PCA and t-SNE Project: Auto MPG

Marks: 30

Context

The shifting market conditions, globalization, cost pressure, and volatility are leading to a change in the automobile market landscape. The emergence of data, in conjunction with machine learning in automobile companies, has paved a way that is helping bring operational and business transformations.

The automobile market is vast and diverse, with numerous vehicle categories being manufactured and sold with varying configurations of attributes such as displacement, horsepower, and acceleration. We aim to find combinations of these features that can clearly distinguish certain groups of automobiles from others through this analysis, as this will inform other downstream processes for any organization aiming to sell each group of vehicles to a slightly different target audience.

You are a Data Scientist at SecondLife which is a leading used car dealership with numerous outlets across the US. Recently, they have started shifting their focus to vintage cars and have been diligently collecting data about all the vintage cars they have sold over the years. The Director of Operations at SecondLife wants to leverage the data to extract insights about the cars and find different groups of vintage cars to target the audience more efficiently.

Objective

The objective of this problem is to **explore the data, reduce the number of features by using dimensionality reduction techniques like PCA and t-SNE, and extract meaningful insights**.

Dataset

There are 8 variables in the data:

- mpg: miles per gallon
- cyl: number of cylinders
- disp: engine displacement (cu. inches) or engine size
- hp: horsepower
- wt: vehicle weight (lbs.)
- acc: time taken to accelerate from 0 to 60 mph (sec.)
- yr: model year
- car name: car model name

Importing the necessary libraries and overview of the dataset

In [11]:
```
# Import all necessary libraries useful for this project:

import pandas as pd

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA # Relevant*

from sklearn.manifold import TSNE # Relevant*

import warnings
warnings.filterwarnings("ignore") # still, important
```

Loading the data

In [12]:
```
cars = pd.read_csv('auto-mpg.csv')
```

In [13]:
```
cars.head(2)
```

Out[13]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name
0	18.0	8	307.0	130	3504	12.0	70	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	buick skylark 320

In [14]:
```
cars.sample(10, random_state = 2)
```

Out[14]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name
94	13.0	8	440.0	215	4735	11.0	73	chrysler new yorker brougham
32	25.0	4	98.0	?	2046	19.0	71	ford pinto
279	29.5	4	98.0	68	2135	16.6	78	honda accord lx
178	23.0	4	120.0	88	2957	17.0	75	peugeot 504
354	34.5	4	100.0	?	2320	15.8	81	renault 18i
25	10.0	8	360.0	215	4615	14.0	70	ford f250
67	11.0	8	429.0	208	4633	11.0	72	mercury marquis
188	16.0	8	318.0	150	4190	13.0	76	dodge coronet brougham
157	15.0	8	350.0	145	4440	14.0	75	chevrolet bel air
225	17.5	6	250.0	110	3520	16.4	77	chevrolet concours

In [15]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 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   car name      398 non-null    object 
dtypes: float64(3), int64(3), object(2)
memory usage: 25.0+ KB
```

In [16]:
```
horsepower2 = pd.DataFrame(cars.horsepower.str.isdigit())
```

```
cars[horsepower2['horsepower'] == False]
```

Out[16]:

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

In [17]:

```
cars = cars.replace('?', np.nan)
```

In [18]:

```
# Imputing the missing values with the median value of the column horsepower
cars.horsepower.fillna(cars.horsepower.median(), inplace = True)

# Converting the horsepower column from object data type to float
cars['horsepower'] = cars['horsepower'].astype('float64')
```

In [19]:

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 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    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64
 7   car name      398 non-null    object
dtypes: float64(4), int64(3), object(1)
memory usage: 25.0+ KB
```

In [20]:

```
cars.duplicated().sum()
```

Out[20]: 0

In [21]:

```
cars["car name"].nunique()
```

Out[21]: 305

In [22]:

```
cars = cars.drop(["car name"], axis = 1)
```

In [23]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 7 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    float64
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
dtypes: float64(4), int64(3)
memory usage: 21.9 KB
```

Observations:

1. We treated the question marks replacing them with np.nan
2. the horsepower column type was changed from obj to float.
3. We dropped the car name column from the dataset. It would not make a difference in the analysis.
4. There are no duplicates in the dataset.

Data Preprocessing and Exploratory Data Analysis

Summary Statistics

In [24]:
```
cars.describe(include = 'all').T
```

Out[24]:

	count	mean	std	min	25%	50%	75%	max
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6
cylinders	398.0	5.454774	1.701004	3.0	4.000	4.0	8.000	8.0
displacement	398.0	193.425879	104.269838	68.0	104.250	148.5	262.000	455.0
horsepower	398.0	104.304020	38.222625	46.0	76.000	93.5	125.000	230.0
weight	398.0	2970.424623	846.841774	1613.0	2223.750	2803.5	3608.000	5140.0
acceleration	398.0	15.568090	2.757689	8.0	13.825	15.5	17.175	24.8
model year	398.0	76.010050	3.697627	70.0	73.000	76.0	79.000	82.0

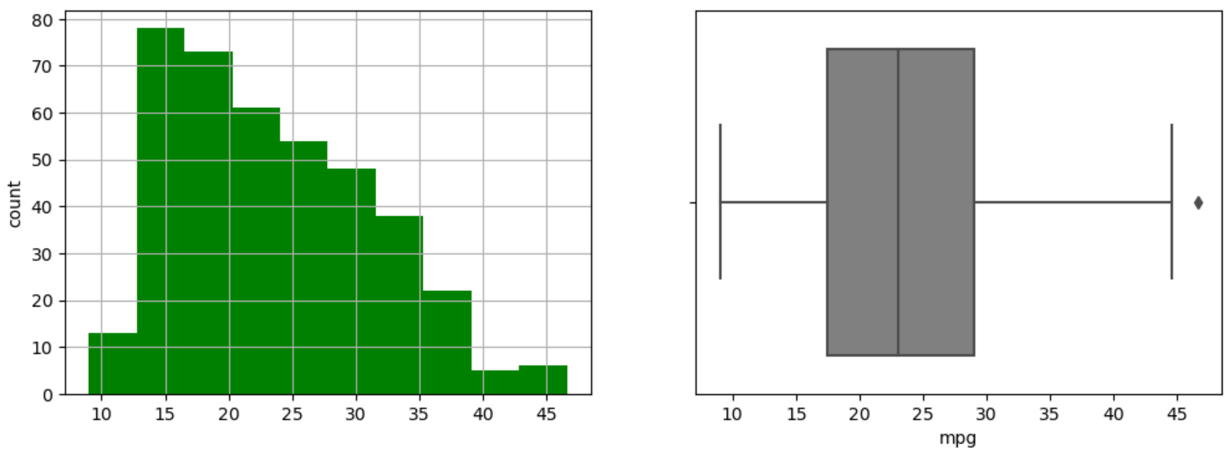
Observations:

1. Model year: we have cars between the year 1970 & 1982 with an average of year 1976
2. Acceleration: minimun 8 - 24 seconds to accelerate to 60mph an average of 15 seconds
3. Weight: average weight of 2970 lbs.

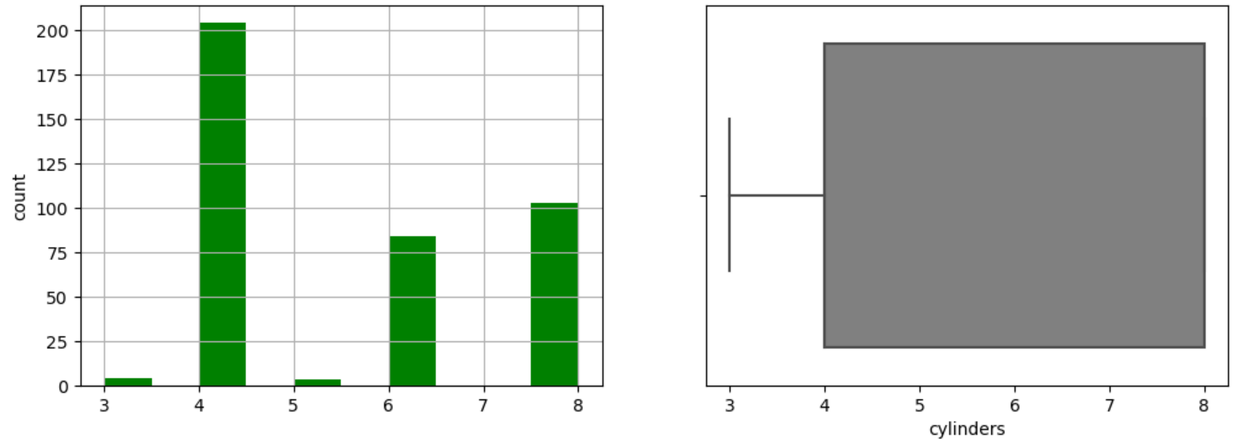
4. Horsepower: average 104.3 with a min of 76horsepower in one foot
5. Displacement: Lower displacement is 68 and an average of 193
6. Cylinders: Minimun 3 cylinders and maximun 8 with an average of 5
7. MPG:9 - 46 miles average of 23 An interesting finding is, desplacement, cylinders and mph.

In [25]:
```
num_cols = list(cars.columns)

