Exploratory Data Analysis for UAE used cars data using Python

June 23, 2024

0.0.1 In this notebook, I will conduct Exploratory Data Analysis on UAE used cars data using Python. I will apply techniques such as Data Cleaning, Feature Engineering, Exploratory Data Visualization, and Hypothesis Testing to gain insights and prepare the data for building a regression price prediction model.

Importing required Libraries

[83]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the data

[84]:
```
df = pd.read_csv("UAE_Used_cars.csv")
```

[85]:
```
## taking a look at the first 5 raws
df.head()
```

[85]:

	Car Brand	Car Model	Production Year	Mileage	Price	Description	Specs	Timestamp	Location
0	Nissan	Altima	2005	445,740 km	3,500	Dubai	GCC Specs	04-03-24 14:49	Dubai
1	Toyota	Camry	1999	200,000 km	5,500	Perfect Condition Toyota Camry	GCC Specs	04-03-24 14:49	Dubai
2	Ford	Focus	2006	366,135 km	5,500	FORD FOCUS	GCC Specs	04-03-24 14:49	Dubai
3	Toyota	Echo	2005	200,000 km	6,000	GCC - TOYOTA ECHO 2005 - Manual, Urgent Sale	GCC Specs	04-03-24 14:49	Dubai
4	Chevrolet	Epica	2009	250,000 km	6000	Chevrolet Epica	American Specs	45354.94097	Abu Dhabi

[86]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8006 entries, 0 to 8005
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Car Brand        8006 non-null   object
 1   Car Model        8006 non-null   object
 2   Production Year  8006 non-null   int64 
 3   Mileage          8006 non-null   object
 4   Price            8006 non-null   object
 5   Description      8006 non-null   object
 6   Specs            8006 non-null   object
 7   Timestamp        8006 non-null   object
 8   Location         8006 non-null   object
dtypes: int64(1), object(8)
memory usage: 563.0+ KB
```

[87]:
```
df.describe(include='all')
```

[87]:

	Car Brand	Car Model	Production Year	Mileage	Price	\
count	8006	8006	8006.000000	8006	8006	
unique	7	183	NaN	2472	1163	
top	Toyota	Patrol	NaN	0 km	25,000	
freq	2522	480	NaN	1375	91	
mean	NaN	NaN	2017.939046	NaN	NaN	
std	NaN	NaN	5.227208	NaN	NaN	
min	NaN	NaN	1929.000000	NaN	NaN	
25%	NaN	NaN	2015.000000	NaN	NaN	
50%	NaN	NaN	2019.000000	NaN	NaN	
75%	NaN	NaN	2022.000000	NaN	NaN	
max	NaN	NaN	2024.000000	NaN	NaN	

	Description	Specs	\
count	8006	8006	
unique	7509	8	
top	Ford Mustang 5.0 GT Premium 2024 MY- V8 Engine…	GCC Specs	
freq	44	5796	
mean	NaN	NaN	
std	NaN	NaN	
min	NaN	NaN	
25%	NaN	NaN	
50%	NaN	NaN	
75%	NaN	NaN	
max	NaN	NaN	

