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]:

```
df=df.drop("Car_Name",axis=1)
```

In [5]:

```
df=df.drop("Model_year",axis=1)
```

In [6]:

```
df.head()
```

Out[6]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Origin
0	8.0	8	307.0	130	3504	12.0	1
1	15.0	8	350.0	165	3693	11.5	1
2	18.0	8	318.0	150	3436	11.0	1
3	16.0	8	304.0	150	3433	12.0	1
4	17.0	8	302.0	140	3449	10.5	1

In [7]:

```
#print row and column count
df.shape
```

Out[7]:

```
(398, 7)
```

In [8]:

```
df.dtypes
```

Out[8]:

```
MPG             float64
Cylinders         int64
Displacement    float64
Horsepower       object
Weight            int64
Acceleration    float64
Origin            int64
dtype: object
```

In [9]:

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

In [10]:

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

In [11]:

```
df.dtypes
```

Out[11]:

```
MPG             float64
Cylinders        object
Displacement    float64
Horsepower      float64
Weight            int64
Acceleration    float64
Origin           object
dtype: object
```

In [12]:

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

Out[12]:

```
MPG             0
Cylinders       0
Displacement    0
Horsepower      6
Weight          0
Acceleration    0
Origin          0
dtype: int64
```

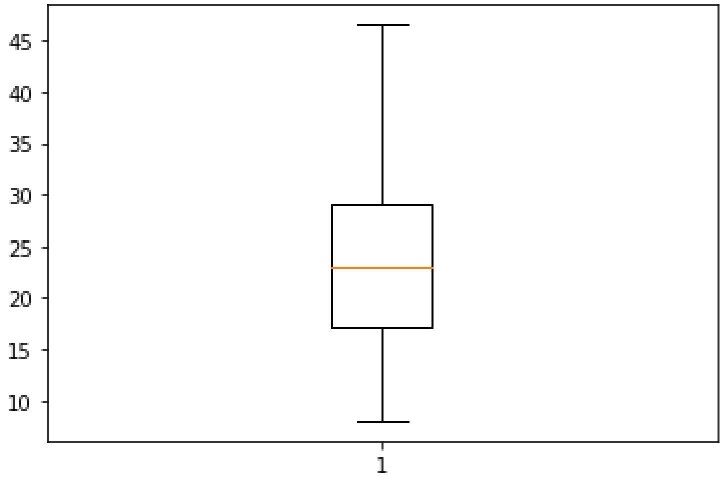
In [13]:

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

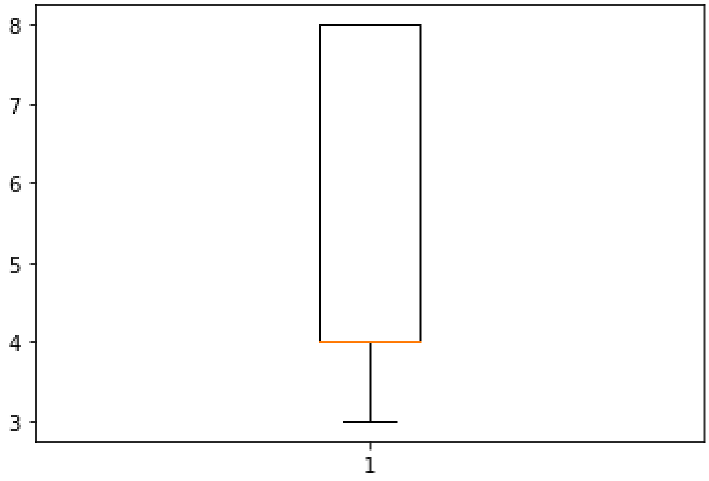
In [14]:
```
df.isnull().sum()
```

Out[14]:
```
MPG             0
Cylinders       0
Displacement    0
Horsepower      0
Weight          0
Acceleration    0
Origin          0
dtype: int64
```

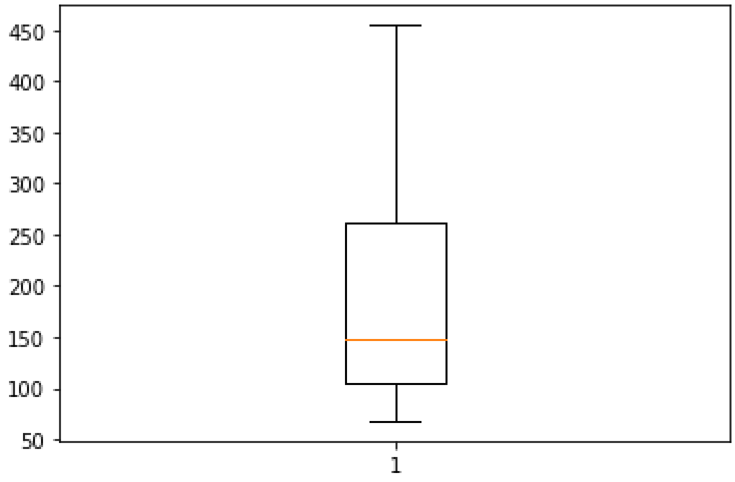
In [15]:
```
#feature engineering-outlier treatment
plt.boxplot(df['MPG'])
plt.show()
```



