Car Price Prediction With Machine Learning

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
```

In [3]:
```
data = pd.read_csv("car data.csv")
data.head()
```

Out[3]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Tran
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
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
```

In [5]:
```
data.info()
```

```
<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
```

In [6]:
```
print(data.describe())
```

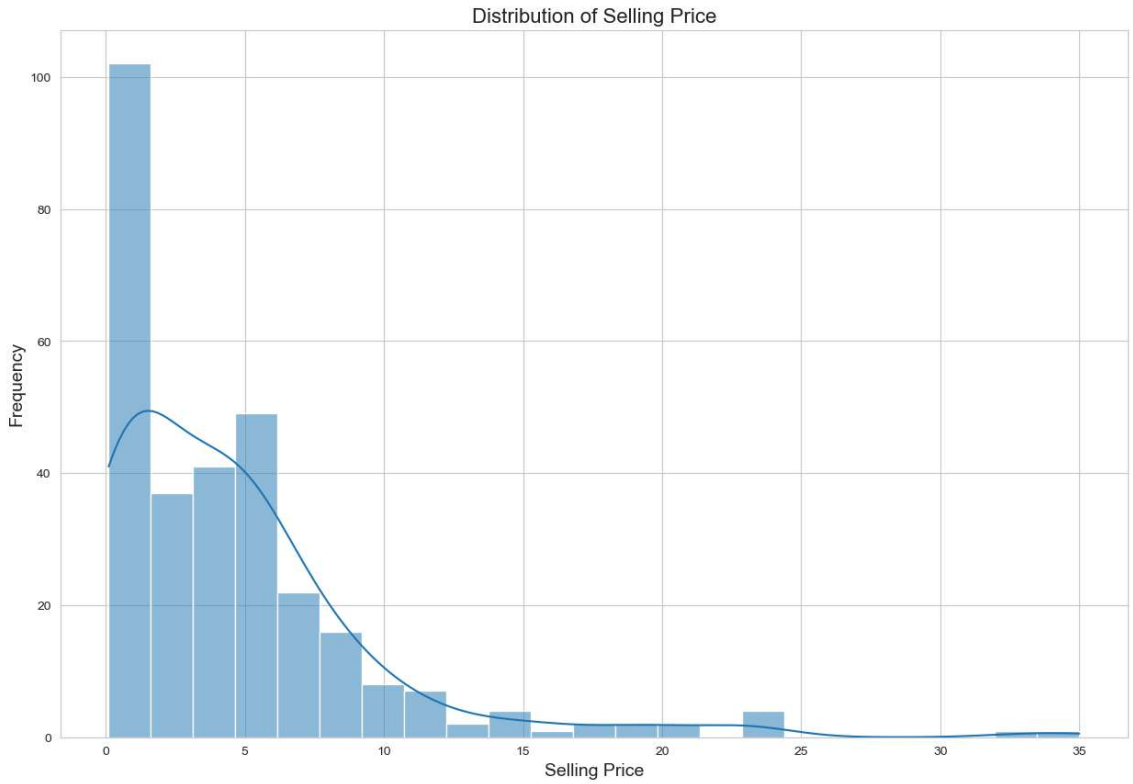
```
              Year  Selling_Price  Present_Price     Driven_kms       Own
er
count   301.000000     301.000000     301.000000     301.000000  301.0000
00
mean   2013.627907       4.661296       7.628472   36947.205980    0.0431
89
std       2.891554       5.082812       8.642584   38886.883882    0.2479
15
min    2003.000000       0.100000       0.320000     500.000000    0.0000
00
25%    2012.000000       0.900000       1.200000   15000.000000    0.0000
00
50%    2014.000000       3.600000       6.400000   32000.000000    0.0000
00
75%    2016.000000       6.000000       9.900000   48767.000000    0.0000
00
max    2018.000000      35.000000      92.600000  500000.000000    3.0000
00
```

