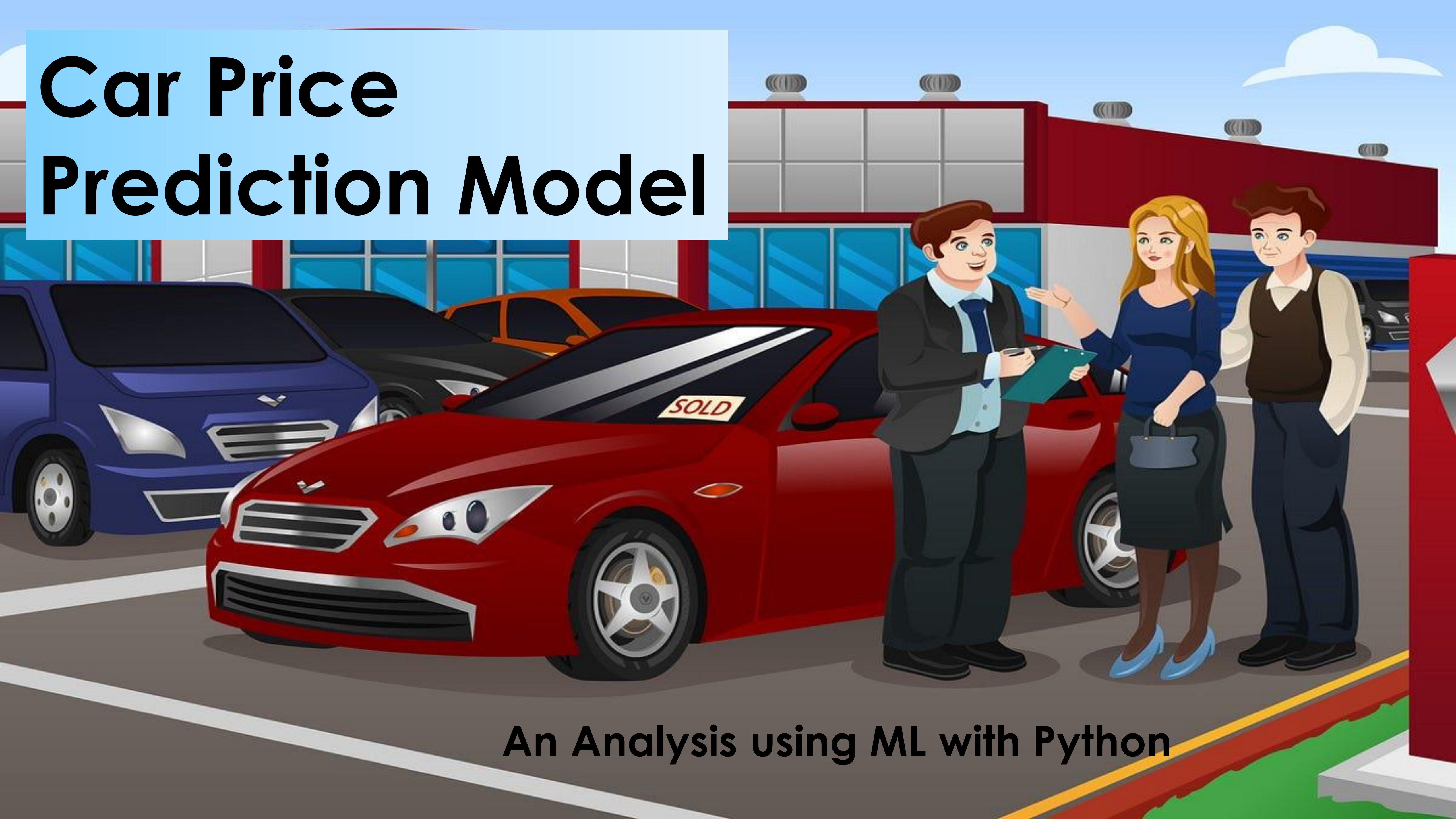
Car Price Prediction Model



An Analysis using ML with Python

OVERVIEW

This project is all about leveraging supervised machine learning to predict car prices, helping businesses and consumers make data-driven decisions.

Project Highlights:

Data Preprocessing: Examined complex datasets by managing missing values, encoding categorical variables, and scaling features like Label Encoder for accurate predictions. Trained and tested data using train test split Get the data ready for further evaluation.