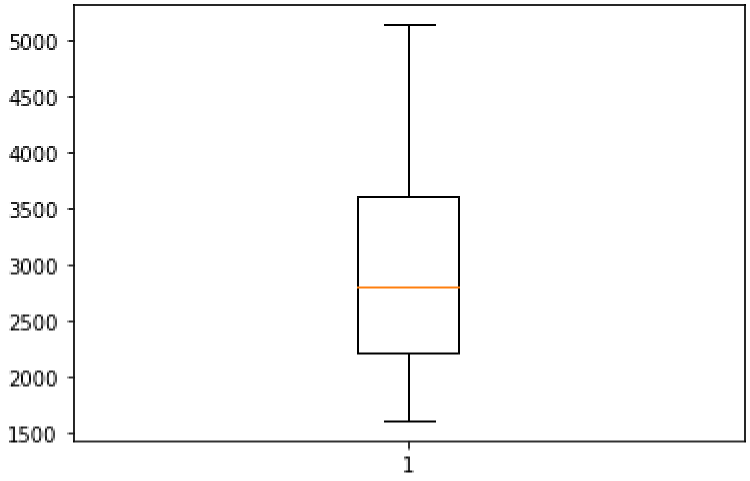
In [16]:
```
plt.boxplot(df['Cylinders'])
plt.show()
```



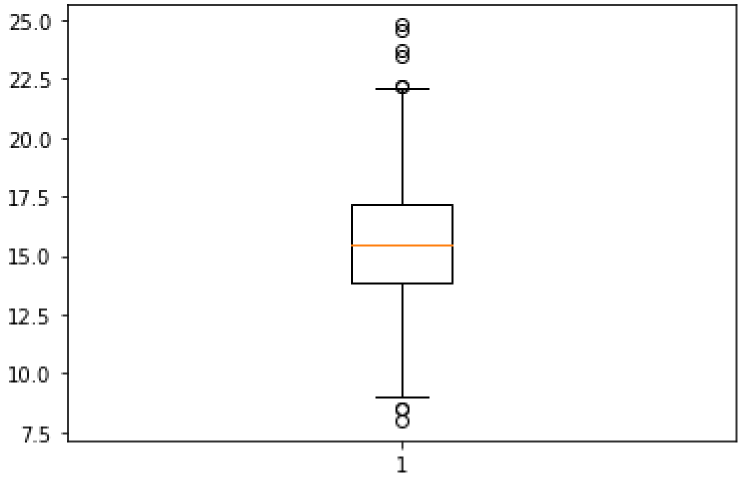
In [17]:
```
plt.boxplot(df['Displacement'])
plt.show()
```



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

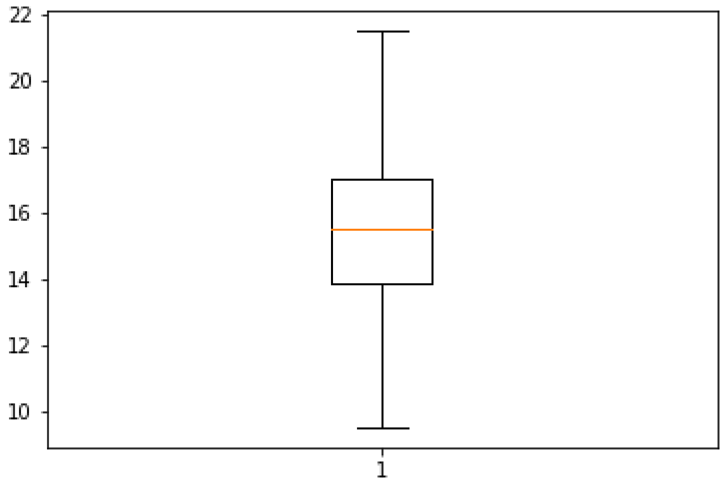


In [19]:
```
plt.boxplot(df['Acceleration'])
plt.show()
```

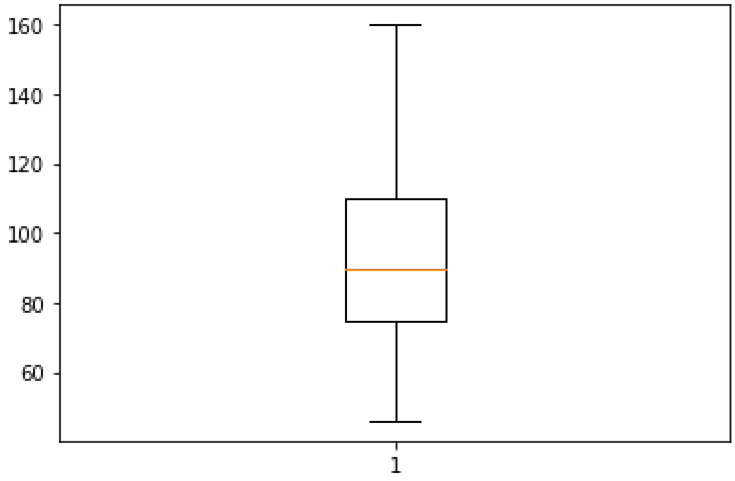


In [20]:
```
#user defined function to remove outliers
def remove_outliers(d,c):
    q1=d[c].quantile(0.25)
    q3=d[c].quantile(0.75)
    iqr=q3-q1
    ub=q3+1.5*iqr
    lb=q1-1.5*iqr
    #remove outliers and store good data in result
    result=d[(d[c]>=lb) & (d[c]<=ub)]
    return result
```

