Name: Satwik Shirpurwar

Roll No: 52

Batch: A3

Subject: Machine Learning Lab

Sec: A

Experiment No-2

In the current automotive market, both buyers and sellers face challenges in determining the fair market value of used cars. Various factors such as the car's age, make, model, mileage, condition, and other features significantly impact its resale value. To facilitate informed decision-making and enhance transparency in the used car market, there is a need for an accurate predictive model that can estimate the price of a used car based on its attributes. Design & and develop a machine learning model to predict the selling price of the car.

Perform the EDA

Apply linear regression

Apply multiple linear regression

Evaluate the performance using MAE, MSE, RMSE, R2Score

Apply hyperparameter tuning to improve performance

In [55]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
```

In [3]:
```
data = pd.read_csv("car data (1).csv")
```

In [4]:
```
data.head(7)
```

Out[4]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0
5	vitara brezza	2018	9.25	9.83	2071	Diesel	Dealer	Manual	0
6	ciaz	2015	6.75	8.12	18796	Petrol	Dealer	Manual	0

In [5]:
```
data.tail(7)
```

Out[5]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
294	amaze	2014	3.75	6.80	33019	Petrol	Dealer	Manual	0
295	city	2015	8.55	13.09	60076	Diesel	Dealer	Manual	0
296	city	2016	9.50	11.60	33988	Diesel	Dealer	Manual	0
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	Manual	0
298	city	2009	3.35	11.00	87934	Petrol	Dealer	Manual	0
299	city	2017	11.50	12.50	9000	Diesel	Dealer	Manual	0
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	Manual	0

In [6]:
```
data.info
```

Out[6]:
```
<bound method DataFrame.info of     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
..       ...   ...            ...            ...         ...       ...
296     city  2016           9.50          11.60       33988    Diesel
297     brio  2015           4.00           5.90       60000    Petrol
298     city  2009           3.35          11.00       87934    Petrol
299     city  2017          11.50          12.50        9000    Diesel
300     brio  2016           5.30           5.90        5464    Petrol

    Selling_type Transmission  Owner
0         Dealer       Manual      0
1         Dealer       Manual      0
2         Dealer       Manual      0
3         Dealer       Manual      0
4         Dealer       Manual      0
..           ...          ...    ...
296       Dealer       Manual      0
297       Dealer       Manual      0
298       Dealer       Manual      0
299       Dealer       Manual      0
300       Dealer       Manual      0

