Principle Component Analysis (PCA) for Data Visualization

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
%matplotlib inline
```

Load, Inspect, & Clean Data Set

In [2]:
```
url = "https://gist.githubusercontent.com/mlissad000/8f38f9938c6cfeb582662ddc2
ef90989/raw/fd0f059a2518dda8efbd5f1820dabcb11cea22ac/imports-85.data"
```

In [3]:
```
# loading dataset into Pandas DataFrame
df = pd.read_csv(url,
                 names=['symboling','normalized-losses','make','fuel-type','as
piration',
                        'num-of-doors','body-style','drive-wheels','engine-loc
ation','wheel-base',
                        'length','width','height','curb-weight','engine-type',
'num-of-cylinders',
                        'engine-size','fuel-system','bore','stroke','compressi
on-ratio','horsepower',
                        'peak-rpm','city-mpg','highway-mpg','price']).replace(
'?',np.NaN)
                                                   #replace all missing values of
'?' in the dataset with NaN
```

In [4]:
```
df.head()
```

Out[4]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 26 columns

In [5]:
```
#Converting object numeric dtypes to floats
obj_cols = ['normalized-losses','bore','stroke','horsepower','peak-rpm','pric
e']
df[obj_cols] = df[obj_cols].apply(pd.to_numeric)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
symboling            205 non-null int64
normalized-losses    164 non-null float64
make                 205 non-null object
fuel-type            205 non-null object
aspiration           205 non-null object
num-of-doors         203 non-null object
body-style           205 non-null object
drive-wheels         205 non-null object
engine-location      205 non-null object
wheel-base           205 non-null float64
length               205 non-null float64
width                205 non-null float64
height               205 non-null float64
curb-weight          205 non-null int64
engine-type          205 non-null object
num-of-cylinders     205 non-null object
engine-size          205 non-null int64
fuel-system          205 non-null object
bore                 201 non-null float64
stroke               201 non-null float64
compression-ratio    205 non-null float64
horsepower           203 non-null float64
peak-rpm             203 non-null float64
city-mpg             205 non-null int64
highway-mpg          205 non-null int64
price                201 non-null float64
dtypes: float64(11), int64(5), object(10)