In [22]:
```
#remove outliers from Acceleration
df=remove_outliers(df,'Acceleration')
plt.boxplot(df['Acceleration'])
plt.show()
```



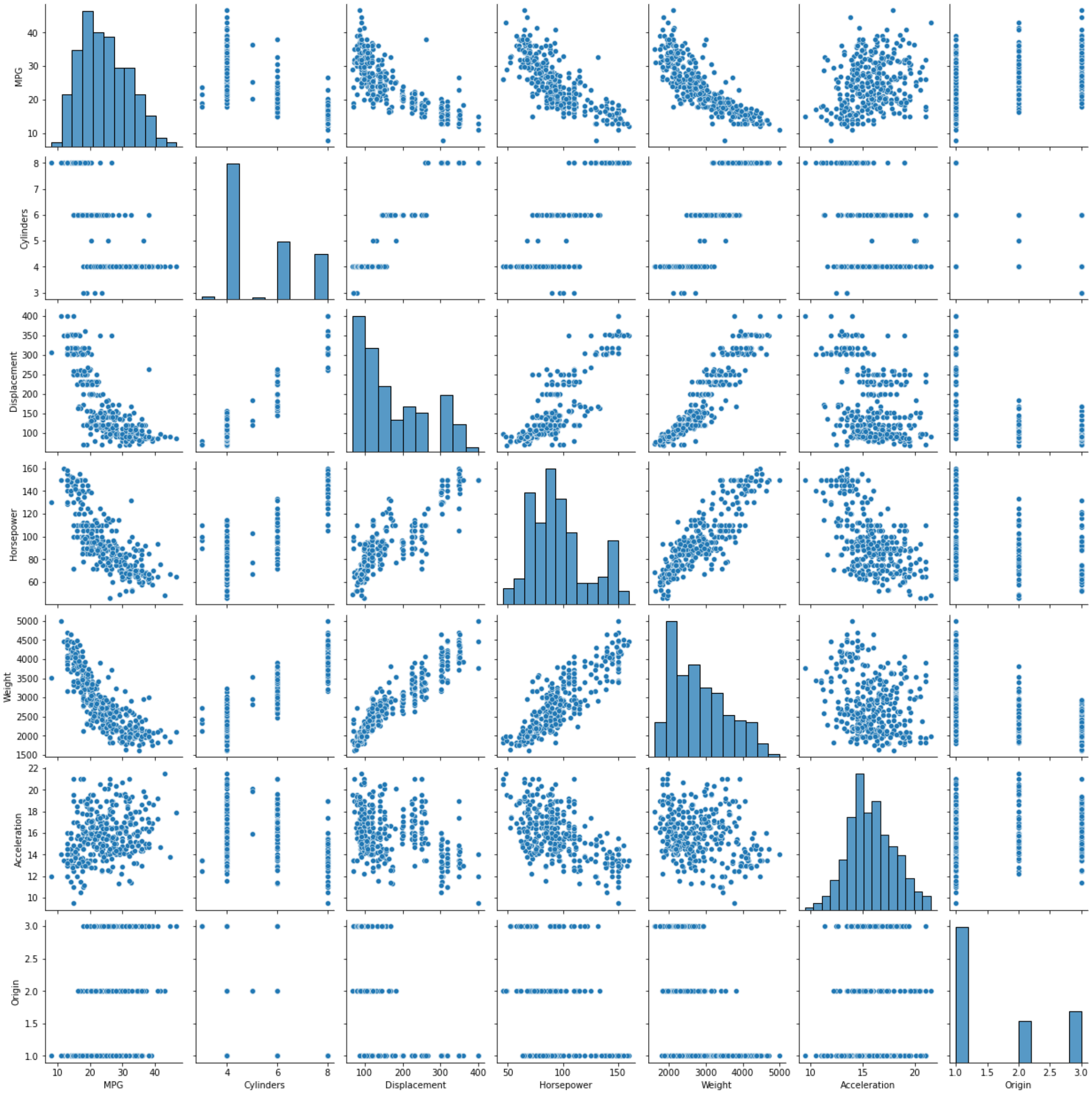
In [25]:
```
#remove outliers from Horsepower
df=remove_outliers(df,'Horsepower')
plt.boxplot(df['Horsepower'])
plt.show()
```



EDA(Exploratory Data Analysis)

