bia-project-1

May 27, 2024

1 Loading the data

[1]:
```
import pandas as pd
import seaborn as sns
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
df= pd.read_csv('/content/drive/MyDrive/data science/dataset/autos_mpg.csv')
```

EDA

[4]:
```
# Checking the data
df.head()
```

[4]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

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

[5]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	\
393	27.0	4	140.0	86	2790	15.6	
394	44.0	4	97.0	52	2130	24.6	
395	32.0	4	135.0	84	2295	11.6	
396	28.0	4	120.0	79	2625	18.6	

```
397  31.0          4         119.0         82    2720          19.4

     model_year  origin         car_name
393          82       1  ford mustang gl
394          82       2        vw pickup
395          82       1    dodge rampage
396          82       1      ford ranger
397          82       1       chevy s-10
```

[6]:
```
df.dtypes
```

[6]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model_year        int64
origin            int64
car_name         object
dtype: object
```

1. float 3
2. int 4
3. object 2

[7]:
```
df.shape
```

[7]:
```
(398, 9)
```

There are 398 rows and 9 columns

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    int64
 8   car_name      398 non-null    object
```

```
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

[9]:
```
df['horsepower'].unique()
```

[9]:
```
array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
       '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
       '200', '210', '193', '?', '100', '105', '175', '153', '180', '110',
       '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
       '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
       '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
       '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
       '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
       '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
       '64', '74', '116', '82'], dtype=object)
```

there is some null values in horsepower

[10]:
```
df[df['horsepower']=='?']
```

[10]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

There is 6 null rows which has a null values

[11]:
```
df = df.drop(df[df['horsepower'] == '?'].index)
```

Droping the all null values

[12]:
```
# Checking the data again
df.isna().sum()
```

[12]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
```

```
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

[13]:
```
df.dtypes
```

[13]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model_year        int64
origin            int64
car_name         object
dtype: object
```

1. Horsepower still have a string values
2. so we convert it into integers value so we import LabelEncoder
3. we importing it before the vasualization becoause we want a clear view of Horsepower

[14]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
```

[15]:
```
df['horsepower'] = le.fit_transform(df['horsepower'])
```

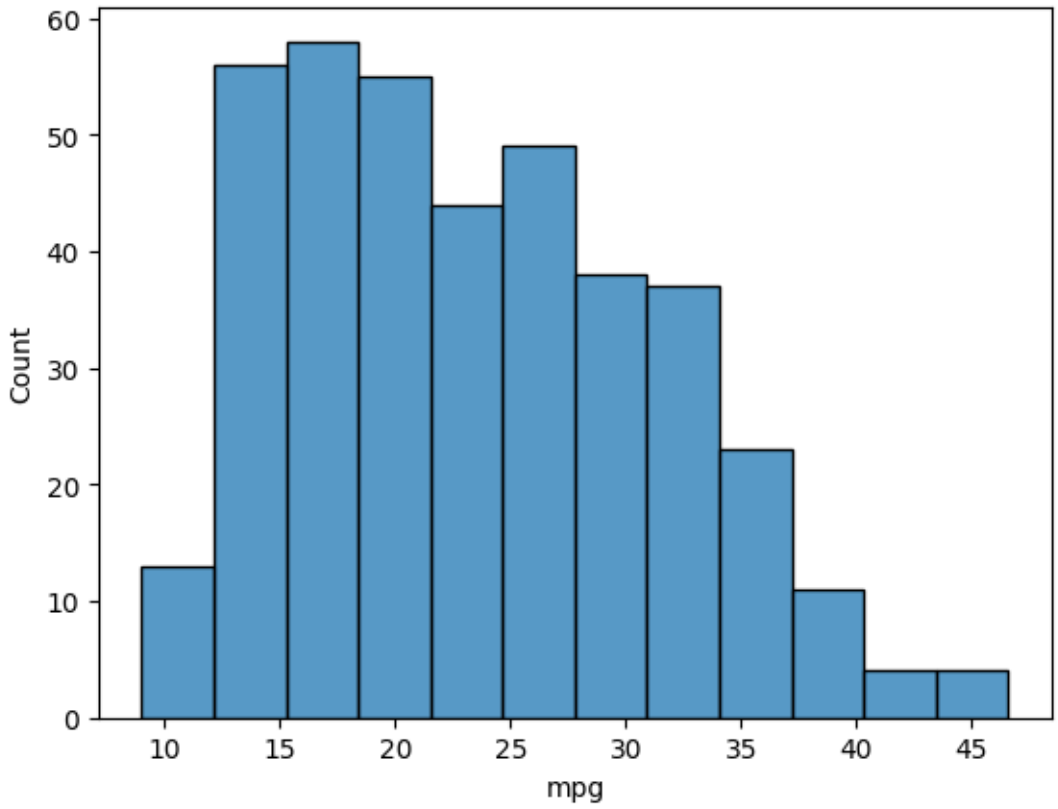
[16]:
```
df.dtypes
```

[16]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower        int64
weight            int64
acceleration    float64
model_year        int64
origin            int64
car_name         object
dtype: object
```

