

for data analysis

```
import pandas as pd
import numpy as np
```

for data visualization

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df= pd.read_csv("Users/Hp/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Anaconda3 (64-bit)/Data_Visualization_with_Python_s2-main/vw.csv")
display(df.shape)
print((number of rows",df.shape[0])
print((number of columns",df.shape[1])
display(df)
print(df)
ui=df.head()
display(ui)
```

(15157, 8)
number of rows= 15157
number of columns= 8
model year price transmission mileage fuelType mpg engineSize
0 T-Roc 2019 25000 Automatic 13904 Diesel 49.6 2.0
1 T-Roc 2019 26883 Automatic 4562 Diesel 49.6 2.0
2 T-Roc 2019 20000 Manual 7414 Diesel 50.4 2.0
3 T-Roc 2019 33482 Automatic 4825 Petrol 32.5 2.0
4 T-Roc 2019 22900 Semi-Auto 6500 Petrol 39.8 1.5
...
15152 Eos 2012 5990 Manual 74009 Diesel 58.9 2.0
15153 Fox 2008 1799 Manual 88102 Petrol 46.3 1.2
15154 Fox 2009 1590 Manual 70000 Petrol 42.0 1.4
15155 Fox 2006 1250 Manual 82704 Petrol 46.3 1.2
15156 Fox 2087 2295 Manual 74609 Petrol 46.3 1.2

[15157 rows x 8 columns]

```
model year price transmission mileage fuelType mpg engineSize
0 T-Roc 2019 25000 Automatic 13904 Diesel 49.6 2.0
1 T-Roc 2019 26883 Automatic 4562 Diesel 49.6 2.0
2 T-Roc 2019 20000 Manual 7414 Diesel 50.4 2.0
3 T-Roc 2019 33482 Automatic 4825 Petrol 32.5 2.0
4 T-Roc 2019 22900 Semi-Auto 6500 Petrol 39.8 1.5
```

```
display(df.head())
```

model year price transmission mileage fuelType mpg engineSize
0 T-Roc 2019 25000 Automatic 13904 Diesel 49.6 2.0
1 T-Roc 2019 26883 Automatic 4562 Diesel 49.6 2.0
2 T-Roc 2019 20000 Manual 7414 Diesel 50.4 2.0
3 T-Roc 2019 33482 Automatic 4825 Petrol 32.5 2.0
4 T-Roc 2019 22900 Semi-Auto 6500 Petrol 39.8 1.5

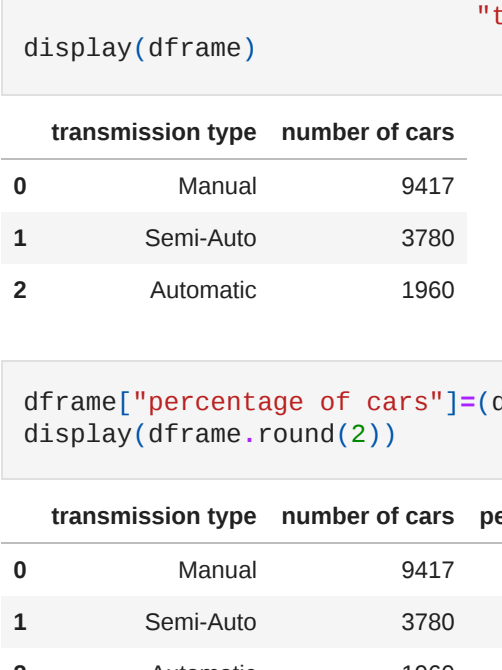
value counts()

```
display(df["transmission"].value_counts())
```

```
Manual    9417
Semi-Auto 3780
Automatic 1960
Name: transmission, dtype: int64
```

Pie-Chart

```
import matplotlib.pyplot as plt
import seaborn as sns
df["transmission"].value_counts().plot(kind="pie", autopct='%1.2f%%', startangle=99)
plt.title("Percentage of Transmission of Cars")
plt.xlabel("")
plt.ylabel("")
plt.show()
```



Creating DataFrame

```
dfFrame=DataFrame(df[["transmission"]].value_counts())
display(dfFrame)
```

print(show)

A bar chart titled "Percentage of Transmiss" (likely "Percentage of Transmissions"). The y-axis has a tick mark at 60. A single green bar is shown, representing a value of 60.

Category	Percentage
Transmissions	60

```
dfFrame=dfFrame.reset_index()
dfFrame=dfFrame.rename(columns={"index":"transmission type",
                                "transmission":"number of cars"})
display(dfFrame)
```

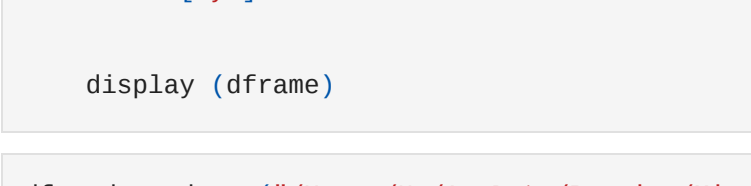
	transmission type	number of cars
0	Manual	9417
1	Semi-Auto	3780
2	Automatic	1960