[301 rows x 9 columns]>
```

In [7]:
```
data.shape
```

Out[7]: (301, 9)

In [8]:
```
data.describe()
```

Out[8]:

	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

In [9]:
```
data[data.duplicated()]
```

Out[9]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
17	ertiga	2016	7.75	10.79	43000	Diesel	Dealer	Manual	0
93	fortuner	2015	23.00	30.61	40000	Diesel	Dealer	Automatic	0

In [11]:
```
data.drop_duplicates(inplace = True)
```

In [12]:
```
data[data.duplicated()]
```

Out[12]:

Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
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 [14]:
```
data.duplicated().sum()
```

Out[14]: 0

In [15]:
```
data.reset_index(drop = True , inplace = True)
```

In [16]:
```
data.shape
```

Out[16]: (299, 9)

In [17]:
```
data
```

Out[17]:

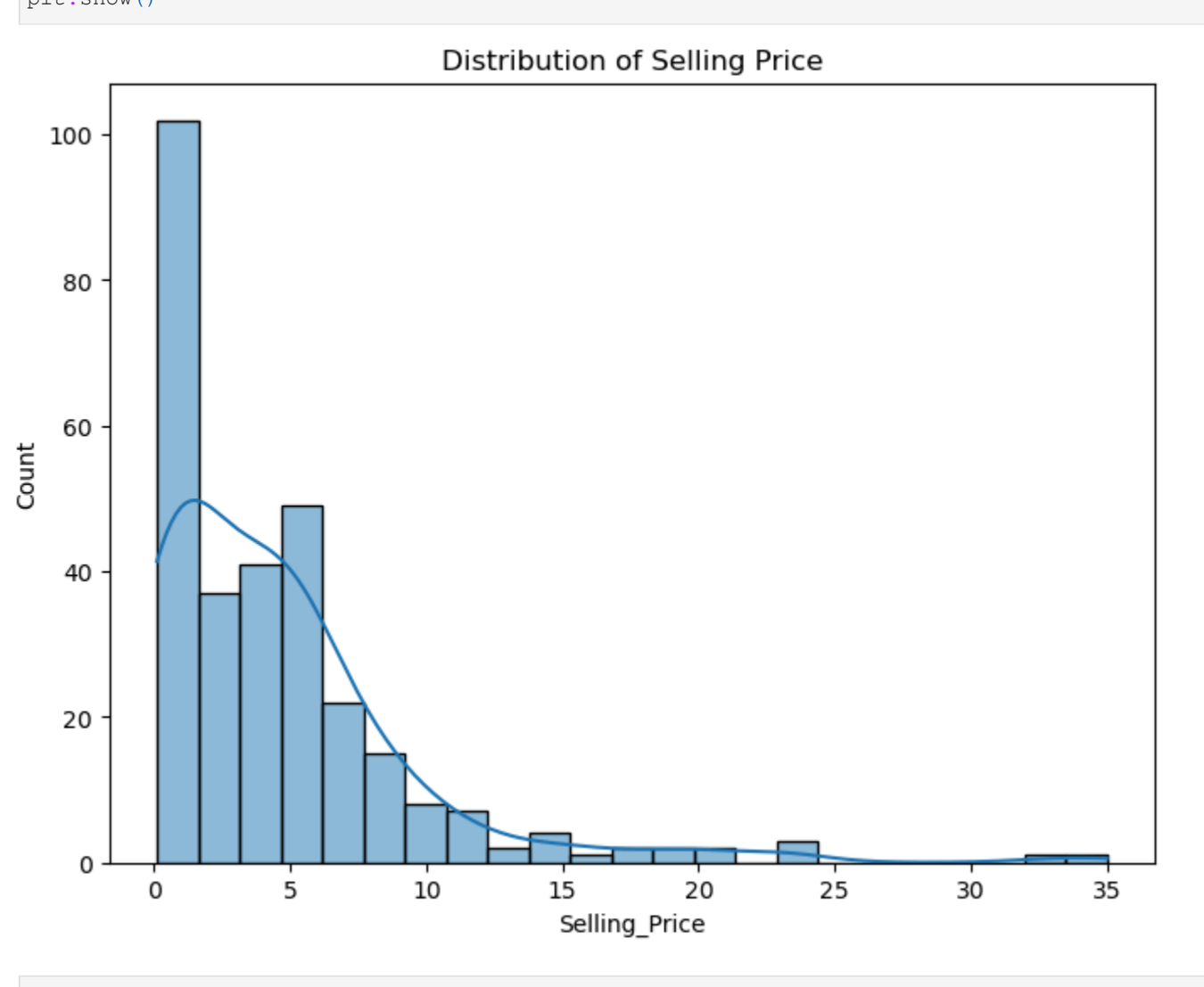
	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0
...	...	...	...	...	...	...	...	...	...
294	city	2016	9.50	11.60	33988	Diesel	Dealer	Manual	0
295	brio	2015	4.00	5.90	60000	Petrol	Dealer	Manual	0
296	city	2009	3.35	11.00	87934	Petrol	Dealer	Manual	0
297	city	2017	11.50	12.50	9000	Diesel	Dealer	Manual	0
298	brio	2016	5.30	5.90	5464	Petrol	Dealer	Manual	0

299 rows × 9 columns

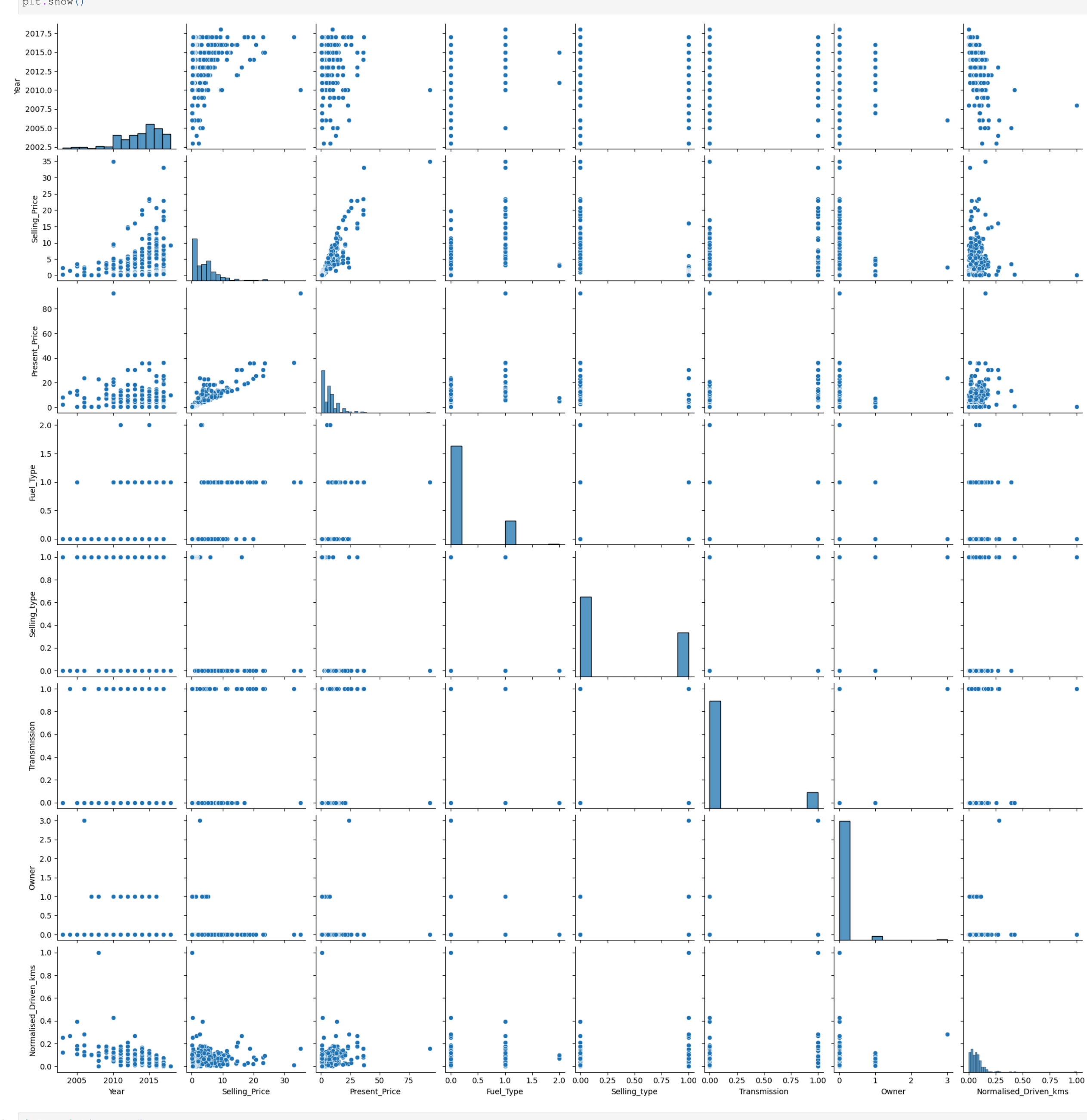
In [18]:
```
data["Car_Name"].unique()
```

Out[18]:
```
array(['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 Jupyter',
       '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 i10', 'i10',
