In [1]:
```
#data preperation and analysis library
import pandas as pd

#plotting libraries
import matplotlib.pyplot as plt
import seaborn as sns

#library for creating random samples
from sklearn.model_selection import train_test_split

#library for buliding linear regression model
from sklearn.linear_model import LinearRegression

#feature selection (to select significant variables)
from sklearn.feature_selection import SelectKBest, f_regression
```

In [2]:
```
#load data
df=pd.read_csv(r"C:\Users\ACER\Desktop\introtallent\python\data\104380_Python
```

In [3]:
```
df.head(2)
```

Out[3]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model_year	Origin	Car_N
0	8.0	8	307.0	130	3504	12.0	2015	1	chev
1	15.0	8	350.0	165	3693	11.5	2015	1	b

In [4]:
```
#print row and column count
df.shape
```

Out[4]: (398, 9)

In [5]:
```
df.dtypes
```

Out[5]:
```
MPG             float64
Cylinders         int64
Displacement    float64
Horsepower       object
Weight            int64
Acceleration    float64
Model_year        int64
Origin            int64
Car_Name         object
dtype: object
```

In [6]:
```
#cyclinders,Model_year,Origin are categorical variables stored as int
#change the datatype to object
df['Cylinders']=df['Cylinders'].astype('object')
df['Model_year']=df['Model_year'].astype('object')
df['Origin']=df['Origin'].astype('object')
```

In [7]:
```
df.dtypes
```

Out[7]:
```
MPG             float64
Cylinders        object
Displacement    float64
Horsepower       object
Weight            int64
Acceleration    float64
Model_year       object
Origin           object
Car_Name         object
dtype: object
```

In [8]:
```
df['Horsepower']=pd.to_numeric(df['Horsepower'],errors='coerce')
```

In [9]:
```
df.dtypes
```

Out[9]:
```
MPG             float64
Cylinders        object
Displacement    float64
Horsepower      float64
Weight            int64
Acceleration    float64
Model_year       object
Origin           object
Car_Name         object
dtype: object
```

In [10]:
```
#Feature engineering-[check and impute missing values,if any]
df.isnull().sum()
```

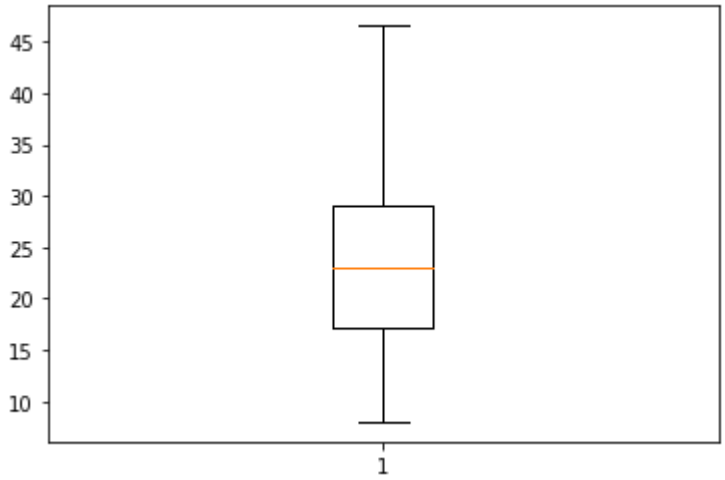
Out[10]:
```
MPG             0
Cylinders       0
Displacement    0
Horsepower      6
Weight          0
Acceleration    0
Model_year      0
Origin          0
Car_Name        0
dtype: int64
```

In [11]:
```
#Impute Horsepower with median
df['Horsepower']=df['Horsepower'].fillna(df['Horsepower'].median())
```

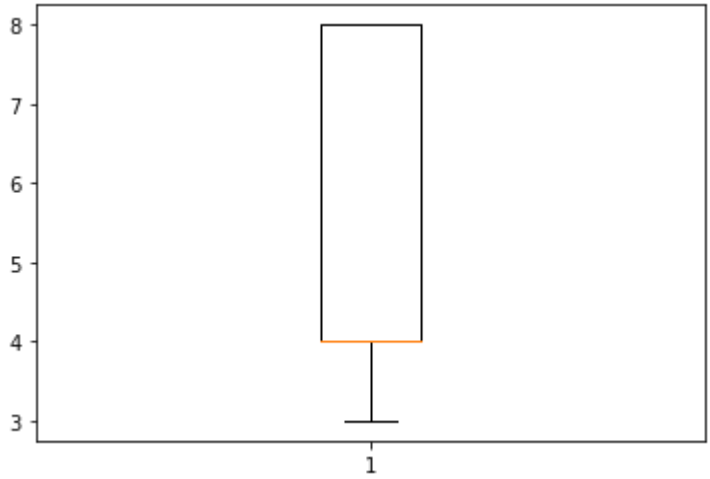
In [12]:
```
df.isnull().sum()
```

Out[12]:
```
MPG             0
Cylinders       0
Displacement    0
Horsepower      0
Weight          0
Acceleration    0
Model_year      0
Origin          0
Car_Name        0
dtype: int64
```

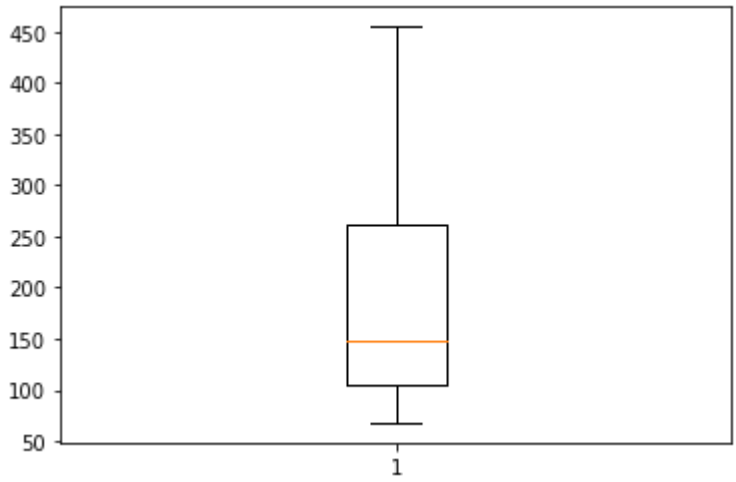
In [13]:
```
#feature engineering-outlier treatment
plt.boxplot(df['MPG'])
plt.show()
```



