Predicting the price ofused cars

Introduction

Buying and selling used cars can be challenging for customers and automobile industry professionals due to the variation in veichels prices in the market. However, When the key factors influencing Used car's price such as, brand, model, milage etc.. are properly analyzed a clear and accurate prediction can be made. by applying data driven analysis methods this challenge can be reduced, leads to fair pricing and better decision making in the used car market.

to address this challenge, this study aims to develop data deriven analysis for predicting used car prices using the key variables that influence the price, the varibales tha we used are brand,model, model-year, milage, fuel type, engine, transmission, external and internal color, number of accident,and clearity of car title. The dataset used is “Used Car Price Prediction Dataset” obtained from kaggle which is originally extracted Cars.com. By analyzing the significant features that affect vehicle pricing, this project intends to build a predictive model that provides more accurate and reliable price estimates for used cars.

Problem Statement

The pricing of used cars is often inconsistent and subjective due to the wide range of factors that influence a vehicle’s market value. Buyers risk overpaying, while sellers may undervalue their cars because price estimation typically depends on personal judgment, incomplete information, or manual comparison across listings. The absence of a reliable, data-driven tool for determining fair used-car prices leads to market inefficiencies and poor decision-making.

Therefore, there is a need for a predictive machine-learning model that can estimate used-car prices accurately using key vehicle attributes.

General Objective:

- my objective is to develop predicting model that accurately predicts the price of used cars based on key vehicle features.

Specific Objectives

1. To identify and analyze the variables that significantly influence used-car prices.
2. To clean, preprocess, and prepare the dataset for modeling.

3. To build and evaluate LinearRegression model for price prediction.
4. To provide insights and recommendations based on the model results.

Research Questions

1. Which vehicle features have the strongest impact on used-car prices?
2. How accurately can our models predict used-car prices? 3.How can predicted prices improve decision-making for buyers and sellers?

Dataset Description

The dataset used in this project is the “Used Car Price Prediction Dataset” from Kaggle, originally sourced from Cars.com. It contains several key attributes of used cars, including price, brand, model, model year, mileage, fuel type, engine specifications, transmission type, interior and exterior color, number of accidents, and title status. The dataset includes a mix of numerical and categorical variables, and it provides enough information to perform meaningful analysis and build a price prediction model.

Methodology

The dataset was loaded into Python using Pandas, and initial data cleaning steps were applied. These included handling missing values, correcting inconsistent entries, and standardizing categorical values such as transmission and engine type. Exploratory Data Analysis (EDA) was then performed to understand the distribution of features and their relationships with car prices. Categorical variables were encoded using one-hot encoding, and numerical variables were normalized where necessary.

After preprocessing, a Linear Regression model was trained using the cleaned dataset. The model learned the relationship between vehicle features and price. The performance of the model was evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R^2 score to determine how accurately it predicts used-car prices.

1 Implementation

- the code is implemented in jupyter lab

1.1 Importing Libraries

In [90]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

1.2 Loading dataset

In [64]:
```
df = pd.read_csv("C:\\Users\\hp eiltebook 1040G8\\Desktop\\used_cars.csv")
df.head()
```

Out[64]:

	brand	model	model_year	milage	fuel_type	engine	transmission	ext_co
0	Ford	Utility Police Interceptor Base	2013	51,000 mi.	E85 Flex Fuel	300.0HP 3.7L V6 Cylinder Engine Flex Fuel Capa...	6-Speed A/T	Black
1	Hyundai	Palisade SEL	2021	34,742 mi.	Gasoline	3.8L V6 24V GDI DOHC	8-Speed Automatic	Moonlight Clou
2	Lexus	RX 350 RX 350	2022	22,372 mi.	Gasoline	3.5 Liter DOHC	Automatic	Blue
3	INFINITI	Q50 Hybrid Sport	2015	88,900 mi.	Hybrid	354.0HP 3.5L V6 Cylinder Engine Gas/Electric H...	7-Speed A/T	Black
4	Audi	Q3 45 S line Premium Plus	2021	9,835 mi.	Gasoline	2.0L I4 16V GDI DOHC Turbo	8-Speed Automatic	Glacie White Metalli

1.3 Cleaning and preprocessing

In [65]:
```
df.isnull().sum()
```

Out[65]:
```
brand             0
model             0
model_year        0
milage            0
fuel_type       170
engine            0
transmission      0
ext_col           0
int_col           0
accident        113
clean_title     596
price             0
dtype: int64
```

we have null values in columns Fuel_type,accident, and clean_title. steps followed to clean it :-