```
dfFrame["percentage of cars"]=(dfFrame["number of cars"]/df.shape[0])*100
display(dfFrame.round(2))
```

	transmission type	number of cars	percentage of cars
0	Manual	9417	62.13
1	Semi-Auto	3780	24.94
2	Automatic	1960	12.93

Bar-plot

```
sns.barplot(x="transmission type",y="percentage of cars",data=dfFrame,alpha=0.50,color="green")
plt.show()
```



Comparison

The pie-chart gives a clear indication that manual type holds greater percentage alone than the other two types combined. If needed, percentages can also be shown as annotations in the bar-chart but the pie-chart as labels are not used in pie-charts.

So, for a more detailed picture, bars can be helpful while pie-charts are applicable to make visualizations in an easier/faster way.

```
def linear_eqn(x, m, c):
    temp_range=[1,11]
    c=1
    y=m*x+c
    dfFrame=pd.DataFrame()
    dfFrame["x"]=x
    dfFrame["y"]=m*x+c
    display(dfFrame)
```

```
df= pd.read_csv("Users/Hp/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Anaconda3 (64-bit)/Data_Visualization_with_Python_s2-main/vw.csv")
display(df.head(10))
```

model year price transmission mileage fuelType mpg engineSize
0 T-Roc 2019 25000 Automatic 13904 Diesel 49.6 2.0
1 T-Roc 2019 26883 Automatic 4562 Diesel 49.6 2.0
2 T-Roc 2019 20000 Manual 7414 Diesel 50.4 2.0
3 T-Roc 2019 33482 Automatic 4825 Petrol 32.5 2.0
4 T-Roc 2019 22900 Semi-Auto 6500 Petrol 39.8 1.5
5 T-Roc 2020 31895 Manual 10 Petrol 42.2 1.5
6 T-Roc 2020 27895 Manual 10 Petrol 42.2 1.5
7 T-Roc 2020 33495 Semi-Auto 10 Petrol 32.5 2.0
8 T-Roc 2019 21995 Manual 10 Petrol 44.1 1.0
9 T-Roc 2019 23285 Manual 10 Petrol 42.2 1.5

```
plt.figure(figsize=(18,14))
```

```
plt.subplot(2, 1, 1)
sns.scatterplot(x="year", y="price", data=df)
```

```
plt.subplot(2, 1, 2)
sns.scatterplot(x="engineSize", y="price", data=df)
```

```
plt.tight_layout()
plt.show()
```

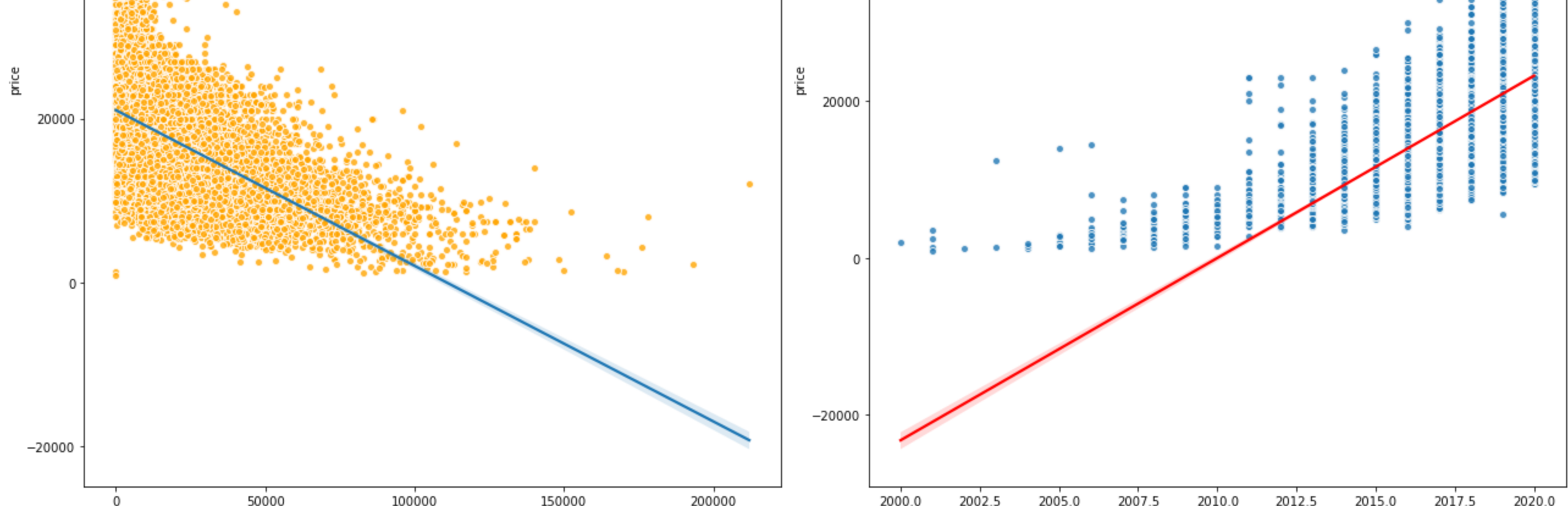


In the price Vs. year scatter plot, it is following a certain trend as the price rises with the passing of years. But the plot doesn't show any distinct relationship between engineSize and price. Price sometimes rises, sometimes stays constant, even sometimes falls as the increase in engine size.

```
plt.figure(figsize=(18,19))
plt.subplot(2, 2, 1)
sns.regplot(x="mileage", y="price", data=df, scatter_kws={"color":"orange", "edgecolor":"white"})
```

```
plt.subplot(2, 2, 2)
sns.regplot(x="year", y="price", data=df, line_kws={"color":"red"}, scatter_kws={"edgecolor":"white"})
```

```
plt.tight_layout()
plt.show()
```



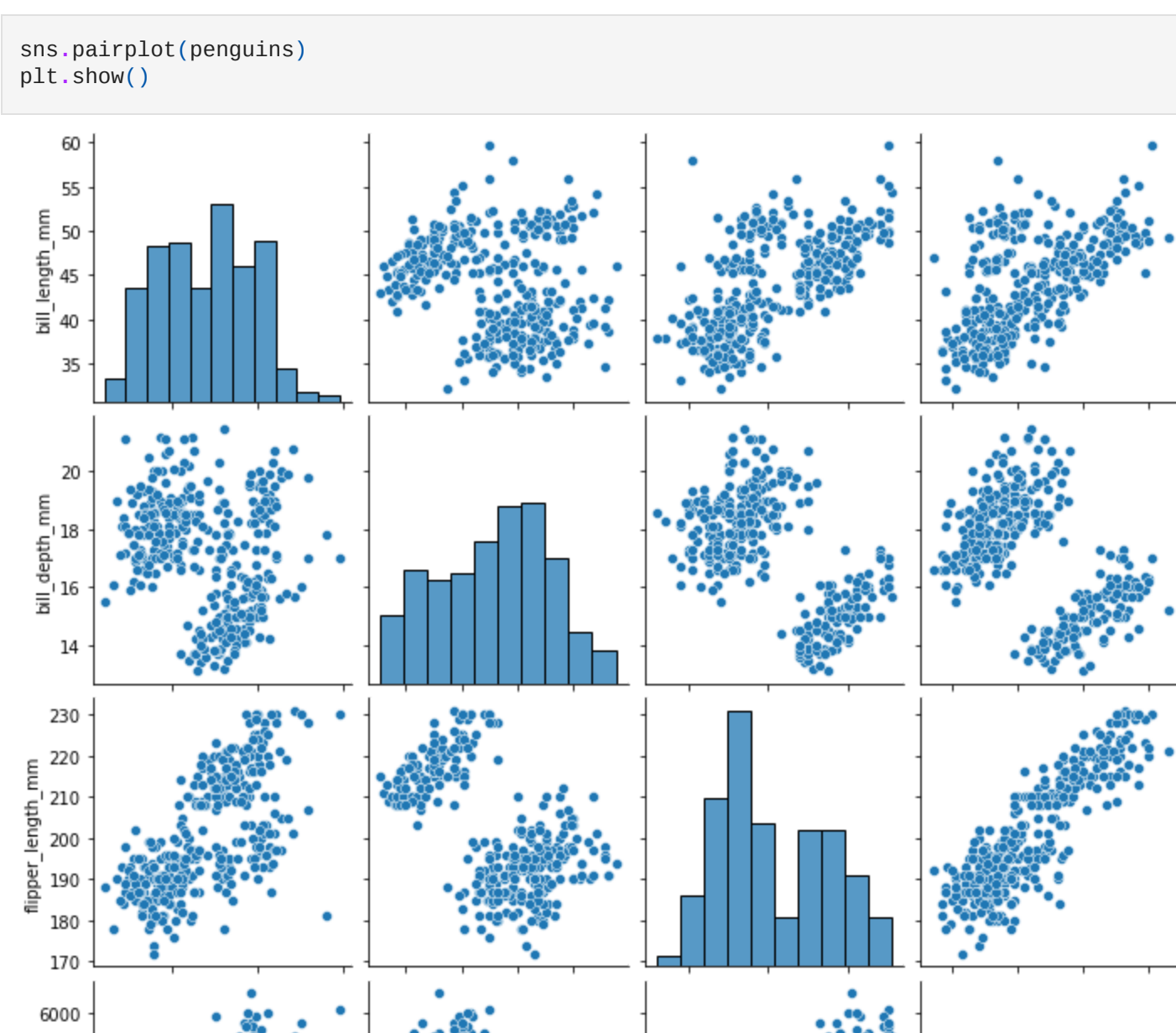
The regression plot also helps us to understand the underlying relation between dependent and independent variables, here, between mileage and price and then year and price. As the graph indicates, price falls with the increase in mileage and rises with the passing of year. The trwregression line helps the audience to understand the trend.