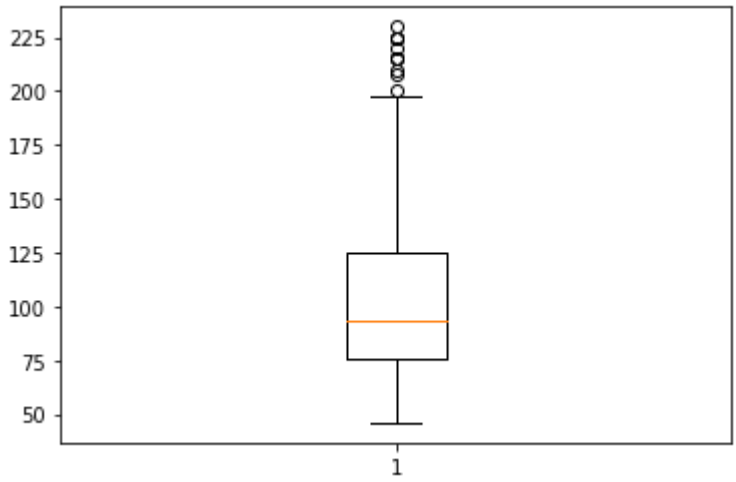
In [14]:
```
plt.boxplot(df['Cylinders'])
plt.show()
```



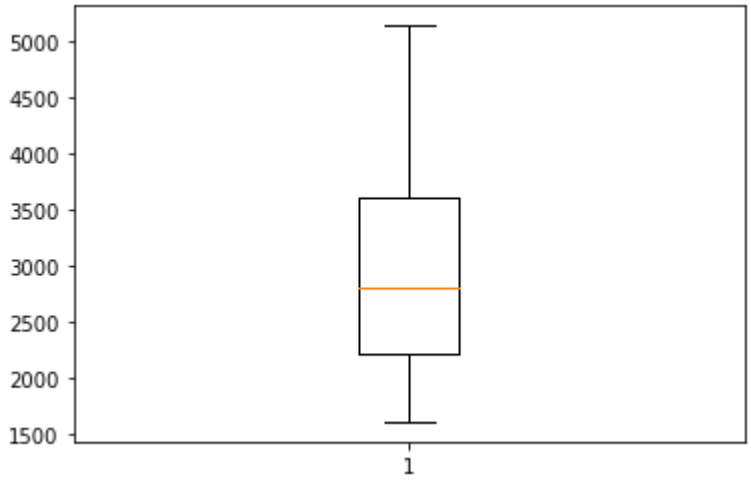
In [15]:
```
plt.boxplot(df['Displacement'])
plt.show()
```



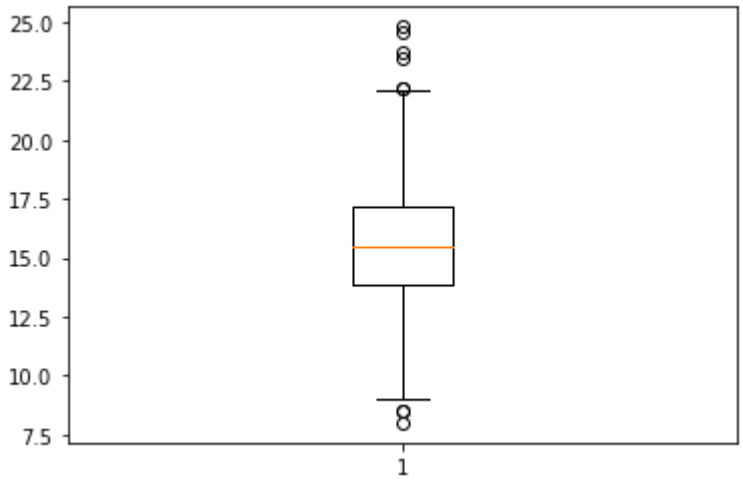
In [16]:
```
plt.boxplot(df['Horsepower'])
plt.show()
```



In [17]:
```
plt.boxplot(df['Weight'])
plt.show()
```

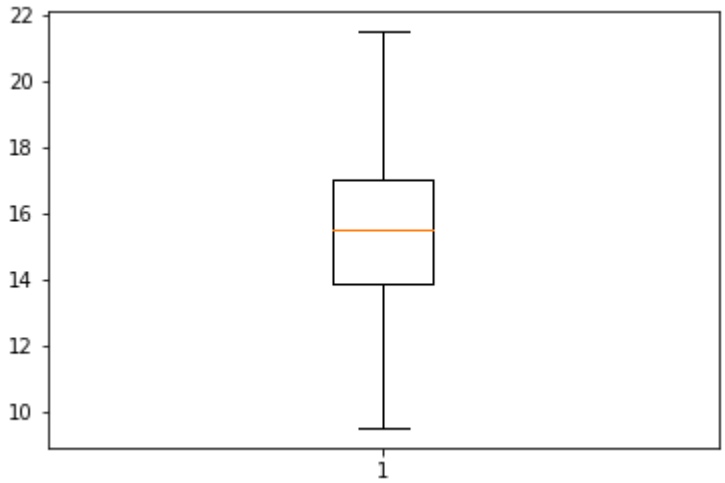


In [18]:
```
plt.boxplot(df['Acceleration'])
plt.show()
```

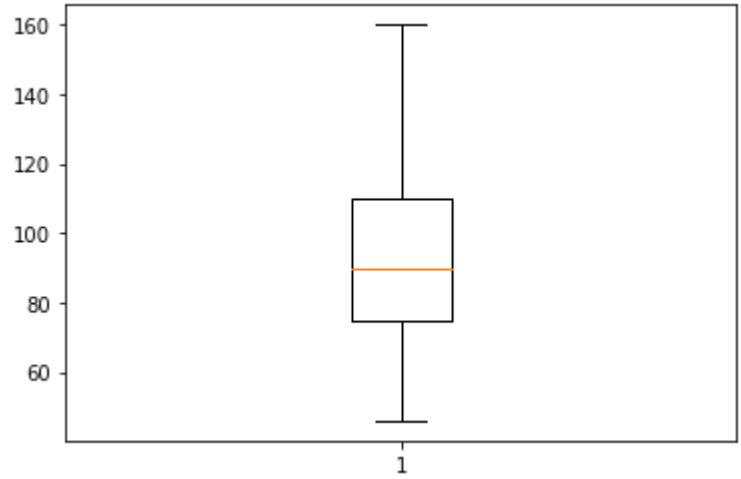


In [19]:
```
#user defined function to remove outliers
def remove_outliers(d,c):
    q1=d[c].quantile(0.25)
    q3=d[c].quantile(0.75)
    iqr=q3-q1
    ub=q3+1.5*iqr
    lb=q1-1.5*iqr
    #remove outliers and store good data in result
    result=d[(d[c]>=lb) & (d[c]<=ub)]
    return result
```