2 Visualizing the data

[17]:
```
sns.histplot(df['mpg'])
```

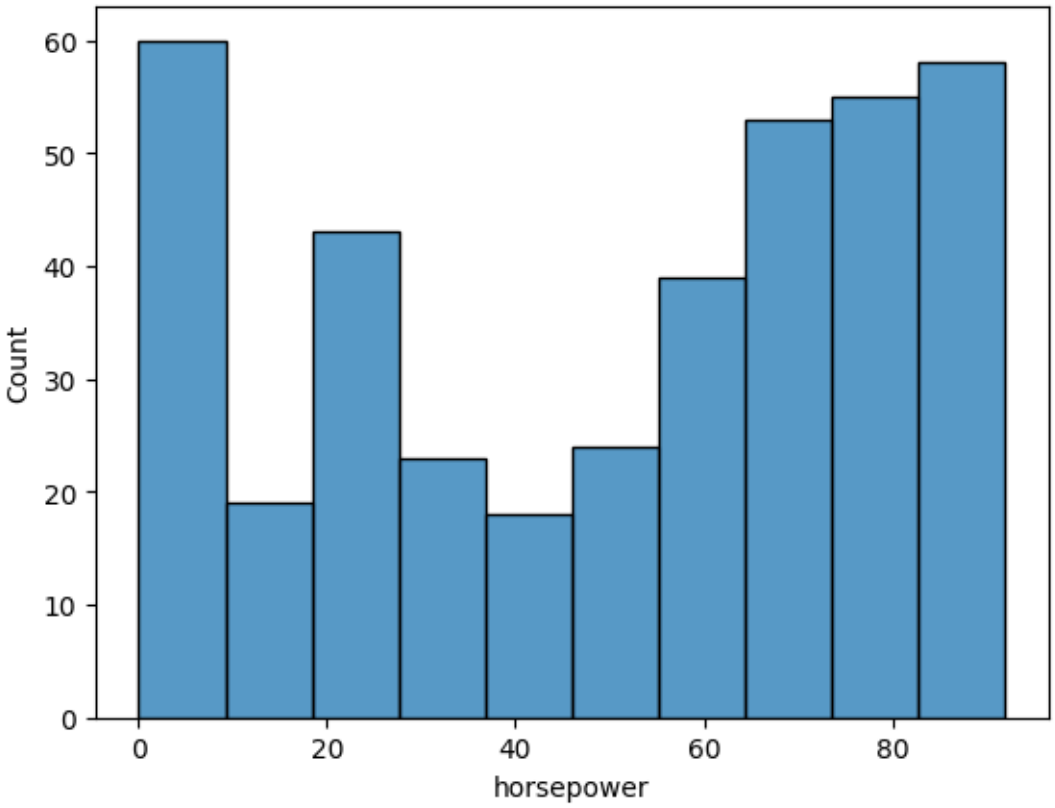
[17]:
```
<Axes: xlabel='mpg', ylabel='Count'>
```