for col in num_cols:

    print(col)

    print('Skew :',round(cars[col].skew(),2))

    plt.figure(figsize = (12, 4))

    plt.subplot(1, 2, 1)

    cars[col].hist(bins = 10, grid = True, color = 'green')# nice color histog

    plt.ylabel('count')

    plt.subplot(1, 2, 2)

    sns.boxplot(x = cars[col], color = 'gray') # another nice color boxplot...

    plt.show()
```

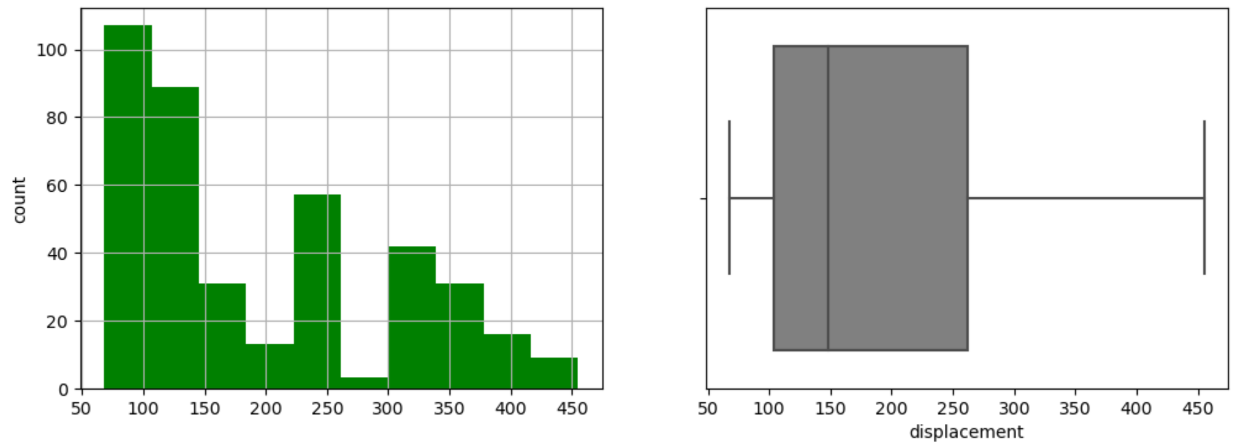
```
mpg
Skew : 0.46
```



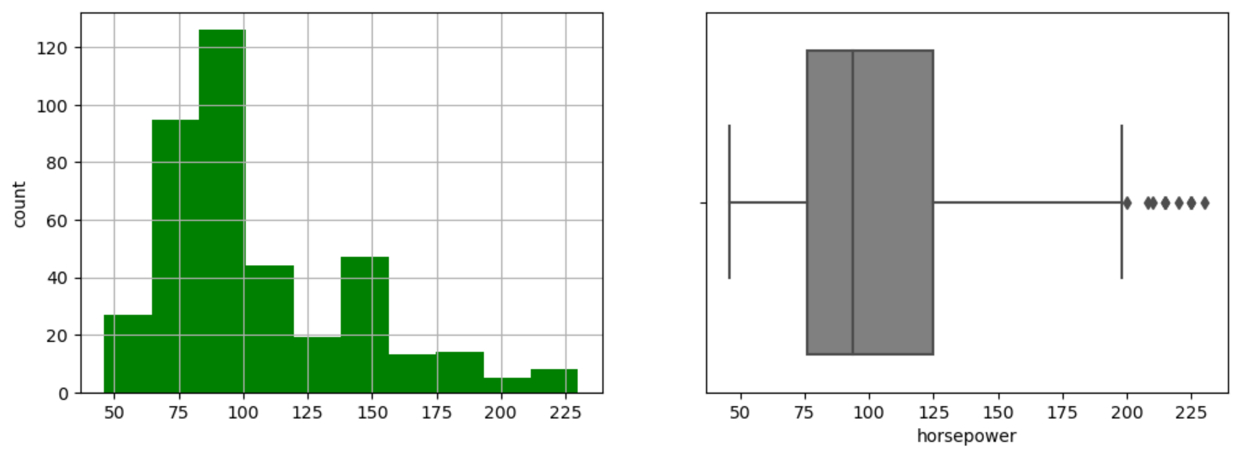