In [21]:
```
#remove outliers from Acceleration
df=remove_outliers(df,'Acceleration')
plt.boxplot(df['Acceleration'])
plt.show()
```



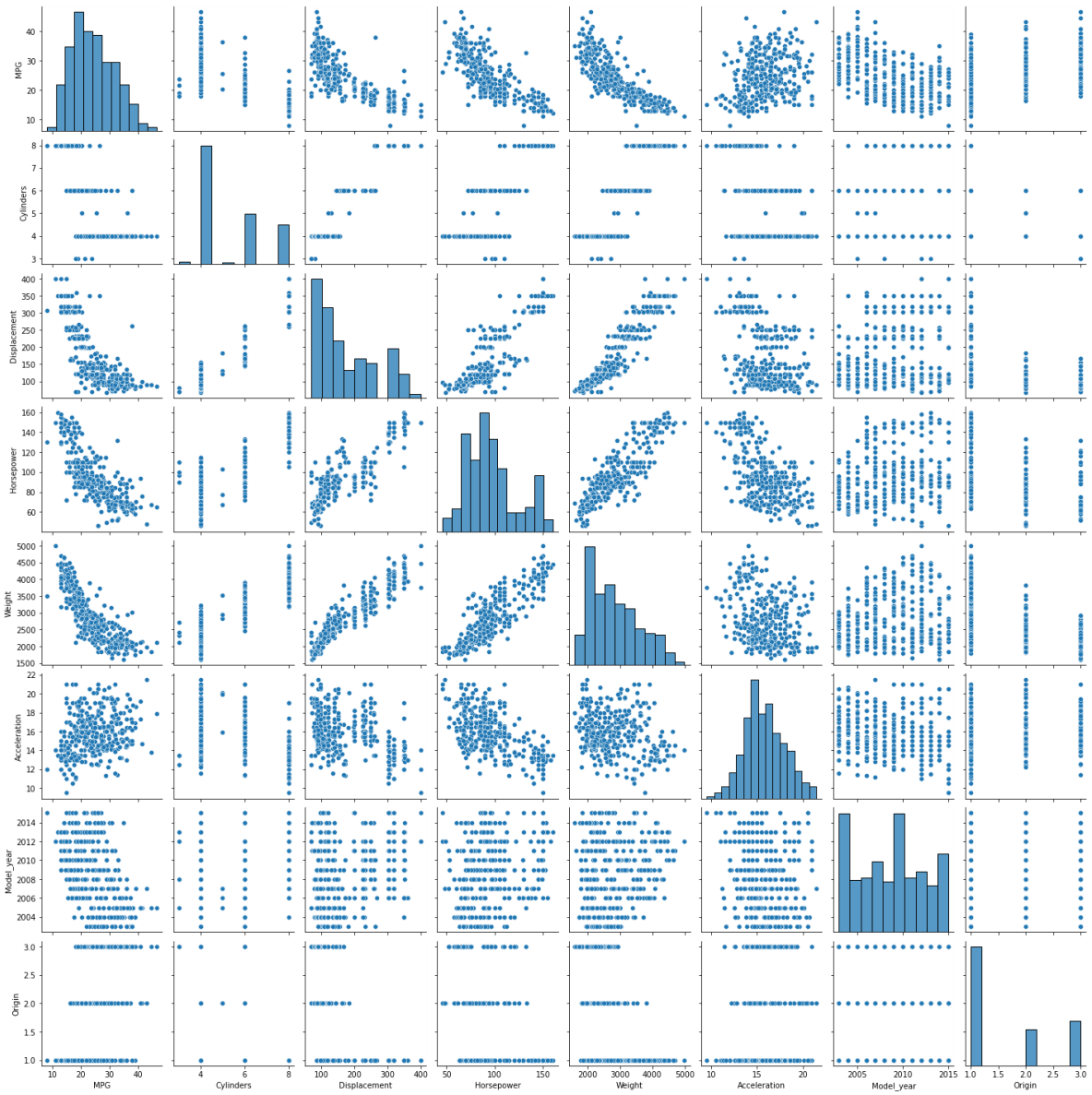
In [24]:
```
#remove outliers from Horsepower
df=remove_outliers(df,'Horsepower')
plt.boxplot(df['Horsepower'])
plt.show()
```



EDA(Exploratory Data Analysis)