memory usage: 41.7+ KB
```

Standardize the Data

In [7]:
```
num_cols = ['length','width','height','curb-weight','engine-size']
x = df.loc[:, num_cols].values
y = df.loc[:,['body-style']].values
```

In [8]:
```
x = StandardScaler().fit_transform(x)
```

In [9]:
```
pd.DataFrame(data = x, columns = num_cols).head()
```

Out[9]:

	length	width	height	curb-weight	engine-size
0	-0.426521	-0.844782	-2.020417	-0.014566	0.074449
1	-0.426521	-0.844782	-2.020417	-0.014566	0.074449
2	-0.231513	-0.190566	-0.543527	0.514882	0.604046
3	0.207256	0.136542	0.235942	-0.420797	-0.431076
4	0.207256	0.230001	0.235942	0.516807	0.218885

PCA Projection to 2D

In [10]:
```
pca = PCA(n_components=2)
```

In [11]:
```
principalComponents = pca.fit_transform(x)
```

In [12]:
```
principalDf = pd.DataFrame(data = principalComponents,
                           columns = ['principal component 1', 'principal comp
onent 2'])
```

In [13]:
```
principalDf.head(5)
```

Out[13]:

	principal component 1	principal component 2
0	-1.044463	1.831512
1	-1.044463	1.831512
2	0.206526	0.800114
3	-0.186852	-0.447660
4	0.630437	-0.079156

In [14]:
```
df[['body-style']].head()
```

Out[14]:

	body-style
0	convertible
1	convertible
2	hatchback
3	sedan
4	sedan

In [15]:
```
finalDf = pd.concat([principalDf, df[['body-style']]], axis = 1)
finalDf.head(5)
```

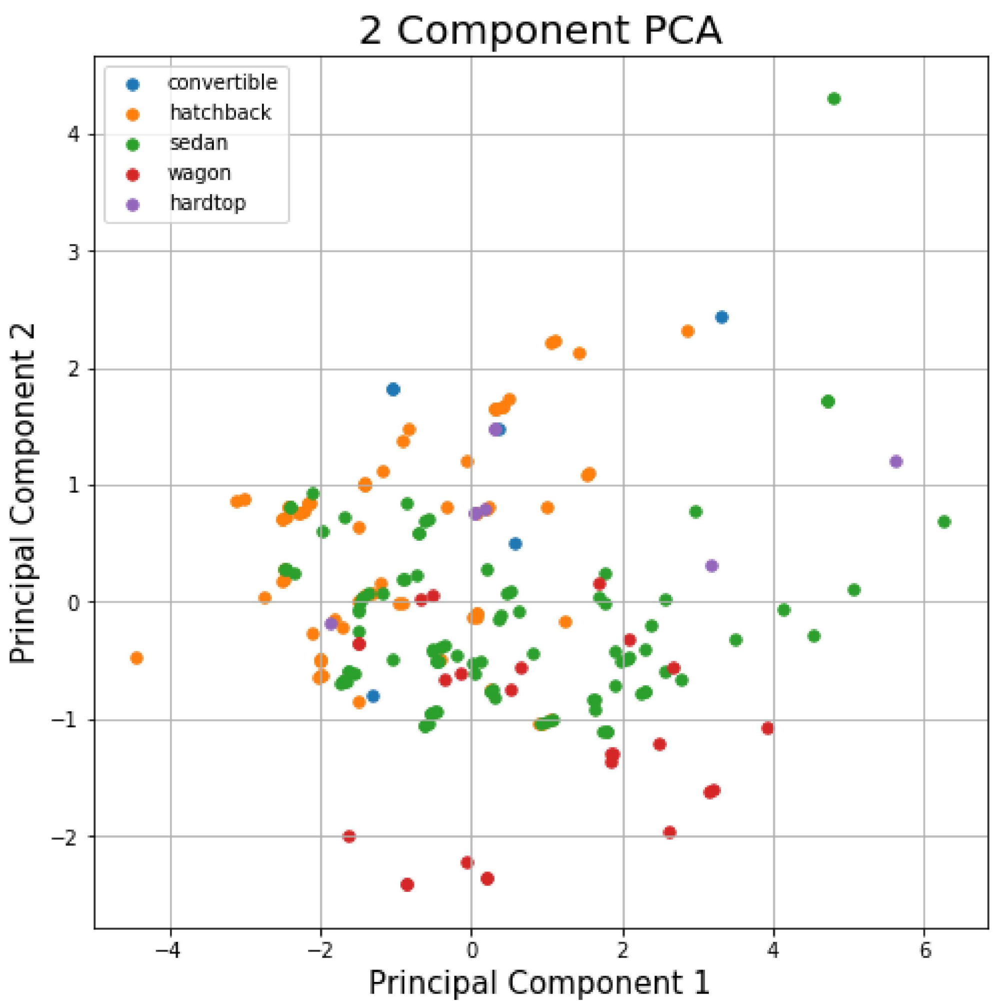
Out[15]:

	principal component 1	principal component 2	body-style
0	-1.044463	1.831512	convertible
1	-1.044463	1.831512	convertible
2	0.206526	0.800114	hatchback
3	-0.186852	-0.447660	sedan
4	0.630437	-0.079156	sedan

Visualize 2D Projection

In [16]:
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 Component PCA', fontsize = 20)


body_styles = list(df['body-style'].unique())
for body in body_styles:
    indicesToKeep = finalDf['body-style'] == body
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , s = 30)
ax.legend(body_styles)
ax.grid()
```



Explained Variance

The explained variance tells us how much information (variance) can be attributed to each of the principal components.

In [17]:
```
pca.explained_variance_ratio_
```

Out[17]:
```
array([0.7120879 , 0.19840549])
```

Together, the first two principal components contain 91.05% of the information. The first principal component contains 71.21% of the variance and the second principal component contains 19.84% of the variance. The other principal components contained the rest of the variance of the dataset.