```
cylinders
Skew : 0.53
```



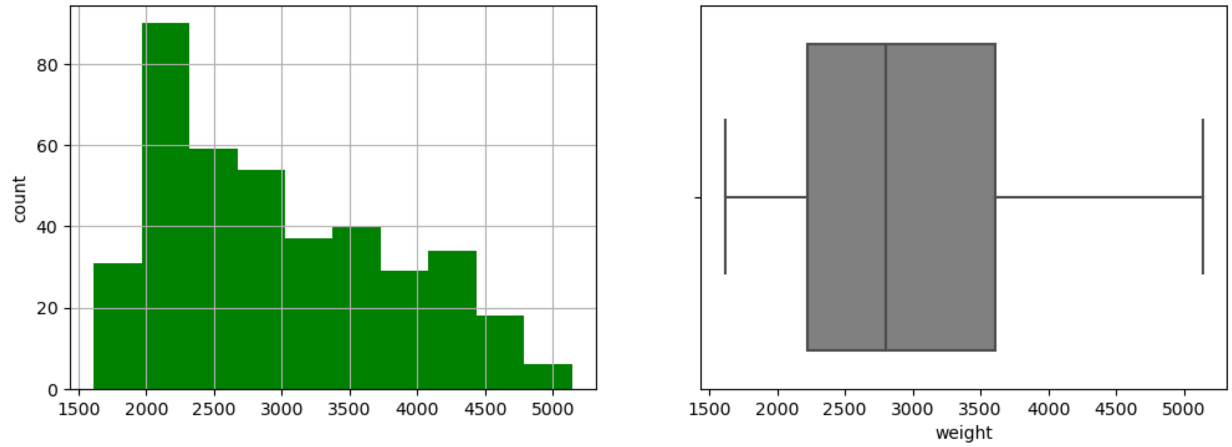

```
displacement
Skew : 0.72
```



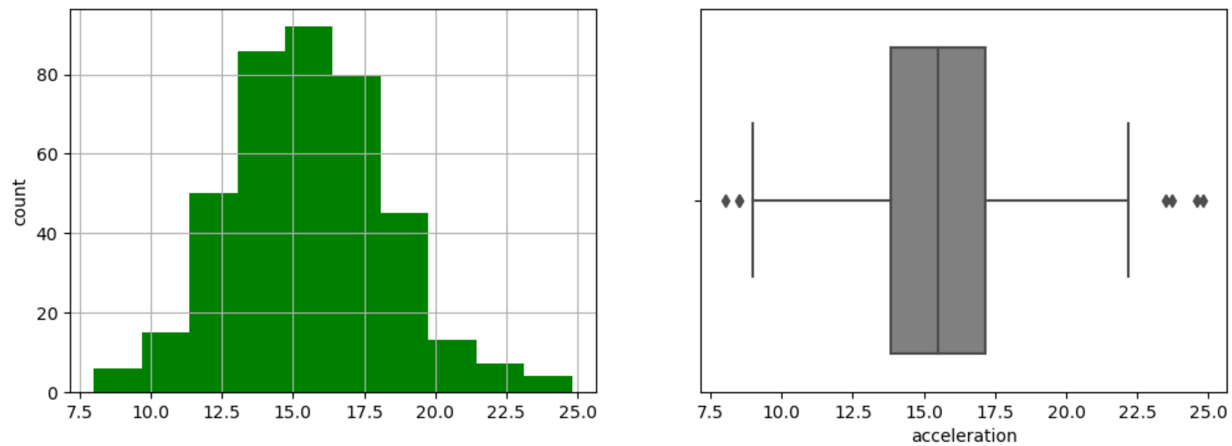

```
horsepower
Skew : 1.11
```



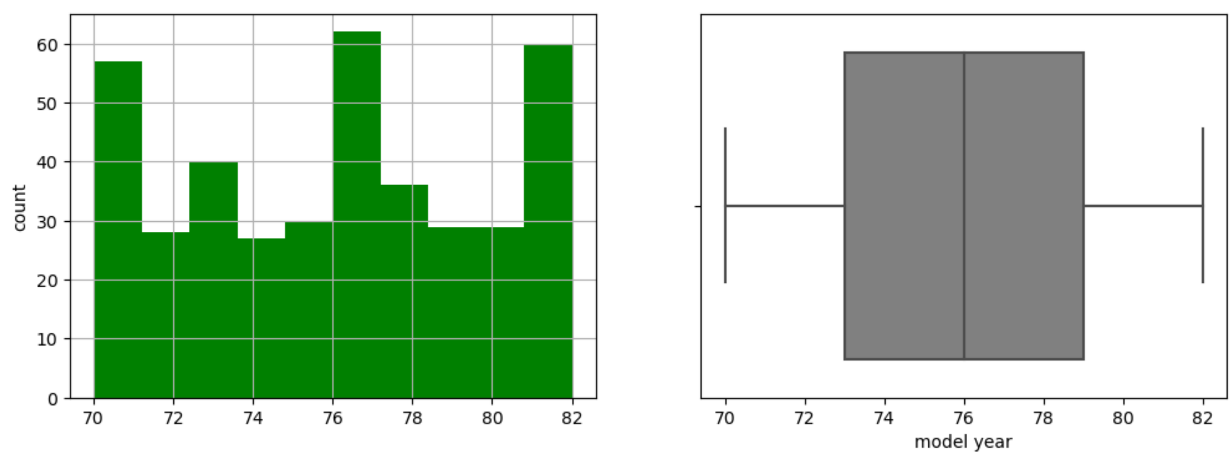

```
weight
Skew : 0.53
```



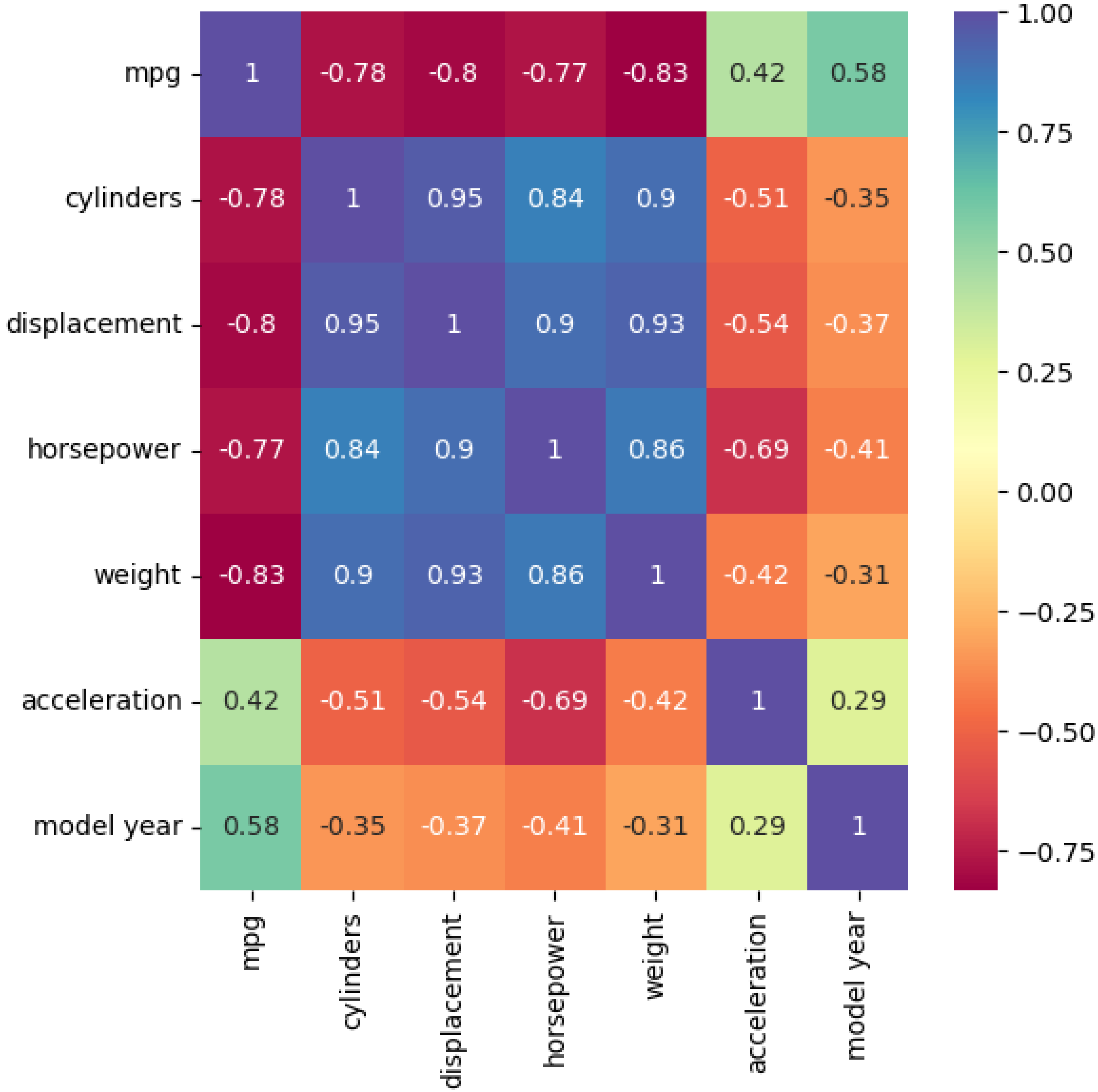
acceleration
Skew : 0.28



model year
Skew : 0.01



In [26]:
```
plt.figure(figsize = (6, 6))
sns.heatmap(cars.corr(), annot = True, cmap = "Spectral")# scectral...yes pleas
plt.xticks(rotation = 90)# easier to read
plt.show()
```



OBSERVATIONS:

High positive correlation:

- 1. cylinders and the displacement- (.95)
- 1. cylinders and the weight - (.9)
- 1. horsepower and the displacement (-.9)
- 1. weight and the displacement - (.9.)

High Negative:

- 1. model year and the rest of the variables correlation ranging from (0.29 -0.58)