- Since its not know which Fuel type it can be, the null rows in Fuel_type are droped
- Also the Accident nan can be know what value it holds so it's droped
- for clean title unique value holds yes and null so instead of null it's replace with Unknow. Here unknow mean two things No and not reported

In [66]:
```
df['clean_title'].unique()
```

Out[66]:
```
array(['Yes', nan], dtype=object)
```

In [67]:
```
df['clean_title'] = df['clean_title'].replace(np.nan,"Unknown")
clean_df = df.dropna()
```

In [68]:
```
clean_df['clean_title'].unique()
```

Out[68]:
```
array(['Yes', 'Unknown'], dtype=object)
```

In [69]:
```
clean_df['brand'].unique()
```

Out[69]:
```
array(['Ford', 'Hyundai', 'Lexus', 'INFINITI', 'Audi', 'Acura', 'BMW',
       'Land', 'Aston', 'Toyota', 'Lincoln', 'Jaguar', 'Mercedes-Benz',
       'Dodge', 'Nissan', 'Chevrolet', 'Kia', 'Jeep', 'Bentley', 'Honda',
       'MINI', 'Porsche', 'Hummer', 'Chrysler', 'Volvo', 'Cadillac',
       'Lamborghini', 'Maserati', 'Genesis', 'Volkswagen', 'GMC', 'RAM',
       'Subaru', 'Alfa', 'Ferrari', 'Scion', 'Mitsubishi', 'Mazda',
       'Saturn', 'Bugatti', 'Rolls-Royce', 'McLaren', 'Buick', 'Lotus',
       'Pontiac', 'FIAT', 'Saab', 'Mercury', 'Plymouth', 'smart',
       'Maybach', 'Suzuki'], dtype=object)
```

- for the brand column to standardize the names to proper market names

In [70]:
```
Clean_brand_name ={
    "Land": "Land Rover",
    "Aston": "Aston Martin",
    "Alfa": "Alfa Romeo",
    "INFINITI": "Infiniti",
    "FIAT": "Fiat",
    "MINI": "Mini",
    "smart": "Smart",
    "Ciechento":"cieChento"
}

clean_df.loc[:,'brand'] = clean_df['brand'].replace(Clean_brand_name)
```

In [71]:
```
clean_df.loc[:,'milage'] = clean_df['milage'].replace(r"[^0-9]","", regex = True)
clean_df['milage'] = clean_df["milage"].astype('int')
```

In [72]:
```
clean_df['fuel_type'].unique()
```

Out[72]:
```
array(['E85 Flex Fuel', 'Gasoline', 'Hybrid', 'Diesel', 'Plug-In Hybrid',
       '–', 'not supported'], dtype=object)
```

In [73]:
```
clean_df['engine'].unique()
```

Out[73]:
```
array(['300.0HP 3.7L V6 Cylinder Engine Flex Fuel Capability',
       '3.8L V6 24V GDI DOHC', '3.5 Liter DOHC', ...,
       '169.0HP 2.5L 4 Cylinder Engine Flex Fuel Capability',
       '136.0HP 1.8L 4 Cylinder Engine Gasoline Fuel',
       '270.0HP 2.0L 4 Cylinder Engine Gasoline Fuel'],
      shape=(1083,), dtype=object)
```

- columns like "engine" holds more than one informations, so we will extract some informations and create several columns which can be helpful for predicting the price

In [74]:
```
#we want to extract HorsePower,Egine size,Cylinder, and Fuel_type
def cleanig_engine_features(clean_df,col="engine"):
    # working with copy file to avoide settingwarning
    clean_df = clean_df.copy()
    #make sure we are working with string datatype
    s= clean_df[col].fillna("").astype('str')


    #extracting Horse power
    clean_df.loc[:,"HorsePower"] = s.str.extract(r'(\d+\.?\d*)\s*HP',expand = False
    # extracting EngineSize
    clean_df.loc[:,"EngineSize"] = s.str.extract(r'(\d+\.?\d*)L', expand = False).p
    #extracting cylinder type: for this we have tow formats like V6 and 4 cyliner s
    # Number follower with cylinder
    CLY_word = s.str.extract(r'(\d+)\s*(?:Cylinder|CYL)', expand = False)
    # Number with V
    CLY_V = s.str.extract(r'V(\d+)\b', expand = False)

    clean_df.loc[:,"Cylinder"] = CLY_word.fillna(CLY_V).pipe(pd.to_numeric, errors
    #extracting Fuel_type
    fuel_types =r'(Gasoline|Flex Fuel|Diesel|Hybrid|Electric|CNG|LPG)'
    clean_df.loc[:,"Fuel_Type"] = s.str.extract(fuel_types,expand = False)

    return clean_df
```