```
              Timestamp Location
count              8006     8006
unique                2        2
top     04-03-24 14:49    Dubai
freq               7311     7311
mean                NaN      NaN
std                 NaN      NaN
min                 NaN      NaN
25%                 NaN      NaN
50%                 NaN      NaN
75%                 NaN      NaN
max                 NaN      NaN
```

Handling Missing Values

[88]:
```
## checking for missing values
df.isnull().sum()
```

[88]:
```
Car Brand          0
Car Model          0
Production Year    0
Mileage            0
Price              0
Description        0
Specs              0
Timestamp          0
Location           0
dtype: int64
```

[89]:
```
# Removing non-numeric characters from the 'Price' column and convert to float
df['Price'] = df['Price'].replace('[\$,]', '', regex=True)
```

[90]:
```
# Checking for any non-numeric entries in the 'Price' column
non_numeric_prices = df[~df['Price'].str.replace('.', '', 1).str.isdigit()]
```

[91]:
```
# Removing rows with non-numeric 'Price' values
df = df[df['Price'].str.replace('.', '', 1).str.isdigit()]
```

[92]:
```
# Converting 'Price' column to float
df['Price'] = df['Price'].astype(float)
```

[93]:
```
# Checking for and remove any remaining NaN values in 'Price'
df.dropna(subset=['Price'], inplace=True)
```

[94]:
```
# dropping rows/columns with too many missing values
df.dropna(subset=['Car Brand', 'Car Model', 'Production Year', 'Price',␣
↪'Location'], inplace=True)
```

[95]:
```
# Filling missing descriptions with an empty string
df['Description'].fillna('', inplace=True)
```

[96]:
```
# Filling missing specs with 'Unknown'
df['Specs'].fillna('Unknown', inplace=True)  # Fill missing specs with 'Unknown'
```

[97]:
```
df['Production Year'] = df['Production Year'].astype(int)
```

[98]:
```
df.head()
```

[98]:

	Car Brand	Car Model	Production Year	Mileage	Price	\
0	Nissan	Altima	2005	445,740 km	3500.0	
1	Toyota	Camry	1999	200,000 km	5500.0	
2	Ford	Focus	2006	366,135 km	5500.0	
3	Toyota	Echo	2005	200,000 km	6000.0	
4	Chevrolet	Epica	2009	250,000 km	6000.0	

	Description	Specs	\
0	Dubai	GCC Specs	
1	Perfect Condition Toyota Camry	GCC Specs	
2	FORD FOCUS	GCC Specs	
3	GCC - TOYOTA ECHO 2005 - Manual, Urgent Sale	GCC Specs	
4	Chevrolet Epica	American Specs	

	Timestamp	Location
0	04-03-24 14:49	Dubai
1	04-03-24 14:49	Dubai
2	04-03-24 14:49	Dubai
3	04-03-24 14:49	Dubai
4	45354.94097	Abu Dhabi

0.1 Engineering the Features

0.1.1 Extracting Features from Description

[99]:
```
import re
```

[100]:
```
mileage_pattern = re.compile(r'(\d{1,3}(?:,\d{3})*|\d+)\s*(km|kilometers)')
color_pattern = re.
 ↪compile(r'\b(white|black|blue|red|silver|grey|green|yellow|brown|gold|orange)\b',␣
 ↪re.IGNORECASE)
```

[101]:
```
def extract_mileage(description):
    match = mileage_pattern.search(description)
    if match:
        mileage = match.group(1).replace(',', '')
        return int(mileage)
```