Most of the mpg is between 15 to 40

[18]:
```
sns.histplot(df['horsepower'])
```

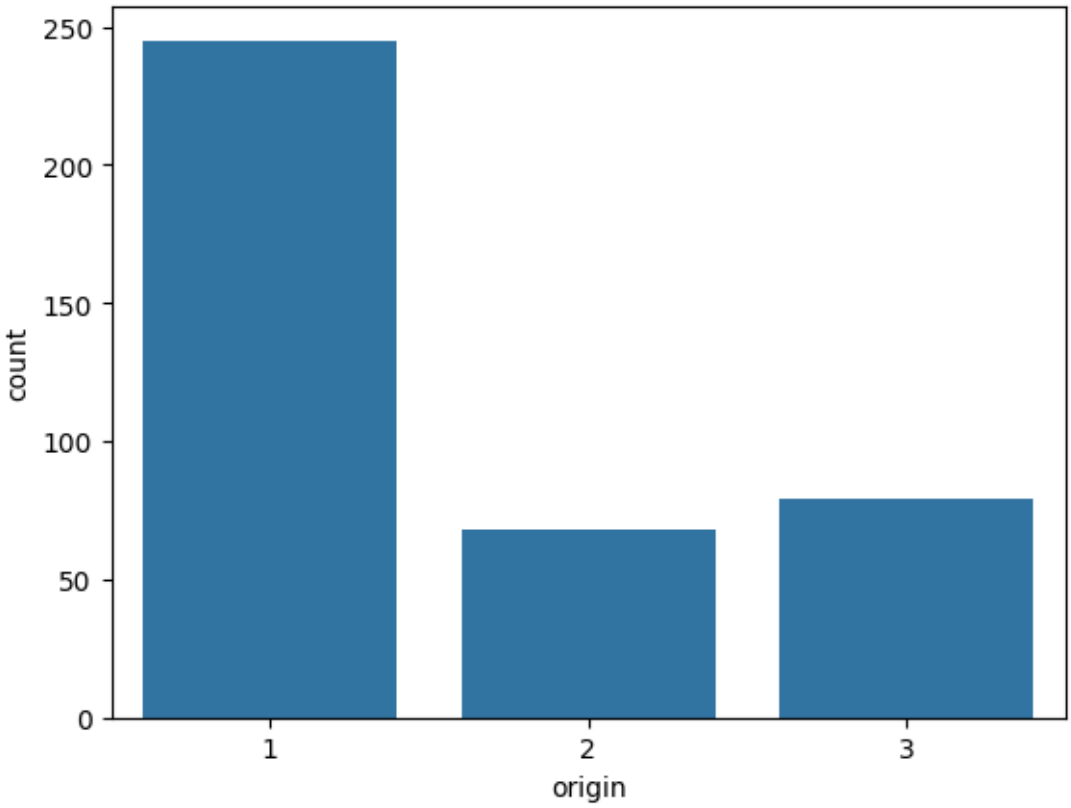
[18]:
```
<Axes: xlabel='horsepower', ylabel='Count'>
```



most of the horsepower count is above 50

[19]:
```
sns.countplot(x=df['origin'])
```

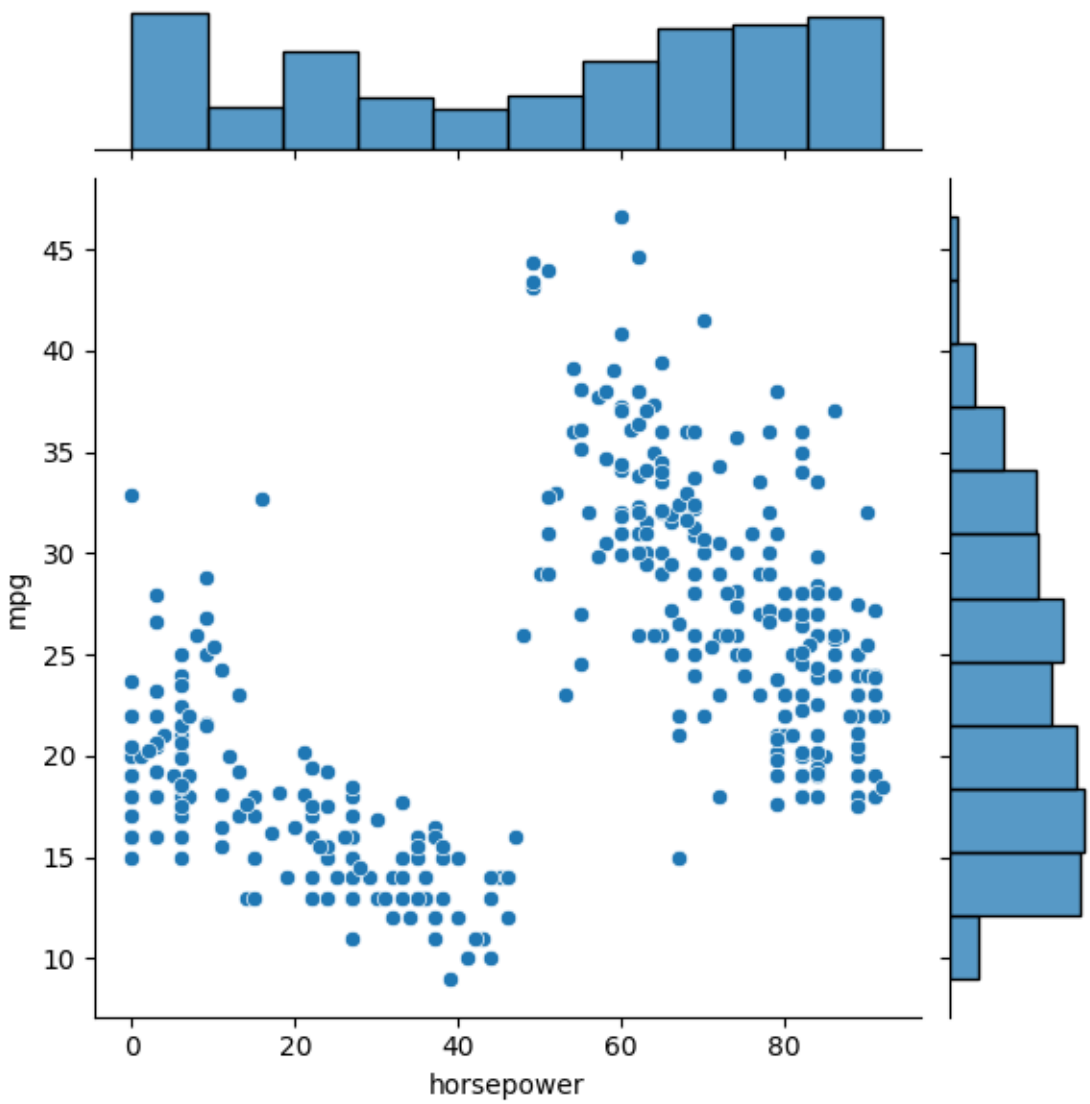
[19]:
```
<Axes: xlabel='origin', ylabel='count'>
```