       'eon', 'xcent', 'elantra', 'creta', 'verna', 'city', 'brio',
       'amaze', 'jazz'], dtype=object)
```

In [51]:
```
# EDA
# Checking the distribution of the target variable (Selling Price)
plt.figure(figsize=(8, 6))
sns.histplot(data['Selling_Price'], kde=True)
plt.title('Distribution of Selling Price')
plt.show()
```

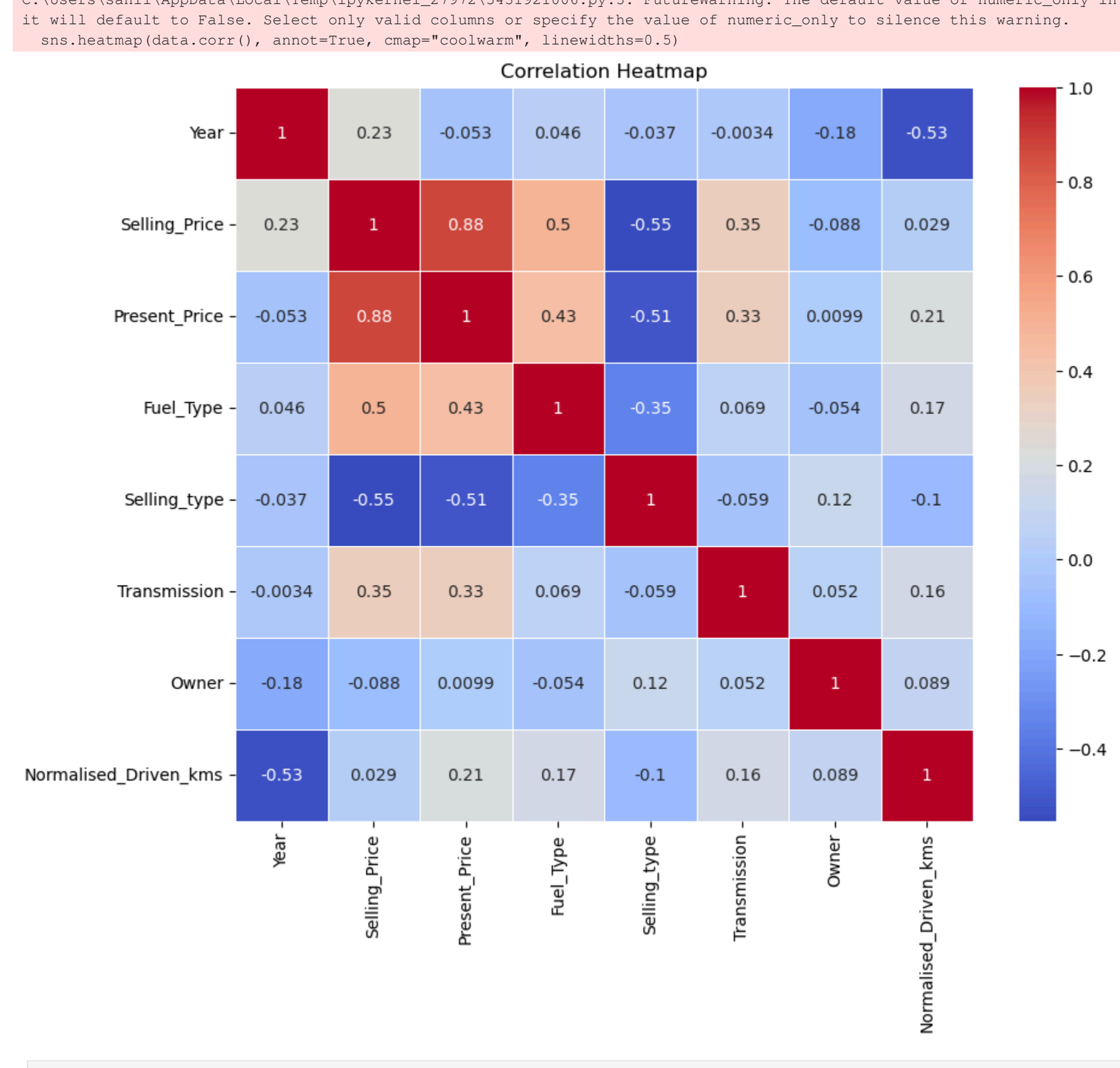


In [52]:
```
# Pairplot to understand relationships between features
sns.pairplot(data)
plt.show()
```



In [53]:
```
# Correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```

```
C:\Users\sahil\AppData\Local\Temp\ipykernel_27972\3431921006.py:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(data.corr(), annot=True, cmap="coolwarm", linewidths=0.5)
```