In [75]:
```
clean_df  = cleanig_engine_features(clean_df,col = "engine")
clean_df.isnull().sum()
```

Out[75]:
```
brand             0
model             0
model_year        0
milage            0
fuel_type         0
engine            0
transmission      0
ext_col           0
int_col           0
accident          0
clean_title       0
price             0
HorsePower      746
EngineSize      213
Cylinder        441
Fuel_Type       717
dtype: int64
```

In [76]:
```
clean_df['clean_title'].unique()
```

Out[76]:
```
array(['Yes', 'Unknown'], dtype=object)
```

In [77]:
```
clean_df = clean_df.dropna()
```

In [ ]:

In [78]:
```
clean_df['transmission']
```

Out[78]:
```
0                          6-Speed A/T
3                          7-Speed A/T
6                          6-Speed A/T
7                                  A/T
8                          6-Speed A/T
                     ...
4002                       9-Speed A/T
4003                               A/T
4005    Transmission w/Dual Shift Mode
4007                               A/T
4008                               A/T
Name: transmission, Length: 2977, dtype: object
```

In [ ]:

Cleaning Transmission column to Automatic and Manual

In [79]:
```
def Transmission_Cleaning(clean_df,col="transmission"):
    s = clean_df.copy()
    s = clean_df[col].fillna("").astype('str').str.lower()
    #Automatic_group
    Automatic =( s.str.contains("a/t")|
             s.str.contains("auto")|
             s.str.contains("cvt")|
             s.str.contains("variable")|
             s.str.contains("single-speed"))

    #manual
    Manual = (s.str.contains("m/t")|
             s.str.contains("manual")|
             s.str.contains("mt"))
    # changeing variables

    clean_df['Transmission_type'] = "Unknown"
    clean_df.loc[Automatic,'Transmission_type'] = "Automatic"
    clean_df.loc[Manual,'Transmission_type'] = "Manual"

    return clean_df
```

In [80]:
```
clean_df = Transmission_Cleaning(clean_df,col="transmission")
```

In [81]:
```
clean_df['accident'] = clean_df['accident'].replace("At least 1 accident or damage
clean_df['accident'] = clean_df['accident'].replace("None reported","Unknown")
clean_df['accident'] = clean_df['accident'].replace(np.nan,"Unknown")
clean_df['accident'].unique()
```

Out[81]:
```
array(['Yes', 'Unknown'], dtype=object)
```

In [82]:
```
clean_df['clean_title'].unique()
```

Out[82]:
```
array(['Yes'], dtype=object)
```

In [83]:
```
clean_df['price'] = clean_df['price'].str.extract(r'(\d[\d,]*)',expand=False).pipe(
```

In [ ]:
```

```

In [84]:
```
clean_df = clean_df[['brand','model','model_year','fuel_type','ext_col','int_col','
clean_df.head()
```

Out[84]:

	brand	model	model_year	fuel_type	ext_col	int_col	HorsePower	EngineSize	Cyl
0	Ford	Utility Police Interceptor Base	2013	E85 Flex Fuel	Black	Black	300.0	3.7	
3	Infiniti	Q50 Hybrid Sport	2015	Hybrid	Black	Black	354.0	3.5	
6	Audi	S3 2.0T Premium Plus	2017	Gasoline	Blue	Black	292.0	2.0	
7	BMW	740 iL	2001	Gasoline	Green	Green	282.0	4.4	
8	Lexus	RC 350 F Sport	2021	Gasoline	Black	Black	311.0	3.5	

1.4 EDA and Pattern finding

- most expensive color internal and external,
- model_years demanded,
- correlation chart of HorsePower,EngineSize,Cylinder in price
- correlation of allvariables over the price
- most expensive brands