1 has a maximum count of origin

[20]:
```
sns.jointplot(x='horsepower',y='mpg',data=df)
```

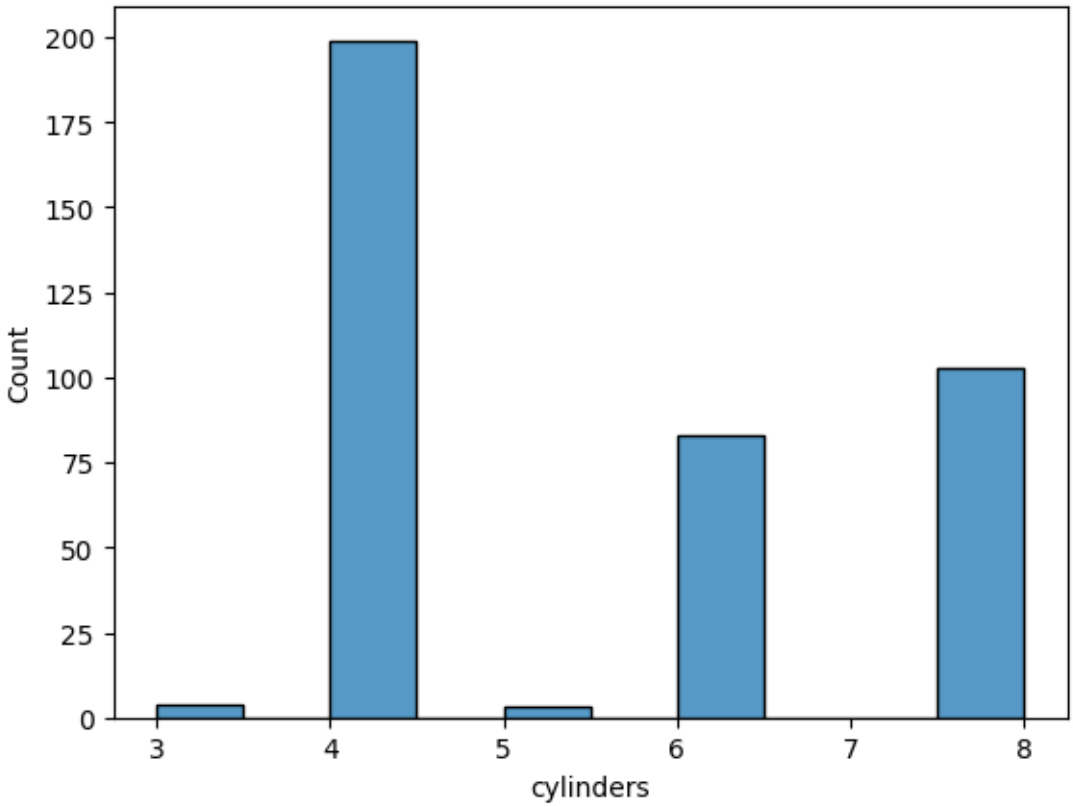
[20]:
```
<seaborn.axisgrid.JointGrid at 0x7d2895ff2aa0>
```