In [19]:
```
data["Fuel_Type"].unique()
```

Out[19]: array(['Petrol', 'Diesel', 'CNG'], dtype=object)

In [20]:
```
data["Fuel_Type"] = data["Fuel_Type"].replace("Petrol", 0)
data["Fuel_Type"] = data["Fuel_Type"].replace("Diesel", 1)
data["Fuel_Type"] = data["Fuel_Type"].replace("CNG", 2)
```

In [21]:
```
data["Selling_type"].unique()
```

Out[21]: array(['Dealer', 'Individual'], dtype=object)

In [23]:
```
data["Selling_type"] = data["Selling_type"].replace("Dealer",0)
data["Selling_type"] = data["Selling_type"].replace("Individual",1)
```

In [24]:
```
data["Transmission"].unique()
```

Out[24]: array(['Manual', 'Automatic'], dtype=object)

In [25]:
```
data["Transmission"] = data["Transmission"].replace("Manual",0)
data["Transmission"] = data["Transmission"].replace("Automatic",1)
```

In [26]:
```
data
```

Out[26]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	0	0	0	0
1	sx4	2013	4.75	9.54	43000	1	0	0	0
2	ciaz	2017	7.25	9.85	6900	0	0	0	0
3	wagon r	2011	2.85	4.15	5200	0	0	0	0
4	swift	2014	4.60	6.87	42450	1	0	0	0
...	...	...	...	...	...	...	...	...	...
294	city	2016	9.50	11.60	33988	1	0	0	0
295	brio	2015	4.00	5.90	60000	0	0	0	0
296	city	2009	3.35	11.00	87934	0	0	0	0
297	city	2017	11.50	12.50	9000	1	0	0	0
298	brio	2016	5.30	5.90	5464	0	0	0	0

