ction-with-machine-learning-oasis

March 4, 2024

1 CAR PRICE PREDICTION WITH MACHINE LEARNING (OASIS)

Problem Statement:

The price of a car depends on a lot of factors like the goodwill of the brand of the car, features of the car, horsepower and the mileage it gives and many more. Car price prediction is one of the major research areas in machine learning. So if you want to learn how to train a car price prediction model then this project is for you.

Importing necessary libraries

[27]:
```
import numpy as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Loading the dataset

[4]:
```
import pandas as pd

Data = pd.read_csv('car data.csv')

Data.head(10)
```

[4]:
```
         Car_Name  Year  Selling_Price  Present_Price  Driven_kms Fuel_Type  \
0            ritz  2014           3.35           5.59       27000    Petrol
1             sx4  2013           4.75           9.54       43000    Diesel
2            ciaz  2017           7.25           9.85        6900    Petrol
3         wagon r  2011           2.85           4.15        5200    Petrol
4           swift  2014           4.60           6.87       42450    Diesel
5   vitara brezza  2018           9.25           9.83        2071    Diesel
6            ciaz  2015           6.75           8.12       18796    Petrol
7         s cross  2015           6.50           8.61       33429    Diesel
```

```
8           ciaz  2016           8.75           8.89       20273     Diesel
9           ciaz  2015           7.45           8.92       42367     Diesel

  Selling_type Transmission  Owner
0       Dealer       Manual      0
1       Dealer       Manual      0
2       Dealer       Manual      0
3       Dealer       Manual      0
4       Dealer       Manual      0
5       Dealer       Manual      0
6       Dealer       Manual      0
7       Dealer       Manual      0
8       Dealer       Manual      0
9       Dealer       Manual      0
```

Data Preprocessing

[13]:
```
# Check for missing values and data types
Data_info = Data.info()

# Summary statistics for numerical features
Data_description = Data.describe()

print('Data Information:')
print(Data_info)
print('\
Summary Statistics for Numerical Features:')
print(Data_description)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Driven_kms     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Selling_type   301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
Data Information:
None
Summary Statistics for Numerical Features:
```

```
              Year  Selling_Price  Present_Price     Driven_kms       Owner
count   301.000000     301.000000     301.000000     301.000000  301.000000
mean   2013.627907       4.661296       7.628472   36947.205980    0.043189
std       2.891554       5.082812       8.642584   38886.883882    0.247915
min    2003.000000       0.100000       0.320000     500.000000    0.000000
25%    2012.000000       0.900000       1.200000   15000.000000    0.000000
50%    2014.000000       3.600000       6.400000   32000.000000    0.000000
75%    2016.000000       6.000000       9.900000   48767.000000    0.000000
max    2018.000000      35.000000      92.600000  500000.000000    3.000000
```

[14]:
```
Data.describe()
```

[14]:
```
              Year  Selling_Price  Present_Price     Driven_kms       Owner
count   301.000000     301.000000     301.000000     301.000000  301.000000
mean   2013.627907       4.661296       7.628472   36947.205980    0.043189
std       2.891554       5.082812       8.642584   38886.883882    0.247915
min    2003.000000       0.100000       0.320000     500.000000    0.000000
25%    2012.000000       0.900000       1.200000   15000.000000    0.000000
50%    2014.000000       3.600000       6.400000   32000.000000    0.000000
75%    2016.000000       6.000000       9.900000   48767.000000    0.000000
max    2018.000000      35.000000      92.600000  500000.000000    3.000000
```

[15]:
```
Data.isnull().sum()
```

[15]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Driven_kms       0
Fuel_Type        0
Selling_type     0
Transmission     0
Owner            0
dtype: int64
```

[16]:
```
Data.duplicated().sum()
```

[16]:
```
2
```

[21]:
```
Data.duplicated().drop
```

[21]:
```
<bound method Series.drop of 0      False
1      False
2      False
3      False
4      False
       …
```

```
296    False
297    False
298    False
299    False
300    False
Length: 301, dtype: bool>
```

EDA

[25]:
```
Data['Owner'].value_counts()
```

[25]:
```
0    290
1     10
3      1
Name: Owner, dtype: int64
```

[26]:
```
Data['Car_Name'].value_counts()
```

[26]:
```
city                        26
corolla altis               16
verna                       14
fortuner                    11
brio                        10
                            ..
Honda CB Trigger             1
Yamaha FZ S                  1
Bajaj Pulsar 135 LS          1
Activa 4g                    1
Bajaj Avenger Street 220     1
Name: Car_Name, Length: 98, dtype: int64
```