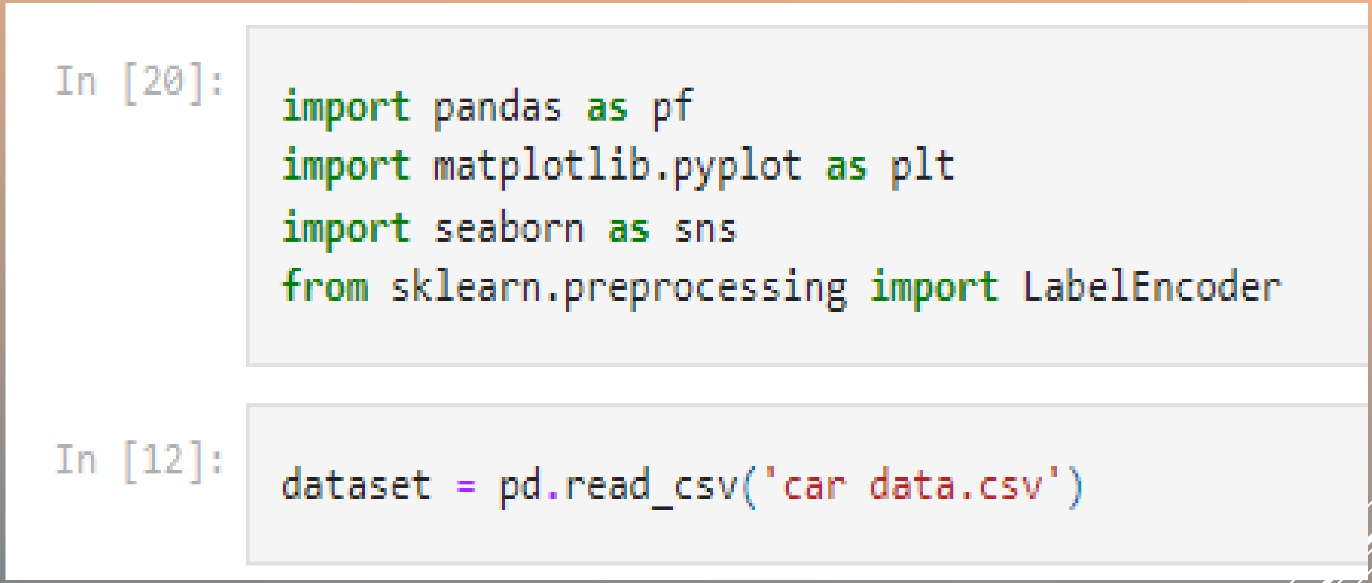
Building: Dived into Random Forest Regressor to create a neural network model that predicts car prices with precision. Explore different scikit-learn algorithms and fine-tune the model for optimal performance.

Model Evaluation: Assess the model's performance with key metrics like rf.fit and rf.score, and checked errors using mean squared error and mean absolute error to ensure accuracy

Feature Analysis: Understand the influence of various features on car prices, helping to uncover the factors that drive market value.