1. There some cars who has a more horsepower and more than 40 mpg as well

[21]:
```
sns.histplot(df['cylinders'])
```

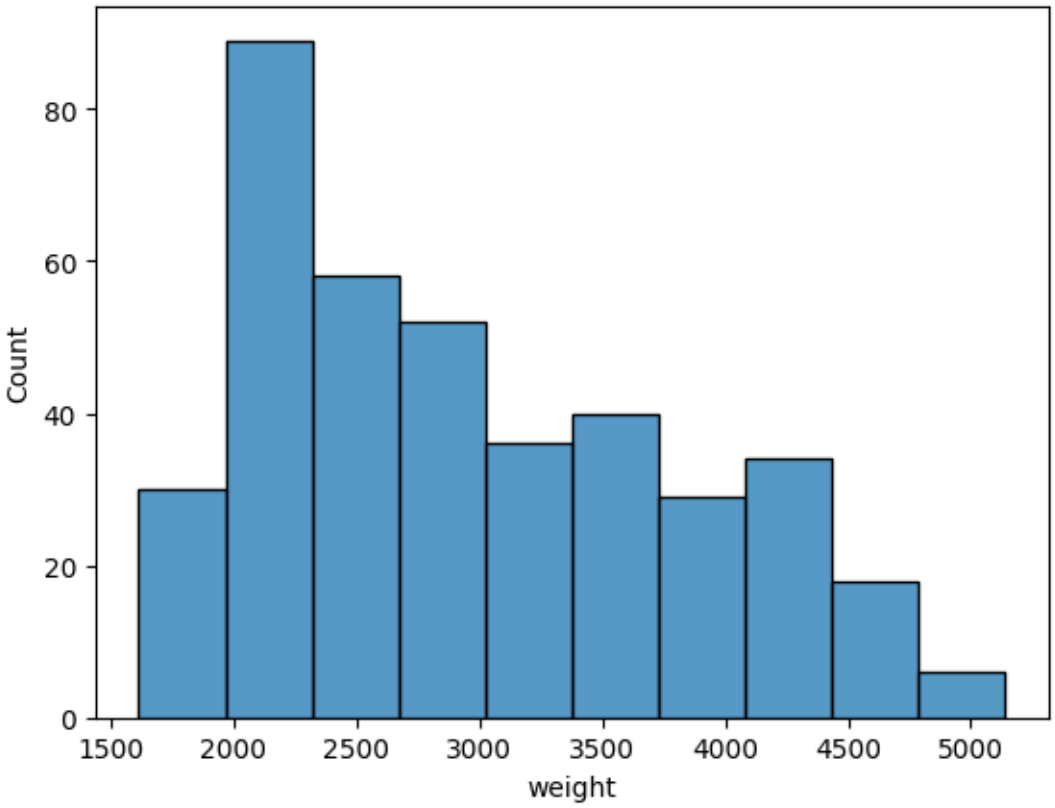
[21]:
```
<Axes: xlabel='cylinders', ylabel='Count'>
```