- 1. acceleration in relation to the other variables is also low.

Scaling the data

In [72]:
```
# Scaling cars
scaler = StandardScaler()
```

```
cars_scaled = pd.DataFrame(scaler.fit_transform(cars), columns = cars.columns)
```

In [73]:
```
cars_scaled.head()
```

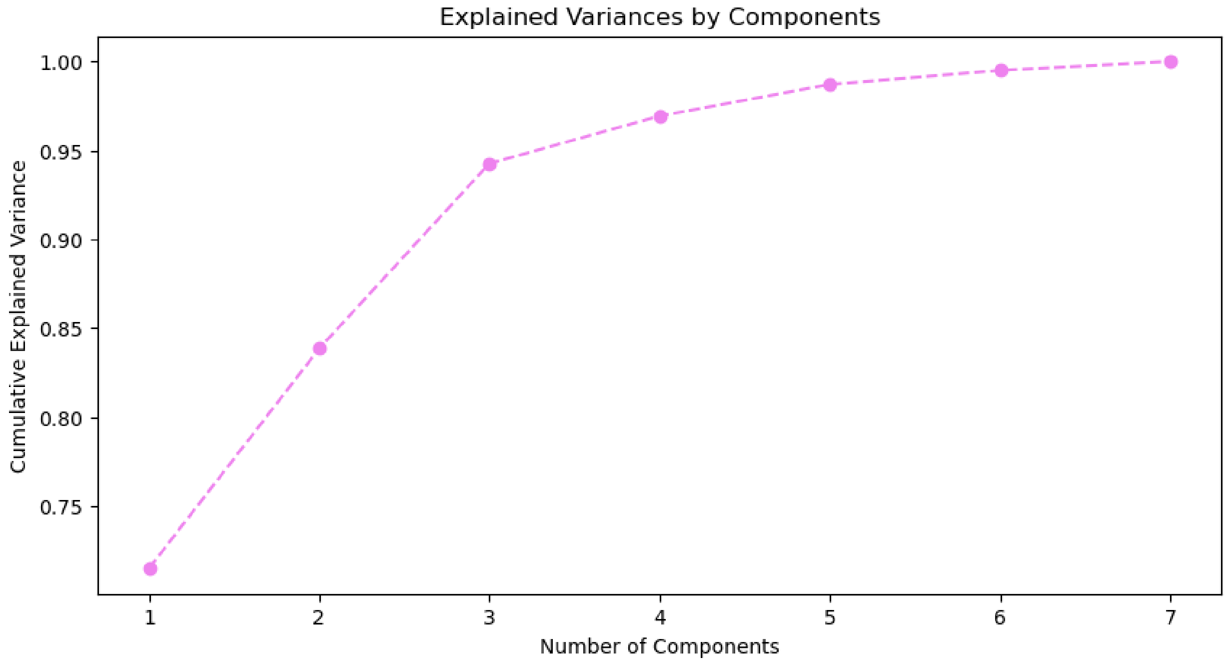
Out[73]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year
0	-0.706439	1.498191	1.090604	0.673118	0.630870	-1.295498	-1.627426
1	-1.090751	1.498191	1.503514	1.589958	0.854333	-1.477038	-1.627426
2	-0.706439	1.498191	1.196232	1.197027	0.550470	-1.658577	-1.627426
3	-0.962647	1.498191	1.061796	1.197027	0.546923	-1.295498	-1.627426
4	-0.834543	1.498191	1.042591	0.935072	0.565841	-1.840117	-1.627426

Principal Component Analysis

In [74]:
```
# Number of principal components
n = cars_scaled.shape[1]