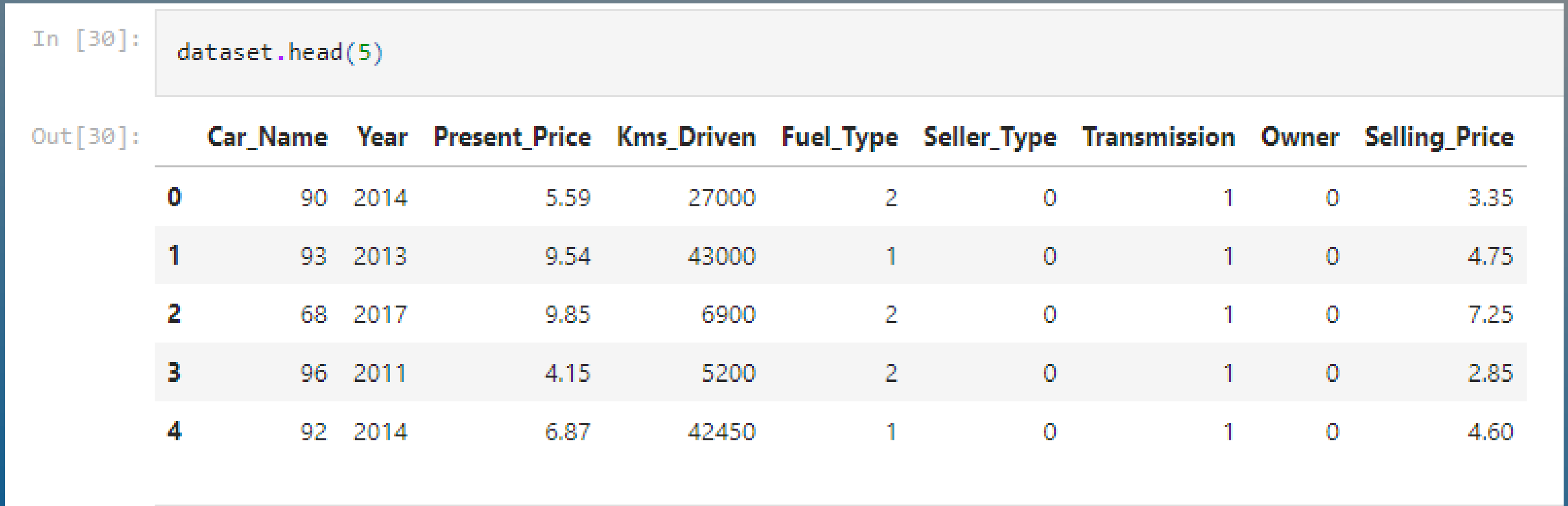
DATA EXPLORATION

Importing required libraries



```
In [20]:  import pandas as pf
          import matplotlib.pyplot as plt
          import seaborn as sns
          from sklearn.preprocessing import LabelEncoder
```

```
In [12]:  dataset = pd.read_csv('car data.csv')
```



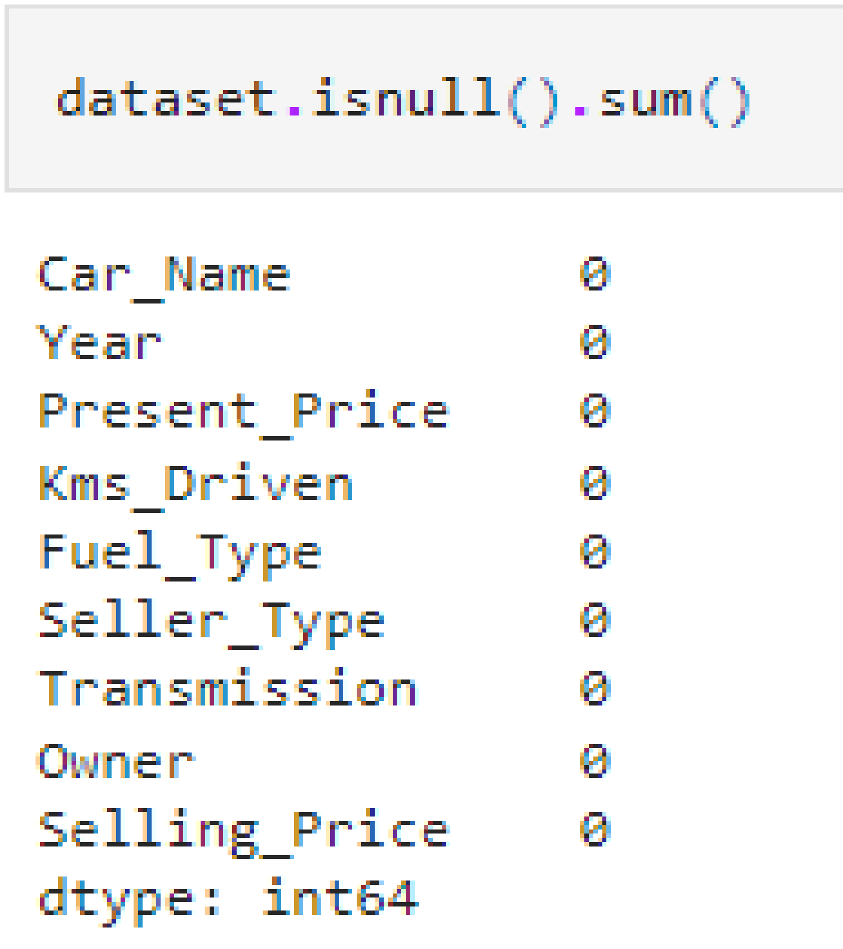
```
In [30]:  dataset.head(5)
```

Out[30]:

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Selling_Price
0	90	2014	5.59	27000	2	0	1	0	3.35
1	93	2013	9.54	43000	1	0	1	0	4.75
2	68	2017	9.85	6900	2	0	1	0	7.25
3	96	2011	4.15	5200	2	0	1	0	2.85
4	92	2014	6.87	42450	1	0	1	0	4.60