most of the cars has a four cylinders

[22]:
```
sns.histplot(df['weight'])
```

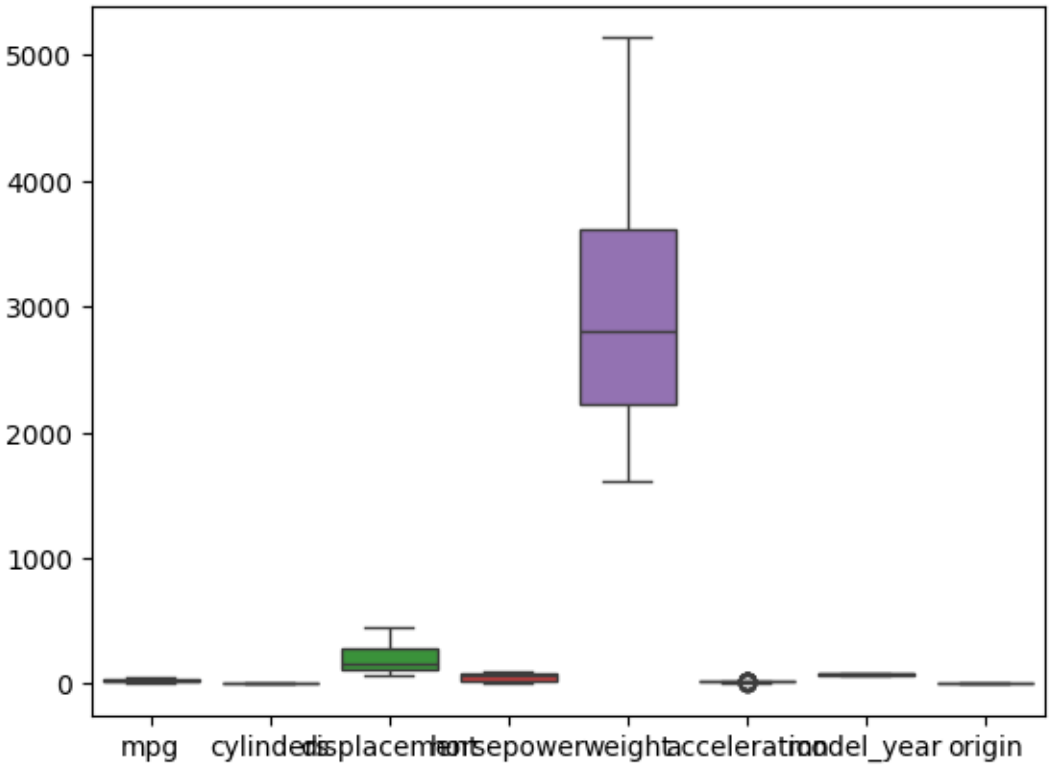
[22]:
```
<Axes: xlabel='weight', ylabel='Count'>
```