# Finding principal components for the data
pca1 = PCA(n_components = n, random_state = 1)
cars_pca = pd.DataFrame(pca1.fit_transform(cars_scaled))

# The percentage of variance explained by each principal component
exp_var = pca1.explained_variance_ratio_
```

In [47]:
```
plt.figure(figsize = (10, 5))

plt.plot(range(1, 8), pca1.explained_variance_ratio_.cumsum(), marker = 'o', li

plt.title("Explained Variances by Components")

plt.xlabel("Number of Components")

plt.ylabel("Cumulative Explained Variance")

plt.show()
```



In [62]:
```
# Finding the least number of components that can explain more than 90% varian
sum = 0

for ab, a in enumerate(exp_var):

    sum = sum + a

    if(sum>0.90):

        print("number of PCs that explain at least 90% variance:", ab + 1)
        break
```

```
number of PCs that explain at least 90% variance: 3
```

Observations:

1.We observe from the plot that the slope of the line goes up in a sharp slope at .95 in the cumulitative explained variance and 3 on the number of components continuing on an upward ortogonally.

Interpret the coefficients of the first three principal components from the below DataFrame

In [63]:
```
pc_comps = ['PC1', 'PC2', 'PC3'] # 3 is the number of componets

data_pca = pd.DataFrame(np.round(pca1.components_[:3,:],2), index = pc_comps,

data_pca.T
```

Out[63]:

	PC1	PC2	PC3
mpg	-0.40	-0.21	-0.26
cylinders	0.42	-0.19	0.14
displacement	0.43	-0.18	0.10
horsepower	0.42	-0.09	-0.17
weight	0.41	-0.22	0.28
acceleration	-0.28	0.02	0.89
model year	-0.23	-0.91	-0.02

Visualize the data in 2 dimensions using the first two principal components

In [64]:
```
#2D for easier vizualization. made it colorful.

def color_high(val):

    if val < -0.25:
        return 'background: gray'

    elif val > 0.25:
        return 'background: orange'

data_pca.T.style.applymap(color_high)
```

Out[64]:

	PC1	PC2	PC3
mpg	-0.400000	-0.210000	-0.260000
cylinders	0.420000	-0.190000	0.140000
displacement	0.430000	-0.180000	0.100000
horsepower	0.420000	-0.090000	-0.170000
weight	0.410000	-0.220000	0.280000
acceleration	-0.280000	0.020000	0.890000
model year	-0.230000	-0.910000	-0.020000

Observations:

1.The first principal component, PC1, seems exibits the highest variability of the features the highest being the amount of cylinders contributes to the displacement of the engine.

2.The second principal component, PC2, seems to be related a negative relationship among the majority of the features but one, the acceleration which it is negative correlated to the rest.

3.The third principal component, PC3, seems to be a negative correlation with PC1(-.28) and a strong positive contribution with PC3 (.89).

1. The positive co-efficient values establishe the underlaying relationship of cylinders, displacement, horsepower and weight **in one word: Speed**

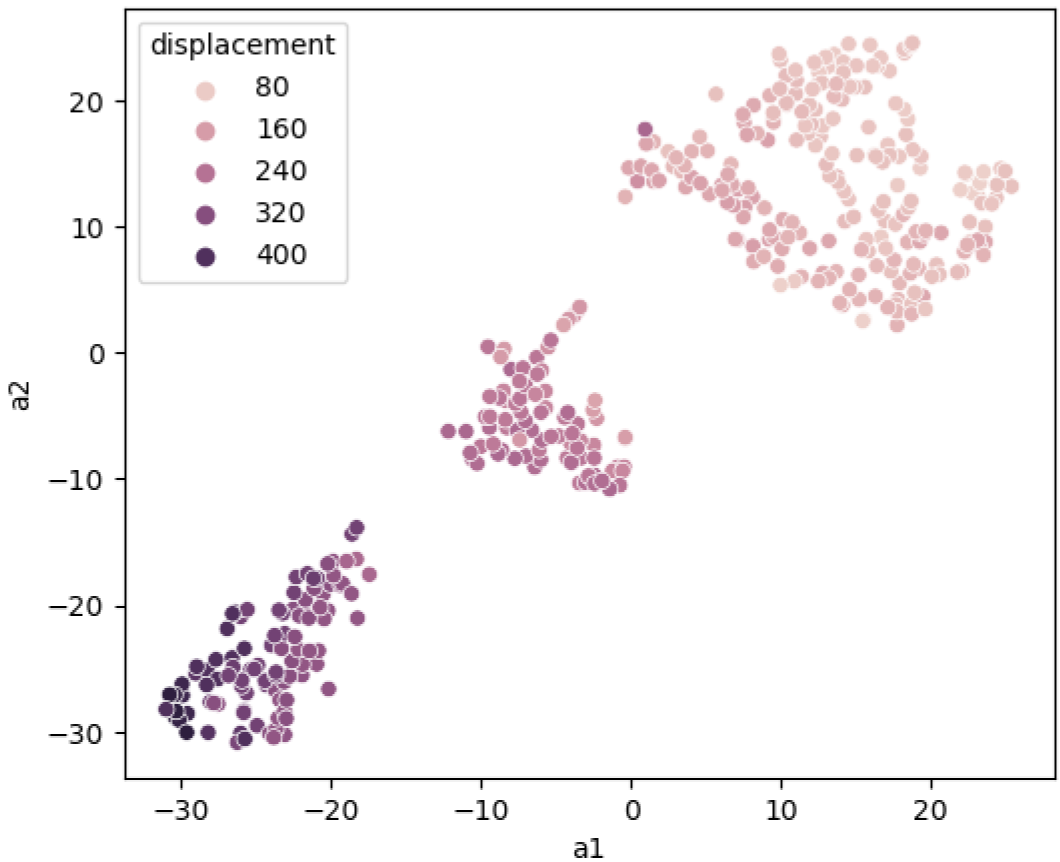
2. The negative co-efficient values indicate that miles per galon and acceleration are similarly negatively co-efficent.

t-SNE

In [79]:
```
# Fitting t-SNE with number of components equal to 2
tsne = TSNE(n_components = 2, random_state = 1)