1.5 Preparing our data for our model

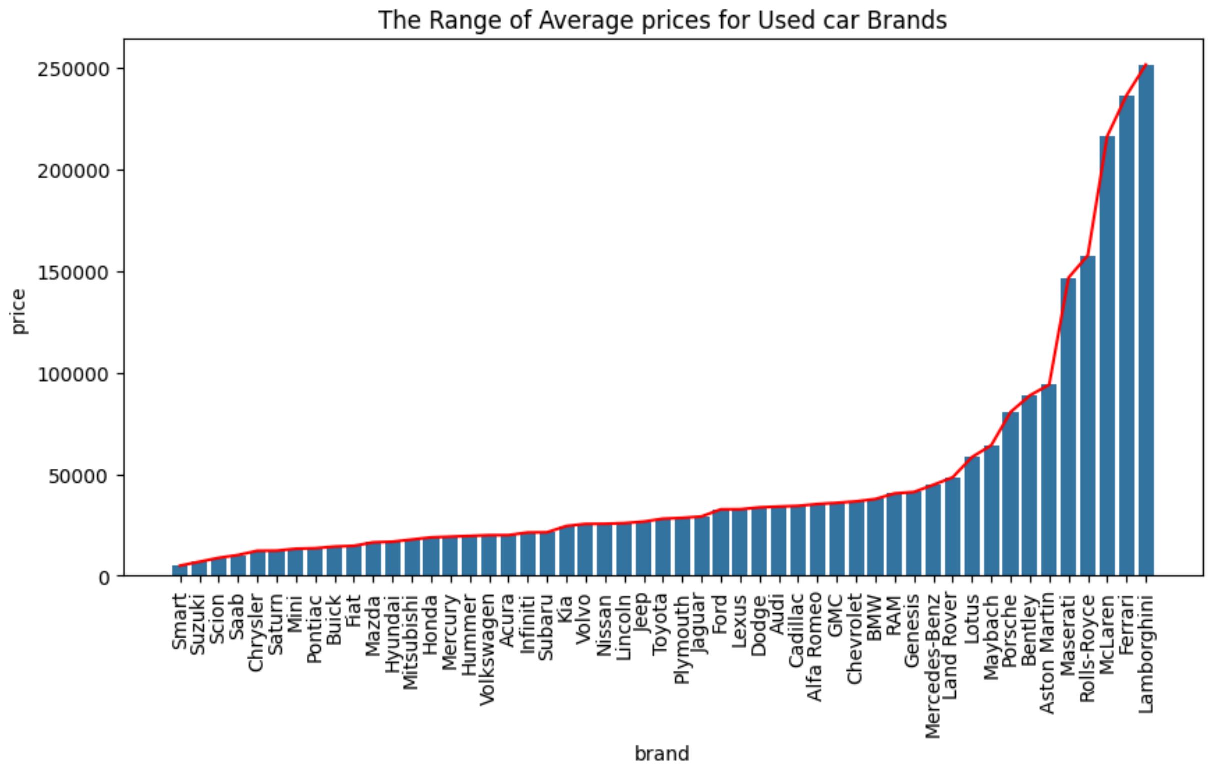
- Standardizeing our variables for our model
- fitting our model
- evaluating our model