```
penguins = sns.load_dataset("penguins")
display(penguins.head())
print(penguins.shape)
```

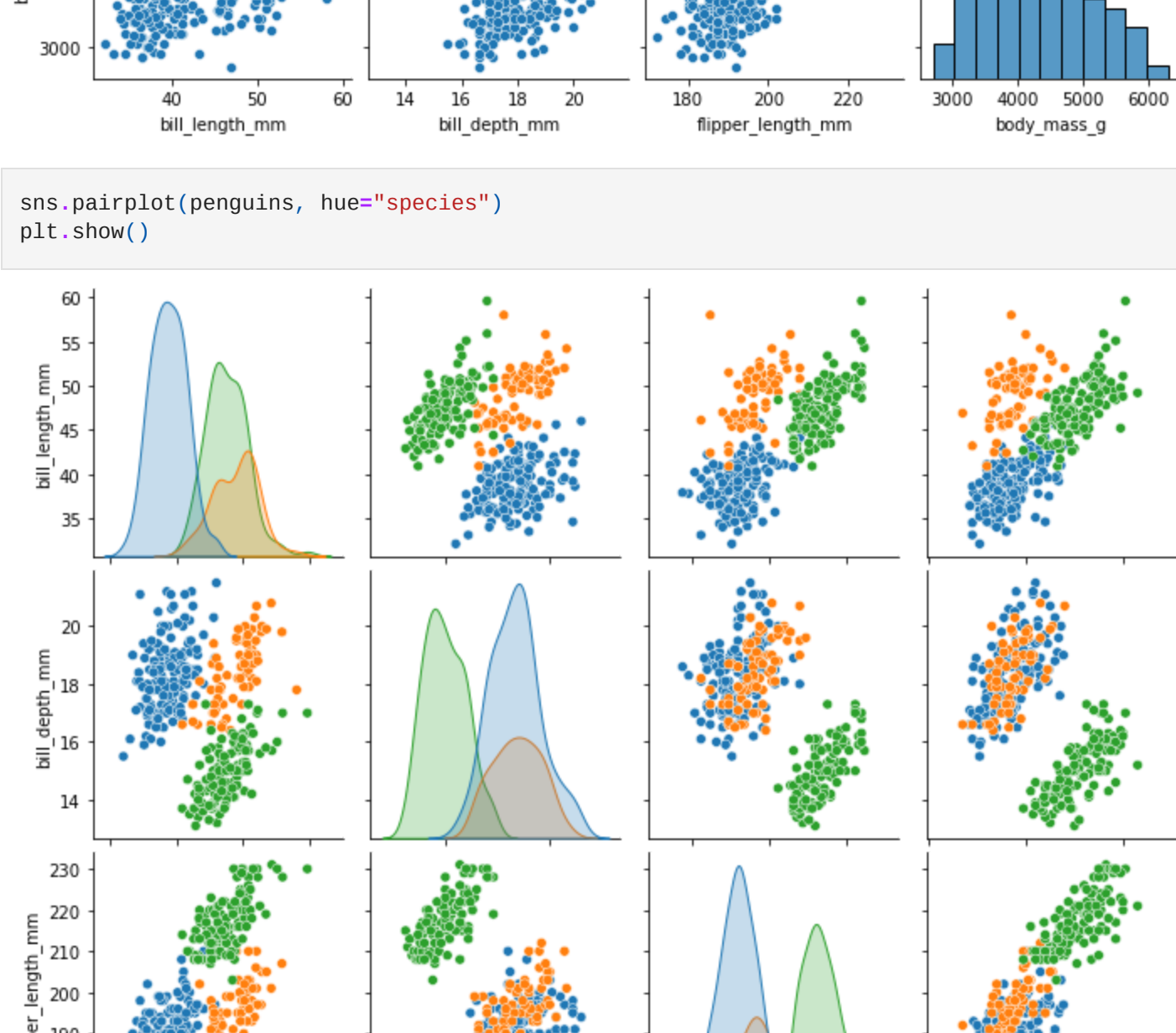
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

(344, 7)