299 rows × 9 columns

In [27]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       299 non-null    object
 1   Year           299 non-null    int64
 2   Selling_Price  299 non-null    float64
 3   Present_Price  299 non-null    float64
 4   Driven_kms     299 non-null    int64
 5   Fuel_Type      299 non-null    int64
 6   Selling_type   299 non-null    int64
 7   Transmission   299 non-null    int64
 8   Owner          299 non-null    int64
dtypes: float64(2), int64(6), object(1)
memory usage: 21.2+ KB
```

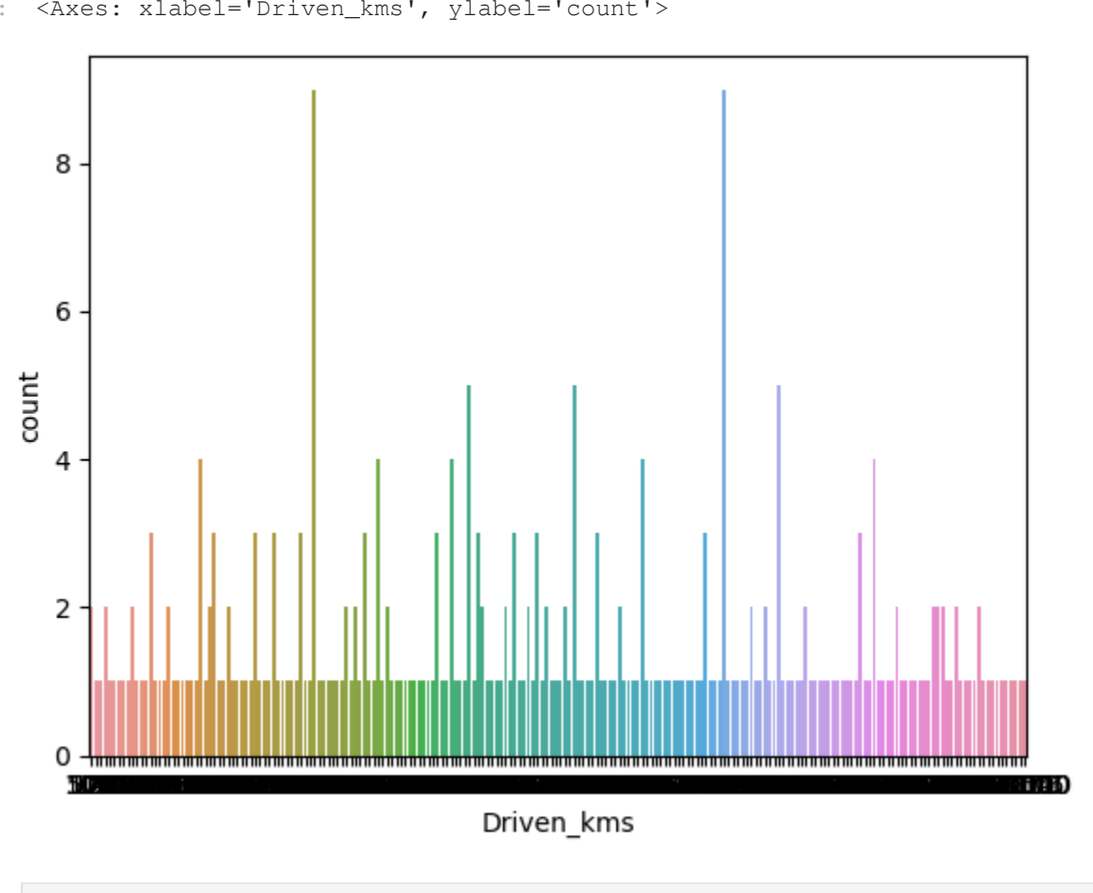
In [28]:
```
data.describe()
```

Out[28]:

	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000
mean	2013.615385	4.589632	7.541037	36916.752508	0.207358	0.354515	0.130435	0.043478
std	2.896868	4.984240	8.566332	39015.170352	0.422297	0.479168	0.337346	0.248720
min	2003.000000	0.100000	0.320000	500.000000	0.000000	0.000000	0.000000	0.000000