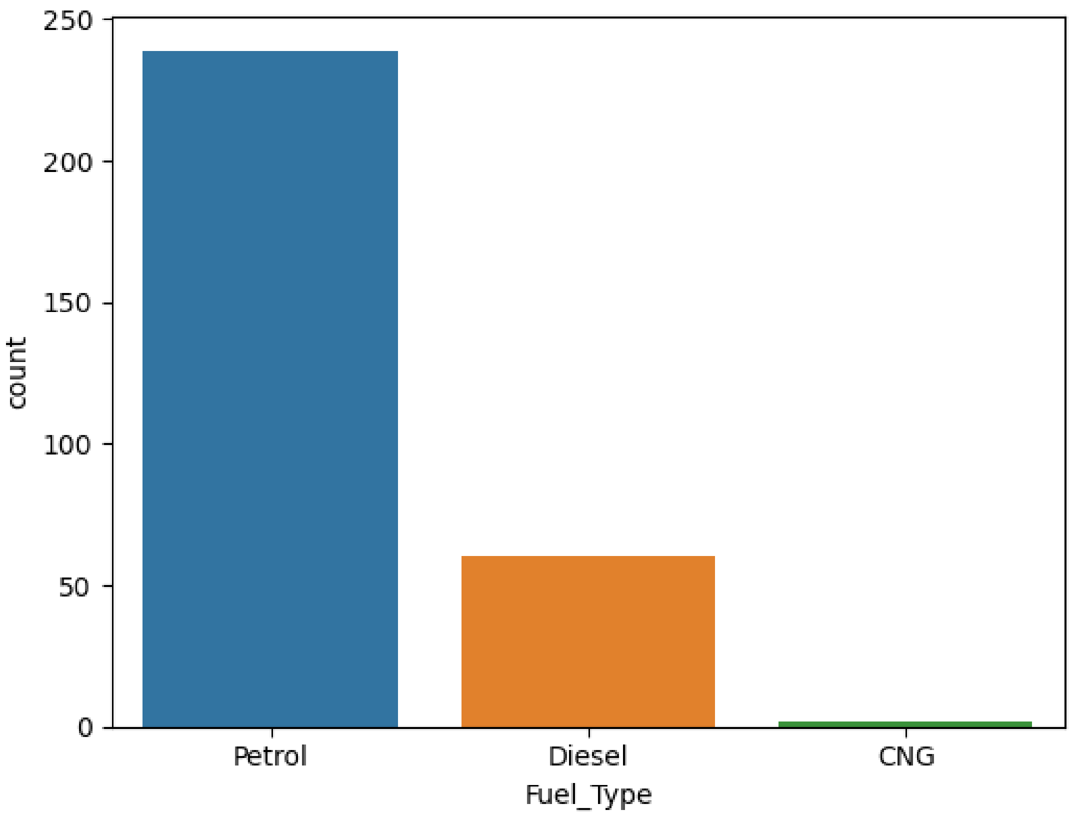
[28]:
```
Data['Fuel_Type'].value_counts()
```

[28]:
```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
```

[36]:
```
import matplotlib.pyplot as plt
import seaborn as sns
Data['Fuel_Type'].value_counts()
```

[36]:
```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
```

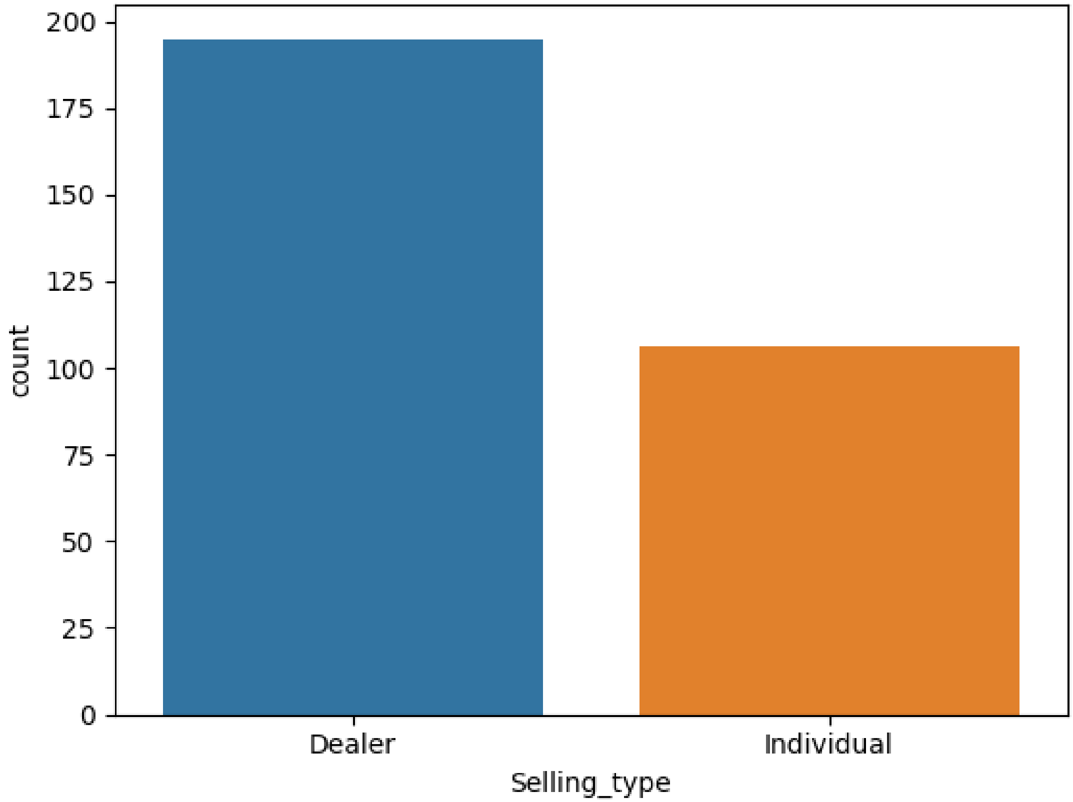
[40]:
```
sns.countplot(x='Fuel_Type', data=Data)
plt.show()
```



[41]:
```
Data['Selling_type'].value_counts()
```

[41]:
```
Dealer        195
Individual    106
Name: Selling_type, dtype: int64
```

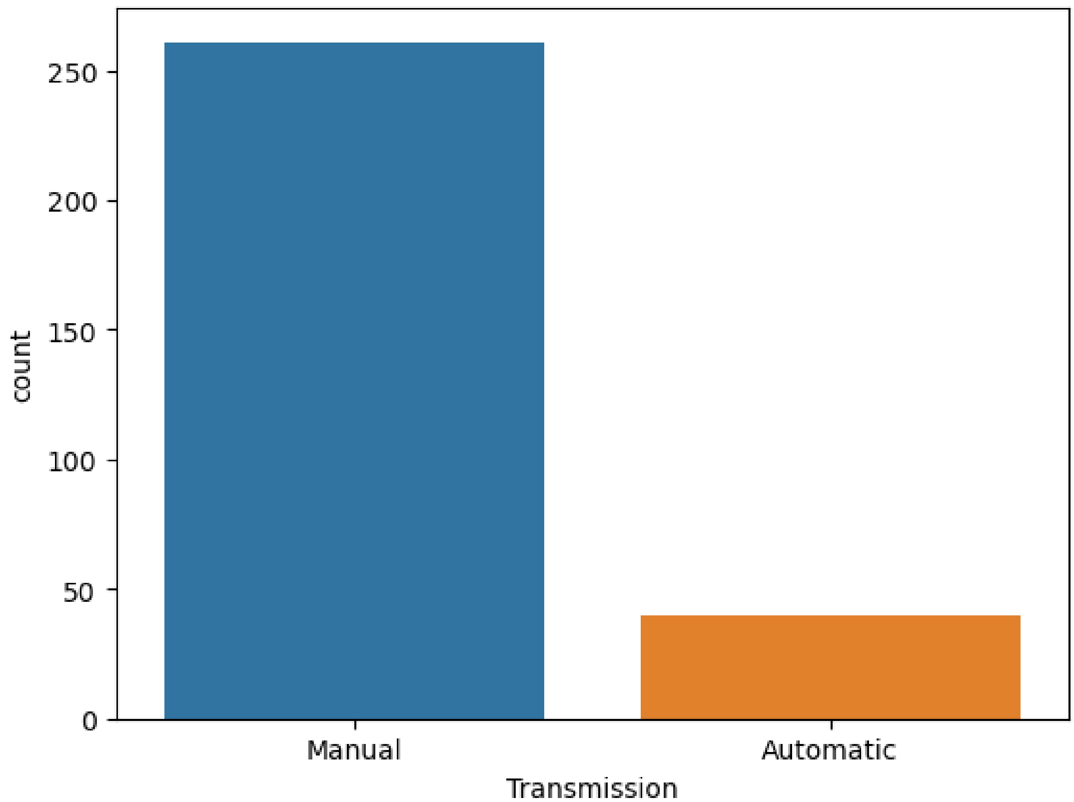
[43]:
```
sns.countplot(x='Selling_type', data=Data)
plt.show()
```