Dataset to be analyzed

DATA EXPLORATION

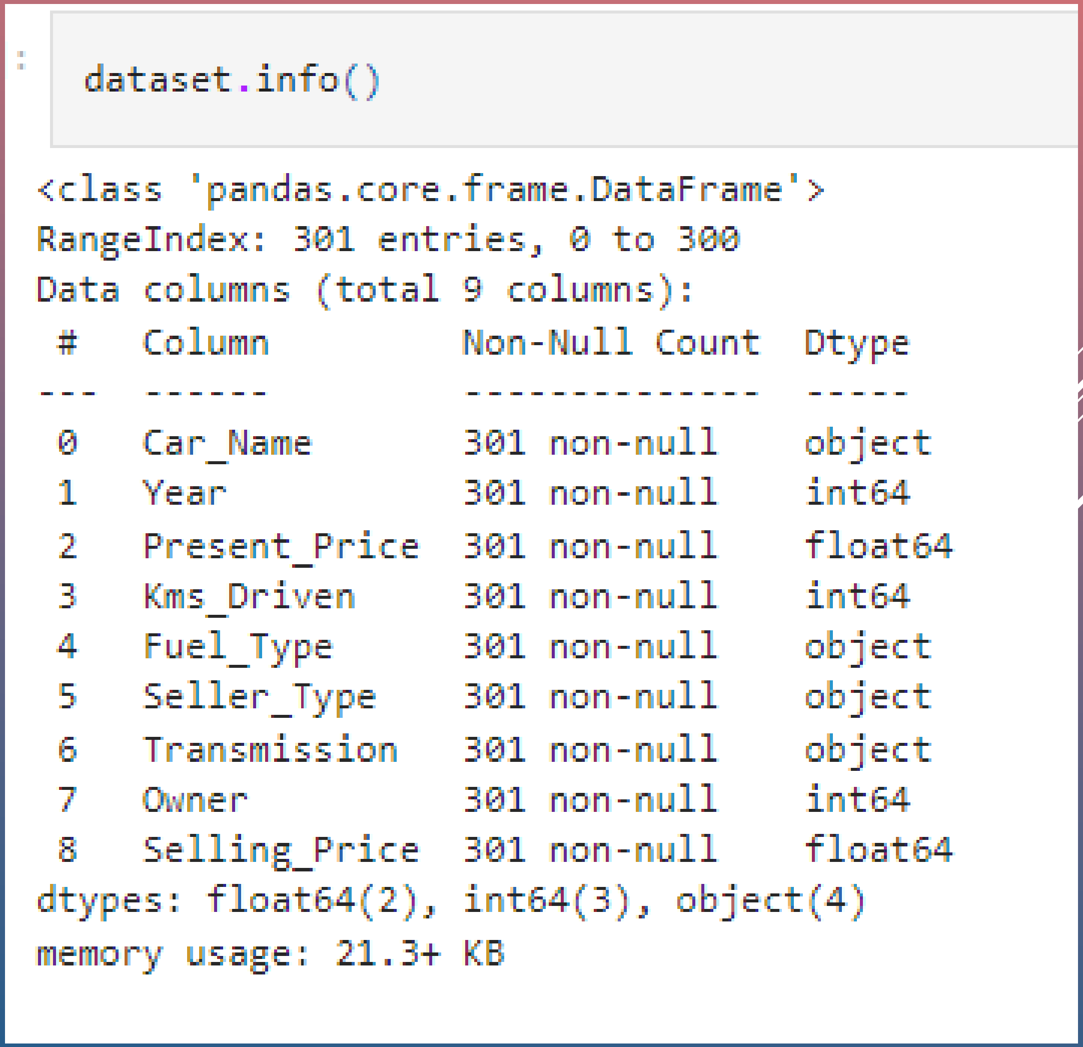


```
dataset.isnull().sum()
```

```
Car_Name         0
Year             0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
Selling_Price    0
dtype: int64
```

There are not null values.

Datatype of each column of dataset.