25%	2012.000000	0.850000	1.200000	15000.000000	0.000000	0.000000	0.000000	0.000000
50%	2014.000000	3.510000	6.100000	32000.000000	0.000000	0.000000	0.000000	0.000000
75%	2016.000000	6.000000	9.840000	48883.500000	0.000000	1.000000	0.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	2.000000	1.000000	1.000000	3.000000

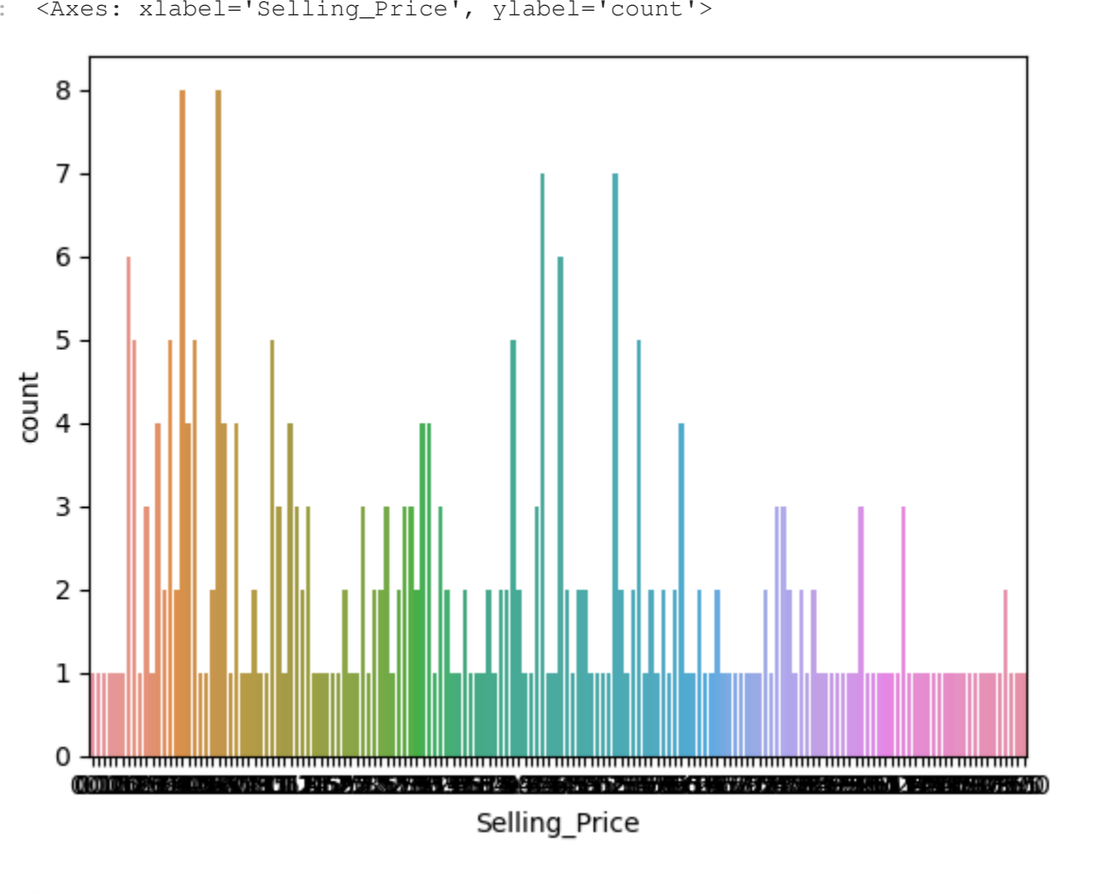
In [29]:
```
sns.countplot(x = data["Driven_kms"])
```

Out[29]: <Axes: xlabel='Driven_kms', ylabel='count'>



In [30]:
```
sns.countplot(x = data["Selling_Price"])
```

Out[30]: <Axes: xlabel='Selling_Price', ylabel='count'>



Scaling

In [31]:
```
data[["Driven_kms"]]
```

Out[31]:

	Driven_kms
0	27000
1	43000
2	6900
3	5200
4	42450
...	...
294	33988
295	60000
296	87934
297	9000
298	5464

299 rows × 1 columns

In [32]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [33]:
```
scaler = MinMaxScaler()
data["Normalised_Driven_kms"] = scaler.fit_transform(data[["Driven_kms"]])
print(data)
```