most of the cars have a weight between 2000 - 4500

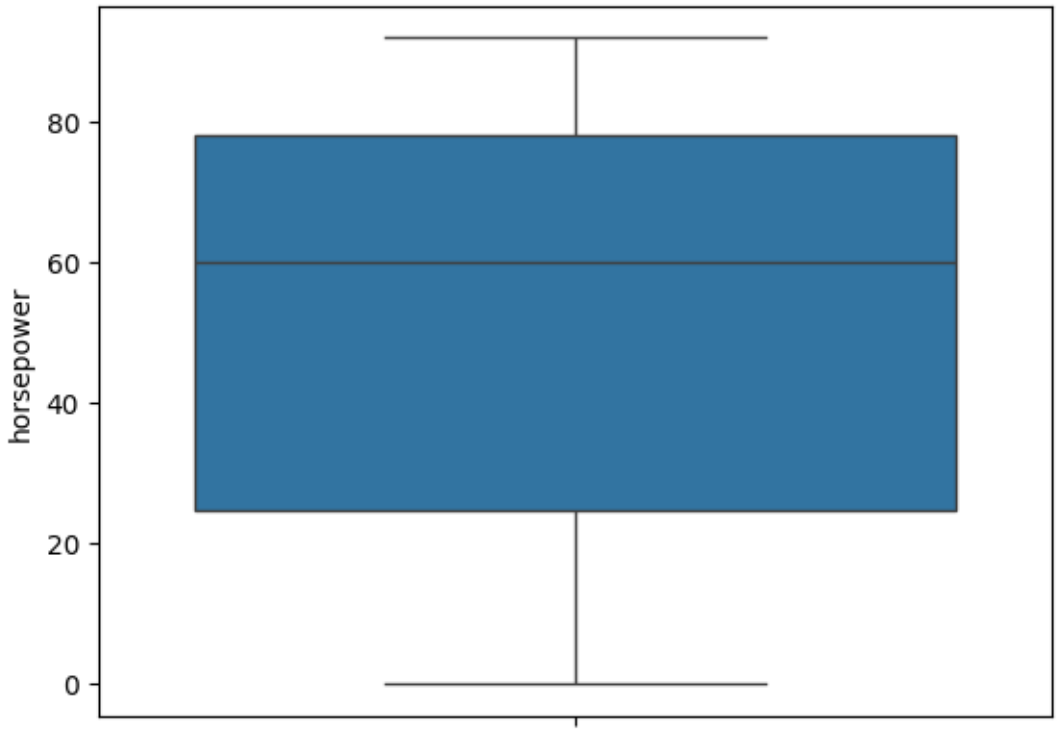
[23]:
```
sns.boxplot(df)
```

[23]:
```
<Axes: >
```



[24]:
```
sns.boxplot(df['horsepower'])
```

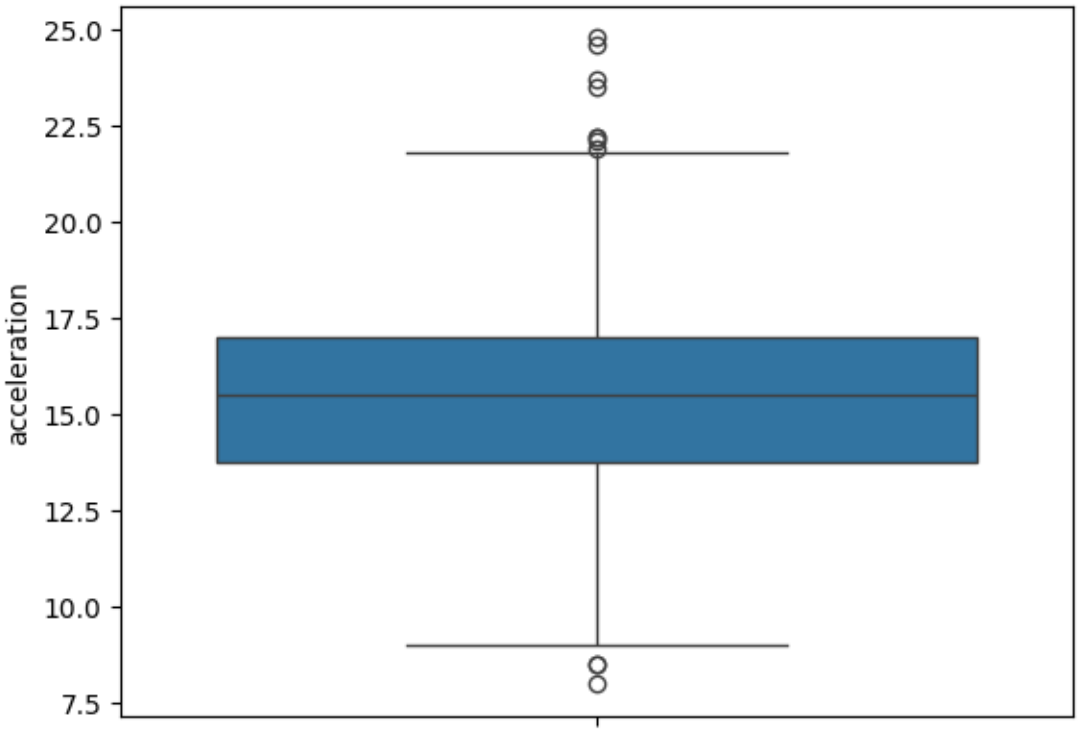
[24]: <Axes: ylabel='horsepower'>



There is no outliers in horsepower

[25]:
```
sns.boxplot(df['acceleration'])
```

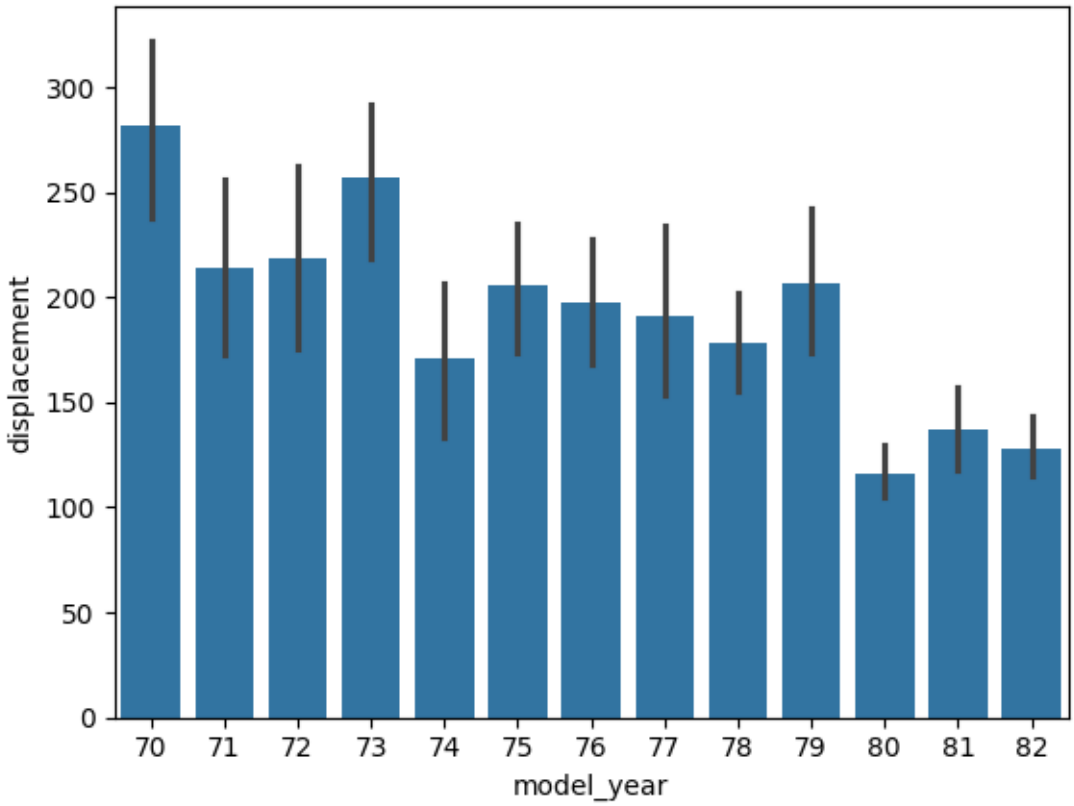
[25]:
```
<Axes: ylabel='acceleration'>
```