In [86]:
```
clean_df.dtypes
```

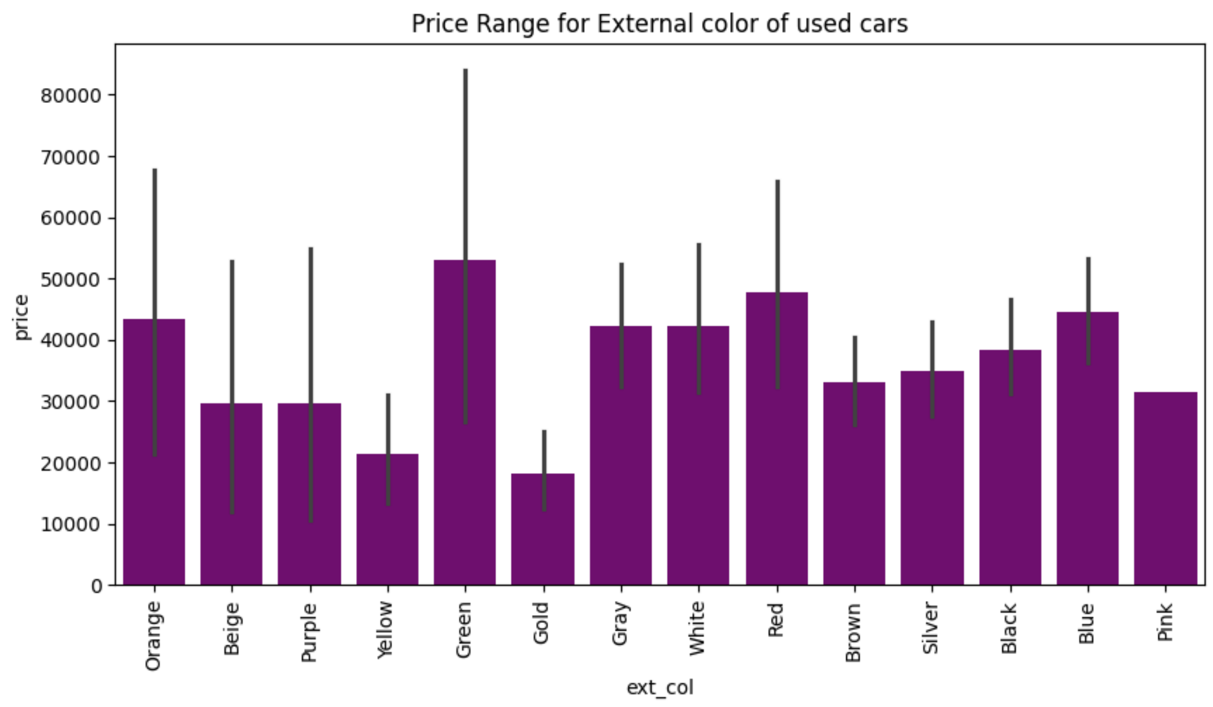
Out[86]:
```
brand                  object
model                  object
model_year              int64
fuel_type              object
ext_col                object
int_col                object
HorsePower            float64
EngineSize            float64
Cylinder                Int64
Transmission_type      object
accident               object
price                 float64
dtype: object
```

```
In [170…  brand_group = clean_df.groupby(['brand'])['price'].mean().reset_index().sort_values

          plt.figure(figsize=(10,5))
          sns.barplot(brand_group,x='brand',y='price')
          sns.lineplot(brand_group,x='brand',y='price',color = 'red')
          plt.title('The Range of Average prices for Used car Brands')
          plt.xticks(rotation=90)
          plt.show()
```

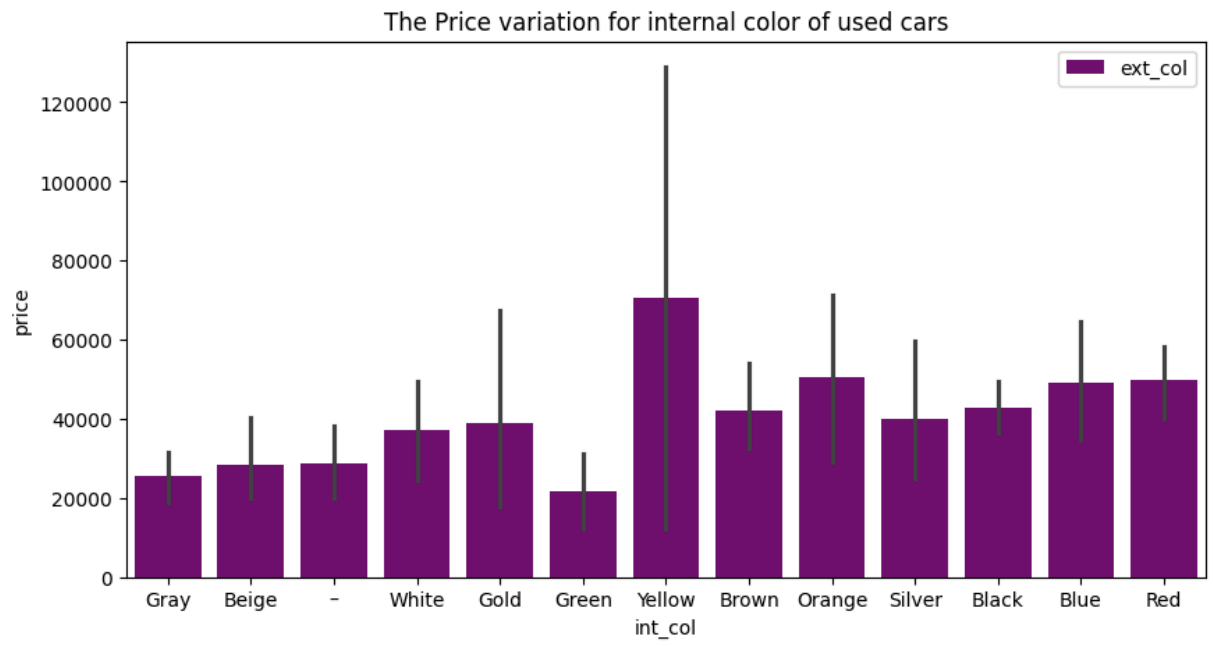


```
In [220…  color = clean_df.groupby(['ext_col','int_col'])['price'].mean().reset_index().sort_
          plt.figure(figsize=(10,5))
          sns.barplot(color,x='ext_col',y='price',color='purple')
          plt.title('Price Range for External color of used cars')
          plt.xticks(rotation=90)
          plt.show()
```

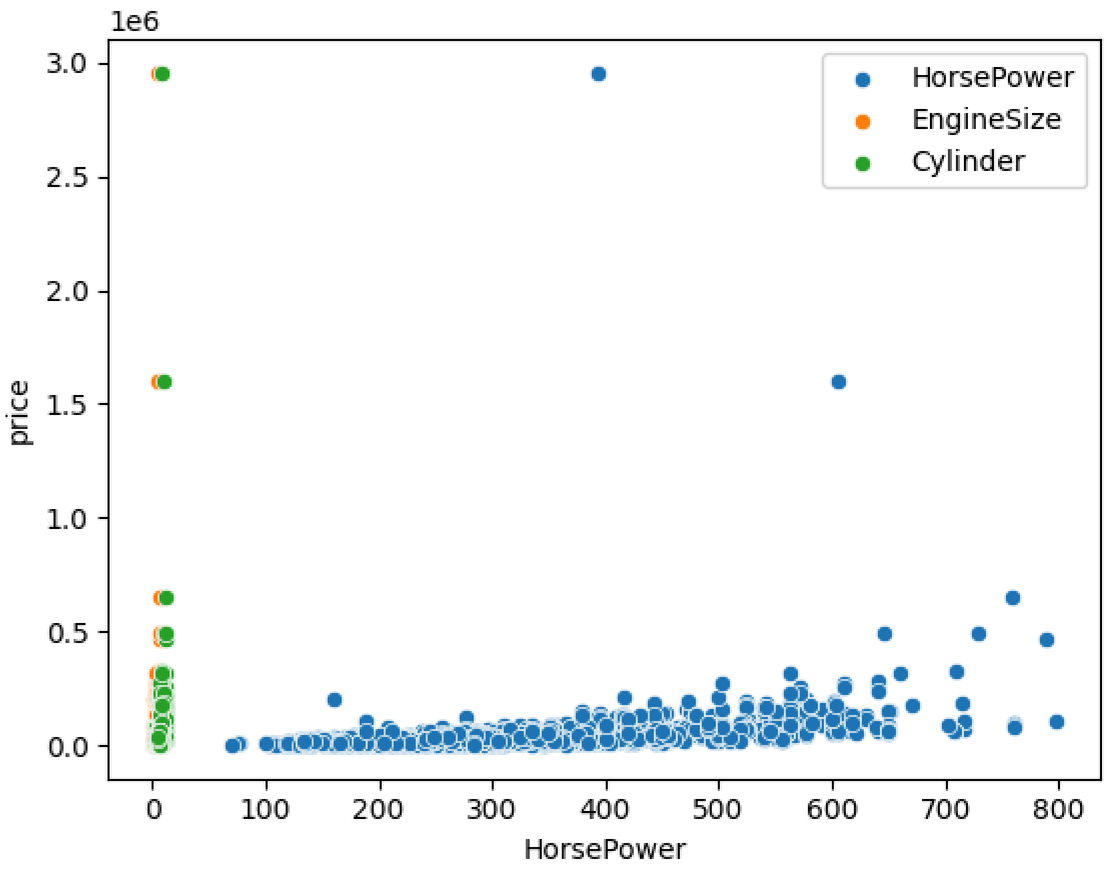


In [221…
```
plt.figure(figsize=(10,5))
plt.figure(figsize=(10,5))
sns.barplot(color,x='int_col',y='price',color='purple',label='ext_col')
plt.title('The Price variation for internal color of used cars')
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```



In [222…
```
sns.scatterplot(data=clean_df,x='HorsePower',y='price',label='HorsePower')
sns.scatterplot(data=clean_df,x='EngineSize',y='price',label='EngineSize')
sns.scatterplot(data=clean_df,x='Cylinder',y='price',label='Cylinder')
plt.legend()
plt.show()
```



In [224…

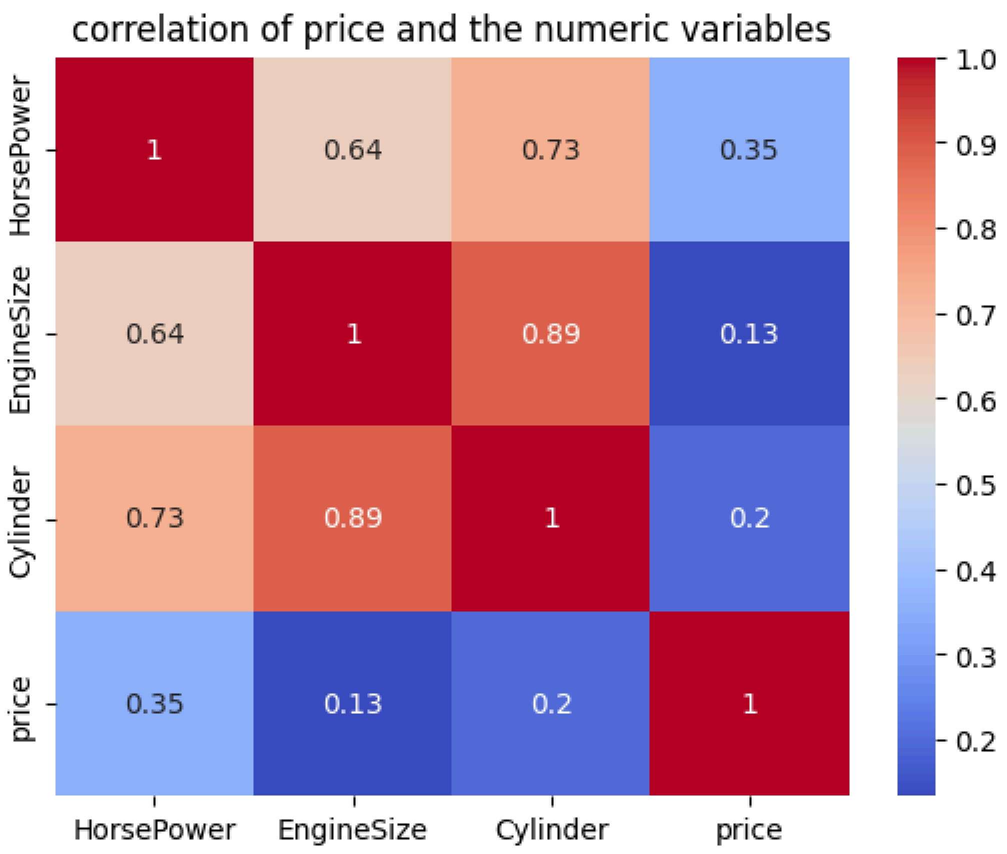
```
clean_df.head()
```

Out[224…

	brand	model	model_year	fuel_type	ext_col	int_col	HorsePower	EngineSize	Cyl
0	Ford	Utility Police Interceptor Base	2013	E85 Flex Fuel	Black	Black	300.0	3.7	
3	Infiniti	Q50 Hybrid Sport	2015	Hybrid	Black	Black	354.0	3.5	
6	Audi	S3 2.0T Premium Plus	2017	Gasoline	Blue	Black	292.0	2.0	
7	BMW	740 iL	2001	Gasoline	Green	Green	282.0	4.4	
8	Lexus	RC 350 F Sport	2021	Gasoline	Black	Black	311.0	3.5	

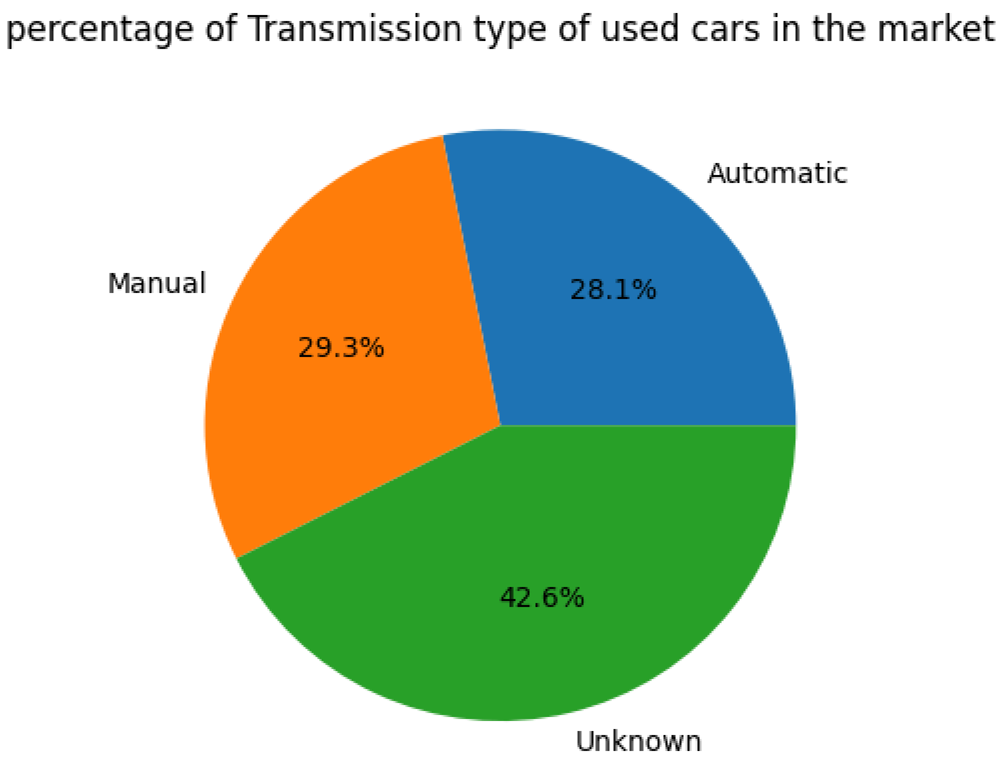
In [225…

```
corr_df = clean_df[['HorsePower','EngineSize','Cylinder','price']]
sns.heatmap(corr_df.corr(),annot=True,cmap='coolwarm')
plt.title('correlation of price and the numeric variables')
plt.show()
```



In [252…

```
Transmission_df = clean_df.groupby('Transmission_type')['price'].mean().reset_index
plt.pie(data = Transmission_df,x='price',labels='Transmission_type',autopct = '%1.1
plt.title('percentage of Transmission type of used cars in the market')
plt.show()
```



Linear Regression

In [253…

```
model = LinearRegression()
```

- before fitting our model we prepare our columns for the model by changing our numeric variables to numeric

In [257…

```
final_df = pd.get_dummies(clean_df,drop_first = True)
final_df = final_df.astype('int')
final_df.columns
```

Out[257…

```
Index(['model_year', 'HorsePower', 'EngineSize', 'Cylinder', 'price',
       'brand_Alfa Romeo', 'brand_Aston Martin', 'brand_Audi', 'brand_BMW',
       'brand_Bentley',
       ...
       'int_col_Green', 'int_col_Orange', 'int_col_Red', 'int_col_Silver',
       'int_col_White', 'int_col_Yellow', 'int_col_–',
       'Transmission_type_Manual', 'Transmission_type_Unknown',
       'accident_Yes'],
      dtype='object', length=1587)
```

In [287…

```
from datetime import datetime
current_year = datetime.now().year
```

In [293…

```
final_df['model_year'] = current_year - final_df['model_year']
```

Out[293…

```
np.int64(1)
```

In [302…

```
x = final_df.drop(columns =['price'])
y= final_df['price']
```

In [303…

```
model.fit(x,y)
```

Out[303…

```
▾ LinearRegression
▶ Parameters
```

In [315…

```
print('Intercept: ',model.intercept_,'\ncofficents:',model.coef_)
```

```
Intercept:  77668.89775567898
cofficents: [-2688.70949432     7.0018044    440.03039144 ...  2725.11446594
  -616.54998997    75.17025358]
```

In [314…

```
print('The RSquare value of our model = ',model.score(x,y))
```

```
The RSquare value of our model =  0.6118313380477749
```

In [ ]:

```

```