```
    Car_Name  Year  Selling_Price  Present_Price  Driven_kms  Fuel_Type  \
0       ritz  2014           3.35           5.59       27000          0
1        sx4  2013           4.75           9.54       43000          1
2       ciaz  2017           7.25           9.85        6900          0
3    wagon r  2011           2.85           4.15        5200          0
4      swift  2014           4.60           6.87       42450          1
..       ...   ...            ...            ...         ...        ...
294     city  2016           9.50          11.60       33988          1
295     brio  2015           4.00           5.90       60000          0
296     city  2009           3.35          11.00       87934          0
297     city  2017          11.50          12.50        9000          1
298     brio  2016           5.30           5.90        5464          0

     Selling_type  Transmission  Owner  Normalised_Driven_kms
0               0             0      0               0.053053
1               0             0      0               0.085085
2               0             0      0               0.012813
3               0             0      0               0.009409
4               0             0      0               0.083984
..            ...           ...    ...                    ...
294             0             0      0               0.067043
295             0             0      0               0.119119
296             0             0      0               0.175043
297             0             0      0               0.017017
298             0             0      0               0.009938

[299 rows x 10 columns]
```

In [34]:
```
data = data.drop(columns = ["Driven_kms"])
```

In [35]:
```
data
```

Out[35]:

	Car_Name	Year	Selling_Price	Present_Price	Fuel_Type	Selling_type	Transmission	Owner	Normalised_Driven_kms
0	ritz	2014	3.35	5.59	0	0	0	0	0.053053
1	sx4	2013	4.75	9.54	1	0	0	0	0.085085
2	ciaz	2017	7.25	9.85	0	0	0	0	0.012813
3	wagon r	2011	2.85	4.15	0	0	0	0	0.009409
4	swift	2014	4.60	6.87	1	0	0	0	0.083984
...	...	...	...	...	...	...	...	...	...
294	city	2016	9.50	11.60	1	0	0	0	0.067043
295	brio	2015	4.00	5.90	0	0	0	0	0.119119
296	city	2009	3.35	11.00	0	0	0	0	0.175043
297	city	2017	11.50	12.50	1	0	0	0	0.017017
298	brio	2016	5.30	5.90	0	0	0	0	0.009938

299 rows × 9 columns

In [36]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

multi-linear regression

In [37]:
```
dependent = ['Selling_Price']
Independent = ['Present_Price' , 'Normalised_Driven_kms']
X = data[dependent] # Independent variable(s)
y = data[Independent] # Dependent variable
```

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [39]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[39]:
```
▾ LinearRegression
LinearRegression()
```

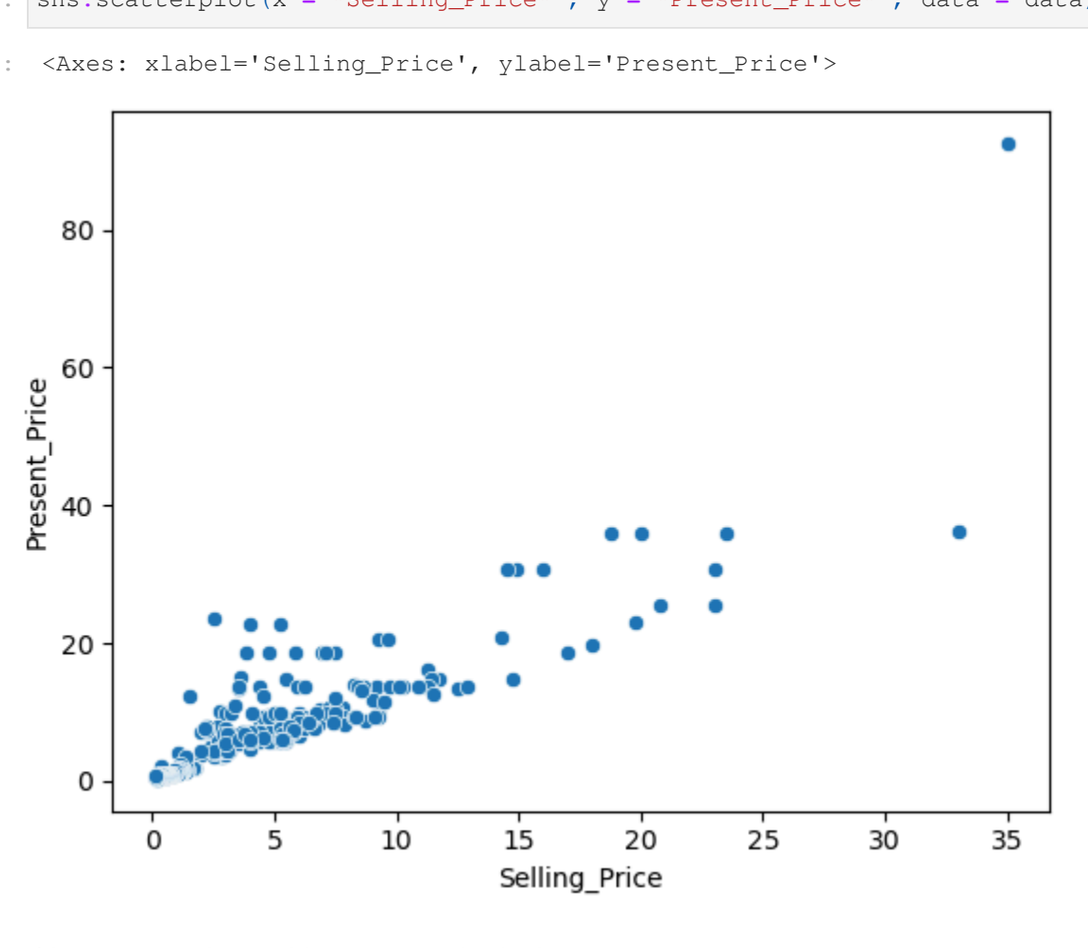
In [40]:
```
y_pred = model.predict(X_test)
```

In [41]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

```
Mean Squared Error: 13.975274131230988
R^2 Score: 0.21612672531342503
```

In [42]:
```
sns.scatterplot(x = "Selling_Price" , y = "Present_Price" , data = data)
```

Out[42]: <Axes: xlabel='Selling_Price', ylabel='Present_Price'>



Linear regression and hyperparameter tuning

In [43]:
```
X = data[['Year', 'Present_Price', 'Fuel_Type', 'Transmission', 'Owner']]
y = data['Selling_Price']
```

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [45]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt
```

In [46]:
```
preprocessor = ColumnTransformer(
transformers=[('cat', OneHotEncoder(drop='first'), ['Fuel_Type', 'Transmission'])],
remainder='passthrough'
)
```

In [47]:
```
model = make_pipeline(preprocessor, LinearRegression())
model.fit(X_train, y_train)
```

Out[47]:
```
▸ Pipeline
  ▸ columntransformer: ColumnTransformer
      ▸ cat               ▸ remainder
      ▸ OneHotEncoder     ▸ passthrough
  ▸ LinearRegression
```

In [48]:
```
y_pred = model.predict(X_test)
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)
```

In [49]:
```
print(f'Training R^2 Score: {train_score}')
print(f'Test R^2 Score: {test_score}')
```

```
Training R^2 Score: 0.892242526706131
Test R^2 Score: 0.7453070498364994
```

In [50]:
```
plt.scatter(y_test, y_test - y_pred)
plt.hlines(y=0, xmin=y_test.min(), xmax=y_test.max(), colors='red')
plt.xlabel('Actual Selling Price')
plt.ylabel('Residuals (Actual - Predicted)')
plt.title('Residual Plot')
plt.show()
```