found some outliers in acceleration but we hold it as it is

[26]:
```
sns.barplot(x='model_year',y='displacement',data=df)
```

[26]:
```
<Axes: xlabel='model_year', ylabel='displacement'>
```



1. Model year number of 70 has a more displacement than others
2. And the second one is model year number of 73
3. no one is below 100

3 Preprocessing the data

[27]:
```
df.dtypes
```

[27]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower        int64
weight            int64
acceleration    float64
model_year        int64
origin            int64
car_name         object
dtype: object
```

1. We already convert the Porsepower column but there is still string values present in the table which is a car_name

2. so before making a predictive model we have to drop string values
3. so we have to drop car_name

[28]:
```
df=df.drop(['car_name'],axis=1)
```

[29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 392 entries, 0 to 397
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    int64  
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   model_year    392 non-null    int64  
 7   origin        392 non-null    int64  
dtypes: float64(3), int64(5)
memory usage: 27.6 KB
```

1. There is one columns called Weight it can be effect on prediction
2. To balance the data we want a consistant data
3. so we use Standard Scalar of that

Importing Standard Scaler

[30]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
```

[31]:
```
df['weight'] = sc.fit_transform(df[['weight']])
```

[32]:
```
df.head()
```

[32]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	15	0.620540	12.0	70	1
1	15.0	8	350.0	33	0.843334	11.5	70	1
2	18.0	8	318.0	27	0.540382	11.0	70	1
3	16.0	8	304.0	27	0.536845	12.0	70	1
4	17.0	8	302.0	22	0.555706	10.5	70	1

```
4           70        1
```

1. The last step of preprocessing is distribute the data into two parts
2. we making predictive model for miles per gallon (mpg)
3. so mpg is our y

[33]:
```
x = df.drop(['mpg'],axis=1)

y = df['mpg']
```

1. we have to divide the the data for prediction
2. so we importing Train_Test_split

Importing Train_test_split

[34]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪3,random_state=123)
```

4 Importing Algorithm

1. The data is linear so we go with LinearRegression first
2. we go with SVR (Support Vector Regression) as well for better accuracy

5 Importing LinearRegression

[35]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

from sklearn.metrics import mean_squared_error, r2_score
```

[36]:
```
lr.fit(x_train,y_train)
```

[36]:
```
LinearRegression()
```

[37]:
```
y_pred = lr.predict(x_test)
```

[38]:
```
rmse = mean_squared_error(y_test,y_pred,squared=False)
```

[39]:
```
rmse
```

[39]:
```
3.358283507518921
```

[40]:
```
r2_score(y_test,y_pred)
```

[40]:
```
0.8023505374803237
```

6 Importing SVR

[41]:
```
from sklearn.svm import SVR

svr=SVR(kernel='linear')
```

[42]:
```
svr.fit(x_train, y_train)
```

[42]:
```
SVR(kernel='linear')
```

[43]:
```
y_pred = svr.predict(x_test)
```

[44]:
```
r2_score(y_test,y_pred)
```

[44]:
```
0.7956554199869587
```

1. LinearRegression gives 80% accuracy for the data
2. SVR gives 79% accuracy for thr data
3. from this data we got a best accuracy in LinearRegression and SVR as well