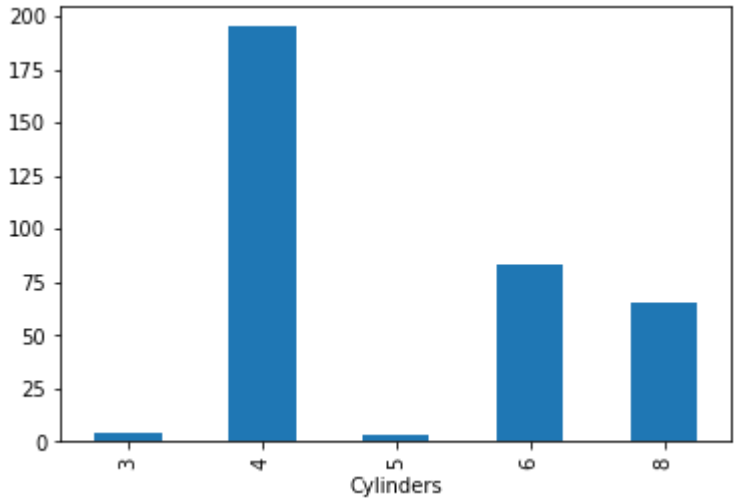
In [25]:
```
#create pairplot
sns.pairplot(df)
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x2af970ddeb0>

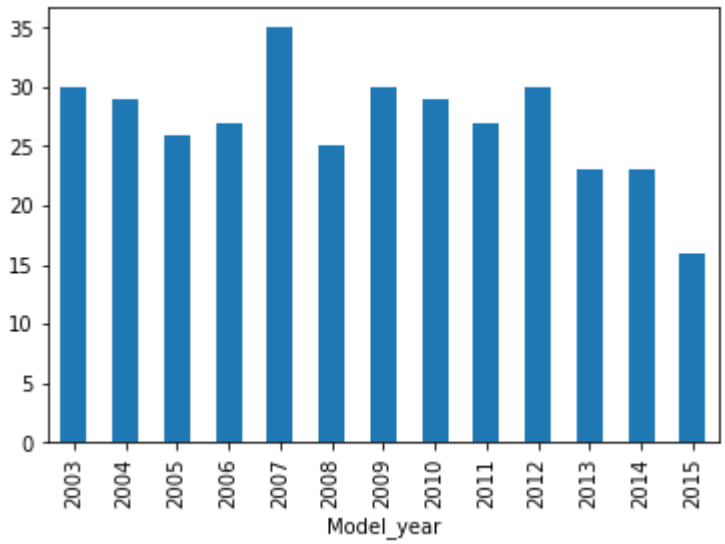


In [26]:
```
#data mix
#'Cylinders','Model_year','Origin','Car_Name'
```

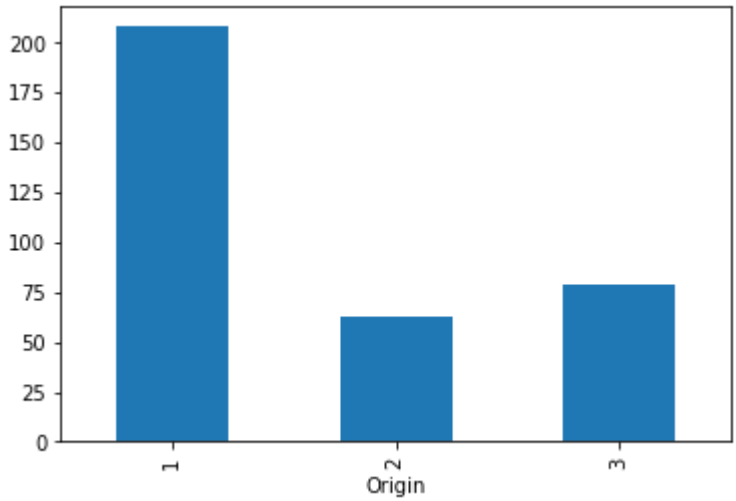
In [27]:
```
df.groupby('Cylinders')['Cylinders'].count().plot(kind='bar')
plt.show()
```



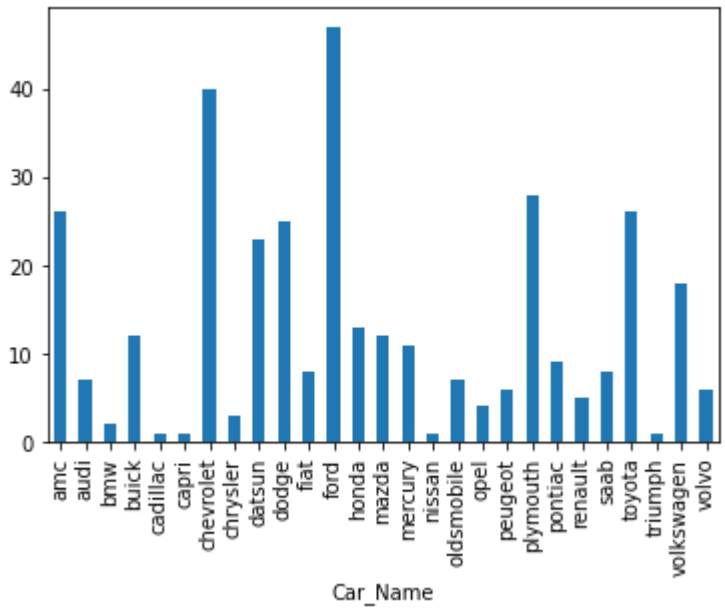
In [28]:
```
df.groupby('Model_year')['Model_year'].count().plot(kind='bar')
plt.show()
```



In [29]:
```
df.groupby('Origin')['Origin'].count().plot(kind='bar')
plt.show()
```

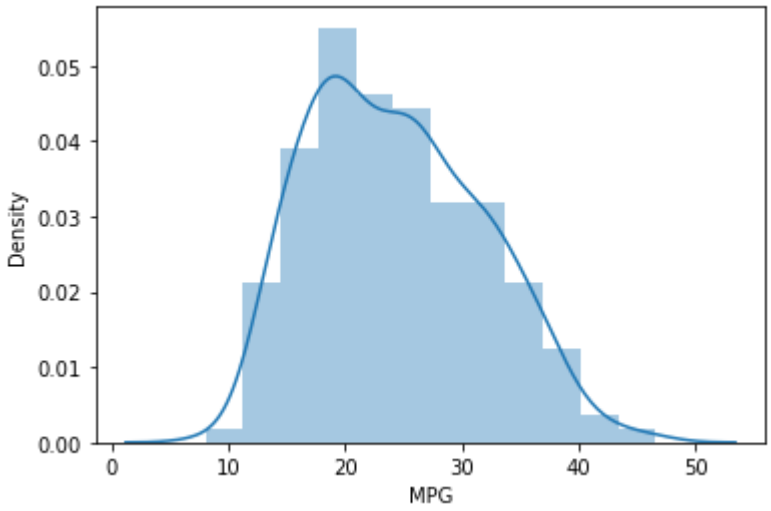


In [30]:
```
df.groupby('Car_Name')['Car_Name'].count().plot(kind='bar')
plt.show()
```



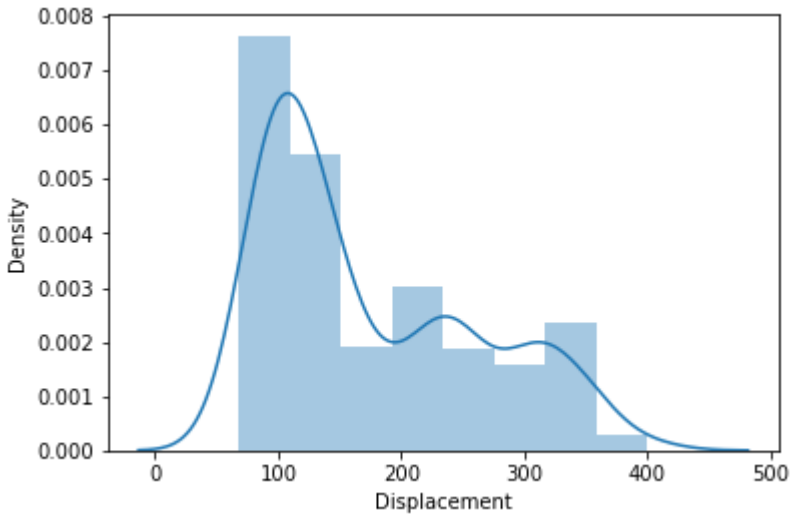
In [31]:
```
#distribution
sns.distplot(df['MPG'])
plt.show()
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```