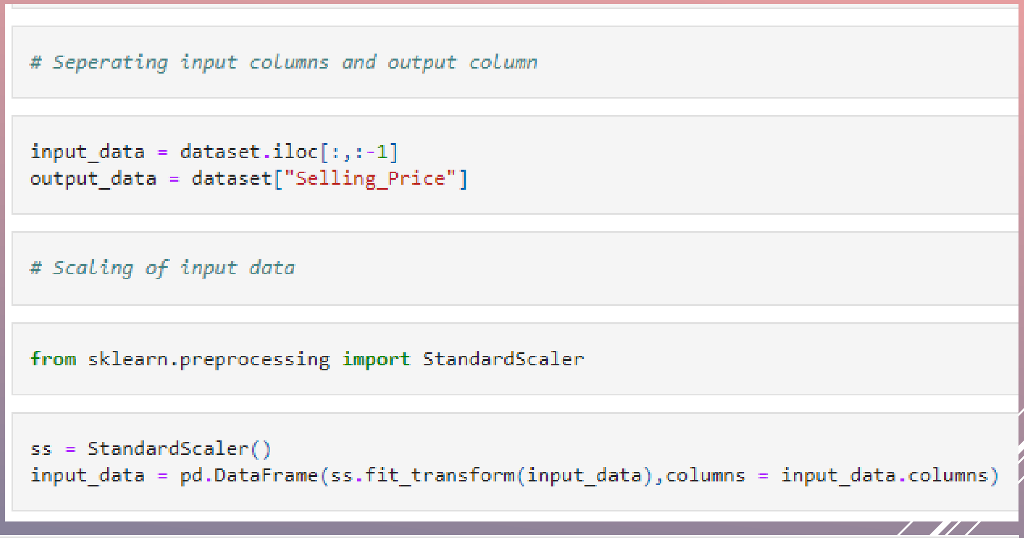
```
dataset.info()
```

```
<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   Present_Price  301 non-null    float64
 3   Kms_Driven     301 non-null    int64  
 4   Fuel_Type      301 non-null    object 
 5   Seller_Type    301 non-null    object 
 6   Transmission   301 non-null    object 
 7   Owner          301 non-null    int64  
 8   Selling_Price  301 non-null    float64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

Changed the datatype of columns like Car Name, Fuel Type, Seller Type, and Transmission from object to integer using Label Encoder and Fir transform.

Separated Input columns and Output columns.

Performed Scaling of Input Data by importing Standard Scaler.



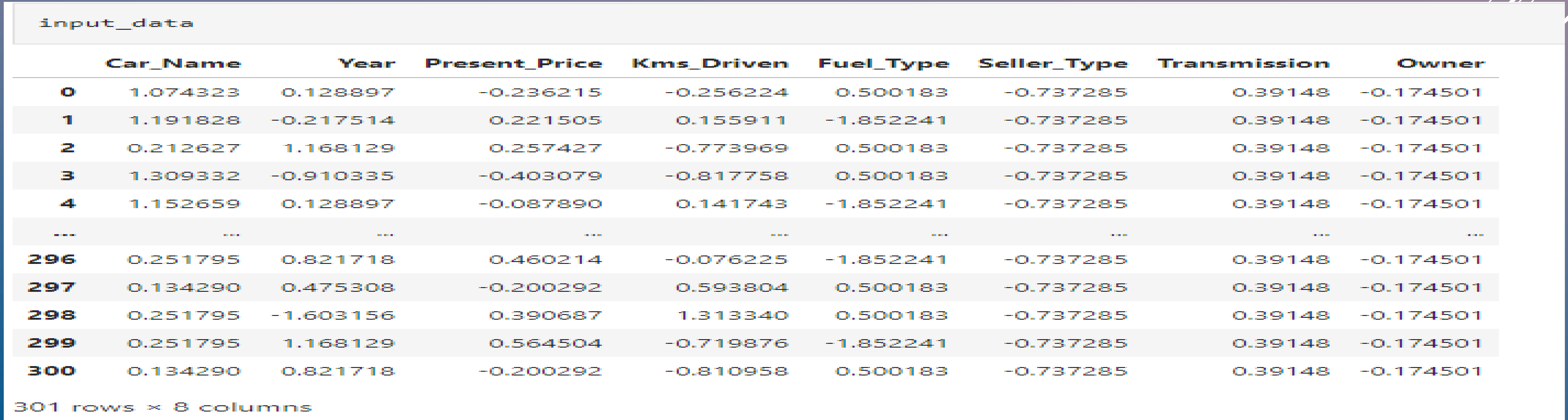
```
# Seperating input columns and output column
```

