```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>

Saving auto.csv to auto.csv
```

```
import pandas as pd
path = 'auto.csv'
df = pd.read_csv(path)
df.tail(5)
```

```
     3   ? alfa-romero     gas    std   two convertible  rwd  front
88.6  \
199 -1  95       volvo     gas    std  four       sedan  rwd  front
109.1
200 -1  95       volvo     gas  turbo  four       sedan  rwd  front
109.1
201 -1  95       volvo     gas    std  four       sedan  rwd  front
109.1
202 -1  95       volvo  diesel  turbo  four       sedan  rwd  front
109.1
203 -1  95       volvo     gas  turbo  four       sedan  rwd  front
109.1

     ...  130  mpfi  3.47  2.68   9.0  111  5000  21  27  13495
199  ...  141  mpfi  3.78  3.15   9.5  114  5400  23  28  16845
200  ...  141  mpfi  3.78  3.15   8.7  160  5300  19  25  19045
201  ...  173  mpfi  3.58  2.87   8.8  134  5500  18  23  21485
202  ...  145   idi  3.01  3.40  23.0  106  4800  26  27  22470
203  ...  141  mpfi  3.78  3.15   9.5  114  5400  19  25  22625

[5 rows x 26 columns]
```

```
headers = ["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"]
print("headers\n", headers)
```

```
headers
 ['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']
```

```
df.columns = headers
df.head(10)
```

```
   symboling normalized-losses         make fuel-type aspiration num-
of-doors  \
0          3                 ?  alfa-romero       gas        std
two
1          1                 ?  alfa-romero       gas        std
two
2          2               164         audi       gas        std
four
3          2               164         audi       gas        std
four
4          2                 ?         audi       gas        std
two
5          1               158         audi       gas        std
four
6          1                 ?         audi       gas        std
four
7          1               158         audi       gas      turbo
four
8          0                 ?         audi       gas      turbo
two
9          2               192          bmw       gas        std
two

    body-style drive-wheels engine-location  wheel-base  ...  engine-
size  \
0  convertible          rwd           front        88.6  ...
130
1    hatchback          rwd           front        94.5  ...
152
2        sedan          fwd           front        99.8  ...
109
3        sedan          4wd           front        99.4  ...
136
4        sedan          fwd           front        99.8  ...
136
5        sedan          fwd           front       105.8  ...
136
6        wagon          fwd           front       105.8  ...
136
7        sedan          fwd           front       105.8  ...
131
8    hatchback          4wd           front        99.5  ...
131
9        sedan          rwd           front       101.2  ...
108

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm
```

```
city-mpg  \
0         mpfi  3.47    2.68                9.0        111      5000
21
1         mpfi  2.68    3.47                9.0        154      5000
19
2         mpfi  3.19    3.40               10.0        102      5500
24
3         mpfi  3.19    3.40                8.0        115      5500
18
4         mpfi  3.19    3.40                8.5        110      5500
19
5         mpfi  3.19    3.40                8.5        110      5500
19
6         mpfi  3.19    3.40                8.5        110      5500
19
7         mpfi  3.13    3.40                8.3        140      5500
17
8         mpfi  3.13    3.40                7.0        160      5500
16
9         mpfi  3.50    2.80                8.8        101      5800
23

  highway-mpg  price
0          27  16500
1          26  16500
2          30  13950
3          22  17450
4          25  15250
5          25  17710
6          25  18920
7          20  23875
8          22      ?
9          29  16430

[10 rows x 26 columns]
```

```
df.dtypes
```

```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
```

```
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

```
df["price"] = df["price"].astype("float")
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-6-564c5f370205> in <module>
----> 1 df["price"] = df["price"].astype("float")

/usr/local/lib/python3.8/dist-packages/pandas/core/generic.py in
astype(self, dtype, copy, errors)
   5813         else:
   5814             # else, only a single dtype is given
-> 5815             new_data = self._mgr.astype(dtype=dtype,
copy=copy, errors=errors)
   5816             return
self._constructor(new_data).__finalize__(self, method="astype")
   5817

/usr/local/lib/python3.8/dist-packages/pandas/core/internals/managers.
py in astype(self, dtype, copy, errors)
    416
    417     def astype(self: T, dtype, copy: bool = False, errors: str
= "raise") -> T:
--> 418         return self.apply("astype", dtype=dtype, copy=copy,
errors=errors)
    419
    420     def convert(

/usr/local/lib/python3.8/dist-packages/pandas/core/internals/managers.
py in apply(self, f, align_keys, ignore_failures, **kwargs)
    325                     applied = b.apply(f, **kwargs)
    326                 else:
--> 327                     applied = getattr(b, f)(**kwargs)
```

```
    328             except (TypeError, NotImplementedError):
    329                 if not ignore_failures:

/usr/local/lib/python3.8/dist-packages/pandas/core/internals/blocks.py
in astype(self, dtype, copy, errors)
    589         values = self.values
    590
--> 591         new_values = astype_array_safe(values, dtype,
copy=copy, errors=errors)
    592
    593         new_values = maybe_coerce_values(new_values)

/usr/local/lib/python3.8/dist-packages/pandas/core/dtypes/cast.py in
astype_array_safe(values, dtype, copy, errors)
   1307
   1308     try:
-> 1309         new_values = astype_array(values, dtype, copy=copy)
   1310     except (ValueError, TypeError):
   1311         # e.g. astype_nansafe can fail on object-dtype of
strings

/usr/local/lib/python3.8/dist-packages/pandas/core/dtypes/cast.py in
astype_array(values, dtype, copy)
   1255
   1256     else:
-> 1257         values = astype_nansafe(values, dtype, copy=copy)
   1258
   1259     # in pandas we don't store numpy str dtypes, so convert to
object

/usr/local/lib/python3.8/dist-packages/pandas/core/dtypes/cast.py in
astype_nansafe(arr, dtype, copy, skipna)
   1199     if copy or is_object_dtype(arr.dtype) or
is_object_dtype(dtype):
   1200         # Explicit copy, or required since NumPy can't view
from / to object.
-> 1201         return arr.astype(dtype, copy=True)
   1202
   1203     return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: '?'
```

```
df.describe()
```

	symboling	wheel-base	length	width	height	\
count	204.000000	204.000000	204.000000	204.000000	204.000000	
mean	0.823529	98.806373	174.075000	65.916667	53.749020	
std	1.239035	5.994144	12.362123	2.146716	2.424901	
min	-2.000000	86.600000	141.100000	60.300000	47.800000	
25%	0.000000	94.500000	166.300000	64.075000	52.000000	

```
50%      1.000000   97.000000  173.200000   65.500000   54.100000
75%      2.000000  102.400000  183.200000   66.900000   55.500000
max      3.000000  120.900000  208.100000   72.300000   59.800000

       curb-weight  engine-size  compression-ratio    city-mpg
highway-mpg
count   204.000000   204.000000         204.000000  204.000000
204.000000
mean   2555.602941   126.892157          10.148137   25.240196
30.769608
std     521.960820    41.744569           3.981000    6.551513
6.898337
min    1488.000000    61.000000           7.000000   13.000000
16.000000
25%    2145.000000    97.000000           8.575000   19.000000
25.000000
50%    2414.000000   119.500000           9.000000   24.000000
30.000000
75%    2939.250000   142.000000           9.400000   30.000000
34.500000
max    4066.000000   326.000000          23.000000   49.000000
54.000000

df.isnull().sum()

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
```

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

```
?      40
161    11
91      8
150     7
134     6
128     6
104     6
85      5
94      5
65      5
102     5
74      5
168     5
103     5
95      5
106     4
93      4
118     4
148     4
122     4
83      3
125     3
154     3
115     3
137     3
101     3
119     2
87      2
89      2
192     2
197     2
158     2
81      2
188     2
194     2
153     2
129     2
108     2
110     2
164     2
145     2
113     2
256     1
107     1
90      1
231     1
```

```
142     1
121     1
78      1
98      1
186     1
77      1
Name: normalized-losses, dtype: int64
```

```
import numpy as np
df.replace("?", np.nan, inplace = True)
df.head()
```

```
   symboling normalized-losses         make fuel-type aspiration num-
of-doors  \
0          3               NaN  alfa-romero       gas        std
two
1          1               NaN  alfa-romero       gas        std
two
2          2               164         audi       gas        std
four
3          2               164         audi       gas        std
four
4          2               NaN         audi       gas        std
two

    body-style drive-wheels engine-location  wheel-base  ...  engine-
size  \
0  convertible          rwd           front        88.6  ...
130
1    hatchback          rwd           front        94.5  ...
152
2        sedan          fwd           front        99.8  ...
109
3        sedan          4wd           front        99.4  ...
136
4        sedan          fwd           front        99.8  ...
136

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm
city-mpg  \
0         mpfi  3.47    2.68               9.0        111      5000
21
1         mpfi  2.68    3.47               9.0        154      5000
19
2         mpfi  3.19    3.40              10.0        102      5500
24
3         mpfi  3.19    3.40               8.0        115      5500
18
4         mpfi  3.19    3.40               8.5        110      5500
19
```

```
  highway-mpg  price
0          27  16500
1          26  16500
2          30  13950
3          22  17450
4          25  15250

[5 rows x 26 columns]
```

```
df.isnull().sum()
```

```
symboling             0
normalized-losses    40
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
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                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 4
dtype: int64
```

```
df.dropna(subset=["normalized-losses"], axis=0)
```

```
     symboling normalized-losses   make fuel-type aspiration num-of-
doors  \
2            2               164   audi       gas        std
four
3            2               164   audi       gas        std
four
5            1               158   audi       gas        std
four
7            1               158   audi       gas      turbo
```

```
four
9            2               192    bmw       gas        std
two
..         ...               ...    ...       ...        ...
...
199         -1                95  volvo       gas        std
four
200         -1                95  volvo       gas      turbo
four
201         -1                95  volvo       gas        std
four
202         -1                95  volvo    diesel      turbo
four
203         -1                95  volvo       gas      turbo
four

    body-style drive-wheels engine-location  wheel-base  ...  engine-
size  \
2        sedan          fwd           front        99.8  ...
109
3        sedan          4wd           front        99.4  ...
136
5        sedan          fwd           front       105.8  ...
136
7        sedan          fwd           front       105.8  ...
131
9        sedan          rwd           front       101.2  ...
108
..         ...          ...             ...         ...  ...
...
199      sedan          rwd           front       109.1  ...
141
200      sedan          rwd           front       109.1  ...
141
201      sedan          rwd           front       109.1  ...
173
202      sedan          rwd           front       109.1  ...
145
203      sedan          rwd           front       109.1  ...
141

     fuel-system  bore  stroke compression-ratio horsepower  peak-rpm
\
2           mpfi  3.19    3.40              10.0        102      5500

3           mpfi  3.19    3.40               8.0        115      5500

5           mpfi  3.19    3.40               8.5        110      5500

7           mpfi  3.13    3.40               8.3        140      5500
```

```
9           mpfi  3.50    2.80               8.8        101      5800

..           ...   ...     ...               ...        ...       ...

199         mpfi  3.78    3.15               9.5        114      5400

200         mpfi  3.78    3.15               8.7        160      5300

201         mpfi  3.58    2.87               8.8        134      5500

202          idi  3.01    3.40              23.0        106      4800

203         mpfi  3.78    3.15               9.5        114      5400


    city-mpg highway-mpg  price
2         24          30  13950
3         18          22  17450
5         19          25  17710
7         17          20  23875
9         23          29  16430
..       ...         ...    ...
199       23          28  16845
200       19          25  19045
201       18          23  21485
202       26          27  22470
203       19          25  22625

[164 rows x 26 columns]
```

```
avg_norm_loss = df["normalized-losses"].astype("float").mean(axis=0)
print("Average of normalized-losses:", avg_norm_loss)
```

```
Average of normalized-losses: 122.0
```

```
df["normalized-losses"].replace(np.nan, avg_norm_loss, inplace=True)
```

```
df.isnull().sum()
```

```
symboling            0
normalized-losses    0
make                 0
fuel-type            0
aspiration           0
num-of-doors         2
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                 4
stroke               4
compression-ratio    0
horsepower           2
peak-rpm             2
city-mpg             0
highway-mpg          0
price                4
dtype: int64
```

```
df['length'].head()
```

```
0    168.8
1    171.2
2    176.6
3    176.6
4    177.3
Name: length, dtype: float64
```

```
df['length'] = df['length']/df['length'].max()
df['width'] = df['width']/df['width'].max()
```

```
df['length'].head()
```

```
0    0.811148
1    0.822681
2    0.848630
3    0.848630
4    0.851994
Name: length, dtype: float64
```

```
df.head()
```

```
   symboling normalized-losses         make  ... city-mpg highway-mpg
price
0          3             122.0  alfa-romero  ...       21          27
16500
1          1             122.0  alfa-romero  ...       19          26
16500
2          2               164         audi  ...       24          30
13950
3          2               164         audi  ...       18          22
17450
4          2             122.0         audi  ...       19          25
```

```
15250

[5 rows x 26 columns]
```

```
pd.get_dummies(df["fuel-type"])
```

```
     diesel  gas
0         0    1
1         0    1
2         0    1
3         0    1
4         0    1
..      ...  ...
199       0    1
200       0    1
201       0    1
202       1    0
203       0    1

[204 rows x 2 columns]
```

```
df["drive-wheels"].value_counts()
```

```
fwd    120
rwd     75
4wd      9
Name: drive-wheels, dtype: int64
```

```
df.corr()
```

```
                   symboling  wheel-base  ...  city-mpg  highway-mpg
symboling           1.000000   -0.525095  ... -0.030557     0.039598
wheel-base         -0.525095    1.000000  ... -0.479633    -0.552897
length             -0.356792    0.877612  ... -0.673251    -0.706635
width              -0.227799    0.795115  ... -0.647177    -0.681169
height             -0.533078    0.582603  ... -0.055659    -0.113995
curb-weight        -0.229281    0.781763  ... -0.758238    -0.798088
engine-size        -0.107229    0.573989  ... -0.654101    -0.677775
compression-ratio  -0.177413    0.249199  ...  0.324186     0.264677
city-mpg           -0.030557   -0.479633  ...  1.000000     0.971311
highway-mpg         0.039598   -0.552897  ...  0.971311     1.000000

[10 rows x 10 columns]
```

```
df["price"] = df["price"].astype(float)
```

```
df.dtypes
```

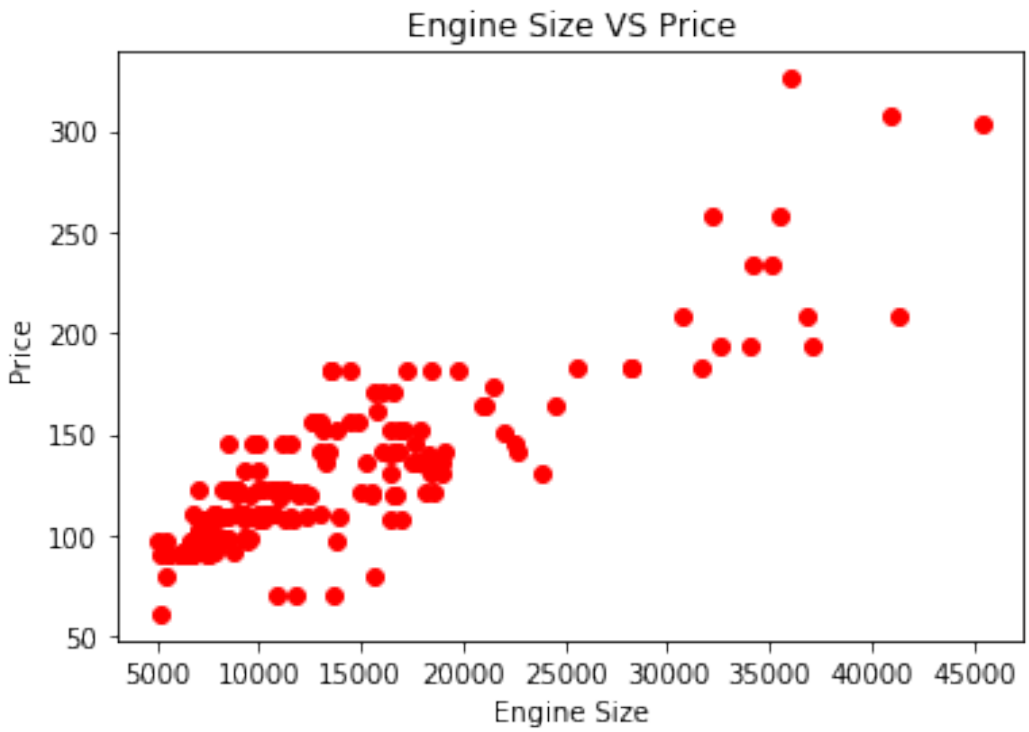
```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
```

```
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                float64
dtype: object
```

```
import matplotlib.pylab as plt
y = df["engine-size"]
x = df["price"]

plt.scatter(x,y, color = "Red")
plt.xlabel("Engine Size")
plt.ylabel("Price")
plt.title("Engine Size VS Price")
```

```
Text(0.5, 1.0, 'Engine Size VS Price')
```

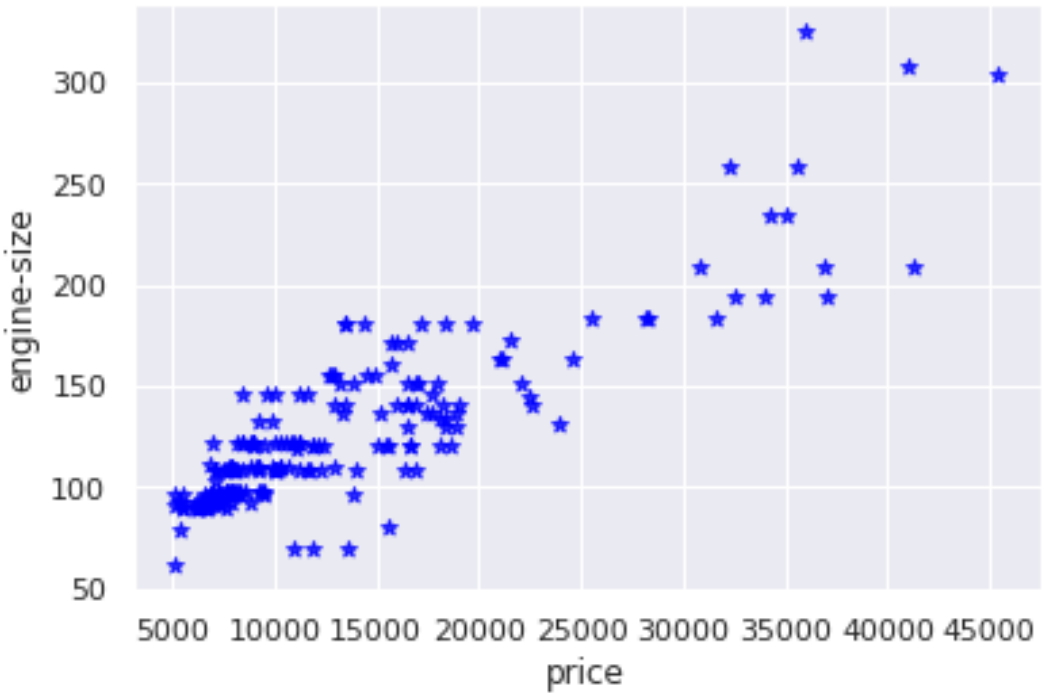



```
import seaborn as sns

sns.set(style = "darkgrid")
sns.regplot(x,y,data=df, fit_reg = False, marker = "*", color =
"blue")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff728e76f90>
```


engine-size
300
250
200
150
100
50
5000
10000
15000
20000
25000
30000
35000
40000
45000
price