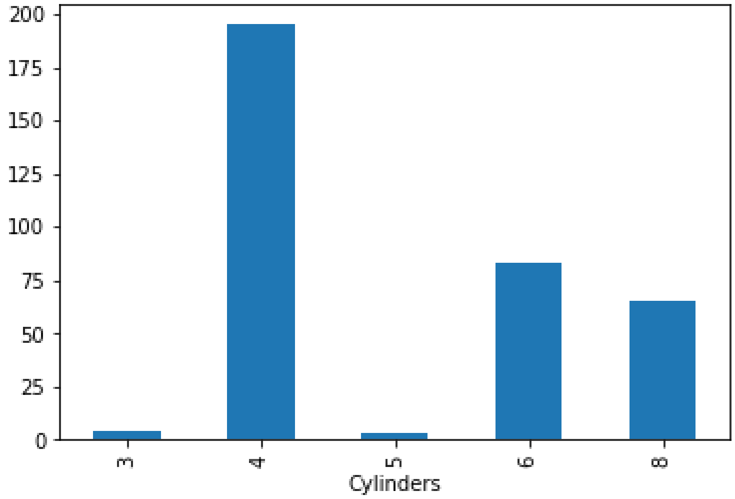
In [26]:
```
#create pairplot
sns.pairplot(df)
```

Out[26]:
```
<seaborn.axisgrid.PairGrid at 0x1683cf5f7f0>
```

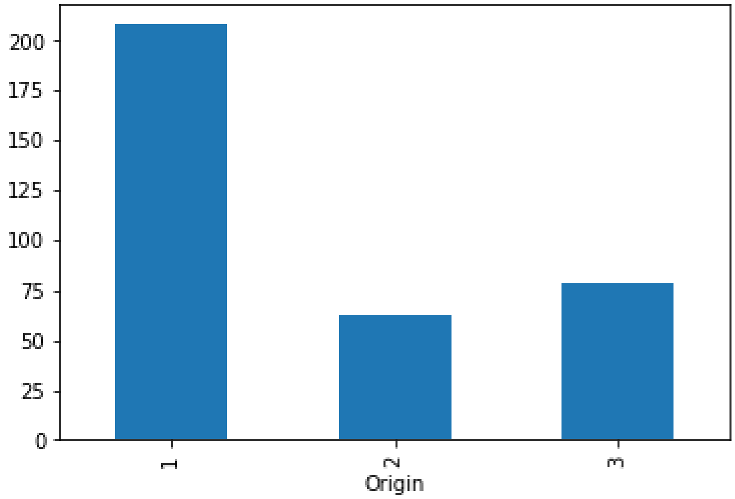


In [27]:
```
#data mix
#'Cylinders','Origin"
```

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



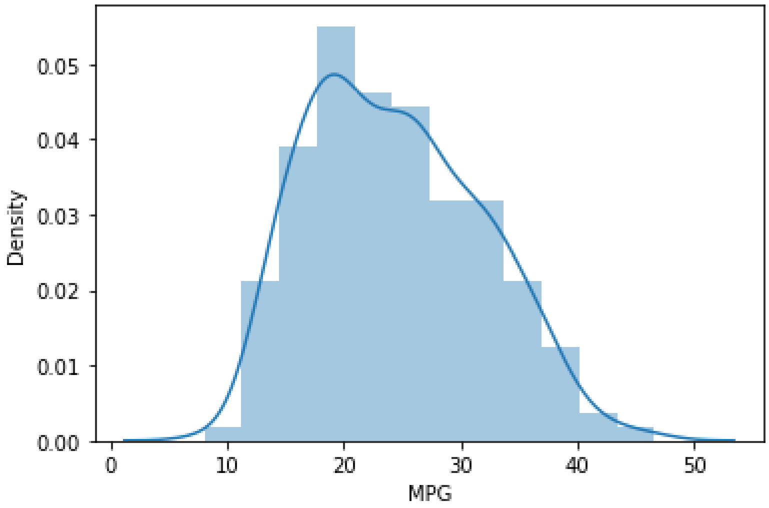
In [29]:
```
df.groupby('Origin')['Origin'].count().plot(kind="bar")
plt.show()
```



In [30]:

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

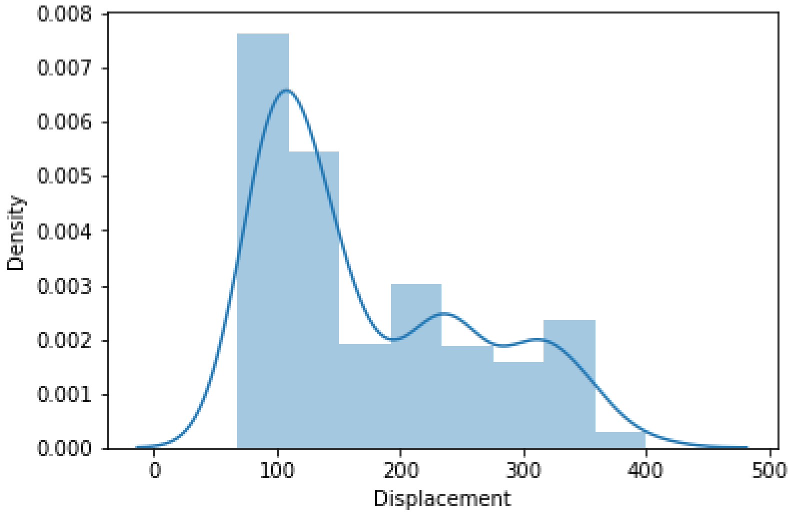
```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function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



In [31]:

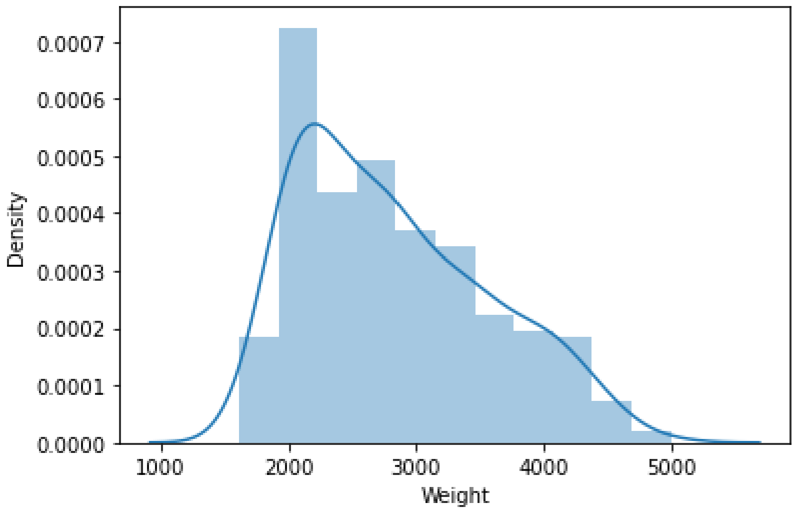
```
sns.distplot(df['Displacement'])
plt.show()
```

```
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)
```



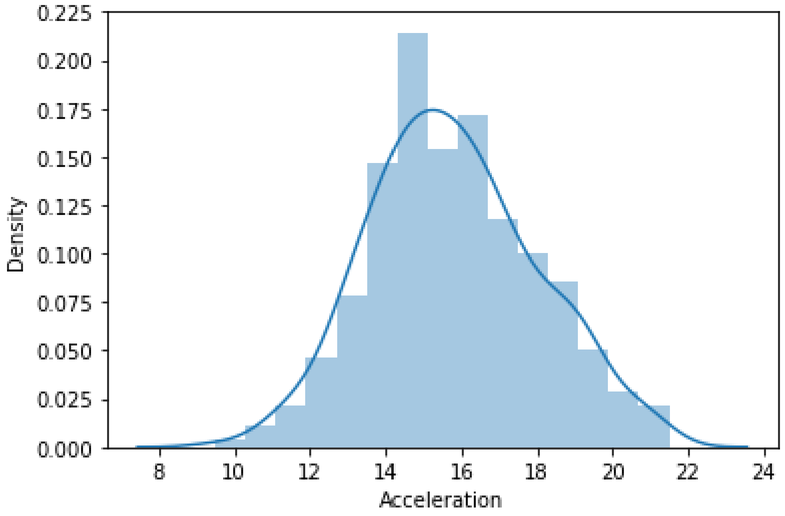
In [32]:
```
sns.distplot(df['Weight'])
plt.show()
```

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

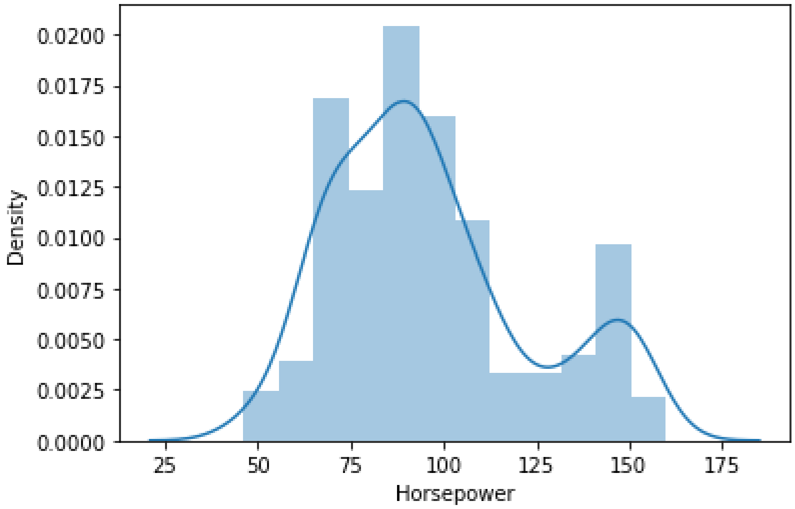
```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function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



In [34]:

```
sns.distplot(df['Horsepower'])
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]:

```
#check object variables for spelling differences, and redundant data
#Cylinders    object
#Origin      object
```

In [36]:

```
df['Cylinders'].unique()
```

Out[36]:

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

In [37]:

```
df['Origin'].unique()
```

Out[37]:

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

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

In [38]:

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

In [39]:
```
df_categorical.head()
```

Out[39]:

	Cylinders	Origin
0	8	1
2	8	1
3	8	1
4	8	1
12	8	1

In [40]:
```
#create dummy
dummy=pd.get_dummies(df_categorical,drop_first=True)
```

In [41]:
```
dummy.head()
```

Out[41]:

	Cylinders_4	Cylinders_5	Cylinders_6	Cylinders_8	Origin_2	Origin_3
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

In [42]:
```
#combine numeric columns from df with dummy columns  to create final data
df_numeric=df.select_dtypes(include=['int64','float64'])
```

In [43]:
```
df_master=pd.concat([df_numeric,dummy],axis=1)
```

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

Out[44]:

	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	

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

```
In [45]: x=df_master.drop('MPG',axis=1)
```

```
In [46]: y=df_master['MPG']
```

Random sampling: create training and test samples

```
In [47]: #create training and test samples
         xtrain,xtest,ytrain,ytest=train_test_split(x,y,train_size=0.7,random_state=0)
```

Feature Selection

```
In [48]: #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)
         #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]
```

```
In [49]: selected_features
```

```
Out[49]: Index(['Displacement', 'Horsepower', 'Weight', 'Cylinders_4', 'Cylinders_8'],
         dtype='object')
```

Instantiate linear regression

```
In [50]: linreg=LinearRegression()
```

Model 1: Build training model using all features

```
In [51]: linreg.fit(xtrain,ytrain)
```

```
Out[51]: LinearRegression()
```

```
In [52]: linreg.score(xtrain,ytrain)
```

```
Out[52]: 0.731954532119161
```

In [53]:
```
#test yours model's learning
#predict mileage using test samples
predicted_mileage=linreg.predict(xtest)
```

In [54]:
```
#check r-squared (accuracy score)
linreg.score(xtest,ytest)
```

Out[54]:
```
0.7329949257086605
```

Model 2: Build model using KBest selected feaures

In [55]:
```
##store KBest columns from xtrain to xtarin_kbest
xtrain_kbest=xtrain[selected_features]
```

In [56]:
```
xtrain_kbest.head()
```

Out[56]:

	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 [57]:
```
linreg.fit(xtrain_kbest,ytrain)
```

Out[57]:
```
LinearRegression()
```

In [58]:
```
linreg.score(xtrain_kbest,ytrain)
```

Out[58]:
```
0.7110901050565608
```

In [59]:
```
#store KBest columns from xtest to xtest_kbest
xtest_kbest=xtest[selected_features]
```

In [60]:
```
pred_y=linreg.predict(xtest_kbest)
```

In [61]:
```
linreg.score(xtest_kbest,ytest)
```

Out[61]:
```
0.6951570744108746
```

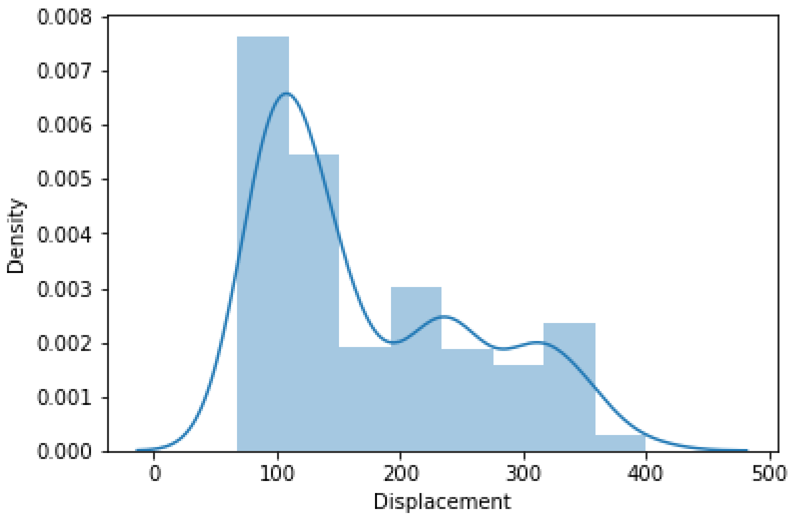
log transformation

BUILT MODEL AFTER REDUCING SKEWNESS IN THE DISPLACEMENT AND WEIGHT VARIABLE

In [63]:

```
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 [64]:

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

In [65]:

```
df.head(3)
```

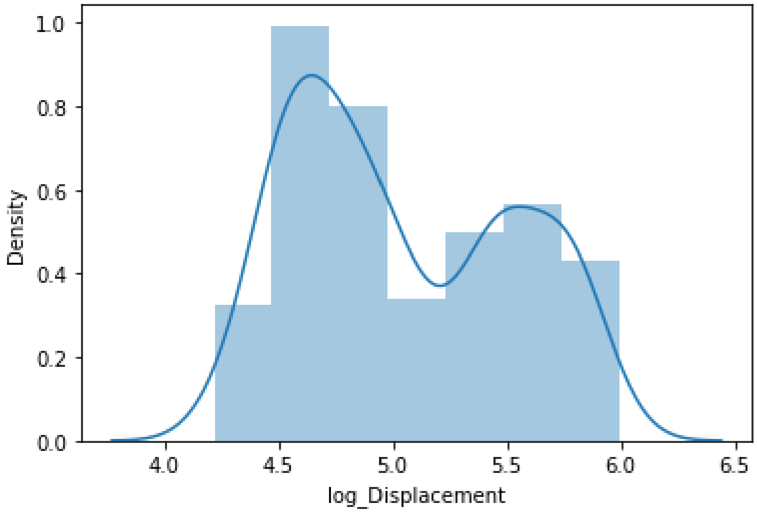
Out[65]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Origin	log_Displacement
0	8.0	8	307.0	130.0	3504	12.0	1	5.726848
2	18.0	8	318.0	150.0	3436	11.0	1	5.762051
3	16.0	8	304.0	150.0	3433	12.0	1	5.717028

In [66]:
```
sns.distplot(df['log_Displacement'])
```

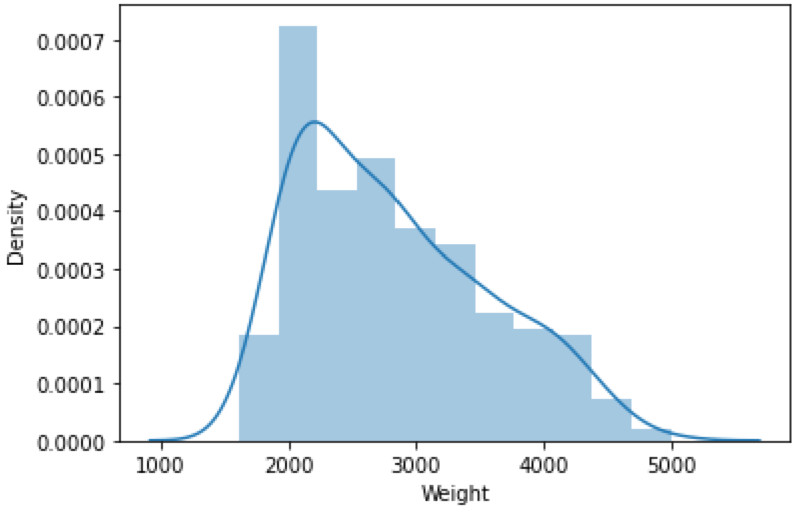
```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futu
reWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function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

Out[66]:
```
<AxesSubplot:xlabel='log_Displacement', ylabel='Density'>
```



In [67]:
```
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 [69]:
```
import numpy as np
df['log_Weight']=np.log(df['Weight'])
```

In [71]:
```
df.head(3)
```

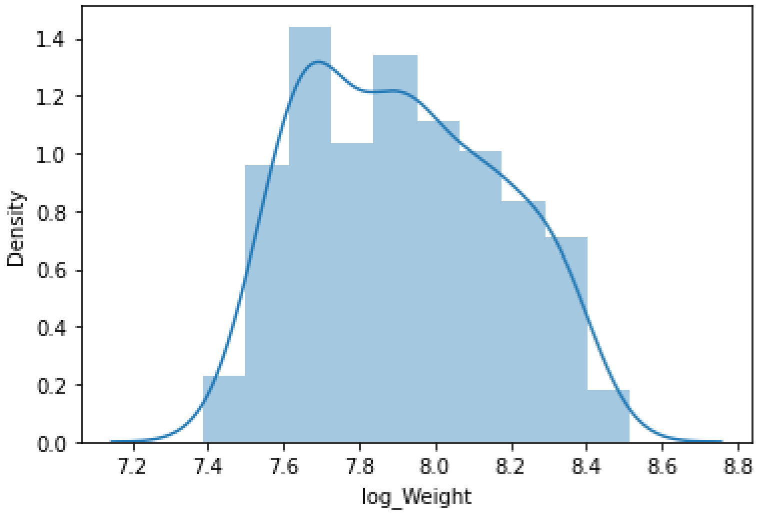
Out[71]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Origin	log_Displacement
0	8.0	8	307.0	130.0	3504	12.0	1	5.726848
2	18.0	8	318.0	150.0	3436	11.0	1	5.762051
3	16.0	8	304.0	150.0	3433	12.0	1	5.717028

In [72]:
```
sns.distplot(df['log_Weight'])
```

```
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[72]:
```
<AxesSubplot:xlabel='log_Weight', ylabel='Density'>
```



In [73]:
```
df=df.drop(['Displacement','Weight'],axis=1)
df.head()
```

Out[73]:

	MPG	Cylinders	Horsepower	Acceleration	Origin	log_Displacement	log_Weight
0	8.0	8	130.0	12.0	1	5.726848	8.161660
2	18.0	8	150.0	11.0	1	5.762051	8.142063
3	16.0	8	150.0	12.0	1	5.717028	8.141190
4	17.0	8	140.0	10.5	1	5.710427	8.145840
12	15.0	8	150.0	9.5	1	5.991465	8.232440

In [74]:
```
df.dtypes
```

Out[74]:
```
MPG                 float64
Cylinders            object
Horsepower          float64
Acceleration        float64
Origin               object
log_Displacement    float64
log_Weight          float64
dtype: object
```

In [75]:
```
dummy.head()
```

Out[75]:

	Cylinders_4	Cylinders_5	Cylinders_6	Cylinders_8	Origin_2	Origin_3
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

In [76]:
```
#combine numeric columns from df with dummy columns  to create final data
df_numeric=df.select_dtypes(include=['int64','float64'])
```

In [77]:
```
df_master=pd.concat([df_numeric,dummy],axis=1)
```

In [78]:
```
df_master.head()
```

Out[78]:

	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	

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

In [79]:
```
x=df_master.drop('MPG',axis=1)
```

In [80]:

```
y=df_master['MPG']
```

Random sampling :create training and test samples

In [81]:

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,train_size=0.7,random_state=0)
```

feature selection

In [82]:

```
#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]
```

In [83]:

```
selected_features
```

Out[83]:

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

Model 3: Build model using KBest selected features

In [84]:

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

In [85]:

```
#train your model
linreg.fit(xtrain_kbest,ytrain)
```

Out[85]:

```
LinearRegression()
```

In [86]:

```
linreg.score(xtrain_kbest,ytrain)
```

Out[86]:

```
0.7118189417837262
```

In [88]:
```
#store KBest columns from xtest to xtest_kbest
xtest_kbest=xtest[selected_features]
```

In [89]:
```
pred_y=linreg.predict(xtest_kbest)
```

In [90]:
```
linreg.score(xtest_kbest,ytest)
```

Out[90]:
```
0.7038559215939076
```

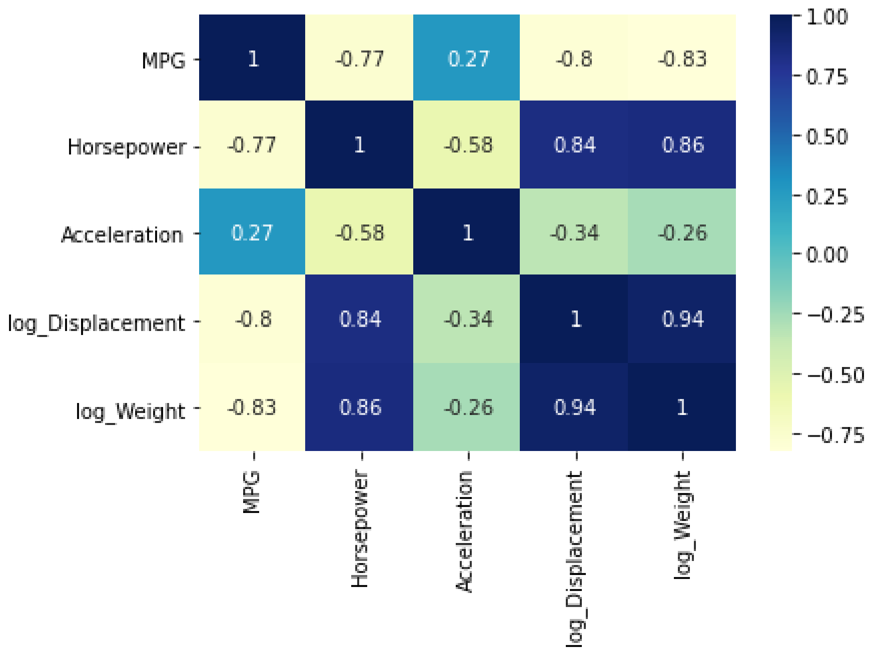
model 4

mutlicollinearity checking

In [91]:
```
new_df_corr=df_numeric.corr()
```

In [92]:
```
sns.heatmap(new_df_corr,cmap='YlGnBu',annot=True) #YlGnBu yellow green blue
```

Out[92]:
```
<AxesSubplot:>
```



In [93]:
```
#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 [94]:
```
#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]
```

In [95]:
```
selected_features
```

Out[95]:
```
Index(['Horsepower', 'Acceleration', 'Cylinders_4', 'Cylinders_8', 'Origin_
3'], dtype='object')
```

Build model using KBest selected features

In [96]:
```
##store KBest columns from xtrain to xtarin_kbest
xtrain_kbest=xtrain[selected_features]
```

In [97]:
```
#train your model
linreg.fit(xtrain_kbest,ytrain)
```

Out[97]:
```
LinearRegression()
```

In [98]:
```
linreg.score(xtrain_kbest,ytrain)
```

Out[98]:
```
0.7110316000206895
```

In [99]:
```
#store KBest columns from xtest to xtest_kbest
xtest_kbest=xtest[selected_features]
```

In [100]:
```
pred_y=linreg.predict(xtest_kbest)
```

In [101]:
```
linreg.score(xtest_kbest,ytest)
```

Out[101]:
```
0.703882872331421
```

In [ ]:
```

```