```
input_data = dataset.iloc[:,:-1]
output_data = dataset["Selling_Price"]
```

```
# Scaling of input data
```

```
from sklearn.preprocessing import StandardScaler
```

```
ss = StandardScaler()
input_data = pd.DataFrame(ss.fit_transform(input_data),columns = input_data.columns)
```

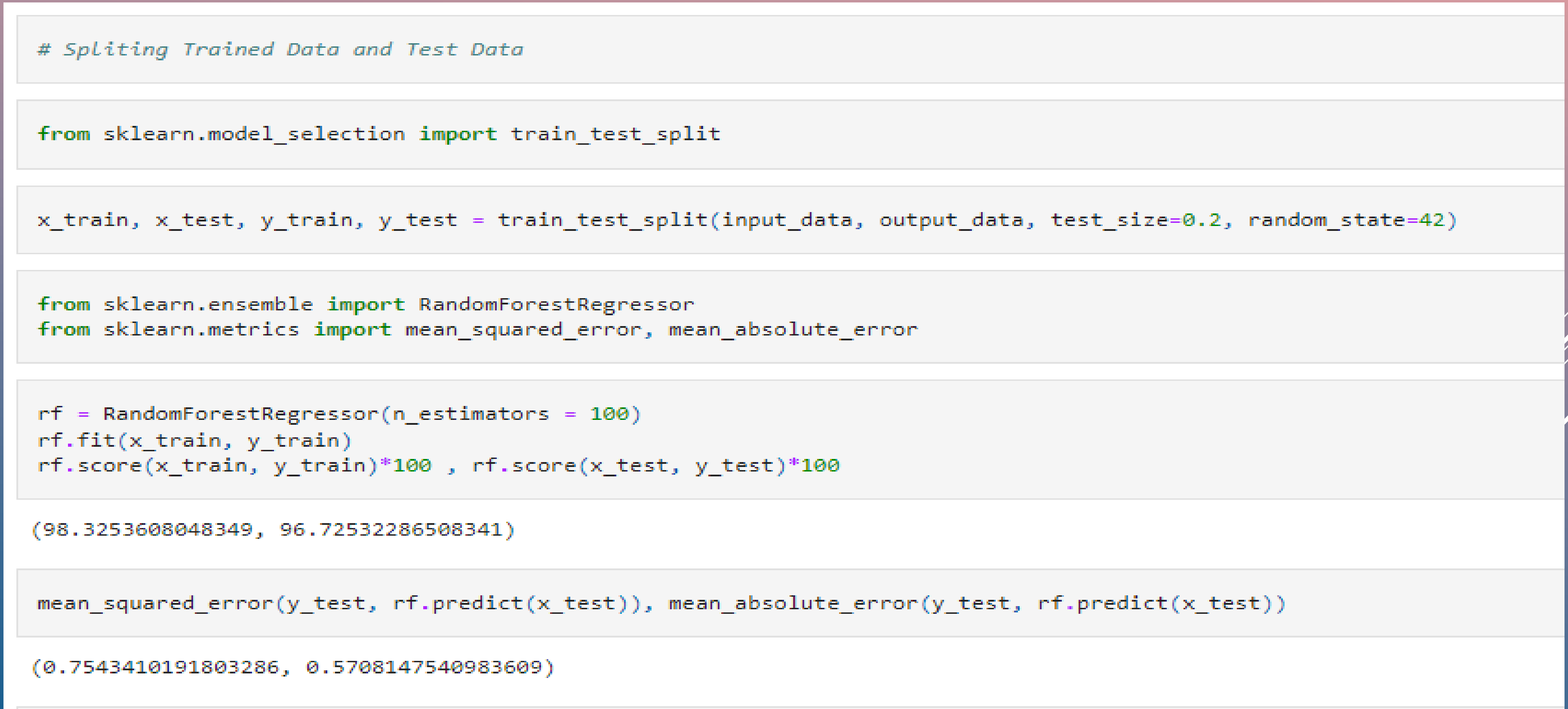


```
input_data
```

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	1.074323	0.128897	-0.236215	-0.256224	0.500183	-0.737285	0.39148	-0.174501
1	1.191828	-0.217514	0.221505	0.155911	-1.852241	-0.737285	0.39148	-0.174501
2	0.212627	1.168129	0.257427	-0.773969	0.500183	-0.737285	0.39148	-0.174501
3	1.309332	-0.910335	-0.403079	-0.817758	0.500183	-0.737285	0.39148	-0.174501
4	1.152659	0.128897	-0.087890	0.141743	-1.852241	-0.737285	0.39148	-0.174501
...	...	...	...	...	...	...	...	...
296	0.251795	0.821718	0.460214	-0.076225	-1.852241	-0.737285	0.39148	-0.174501
297	0.134290	0.475308	-0.200292	0.593804	0.500183	-0.737285	0.39148	-0.174501
298	0.251795	-1.603156	0.390687	1.313340	0.500183	-0.737285	0.39148	-0.174501
299	0.251795	1.168129	0.564504	-0.719876	-1.852241	-0.737285	0.39148	-0.174501
300	0.134290	0.821718	-0.200292	-0.810958	0.500183	-0.737285	0.39148	-0.174501

301 rows × 8 columns

Splitting Data and Checking Errors For Accurate Predictions.