cars_tsne = tsne.fit_transform(cars_scaled)
```

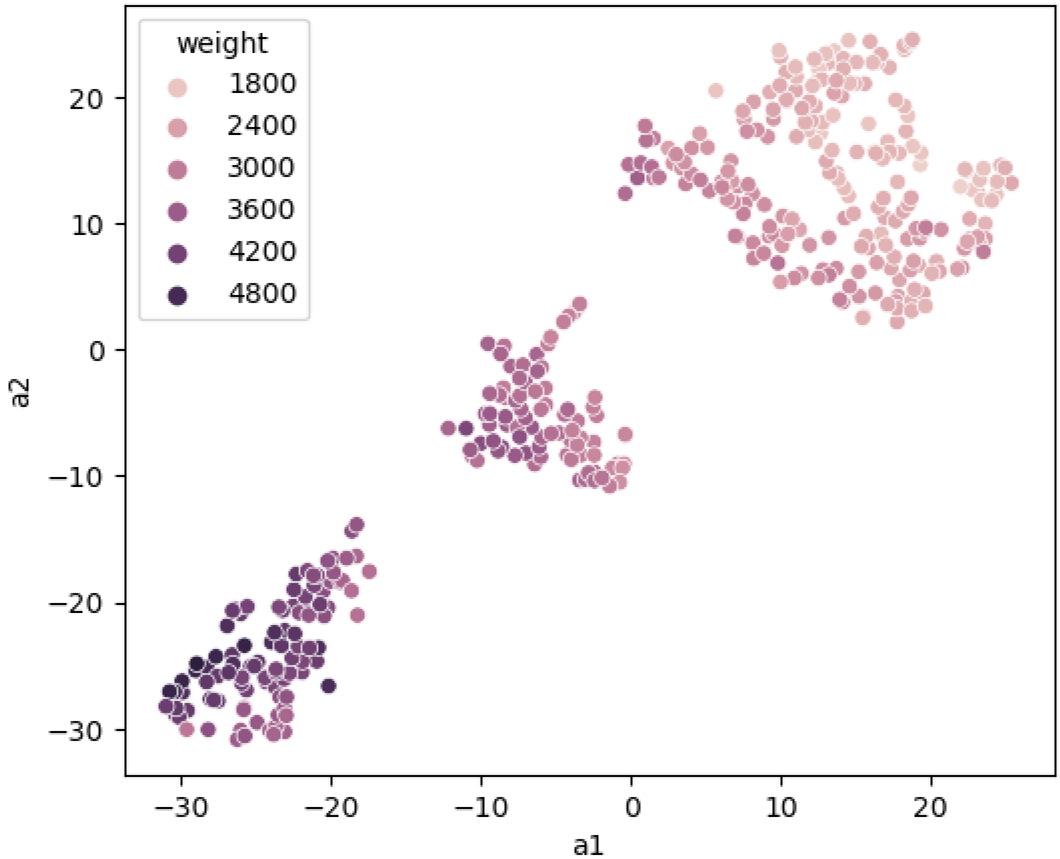
In [80]:
```
cars_tsne = pd.DataFrame(cars_tsne, columns = ['a1', 'a2']) # chose this a ins
```

In [81]:
```
# Scatter plot for two components
plt.figure(figsize = (6,5))

sns.scatterplot(x = 'a1', y = 'a2', data = cars_tsne, hue=cars.displacement)

plt.show()
```



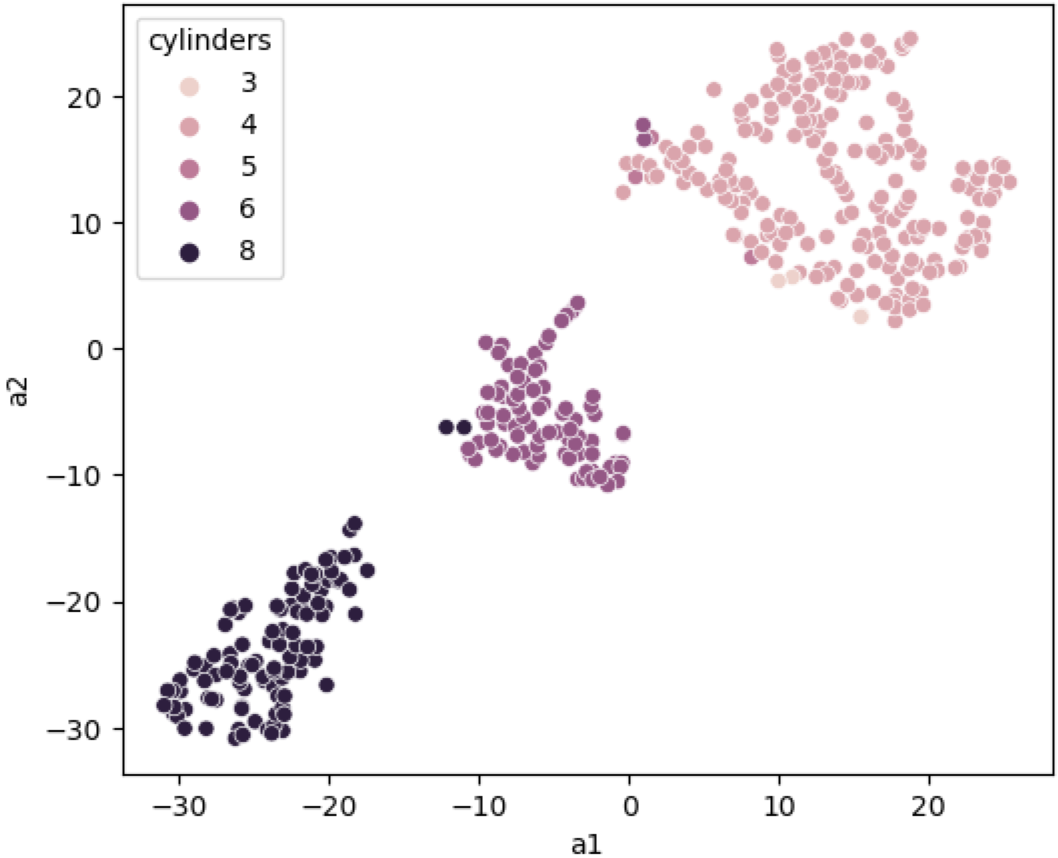
In [296…
```
# Scatter plot for two components
plt.figure(figsize = (6,5))

sns.scatterplot(x = 'a1', y = 'a2', data = cars_tsne, hue=cars.weight)
```

```
plt.show()
```



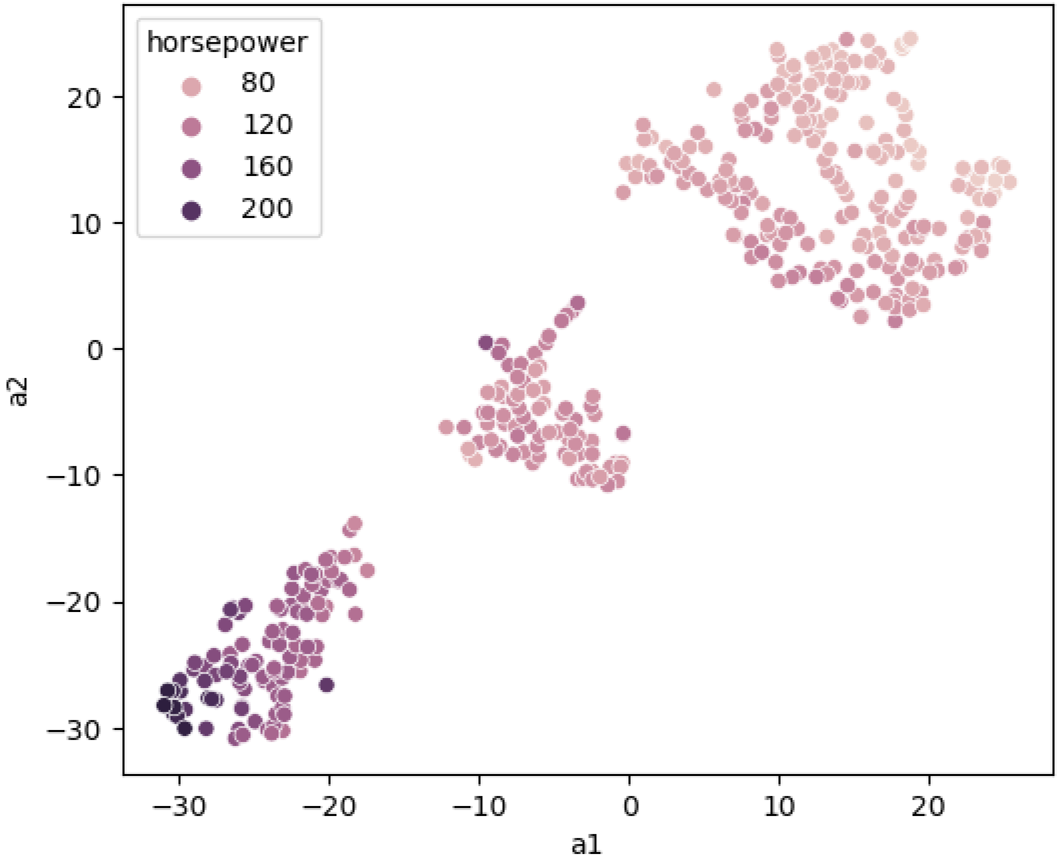
In [297…

```
# Scatter plot for two components
plt.figure(figsize = (6,5))

sns.scatterplot(x = 'a1', y = 'a2', data = cars_tsne, hue=cars.cylinders)

plt.show()
```



In [280…

```
# Scatter plot for two components
plt.figure(figsize = (6,5))

sns.scatterplot(x = 'a1', y = 'a2', data = cars_tsne, hue=cars.horsepower)

plt.show()
```



Observations:

3 main groups with 4 correlated features

1.Displacement:low of 80 high of 400

2.Weight: low 1800 high 4800

3.Cylinders: low 3 high 8

4.Horsepower: low of 80 high of 200

The majority of the cars are the lower numbers of the spectrum. The second place is the highest of the spectrum meaning heavier, more cylinders...etc The fewer cars fall in the middle of the scales.

In [84]:
```
cars_tsne = pd.DataFrame(data = cars_tsne, columns = ['a1', 'a2'])
```

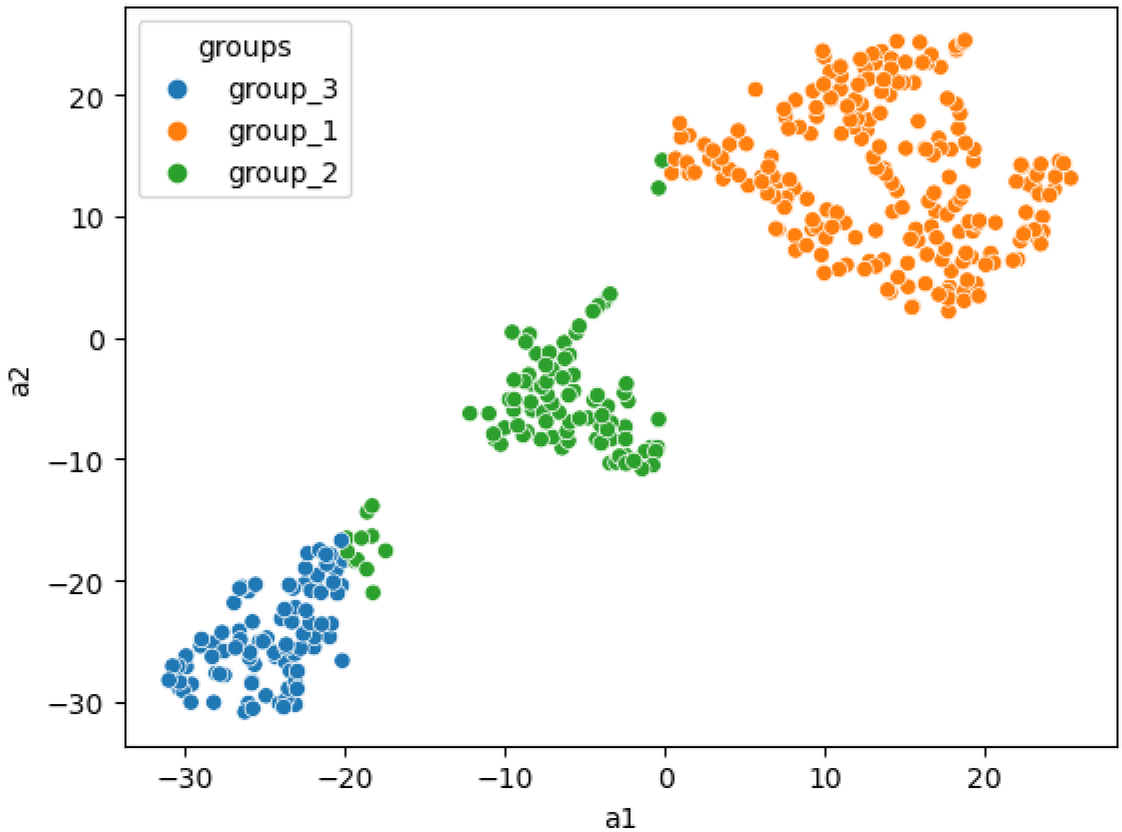
In [85]:
```
# Let's assign points to 3 different groups
def grouping(x):
    first_component = x['a1']

    second_component = x['a2']

    if (first_component > 0) and (second_component > -5):
        return 'group_1'
```