[44]:
```
Data['Transmission'].value_counts()
```

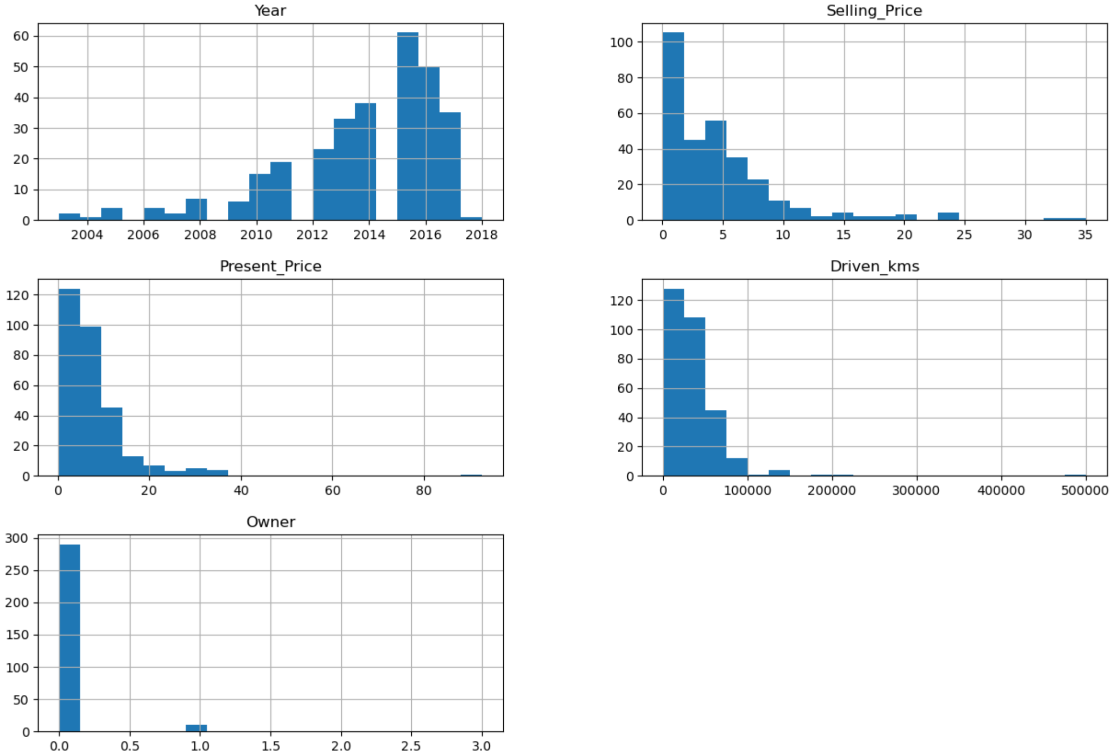
[44]:
```
Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

[47]:
```
sns.countplot(x='Transmission', data=Data)
plt.show()
```



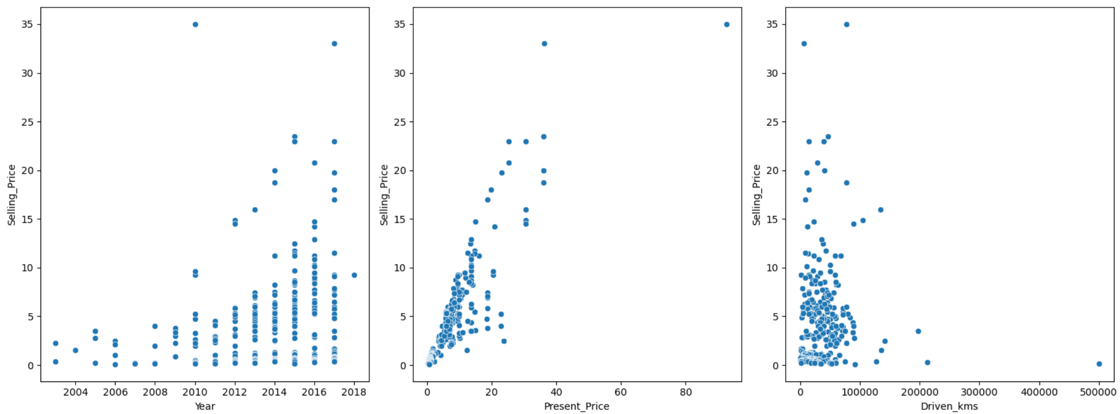
[48]:
```
Data.hist(bins=20, figsize=(15, 10))
plt.show
```

[48]: <function matplotlib.pyplot.show(close=None, block=None)>