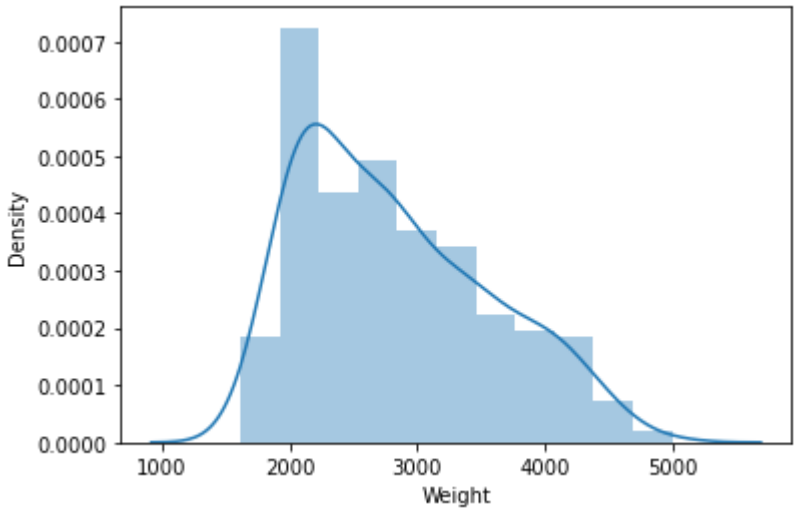
In [32]:
```
sns.distplot(df['Displacement'])
plt.show()
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```



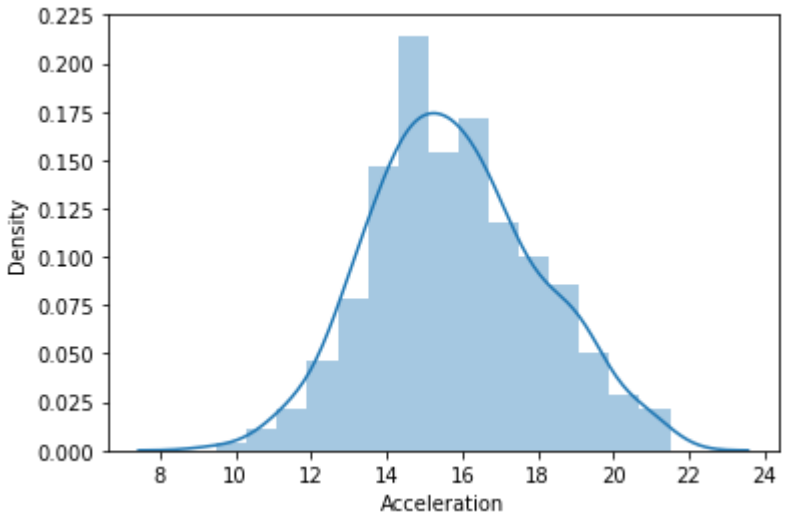
In [33]:
```
sns.distplot(df['Weight'])
plt.show()
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```



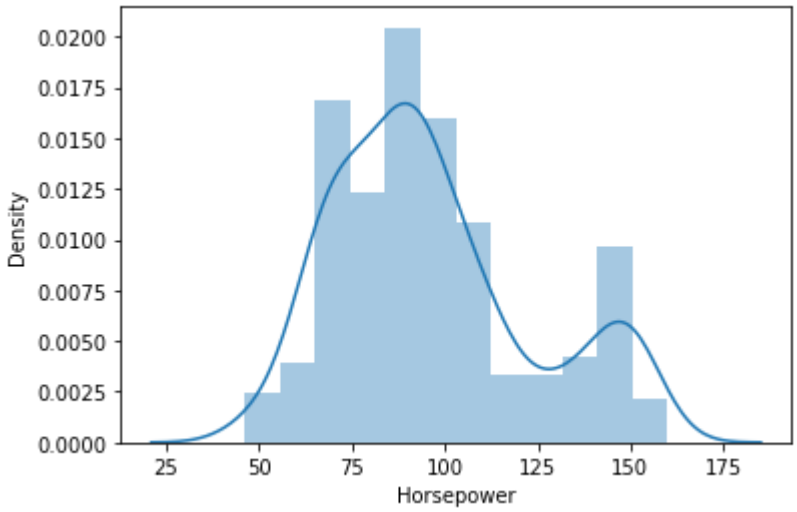
In [34]:
```
sns.distplot(df['Acceleration'])
plt.show()
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```



In [35]:
```
sns.distplot(df['Horsepower'])
plt.show()
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [36]:
```
#check object variables for spelling differences, and redundant data
#Cylinders   object
#Model_year  object
#Origin     object
#Car_Name     object
```

In [37]:
```
df['Cylinders'].unique()
```

Out[37]:
```
array([8, 4, 6, 3, 5], dtype=object)
```

In [38]:
```
df['Model_year'].unique()
```

Out[38]:
```
array([2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 2006, 2005,
       2004, 2003], dtype=object)
```

In [39]:
```
df['Origin'].unique()
```

Out[39]:
```
array([1, 3, 2], dtype=object)
```

In [40]:
```
df['Car_Name'].unique()
```

Out[40]:
```
array(['chevrolet', 'plymouth', 'amc', 'ford', 'toyota', 'datsun',
       'volkswagen', 'peugeot', 'audi', 'saab', 'bmw', 'pontiac',
       'mercury', 'opel', 'fiat', 'dodge', 'buick', 'oldsmobile', 'mazda',
       'volvo', 'renault', 'honda', 'capri', 'chrysler', 'cadillac',
       'triumph', 'nissan'], dtype=object)
```

Feature Engineering:One-hot-encoding(dummy conversion)

In [41]:
```
#store all categorical variables in a new dataframe
df_categorical=df.select_dtypes(include=['object'])
```

In [42]:
```
df_categorical.head()
```

Out[42]:

	Cylinders	Model_year	Origin	Car_Name
0	8	2015	1	chevrolet
2	8	2015	1	plymouth
3	8	2015	1	amc
4	8	2015	1	ford
12	8	2015	1	chevrolet

In [43]:
```
#create dummy
dummy=pd.get_dummies(df_categorical,drop_first=True)
```

In [44]:
```
dummy.head()
```

Out[44]:

	Cylinders_4	Cylinders_5	Cylinders_6	Cylinders_8	Model_year_2004	Model_year_2005	Mod
0	0	0	0	1	0	0	
2	0	0	0	1	0	0	
3	0	0	0	1	0	0	
4	0	0	0	1	0	0	
12	0	0	0	1	0	0	

5 rows × 44 columns

In [45]:
```
#combine numeric columns from df with dummy columns  to create final data
df_numeric=df.select_dtypes(include=['int64','float64'])
```

In [46]:
```
df_master=pd.concat([df_numeric,dummy],axis=1)
```

In [47]:
```
df_master.head()
```

Out[47]:

	MPG	Displacement	Horsepower	Weight	Acceleration	Cylinders_4	Cylinders_5	Cylinders_
0	8.0	307.0	130.0	3504	12.0	0	0	
2	18.0	318.0	150.0	3436	11.0	0	0	
3	16.0	304.0	150.0	3433	12.0	0	0	
4	17.0	302.0	140.0	3449	10.5	0	0	
12	15.0	400.0	150.0	3761	9.5	0	0	

5 rows × 49 columns

In [48]:
```
#export final data to excel
df_master.to_excel(r"C:\Users\ACER\Desktop\introtallent\python\regression data
```

Create X (with all independent variables) and Y (With the target variable)

In [49]:
```
x=df_master.drop('MPG',axis=1)
```

In [50]:
```
y=df_master['MPG']
```

Random sampling: create training and test samples

In [51]:
```
#create training and test samples
xtrain,xtest,ytrain,ytest=train_test_split(x,y,train_size=0.7,random_state=0)
```

Feature Selection

In [52]:
```
#create key_features object to select the top k features
# key_features = SelectKBest(score_func=f_regression,k='all')
key_features=SelectKBest(score_func=f_regression,k=5)
#to select 5 significant features

#Fit the key_features to the training data and transform it
xtrain_selected=key_features.fit_transform(xtrain,ytrain)

#Get the indices of the selected features
selected_indices=key_features.get_support(indices=True)

#Get the names of the selected features
selected_features=xtrain.columns[selected_indices]
```

```
C:\Users\ACER\anaconda3\lib\site-packages\sklearn\feature_selection\_univaria
te_selection.py:302: RuntimeWarning: invalid value encountered in true_divide
  corr /= X_norms
```

In [53]:
```
selected_features
```

Out[53]:
```
Index(['Displacement', 'Horsepower', 'Weight', 'Cylinders_4', 'Cylinders_8'],
dtype='object')
```

Instantiate linear regression

In [54]:
```
linreg=LinearRegression()
```

Model 1: Build training model using all features

In [55]:
```
#train your model
linreg.fit(xtrain,ytrain)
```

Out[55]:
```
LinearRegression()
```

In [56]:
```
#check accuracy of the model
linreg.score(xtrain,ytrain)
```

Out[56]:
```
0.8810719879660667
```

In [57]:
```
#test yours model's learning
#predict mileage using test samples
predicted_mileage=linreg.predict(xtest)
```

In [58]:
```
#check r-squared (accuracy score)
linreg.score(xtest,ytest)
```

Out[58]: 0.8698795919221128

Model 2: Build model using KBest selected feaures

In [59]:
```
##store KBest columns from xtrain to xtarin_kbest
xtrain_kbest=xtrain[selected_features]
```

In [60]:
```
xtrain_kbest.head()
```

Out[60]:

	Displacement	Horsepower	Weight	Cylinders_4	Cylinders_8
334	70.0	100.0	2420	0	0
175	90.0	70.0	1937	1	0
112	122.0	85.0	2310	1	0
2	318.0	150.0	3436	0	1
198	91.0	53.0	1795	1	0

In [61]:
```
#train your model
linreg.fit(xtrain_kbest,ytrain)
```

Out[61]: LinearRegression()

In [62]:
```
linreg.score(xtrain_kbest,ytrain)
```

Out[62]: 0.7110901050565608

In [63]:
```
#store KBest columns from xtest to xtest_kbest
xtest_kbest=xtest[selected_features]
```