```
sns.pairplot(penguins)
plt.show()
```

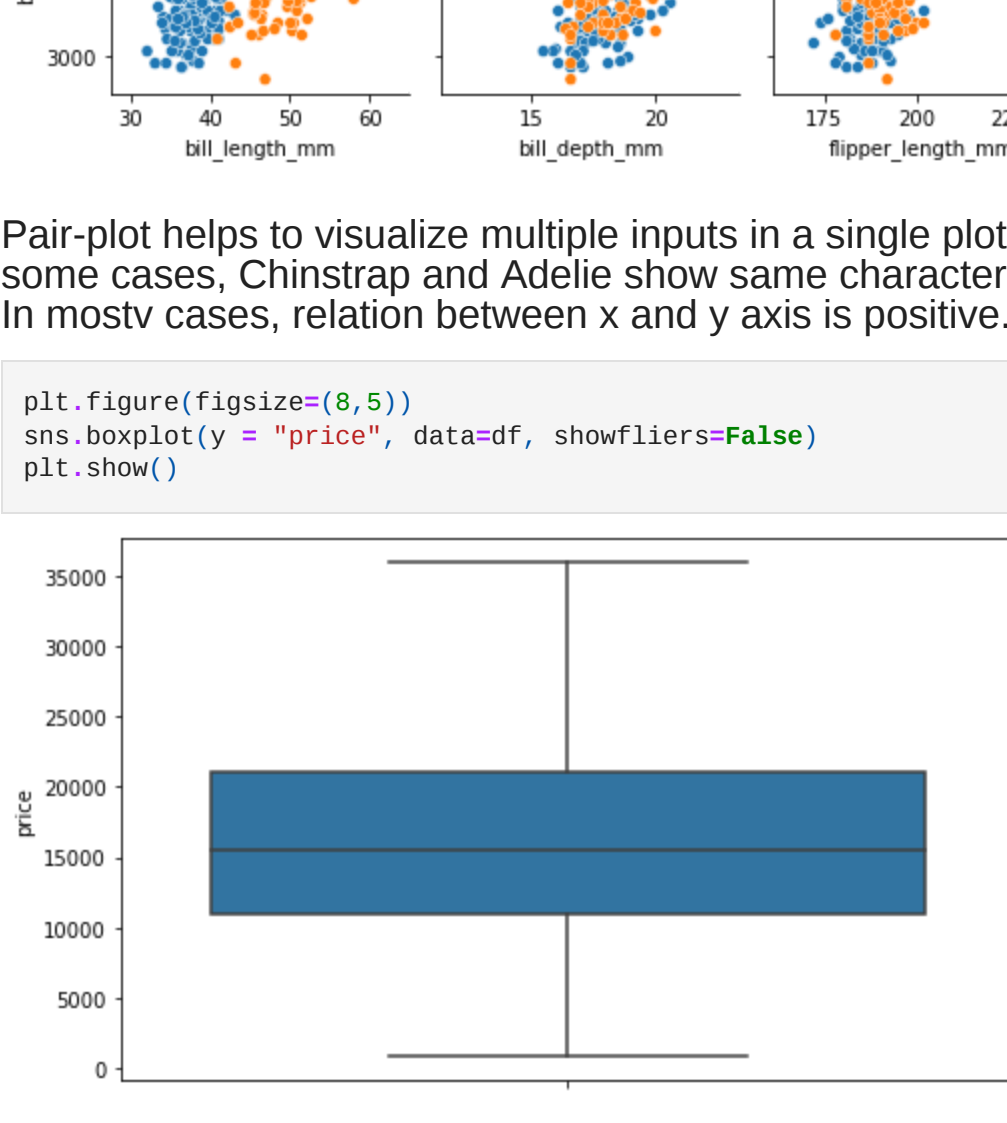


```
sns.pairplot(penguins, hue="species")
plt.show()
```



Pair-plot helps to visualize multiple inputs in a single plot. This also helps to compare those inputs among themselves. For example, here, it is shown that in some cases, Chinstrap and Adelie show same characteristics as blue and orange dots are overlapping while the green dots are showing distinct characters. In mostv cases, relation between x and y axis is positive. Adding a categorical column as hue was important as different color points indicate different inputs.

```
sns.figure(figsize=(8,5))
sns.boxplot(y = "price", data=df, showfliers=False)
plt.show()
```



Box-plot gives a summary of the visualization and can handle large amount of data. Also, it indicates the min and max values of the selected data. But, it comes with some disadvantages. It cannot return the exact value of data.

```
df.columns
Index(['model', 'year', 'price', 'transmission', 'mileage', 'fuelType', 'mpg',
       'engineSize'],
      dtype='object')
```