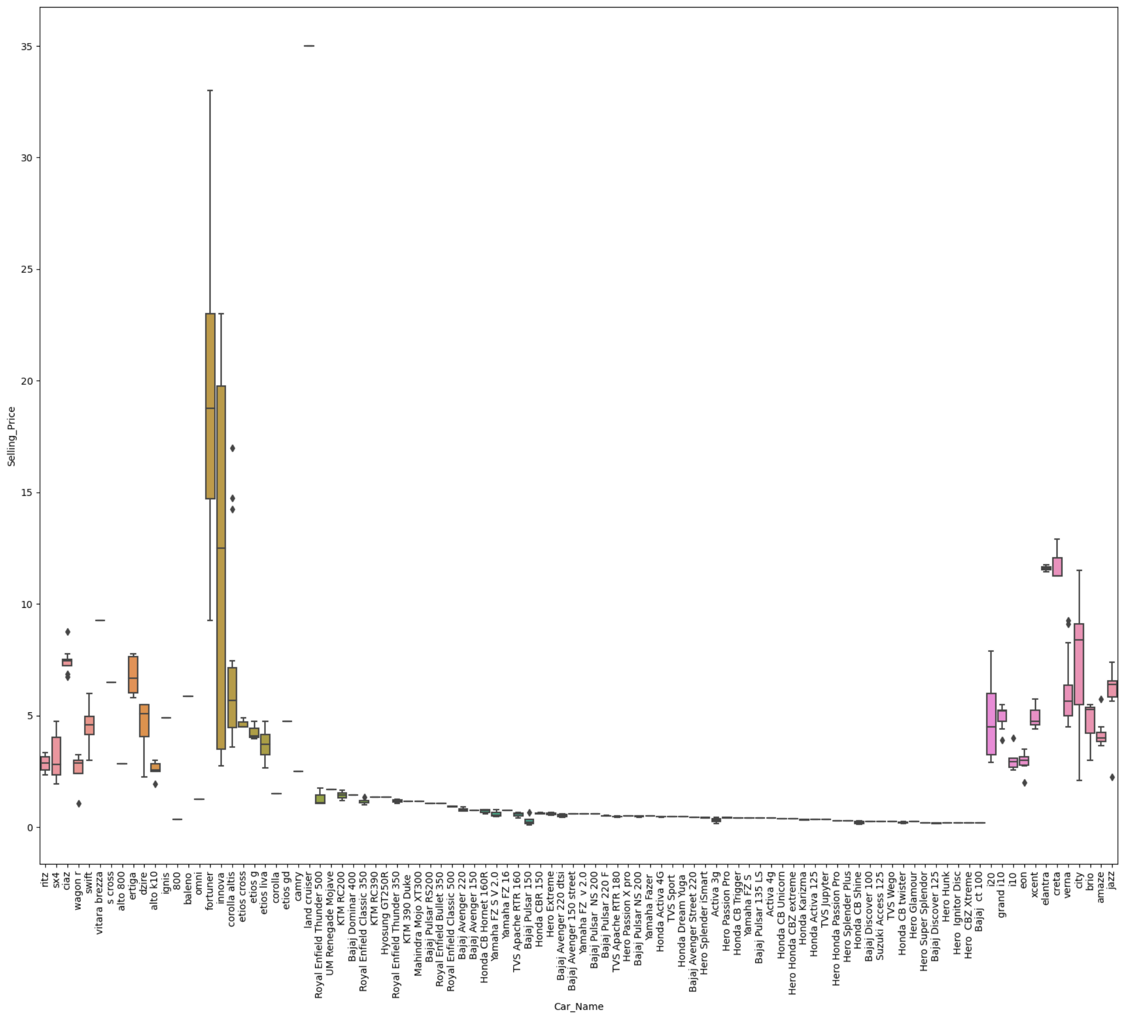
[52]:
```
#Finding relationships between different numerical features and our target
↪features
```

[51]:
```
plt.figure(figsize=(16, 6))
plt.subplot(1, 3, 1)
sns.scatterplot(x='Year', y='Selling_Price', data=Data)
plt.subplot(1, 3, 2)
sns.scatterplot(x='Present_Price', y='Selling_Price', data=Data)
plt.subplot(1, 3, 3)