```
    if (first_component > -20 ) and (first_component < 5):
        return 'group_2'

    else:
        return 'group_3'
```

Visualize the clusters w.r.t different variables using scatter plot and box plot

In [86]:
```
cars_tsne['groups'] = cars_tsne.apply(grouping, axis = 1)
```

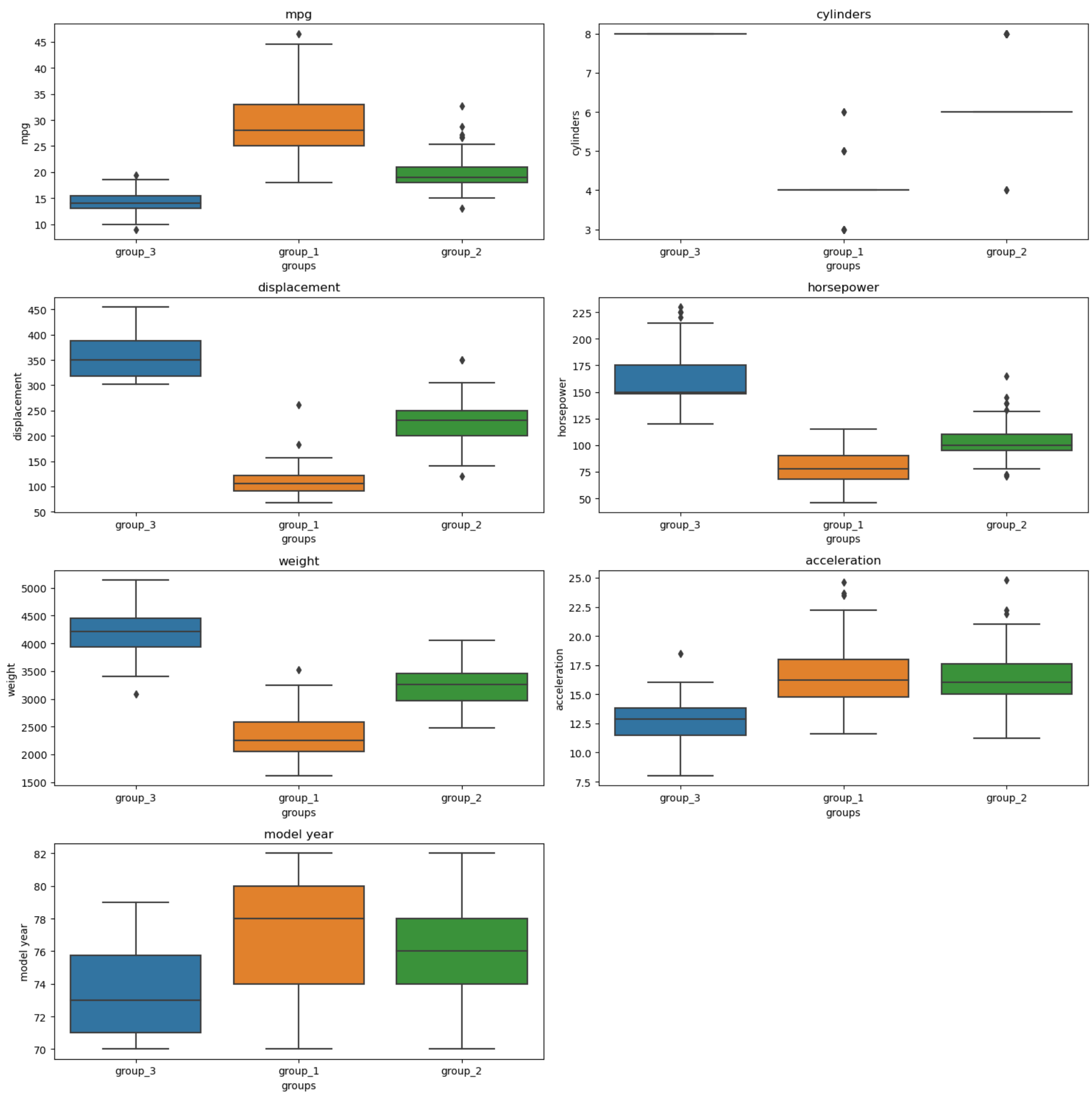
In [87]:
```
sns.scatterplot(x = cars_tsne.iloc[:,0], y = cars_tsne.iloc[:,1], hue = cars_t

plt.show()
```



In [88]:
```
cars['groups'] = cars_tsne['groups']
```

In [91]:
```
all_col = cars.columns.tolist()

plt.figure(figsize = (15, 15))

for i, variable in enumerate(all_col):
    if i == 7:
        break

    plt.subplot(4, 2, i + 1)

    sns.boxplot(y=cars[variable], x=cars_tsne['groups'])

    plt.tight_layout()
```

```
    plt.title(variable)

plt.show()
```



Actionable Insights and Recommendations

Vintage cars have an appeal to different audiences

According to the data these are the findings:

There are 3 groups in the data:

There are certain combinations of features that seem to be positively correlated.

1. Displacement:The lower the displacement the less fuel used.
2. Weight: The heavier the car that is most taxing on all the features.

3. Cylinder: generate power, efficiency and smoothness the more cylinders the heavier and more fuel consumed.
4. Horsepower: power speed and overall performance.
5. Acceleration:measurement of movement from 1-60 expressed in seconds.

Less relevant: model:specific design or version of a vehicle produced by a particular manufacturer. mpg: fuel efficiency traveled per gallon.

These are characteristics of each group:

Group1: Average values: 4 cylinders that go 30 mpg displacement of 100, and horsepower of 80. Weighing approximately 2200 lbs. year model 78. **Smaller, fuel efficient, better mileage. The majority of cars in the database fall under this category.**

Group2: Average values: 6 cylinders that go 20 mpg displacement of 220, and horsepower of 125. Weighing approximately 3300 lbs. year model 76. **Transition to more fuel efficient vehicles.**

Group3: Average values: 8 cylinders that go 15 mpg displacement of 350, and horsepower of 150. Weighing approximately 4200 lbs. year model 73. **powerful engine 8 cylinder but heavy, less fuel efficiency.**