```
df.info()
```

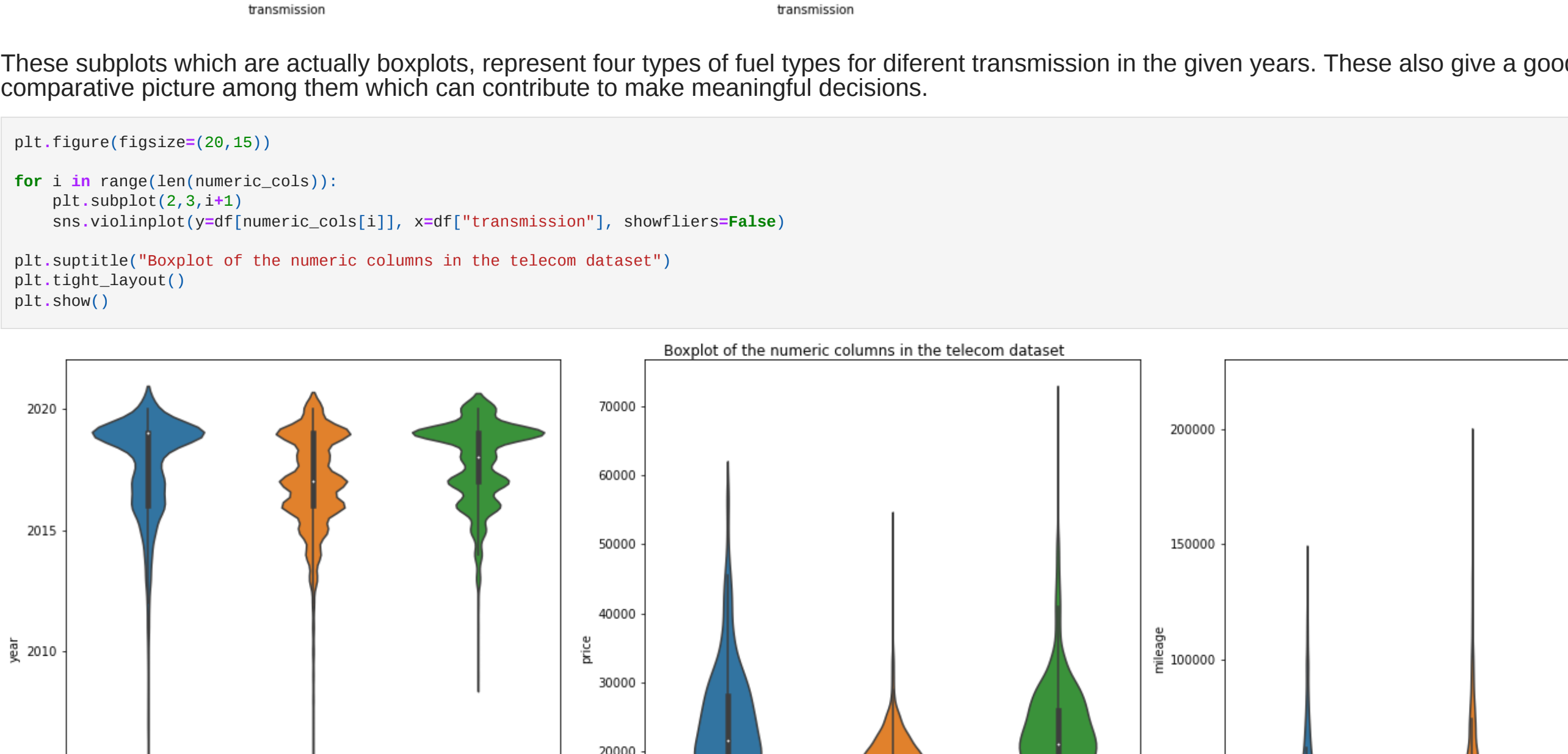
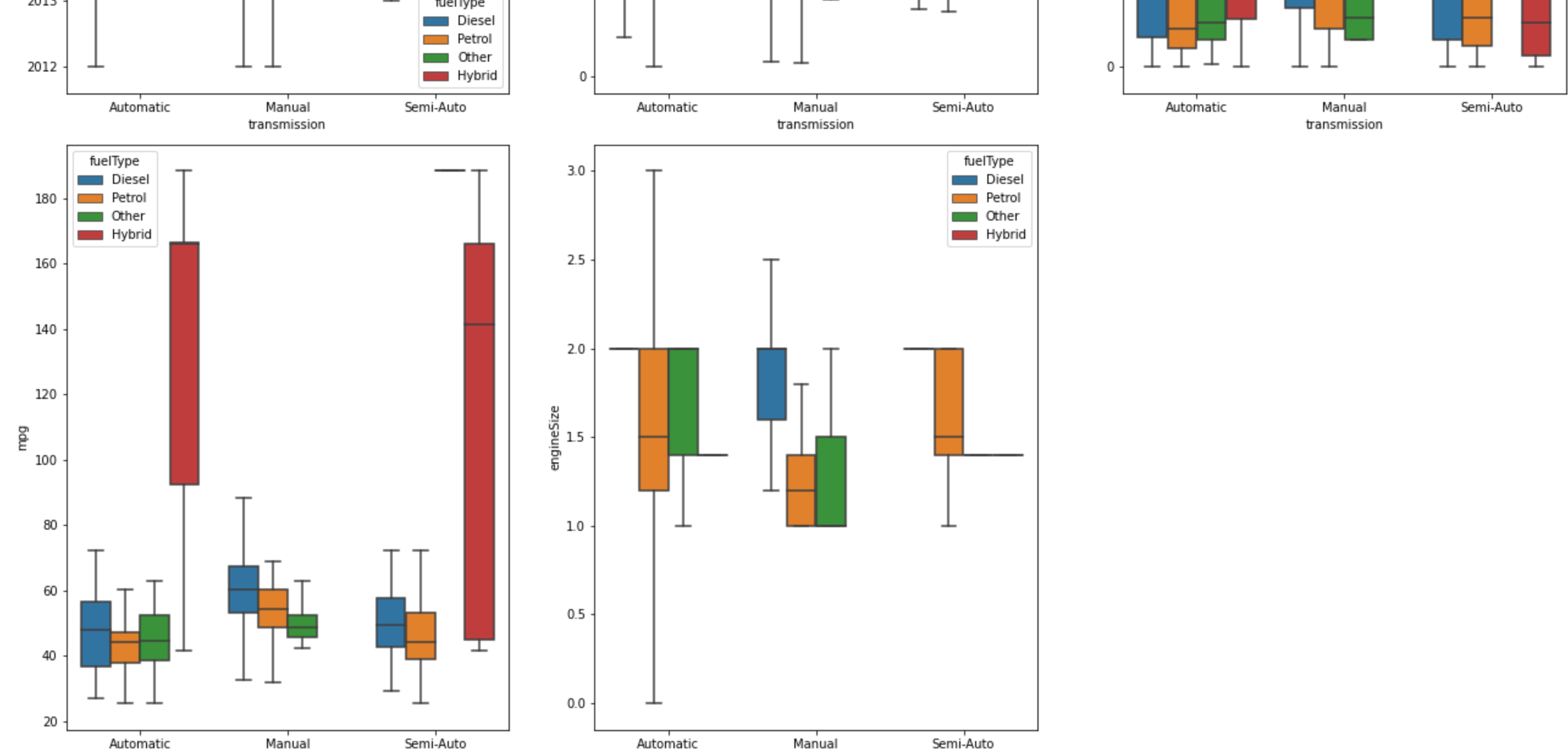
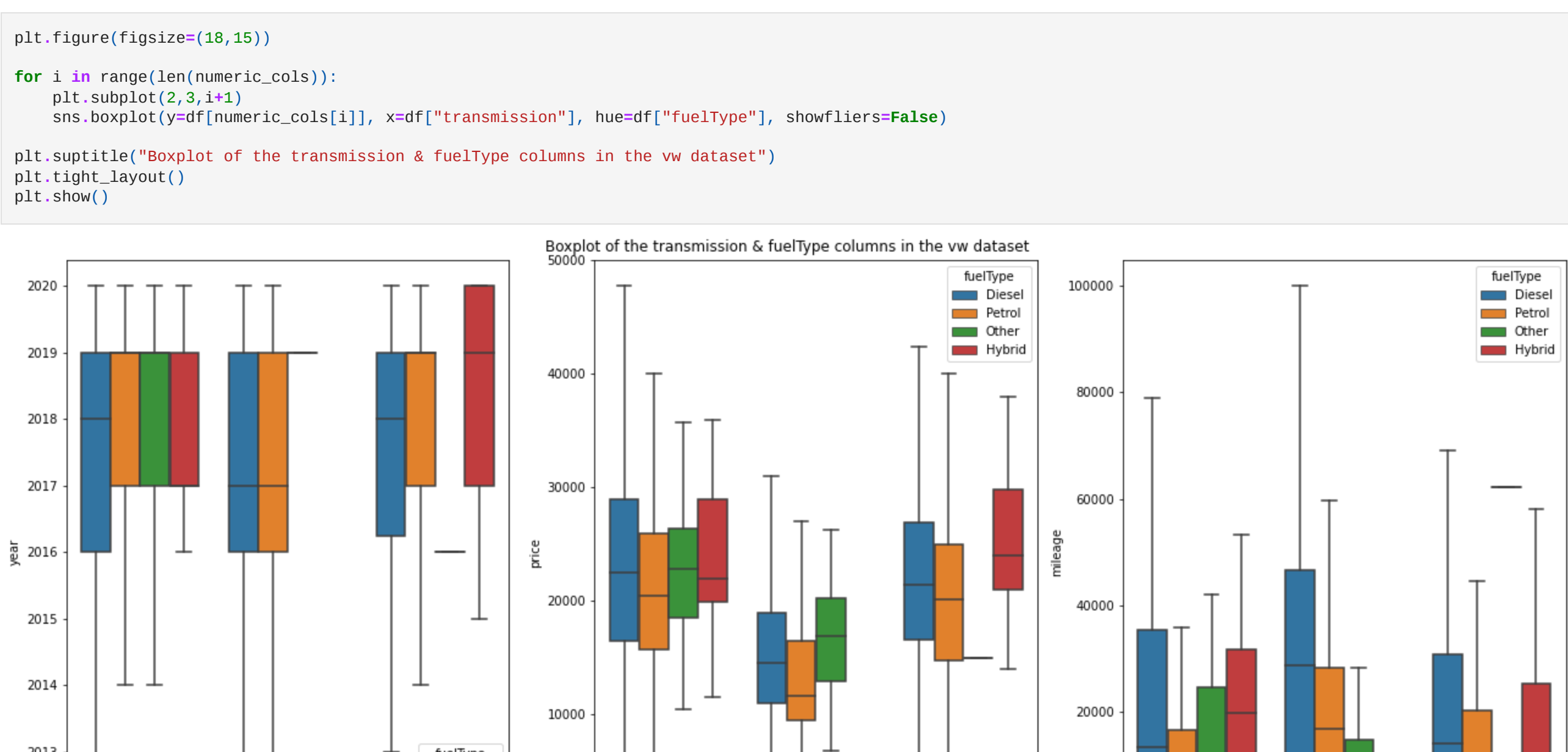
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15157 entries, 0 to 15156
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
--  --
 0   model      15157 non-null   object
 1   year       15157 non-null   int64
 2   price      15157 non-null   object
 3   transmission 15157 non-null   object
 4   mileage    15157 non-null   int64
 5   fuelType   15157 non-null   object
 6   mpg        15157 non-null   float64
 7   engineSize 15157 non-null   float64
dtypes: float64(2), int64(3), object(3)
memory usage: 847.4+ KB
```

```
numeric_cols = []
for cols in df.columns:
    if df[cols].dtypes == "o":
        numeric_cols.append(cols)
print(numeric_cols)
```

```
['year', 'price', 'mileage', 'mpg', 'engineSize']
```

```
plt.figure(figsize=(18,15))
for i in range(len(numeric_cols)):
    plt.subplot(2,3,i+1)
    sns.boxplot(y=df[numeric_cols[i]], x=df["transmission"], showfliers=False)
```

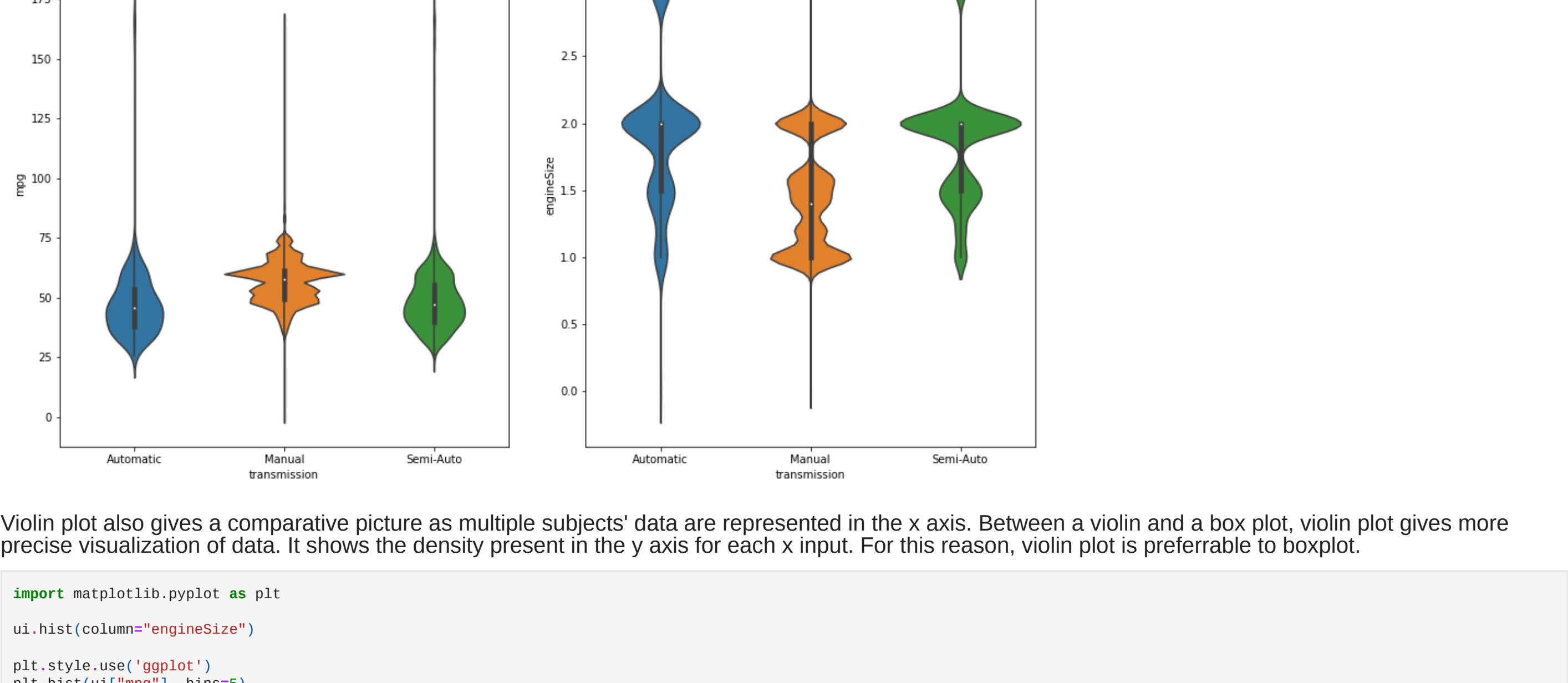
```
plt.suptitle("Boxplot of the transmission & fuelType columns in the vw dataset")
plt.tight_layout()
plt.show()
```



These subplots which are actually boxplots, represent four types of fuel types for different transmission in the given years. These also give a good comparative picture among them which can contribute to make meaningful decisions.

```
plt.figure(figsize=(20,15))
for i in range(len(numeric_cols)):
    plt.subplot(2,3,i+1)
    sns.violinplot(y=df[numeric_cols[i]], x=df["transmission"], showfliers=False)
```

```
plt.suptitle("Boxplot of the numeric columns in the telecom dataset")
plt.tight_layout()
plt.show()
```



Violin plot also gives a comparative picture as multiple subjects' data are represented in the x axis. Between a violin and a box plot, violin plot gives more precise visualization of data. It shows the density present in the y axis for each x input. For this reason, violin plot is preferable to boxplot.

```
import matplotlib.pyplot as plt
ui.hist(columns="engineSize")
```

```
plt.style.use('ggplot')
plt.hist(ui["mpg"], bins=5)
plt.tight_layout()
plt.show()
```



Histogram is used for continuous data and data distribution. It cannot compare among individual pieces of data as it is used for categorized data.