sns.scatterplot(x='Driven_kms', y='Selling_Price', data=Data)
plt.tight_layout()
plt.show()
```



[53]:
```
#Finding Relationship between Cars and it's Selling price using BOXPlot
```

[55]:
```
plt.figure(figsize=(20,16))
sns.boxplot(x='Car_Name', y='Selling_Price', data=Data)
plt.xticks(rotation=90)
plt.show()
```

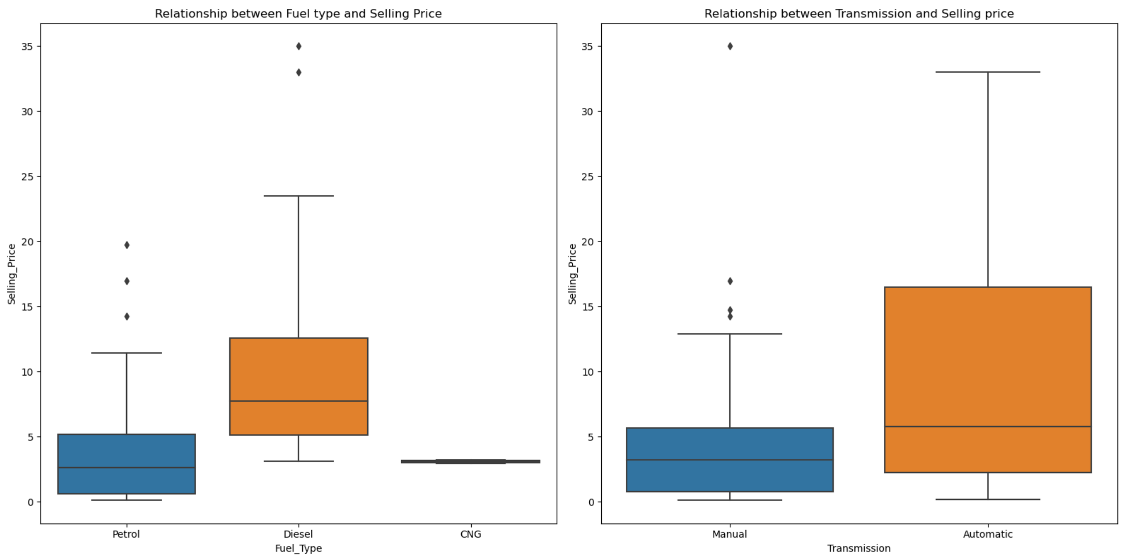


[ ]:

```
# Categorical Feature vs. Price
```

[58]:

```
plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
sns.boxplot(x='Fuel_Type', y='Selling_Price', data=Data)
plt.title('Relationship between Fuel type and Selling Price')
plt.subplot(1, 2, 2)
sns.boxplot(x='Transmission', y='Selling_Price', data=Data)
plt.title('Relationship between Transmission and Selling price')
plt.tight_layout()
plt.show()
```



Model Building

```
[73]: #Split the datset into features
```

```
[74]: X = Data.drop('Selling_Price', axis=1)
y = Data['Selling_Price']
```

```
[76]: # One-hot encoding categorical values into numerical values
```

```
[78]: X_encoded = pd.get_dummies(X, columns=['Fuel_Type', 'Selling_type',␣
      ↪'Transmission','Car_Name'], prefix=['Fuel', 'Selling',␣
      ↪'Transmission','Cars'])
```

```
[79]: #Splitting the dataset
```

```
[80]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.
      ↪2, random_state=42)
```

```
[81]: #Train a Regression Model
```

```
[82]: from sklearn.linear_model import LinearRegression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

```
[82]: LinearRegression()
```

```
[83]: y_pred_linear = linear_model.predict(X_test)
```

[84]:
```
#Evaluating the Regression Model
```

[85]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
mse_linear = mean_squared_error(y_test, y_pred_linear)
rmse_linear = sqrt(mse_linear)
print(f'Linear Regression RMSE: {rmse_linear}')
```

```
Linear Regression RMSE: 1.5125556296301736
```

[86]:
```
#Train a Random Forest Model
```

[87]:
```
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
```

[87]:
```
RandomForestRegressor(random_state=42)
```

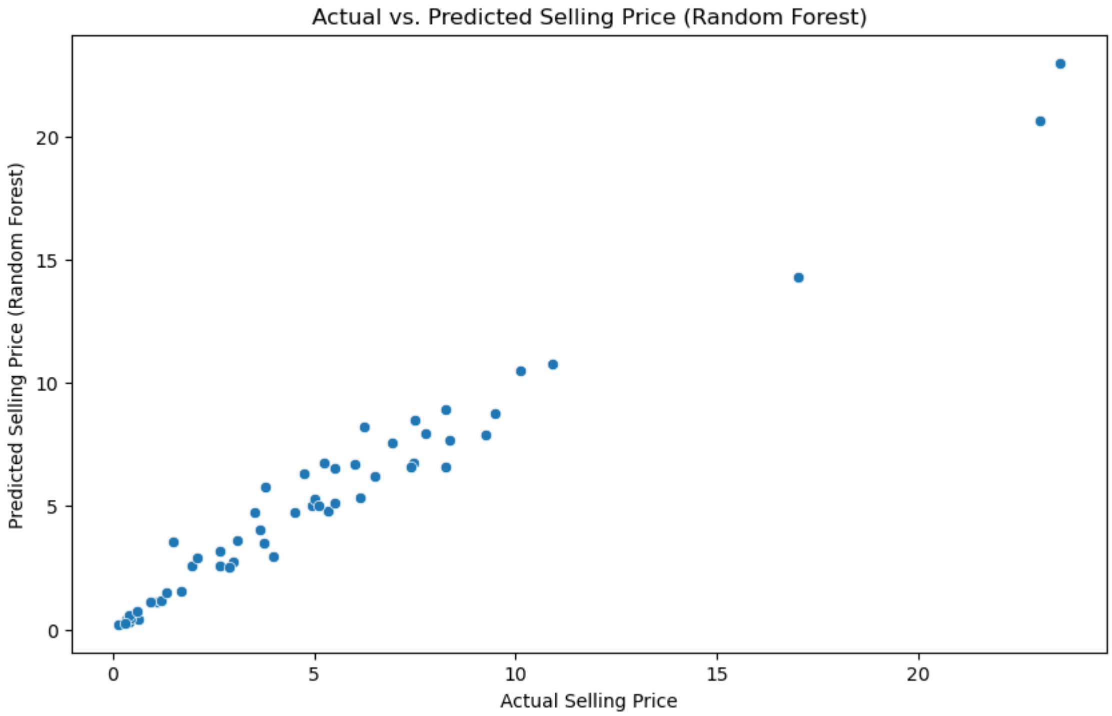
[88]:
```
y_pred_rf = rf_model.predict(X_test)
```

[89]:
```
#Evaluating the Random Forest Model
```

[90]:
```
mse_rf = mean_squared_error(y_test, y_pred_rf)
rmse_rf = sqrt(mse_rf)
print(f'Random Forest RMSE: {rmse_rf}')
```

```
Random Forest RMSE: 0.8724393833985933
```

[91]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred_rf)
plt.xlabel('Actual Selling Price')
plt.ylabel('Predicted Selling Price (Random Forest)')
plt.title('Actual vs. Predicted Selling Price (Random Forest)')
plt.show()
```


Actual vs. Predicted Selling Price (Random Forest)
Predicted Selling Price (Random Forest)
Actual Selling Price
0
5
10
15
20