```
# Spliting Trained Data and Test Data
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(input_data, output_data, test_size=0.2, random_state=42)
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

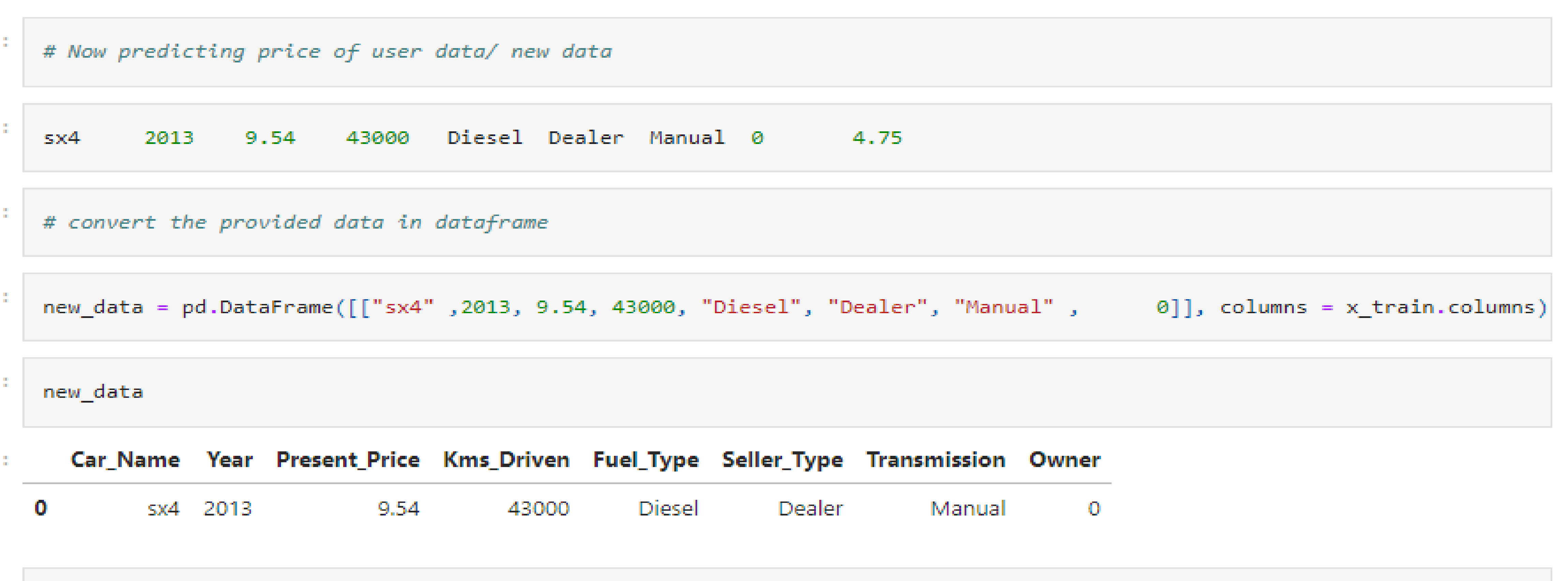
```
rf = RandomForestRegressor(n_estimators = 100)
rf.fit(x_train, y_train)
rf.score(x_train, y_train)*100 , rf.score(x_test, y_test)*100
```

```
(98.3253608048349, 96.72532286508341)
```

```
mean_squared_error(y_test, rf.predict(x_test)), mean_absolute_error(y_test, rf.predict(x_test))
```

```
(0.7543410191803286, 0.5708147540983609)
```

Prediction Model



```
# Now predicting price of user data/ new data
```

```
sx4      2013     9.54     43000    Diesel   Dealer   Manual   0        4.75
```

```
# convert the provided data in dataframe
```

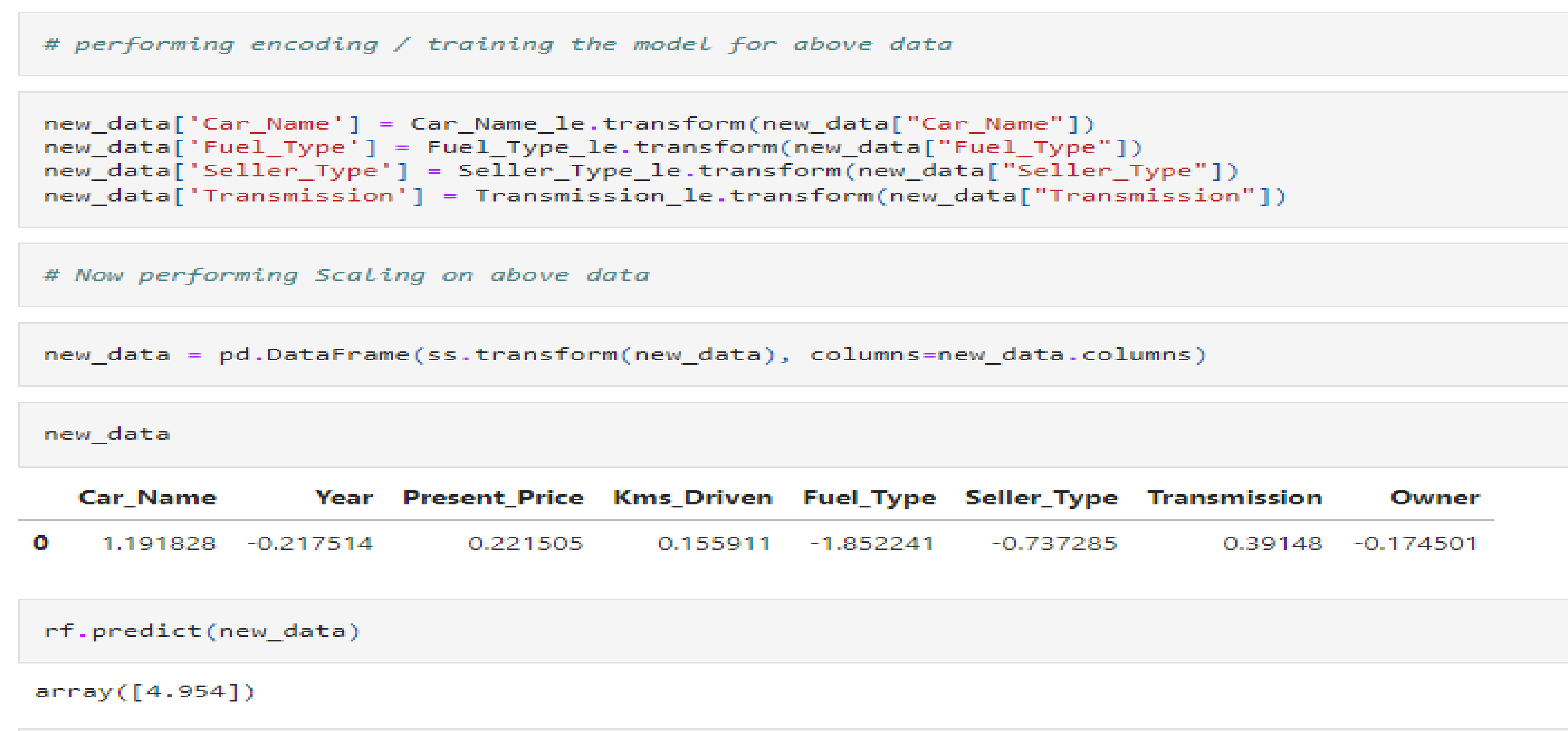
```
new_data = pd.DataFrame([["sx4" ,2013, 9.54, 43000, "Diesel", "Dealer", "Manual" ,      0]], columns = x_train.columns)
```

```
new_data
```

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	sx4	2013	9.54	43000	Diesel	Dealer	Manual	0

Converted the user data into Data Frame using Pandas.

Prediction Model



```
# performing encoding / training the model for above data
```

```
new_data['Car_Name'] = Car_Name_le.transform(new_data["Car_Name"])
new_data['Fuel_Type'] = Fuel_Type_le.transform(new_data["Fuel_Type"])
new_data['Seller_Type'] = Seller_Type_le.transform(new_data["Seller_Type"])
new_data['Transmission'] = Transmission_le.transform(new_data["Transmission"])
```

```
# Now performing Scaling on above data
```

```
new_data = pd.DataFrame(ss.transform(new_data), columns=new_data.columns)
```

```
new_data
```

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	1.191828	-0.217514	0.221505	0.155911	-1.852241	-0.737285	0.39148	-0.174501

```
rf.predict(new_data)
```

```
array([4.954])
```

Predicting price of user data using rf.predict()