In [64]:
```
pred_y=linreg.predict(xtest_kbest)
```

In [65]:
```
linreg.score(xtest_kbest,ytest)
```

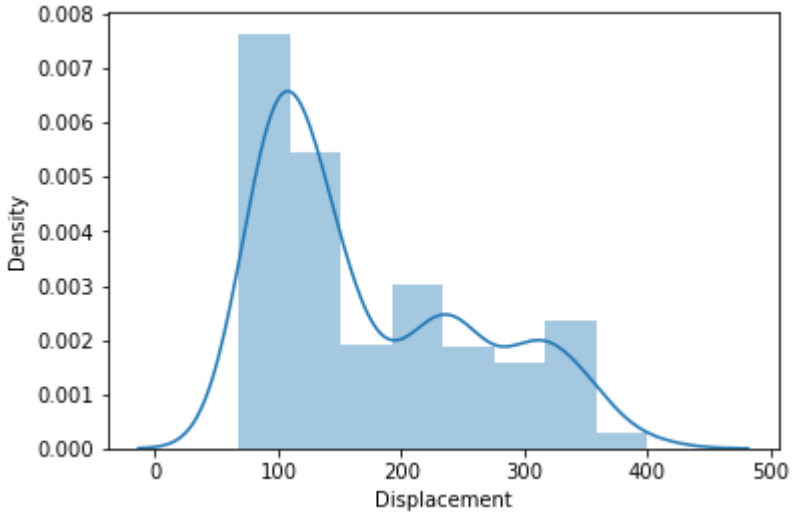
Out[65]: 0.6951570744108746

log transformation

BUILT MODEL AFTER REDUCING SKEWNESS IN THE DISPLACEMENT AND WEIGHT VARIABLE

In [66]:
```
sns.distplot(df['Displacement'])
plt.show()
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [67]:
```
import numpy as np
df['log_Displacement']=np.log(df['Displacement'])
```

In [68]:
```
df.head(3)
```

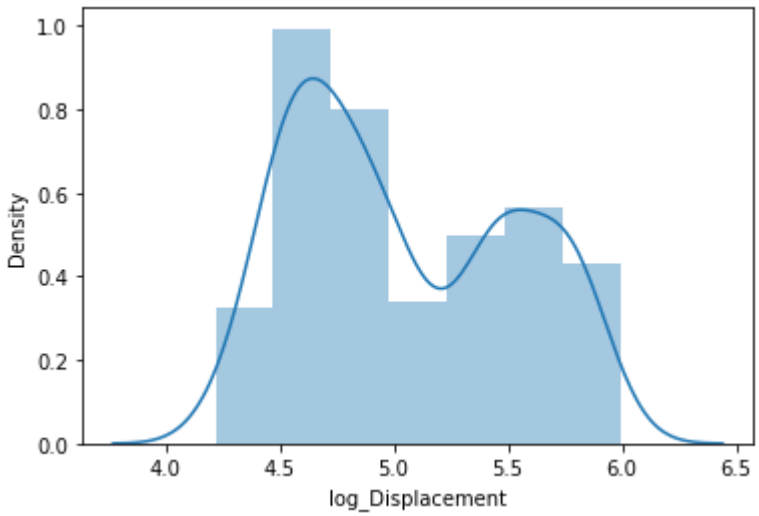
Out[68]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model_year	Origin	Car_N
0	8.0	8	307.0	130.0	3504	12.0	2015	1	chev
2	18.0	8	318.0	150.0	3436	11.0	2015	1	plym
3	16.0	8	304.0	150.0	3433	12.0	2015	1	

In [69]:
```
sns.distplot(df['log_Displacement'])
```

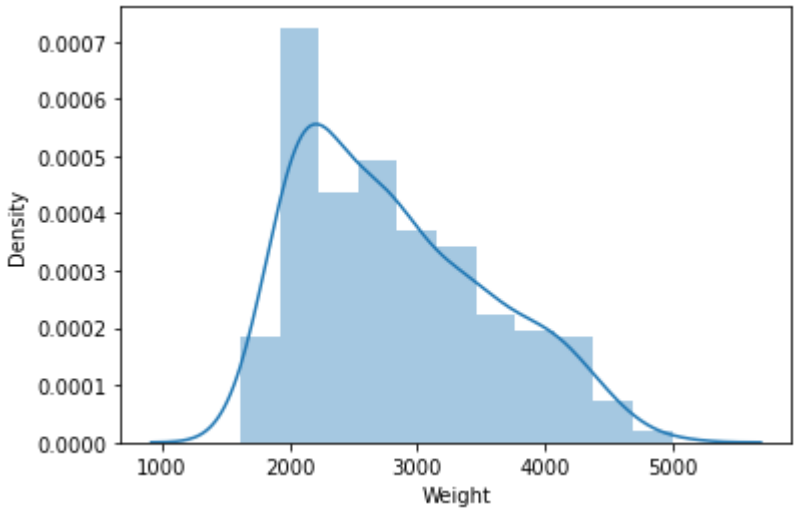
```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```

Out[69]: <AxesSubplot:xlabel='log_Displacement', ylabel='Density'>



In [70]:
```
sns.distplot(df['Weight'])
plt.show()
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```



In [71]:
```
df['log_Weight']=np.log(df['Weight'])
```

In [72]:
```
df.head()
```

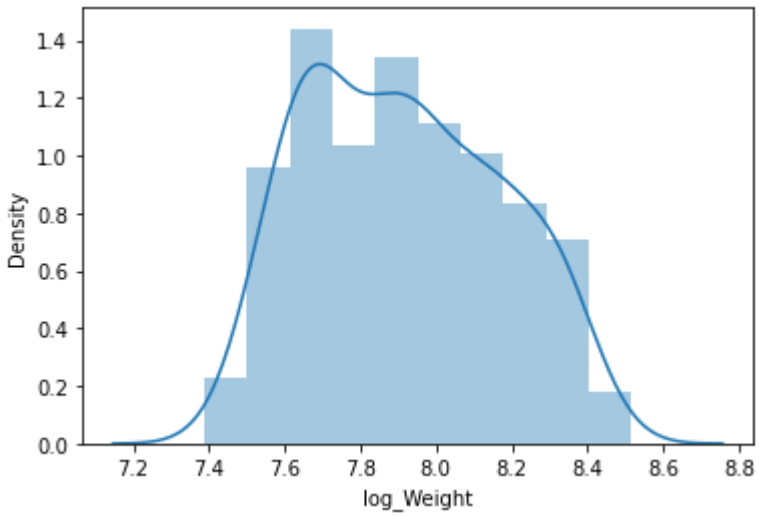
Out[72]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model_year	Origin	Car_I
0	8.0	8	307.0	130.0	3504	12.0	2015	1	che
2	18.0	8	318.0	150.0	3436	11.0	2015	1	plyr
3	16.0	8	304.0	150.0	3433	12.0	2015	1	
4	17.0	8	302.0	140.0	3449	10.5	2015	1	
12	15.0	8	400.0	150.0	3761	9.5	2015	1	che

In [73]:
```
sns.distplot(df['log_Weight'])
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[73]:
```
<AxesSubplot:xlabel='log_Weight', ylabel='Density'>
```



In [74]:
```
df=df.drop(['Displacement','Weight'],axis=1)
df.head()
```

Out[74]:

	MPG	Cylinders	Horsepower	Acceleration	Model_year	Origin	Car_Name	log_Displacemen
0	8.0	8	130.0	12.0	2015	1	chevrolet	5.726848
2	18.0	8	150.0	11.0	2015	1	plymouth	5.762051
3	16.0	8	150.0	12.0	2015	1	amc	5.717028
4	17.0	8	140.0	10.5	2015	1	ford	5.710427
12	15.0	8	150.0	9.5	2015	1	chevrolet	5.991465

In [75]:
```
df.dtypes
```

Out[75]:
```
MPG                float64
Cylinders           object
Horsepower         float64
Acceleration       float64
Model_year          object
Origin              object
Car_Name            object
log_Displacement   float64
log_Weight         float64
dtype: object
```

In [76]:
```
dummy.head()
```

Out[76]:

	Cylinders_4	Cylinders_5	Cylinders_6	Cylinders_8	Model_year_2004	Model_year_2005	Mod
0	0	0	0	1	0	0	
2	0	0	0	1	0	0	
3	0	0	0	1	0	0	
4	0	0	0	1	0	0	
12	0	0	0	1	0	0	

5 rows × 44 columns

In [77]:
```
#combine numeric columns from df with dummy columns  to create final data
df_numeric=df.select_dtypes(include=['int64','float64'])
```

In [78]:
```
df_master=pd.concat([df_numeric,dummy],axis=1)
```

In [79]:
```
df_master.head()
```

Out[79]:

	MPG	Horsepower	Acceleration	log_Displacement	log_Weight	Cylinders_4	Cylinders_5	Cy
0	8.0	130.0	12.0	5.726848	8.161660	0	0	
2	18.0	150.0	11.0	5.762051	8.142063	0	0	
3	16.0	150.0	12.0	5.717028	8.141190	0	0	
4	17.0	140.0	10.5	5.710427	8.145840	0	0	
12	15.0	150.0	9.5	5.991465	8.232440	0	0	

5 rows × 49 columns

create x (with all independent variable) and y(with all target variable)

In [80]:
```
x=df_master.drop('MPG',axis=1)
```

In [81]:
```
y=df_master['MPG']
```

Random sampling :create training and test samples

In [82]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,train_size=0.7,random_state=0)
```

feature selection

In [83]:
```
#create key_features object to select the top k features
# key_features = SelectKBest(score_func=f_regression,k='all')
key_features=SelectKBest(score_func=f_regression,k=5)
#to select 5 significant features

#Fit the key_features to the training data and transform it
xtrain_selected=key_features.fit_transform(xtrain,ytrain)

#Get the indices of the selected features
selected_indices=key_features.get_support(indices=True)

#Get the names of the selected features
selected_features=xtrain.columns[selected_indices]
```

```
C:\Users\ACER\anaconda3\lib\site-packages\sklearn\feature_selection\_univaria
te_selection.py:302: RuntimeWarning: invalid value encountered in true_divide
  corr /= X_norms
```

In [84]:
```
selected_features
```

Out[84]:
```
Index(['Horsepower', 'log_Displacement', 'log_Weight', 'Cylinders_4',
       'Cylinders_8'],
      dtype='object')
```

Model 3: Build model using KBest selected features

In [90]:
```
##store KBest columns from xtrain to xtarin_kbest
xtrain_kbest=xtrain[selected_features]
```

In [91]:
```
#train your model
linreg.fit(xtrain_kbest,ytrain)
```

Out[91]:
```
LinearRegression()
```

In [92]:
```
linreg.score(xtrain_kbest,ytrain)
```

Out[92]:
```
0.7118189417837262
```

In [93]:
```
#store KBest columns from xtest to xtest_kbest
xtest_kbest=xtest[selected_features]
```

In [94]:
```
pred_y=linreg.predict(xtest_kbest)
```

In [95]:
```
linreg.score(xtest_kbest,ytest)
```

Out[95]:
```
0.7038559215939076
```

In [ ]:
```
#slightly changes after log transformation
```

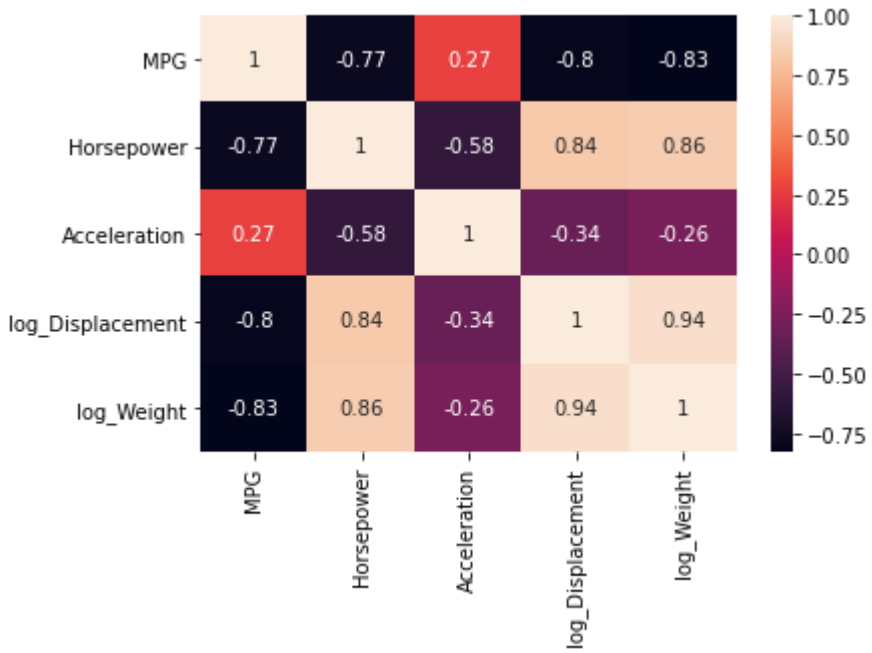
model 4

mutlicollinearity checking

In [100]:
```
new_df_corr=df_numeric.corr()
```

In [101]:
```
sns.heatmap(new_df_corr,cmap='YlGnBu',annot=True) #YlGnBu yellow green blue
```

Out[101]: <AxesSubplot:>



In [102]:
```
#drop mutlicollunear variables from xtarin and xtest
xtrain=xtrain.drop(['log_Weight','log_Displacement'],axis=1)
xtest=xtest.drop(['log_Weight','log_Displacement'],axis=1)
```

feature selection

In [103]:
```
#create key_features object to select the top k features
# key_features = SelectKBest(score_func=f_regression,k='all')
key_features=SelectKBest(score_func=f_regression,k=5)
#to select 5 significant features

#Fit the key_features to the training data and transform it
xtrain_selected=key_features.fit_transform(xtrain,ytrain)

#Get the indices of the selected features
selected_indices=key_features.get_support(indices=True)

#Get the names of the selected features
selected_features=xtrain.columns[selected_indices]
```

```
C:\Users\ACER\anaconda3\lib\site-packages\sklearn\feature_selection\_univaria
te_selection.py:302: RuntimeWarning: invalid value encountered in true_divide
  corr /= X_norms
```

In [104]:
```
selected_features
```

Out[104]:
```
Index(['Horsepower', 'Cylinders_4', 'Cylinders_8', 'Model_year_2005',
       'Origin_3'],
      dtype='object')
```

Build model using KBest selected features

In [105]:
```
##store KBest columns from xtrain to xtarin_kbest
xtrain_kbest=xtrain[selected_features]
```

In [106]:
```
#train your model
linreg.fit(xtrain_kbest,ytrain)
```

Out[106]:
```
LinearRegression()
```

In [107]:
```
linreg.score(xtrain_kbest,ytrain)
```

Out[107]:
```
0.7191991282910362
```

In [108]:
```
#store KBest columns from xtest to xtest_kbest
xtest_kbest=xtest[selected_features]
```

In [109]:
```
pred_y=linreg.predict(xtest_kbest)
```

In [110]:
```
linreg.score(xtest_kbest,ytest)
```

Out[110]:
```
0.7288670006944613
```

In [ ]: