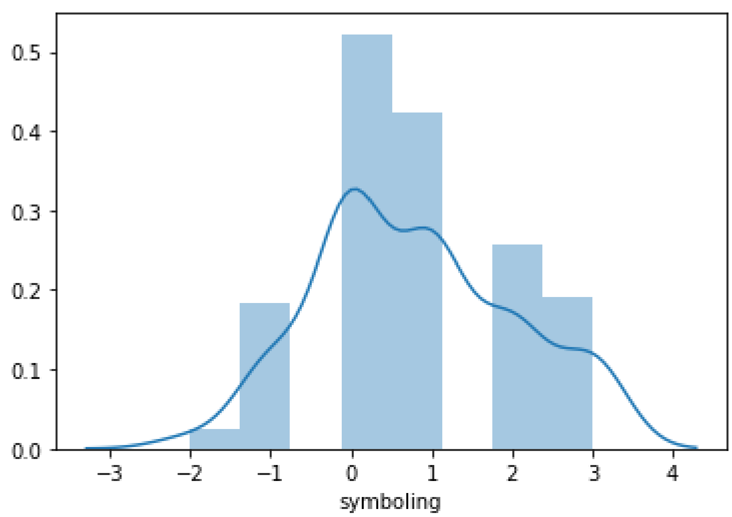
In [401]:
```
plt.hist(df['symboling'].value_counts())
```

Out[401]:
```
(array([1., 0., 1., 1., 1., 0., 0., 1., 0., 1.]),
 array([ 3. ,  9.4, 15.8, 22.2, 28.6, 35. , 41.4, 47.8, 54.2, 60.6, 67. ]),
 <a list of 10 Patch objects>)
```



In [466]:
```
sns.distplot(df['symboling'])  # visualizing distribution of targert columns
```

Out[466]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19b55d6a460>
```



In [403]:
```
df.head()
```

Out[403]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...

5 rows × 26 columns

In [404]:
```
df
```

Out[404]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4
...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1

205 rows × 26 columns

In [405]:
```
df['normalized-losses'].value_counts()
```

Out[405]:
```
?      41
161    11
91      8
150     7
134     6
104     6
128     6
102     5
103     5
168     5
95      5
65      5
94      5
85      5
74      5
93      4
122     4
106     4
118     4
148     4
154     3
101     3
125     3
83      3
137     3
115     3
119     2
194     2
192     2
145     2
108     2
87      2
129     2
110     2
158     2
81      2
164     2
89      2
188     2
153     2
197     2
113     2
90      1
186     1
142     1
256     1
98      1
121     1
231     1
107     1
78      1
77      1
Name: normalized-losses, dtype: int64
```

In [406]:
```
## now we have to remove all the ? values with mean
```

In [407]:
```
df['normalized-losses'].replace('?',np.NaN,inplace=True) # handling '?'
```

In [408]:
```
from sklearn.impute import SimpleImputer
```

In [467]:
```
imp = SimpleImputer(missing_values=np.nan,strategy='mean')  # imputing columns wi
```

In [410]:
```
df.replace('?',np.NaN,inplace=True)
```

In [411]:
```
# imp.fit_transform(df['normalized-losses'])
```

In [412]:
```
df['normalized-losses'].value_counts()
```

Out[412]:
```
161    11
91      8
150     7
134     6
104     6
128     6
65      5
94      5
103     5
168     5
95      5
102     5
85      5
74      5
122     4
93      4
118     4
148     4
106     4
125     3
101     3
154     3
83      3
137     3
115     3
87      2
145     2
192     2
108     2
119     2
129     2
194     2
110     2
197     2
113     2
81      2
164     2
188     2
89      2
153     2
158     2
186     1
142     1
98      1
90      1
256     1
231     1
77      1
107     1
78      1
121     1
Name: normalized-losses, dtype: int64
```

In [413]:
```
num_count =0  # getting the mean of the df column
summ=0
for i in df['normalized-losses']:
    if i is np.NaN:
        continue
    else:
        summ+=int(i)
        num_count+=1
mean= summ/num_count
print(mean)
```

```
122.0
```

In [414]:
```
df['normalized-losses'].replace(np.NaN,122,inplace=True)
```

In [415]:
```
df['fuel-system'].value_counts()
```

Out[415]:
```
mpfi    94
2bbl    66
idi     20
1bbl    11
spdi     9
4bbl     3
spfi     1
mfi      1
Name: fuel-system, dtype: int64
```

In [416]:
```
col_list =df.columns.tolist()
```

In [417]:
```
categorical_cols = [i for i in df.columns if df[i].dtype=='object' and df[i].nuni
```

In [418]:
```
categorical_cols   ## getting all categorical columns
```

Out[418]:
```
['make',
 'fuel-type',
 'aspiration',
 'num-of-doors',
 'body-style',
 'drive-wheels',
 'engine-location',
 'engine-type',
 'num-of-cylinders',
 'fuel-system']
```

In [419]:
```
df[categorical_cols].nunique()
```

Out[419]:
```
make                22
fuel-type            2
aspiration           2
num-of-doors         2
body-style           5
drive-wheels         3
engine-location      2
engine-type          7
num-of-cylinders     7
fuel-system          8
dtype: int64
```

In [420]:
```
df
```

Out[420]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-bas
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.(
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.(
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.!
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4
...	...	...	...	...	...	...	...	...	...	..
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1

205 rows × 26 columns

In [421]:
```
#from sklearn.preprocessing import LabelEncoder,LabelBinarizer
```

In [422]:
```
df['price'].isna().sum()
```

Out[422]: 4

In [423]:
```
df.dropna(inplace=True)
```

In [424]:
```
df.isna().sum()
```

Out[424]:
```
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 [425]:
```
df_numeric = pd.get_dummies(df,columns=categorical_cols)  # one hot encoding datc
```

In [426]:
```
df_numeric
```

Out[426]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	...	cyli
0	3	122	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	...	
1	3	122	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	...	
2	1	122	94.5	171.2	65.5	52.4	2823	152	2.68	3.47	...	
3	2	164	99.8	176.6	66.2	54.3	2337	109	3.19	3.4	...	
4	2	164	99.4	176.6	66.4	54.3	2824	136	3.19	3.4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95	109.1	188.8	68.9	55.5	2952	141	3.78	3.15	...	
201	-1	95	109.1	188.8	68.8	55.5	3049	141	3.78	3.15	...	
202	-1	95	109.1	188.8	68.9	55.5	3012	173	3.58	2.87	...	
203	-1	95	109.1	188.8	68.9	55.5	3217	145	3.01	3.4	...	
204	-1	95	109.1	188.8	68.9	55.5	3062	141	3.78	3.15	...	

193 rows × 71 columns

In [427]:
```
df_numeric.astype('float64').dtypes  ## typecasting all numeric columns to float6
```

Out[427]:
```
symboling            float64
normalized-losses    float64
wheel-base           float64
length               float64
width                float64
                      ...
fuel-system_idi      float64
fuel-system_mfi      float64
fuel-system_mpfi     float64
fuel-system_spdi     float64
fuel-system_spfi     float64
Length: 71, dtype: object
```

In [428]:
```
## splitting dataset into feature and target columns
```

In [429]:
```
y = df_numeric['symboling']   # target columns
```

In [430]:
```
y
```

Out[430]:
```
0      3
1      3
2      1
3      2
4      2
      ..
200   -1
201   -1
202   -1
203   -1
204   -1
Name: symboling, Length: 193, dtype: int64
```

In [431]:
```
df_numeric.drop(['symboling'],axis=1,inplace=True)
```

In [432]:
```
X = df_numeric  # feature columns
```

In [433]:
```
X
```

Out[433]:

	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	...
0	122	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	...
1	122	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	...
2	122	94.5	171.2	65.5	52.4	2823	152	2.68	3.47	9.0	...
3	164	99.8	176.6	66.2	54.3	2337	109	3.19	3.4	10.0	...
4	164	99.4	176.6	66.4	54.3	2824	136	3.19	3.4	8.0	...
...	...	...	...	...	...	...	...	...	...	...	...
200	95	109.1	188.8	68.9	55.5	2952	141	3.78	3.15	9.5	...
201	95	109.1	188.8	68.8	55.5	3049	141	3.78	3.15	8.7	...
202	95	109.1	188.8	68.9	55.5	3012	173	3.58	2.87	8.8	...
203	95	109.1	188.8	68.9	55.5	3217	145	3.01	3.4	23.0	...
204	95	109.1	188.8	68.9	55.5	3062	141	3.78	3.15	9.5	...

193 rows × 70 columns

In [ ]:

In [434]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state =
```

In [435]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[435]:
```
((135, 70), (58, 70), (135,), (58,))
```

In [436]:
```
## buidling model
```

In [437]:
```
df['normalized-losses'].value_counts()
```

Out[437]:
```
122    34
161    11
91      8
104     6
134     6
128     6
85      5
94      5
103     5
102     5
168     5
95      5
74      5
65      5
93      4
118     4
122     4
106     4
148     3
150     3
101     3
154     3
83      3
137     3
115     3
125     3
192     2
197     2
108     2
194     2
87      2
119     2
129     2
188     2
158     2
113     2
81      2
89      2
153     2
164     2
145     2
110     2
231     1
256     1
142     1
121     1
77      1
107     1
78      1
98      1
186     1
90      1
Name: normalized-losses, dtype: int64
```

In [438]:
```
gnb = GaussianNB()   # initialize model
```

In [439]:
```
gnb.fit(X_train,y_train)   ## fit model on train data
```

Out[439]:
```
GaussianNB()
```

In [440]:
```
y_pred = gnb.predict(X_test)  # get y_pred
```

In [441]:
```
print(accuracy_score(y_test,y_pred))   # accuracy score
```

```
0.6379310344827587
```

In [445]:
```
print(confusion_matrix(y_test,y_pred))  ## our multiclass confusion metrics
```

```
[[ 6  0  0  1  0]
 [ 2  9  1  3  1]
 [ 2  0 13  6  0]
 [ 0  1  2  5  0]
 [ 0  0  0  2  4]]
```

In [449]:
```
y_pred  ## y_pred
```

Out[449]:
```
array([ 1,  3, -1,  2,  0,  1,  0, -1, -1, -1, -1,  2,  2,  0,  0,  2,  2,
        1,  1,  1,  0,  2,  0,  1,  1,  0,  2,  0,  1,  3,  0, -1,  1,  1,
        1,  2, -1,  2,  1,  2,  2, -1,  1, -1,  2,  3,  2,  2,  3,  0, -1,
        3,  1,  1,  2,  1,  2,  2], dtype=int64)
```

In [451]:
```
print(recall_score(y_test,y_pred,average=None)) # recall score
```

```
[0.85714286 0.5625     0.61904762 0.625      0.66666667]
```

In [453]:
```
print(precision_score(y_test,y_pred,average=None))  # precision score
```

```
[0.6        0.9        0.8125     0.29411765 0.8       ]
```

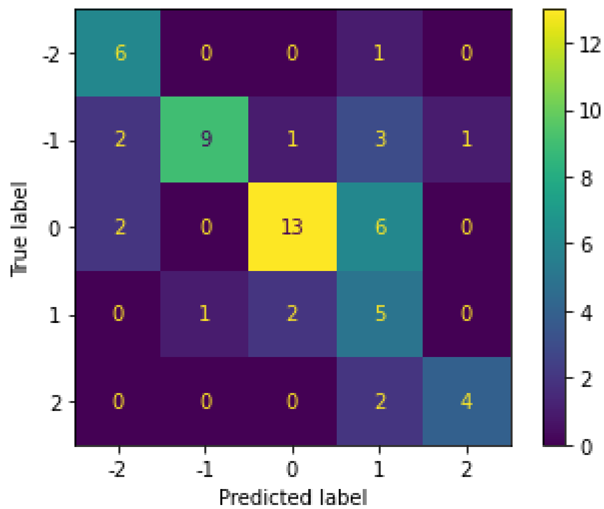
In [456]:
```
print(f1_score(y_test,y_pred,average=None))  # f1_score
```

```
[0.70588235 0.69230769 0.7027027  0.4        0.72727273]
```

In [459]:
```
from sklearn.metrics import plot_confusion_matrix
```

In [460]:
```
plot_confusion_matrix(gnb,X_test,y_test)
```

Out[460]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x19b541c5d30
>
```



In [461]:
```
## thus we successfully build GaussianNB model on Automobile dataset
```