```
    return None

def extract_color(description):
    match = color_pattern.search(description)
    if match:
        return match.group(0).capitalize()
    return 'Unknown'
```

[102]:
```
df['Mileage'] = df['Description'].apply(extract_mileage)
df['Color'] = df['Description'].apply(extract_color)
```

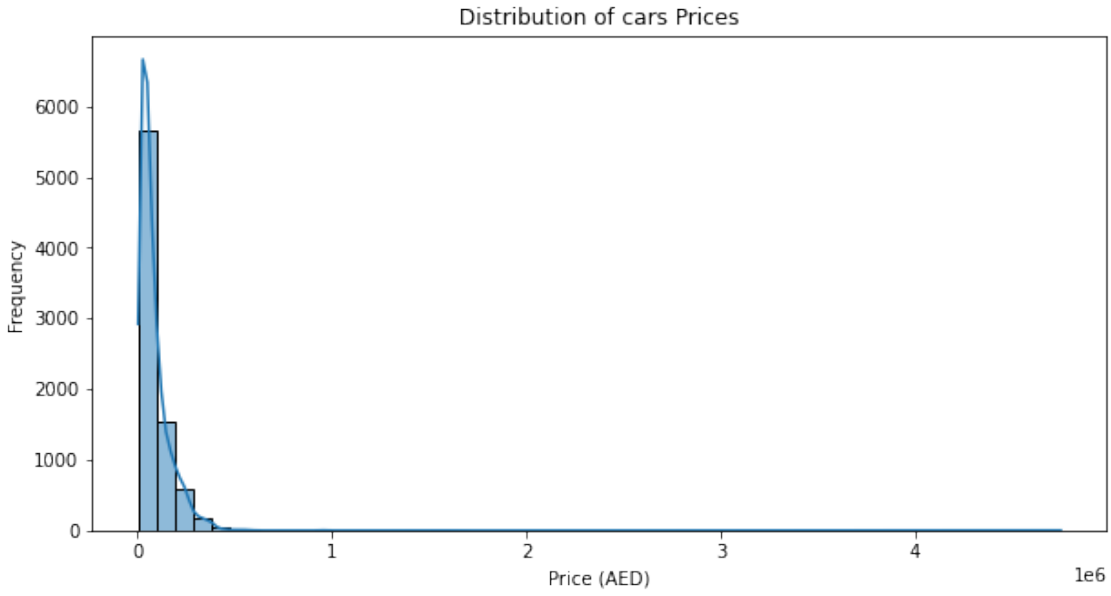
[103]:
```
df['Mileage'].fillna(df['Mileage'].median(), inplace=True)
```

0.1.2 Performing Categorical Encoding

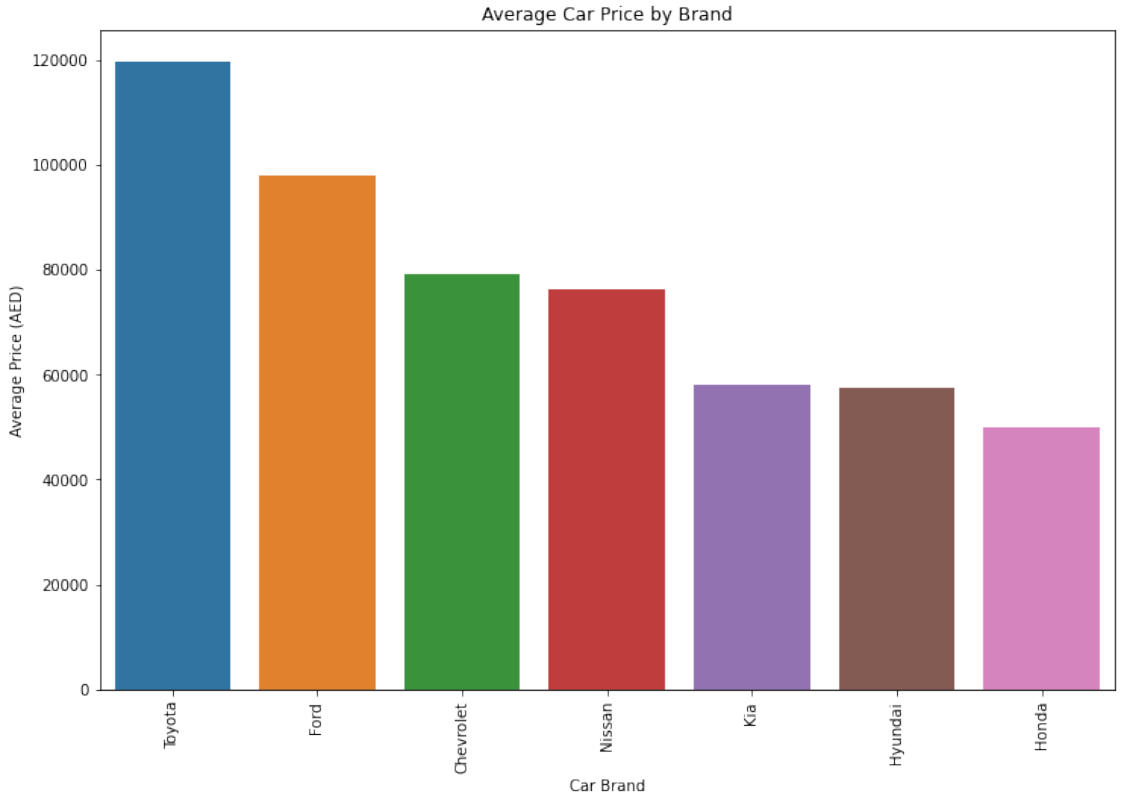
0.2 Exploratory Data Visualization

[104]:
```
# Plotting the destribution of cars prices
plt.figure(figsize=(10,5))
sns.histplot(df['Price'], bins=50, kde= True)
plt.title('Distribution of cars Prices')
plt.xlabel('Price (AED)')
plt.ylabel('Frequency')
plt.show
```

[104]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



[105]:
```
# Plot average car price by brand
plt.figure(figsize=(12, 8))
avg_price_by_brand = df.groupby('Car Brand')['Price'].mean().
 ↪sort_values(ascending=False)
sns.barplot(x=avg_price_by_brand.index, y=avg_price_by_brand.values)
plt.title('Average Car Price by Brand')
plt.xlabel('Car Brand')
plt.ylabel('Average Price (AED)')
plt.xticks(rotation=90)
plt.show()
```



0.3 Performing Hypothesis Testing

0.3.1 Comparing Car Prices in Dubai and Abu Dhabi

[106]:
```
from scipy.stats import ttest_ind
```

[107]:
```
# Separate the prices by location
dubai_prices = df[df['Location'] == 'Dubai']['Price']
abu_dhabi_prices = df[df['Location'] == 'Abu Dhabi']['Price']
```

0.3.2 Performing t-test

[108]:
```
t_stat, p_val = ttest_ind(dubai_prices, abu_dhabi_prices, equal_var=False)
```

[109]:
```
# Displaying the test results
print(f'T-test statistic: {t_stat:.2f}')
print(f'P-value: {p_val:.4f}')
```

```
T-test statistic: 2.48
P-value: 0.0132
```

[110]:
```
# Interpretation
if p_val < 0.05:
    print("There is a significant difference in car prices between Dubai and␣
 ↪Abu Dhabi.")
else:
    print("There is no significant difference in car prices between Dubai and␣
 ↪Abu Dhabi.")
```

```
There is a significant difference in car prices between Dubai and Abu Dhabi.
```

Author: Bahraleloom Abdalrahem **Email:** bahraleloom@gmail.com **GitHub:** https://github.com/Bahraleloom **Kaggle:** https://www.kaggle.com/bahraleloom **Date:** 23/06/2024