In [10]:
```
print(data.Car_Name.unique())
```

```
['ritz' 'sx4' 'ciaz' 'wagon r' 'swift' 'vitara brezza' 's cross'
 'alto 800' 'ertiga' 'dzire' 'alto k10' 'ignis' '800' 'baleno' 'omni'
 'fortuner' 'innova' 'corolla altis' 'etios cross' 'etios g' 'etios liva'
 'corolla' 'etios gd' 'camry' 'land cruiser' 'Royal Enfield Thunder 500'
 'UM Renegade Mojave' 'KTM RC200' 'Bajaj Dominar 400'
 'Royal Enfield Classic 350' 'KTM RC390' 'Hyosung GT250R'
 'Royal Enfield Thunder 350' 'KTM 390 Duke ' 'Mahindra Mojo XT300'
 'Bajaj Pulsar RS200' 'Royal Enfield Bullet 350'
 'Royal Enfield Classic 500' 'Bajaj Avenger 220' 'Bajaj Avenger 150'
 'Honda CB Hornet 160R' 'Yamaha FZ S V 2.0' 'Yamaha FZ 16'
 'TVS Apache RTR 160' 'Bajaj Pulsar 150' 'Honda CBR 150' 'Hero Extreme'
 'Bajaj Avenger 220 dtsi' 'Bajaj Avenger 150 street' 'Yamaha FZ  v 2.0'
 'Bajaj Pulsar  NS 200' 'Bajaj Pulsar 220 F' 'TVS Apache RTR 180'
 'Hero Passion X pro' 'Bajaj Pulsar NS 200' 'Yamaha Fazer '
 'Honda Activa 4G' 'TVS Sport ' 'Honda Dream Yuga '
 'Bajaj Avenger Street 220' 'Hero Splender iSmart' 'Activa 3g'
 'Hero Passion Pro' 'Honda CB Trigger' 'Yamaha FZ S '
 'Bajaj Pulsar 135 LS' 'Activa 4g' 'Honda CB Unicorn'
 'Hero Honda CBZ extreme' 'Honda Karizma' 'Honda Activa 125' 'TVS Jupyte
r'
 'Hero Honda Passion Pro' 'Hero Splender Plus' 'Honda CB Shine'
 'Bajaj Discover 100' 'Suzuki Access 125' 'TVS Wego' 'Honda CB twister'
 'Hero Glamour' 'Hero Super Splendor' 'Bajaj Discover 125' 'Hero Hunk'
 'Hero  Ignitor Disc' 'Hero  CBZ Xtreme' 'Bajaj  ct 100' 'i20' 'grand i1
0'
 'i10' 'eon' 'xcent' 'elantra' 'creta' 'verna' 'city' 'brio' 'amaze'
 'jazz']
```

In [12]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(15, 10))
sns.histplot(data['Selling_Price'], kde=True)
plt.xlabel('Selling Price', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.title('Distribution of Selling Price', fontsize=16)
plt.show()
```



In [17]:

```
print(numeric_data.corr())
```

```
                Year  Selling_Price  Present_Price  Driven_kms     Own
er
Year        1.000000       0.236141      -0.047192   -0.524342 -0.1821
04
Selling_Price  0.236141       1.000000       0.878914    0.029187 -0.0883
44
Present_Price -0.047192       0.878914       1.000000    0.203618  0.0080
58
Driven_kms    -0.524342       0.029187       0.203618    1.000000  0.0892
16
Owner         -0.182104      -0.088344       0.008058    0.089216  1.0000
00
```

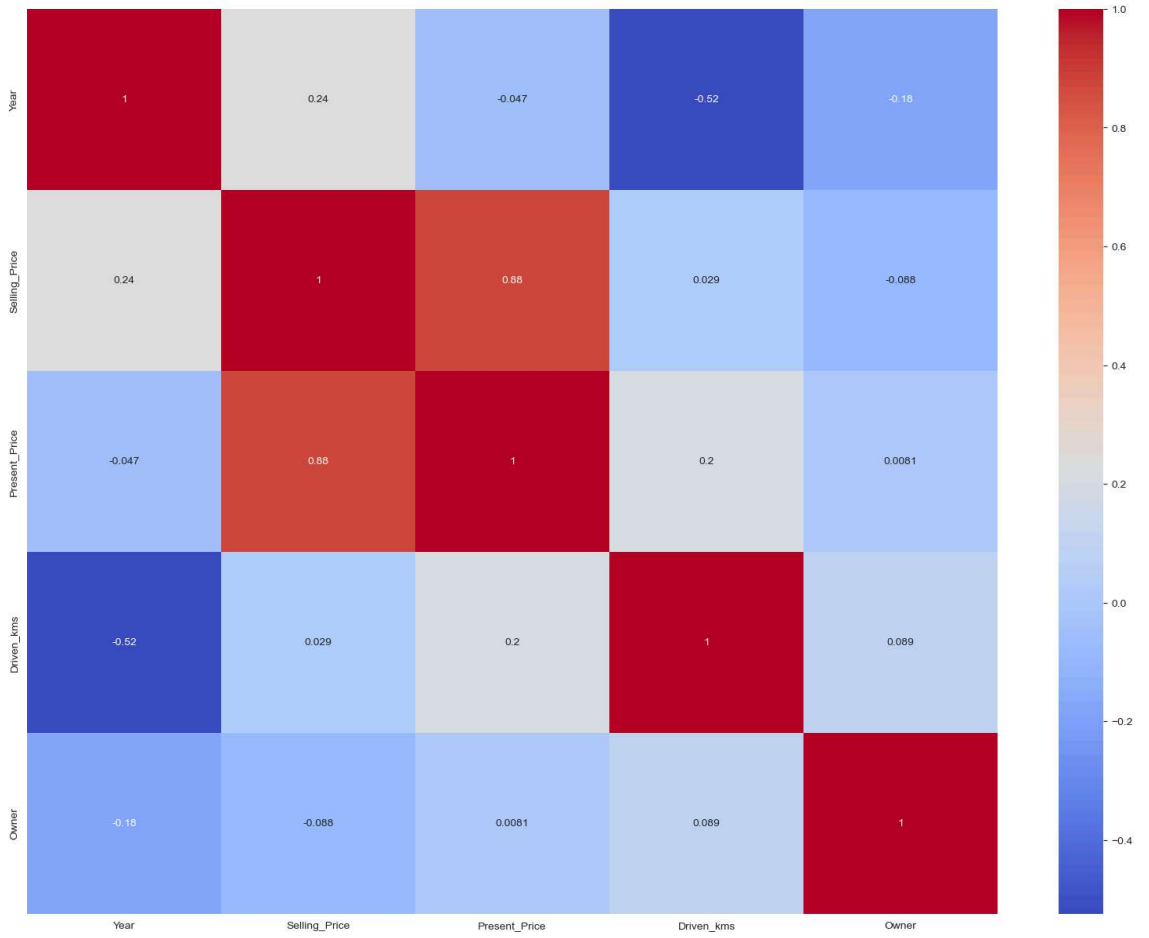
In [24]:

```
plt.figure(figsize=(20, 15))

# Calculate the correlation matrix
correlations = numeric_data.corr()

# Create the heatmap
sns.heatmap(correlations, cmap="coolwarm", annot=True)

# Display the plot
plt.show()
```



Training a Car Price Prediction Model

In [41]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error

# Define the target variable
predict = "Selling_Price"

# Select relevant columns based on actual DataFrame columns
data = data[["Year", "Selling_Price", "Present_Price", "Driven_kms",
             "Owner", "Fuel_Type_Diesel", "Fuel_Type_Petrol",
             "Selling_type_Individual", "Transmission_Manual"]]

# Prepare features and target variable
x = data.drop([predict], axis=1)
y = data[predict]

# Splitting the data
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random

# Train the model
model = DecisionTreeRegressor()
model.fit(xtrain, ytrain)

# Make predictions
predictions = model.predict(xtest)

# Evaluate the model
mae = mean_absolute_error(ytest, predictions)
score = model.score(xtest, ytest)

print(f"Mean Absolute Error: {mae}")
print(f"R^2 Score: {score}")
```

```
Mean Absolute Error: 0.7181967213114755
R^2 Score: 0.9457356302676496
```

In [ ]: