Data Collection & Pre-processing

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re

df = pd.read_csv("./used_car_dataset.csv")
df.head()
```

```
                                              car_name
car_price_in_rupees  \
0  Hyundai Grand i10 Magna 1.2 Kappa VTVT [2017-2...          ₹ 4.45
Lakh
1                          Maruti Suzuki Alto 800 Lxi          ₹ 2.93
Lakh
2                             Tata Safari XZ Plus New        ₹ 22.49
Lakh
3                             Maruti Suzuki Ciaz ZXI+          ₹ 6.95
Lakh
4     Jeep Compass Sport Plus 1.4 Petrol [2019-2020]            ₹ 12
Lakh

  kms_driven fuel_type       city  year_of_manufacture
0  22,402 km    Petrol     Mumbai                 2016
1  10,344 km    Petrol    Kolkata                 2019
2  12,999 km    Diesel  Bangalore                 2021
3  45,000 km    Petrol      Thane                 2016
4  11,193 km    Petrol    Kolkata                 2019
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2105 entries, 0 to 2104
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   car_name             2105 non-null   object
 1   car_price_in_rupees  2105 non-null   object
 2   kms_driven           2105 non-null   object
 3   fuel_type            2105 non-null   object
 4   city                 2105 non-null   object
 5   year_of_manufacture  2105 non-null   int64
dtypes: int64(1), object(5)
memory usage: 98.8+ KB
```

```
df['Price'] = df['car_price_in_rupees'].str.extract('([\d.]
+)').astype(float) * 100000
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2105 entries, 0 to 2104
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   car_name             2105 non-null   object 
 1   car_price_in_rupees  2105 non-null   object 
 2   kms_driven           2105 non-null   object 
 3   fuel_type            2105 non-null   object 
 4   city                 2105 non-null   object 
 5   year_of_manufacture  2105 non-null   int64  
 6   Price                2105 non-null   float64
dtypes: float64(1), int64(1), object(5)
memory usage: 115.2+ KB
```

Feature Engineering

```
df['Age'] = 2024-df['year_of_manufacture']

print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
print(df.duplicated().sum())
```

```
92
0
```

```
df['KM'] = df['kms_driven'].apply(lambda x: int(re.sub(r'\D', '', x)))
df['KM'].head()
```

```
0    22402
1    10344
2    12999
3    45000
4    11193
Name: KM, dtype: int64
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

df['city'] = label_encoder.fit_transform(df['city'])
df['fuel_type'] = label_encoder.fit_transform(df['fuel_type'])
```

```
df = df.drop(['car_name', 'car_price_in_rupees',
'year_of_manufacture', 'kms_driven'], axis=1)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2013 entries, 0 to 2104
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   fuel_type  2013 non-null   int64
 1   city       2013 non-null   int64
 2   Price      2013 non-null   float64
 3   Age        2013 non-null   int64
 4   KM         2013 non-null   int64
dtypes: float64(1), int64(4)
memory usage: 94.4 KB
```

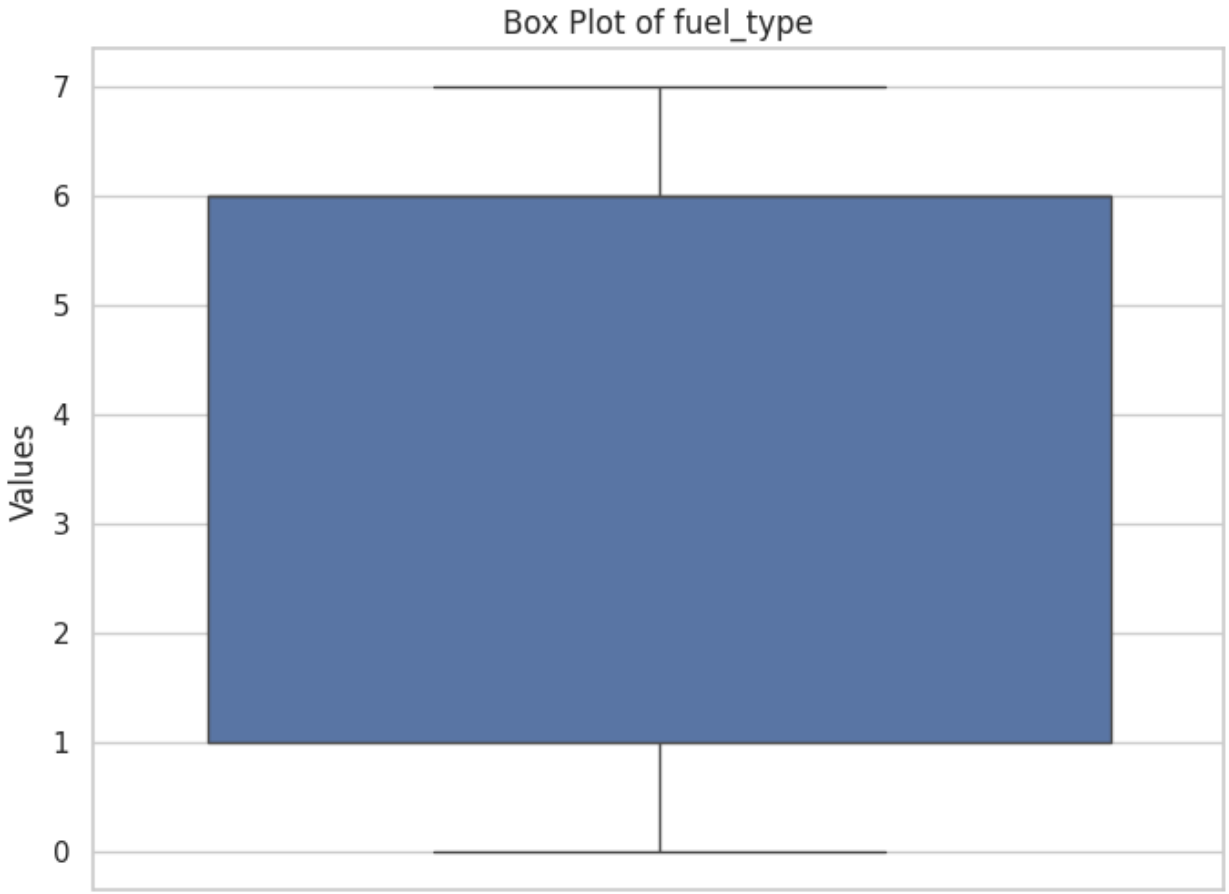
```
df.describe()
```

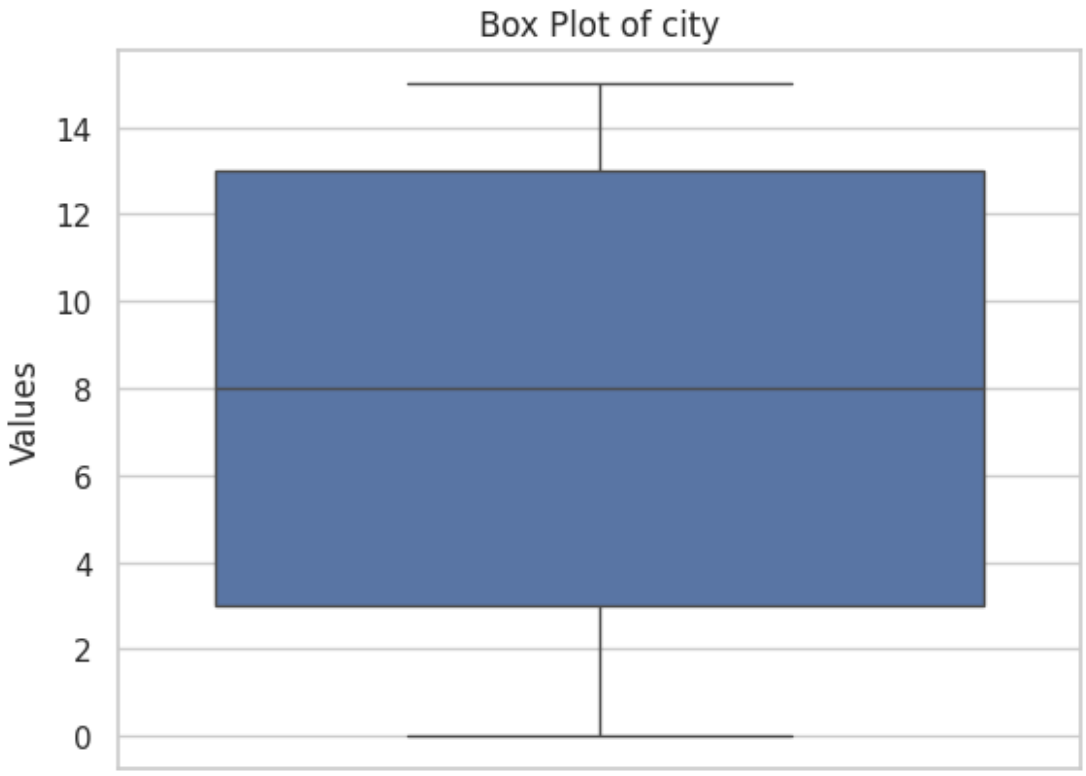
```
          fuel_type          city          Price           Age
KM
count  2013.000000  2013.000000  2.013000e+03  2013.000000
2.013000e+03
mean      4.220566     7.122206  1.111732e+06     6.976155
4.843062e+04
std       2.443809     4.676192  1.210148e+06     2.823402
8.542756e+04
min       0.000000     0.000000  1.100000e+05     2.000000
6.400000e+01
25%       1.000000     3.000000  4.750000e+05     5.000000
2.500000e+04
50%       6.000000     8.000000  6.990000e+05     7.000000
4.282500e+04
75%       6.000000    11.000000  1.150000e+06     9.000000
6.300000e+04
max       7.000000    15.000000  9.900000e+06    20.000000
3.600000e+06
```

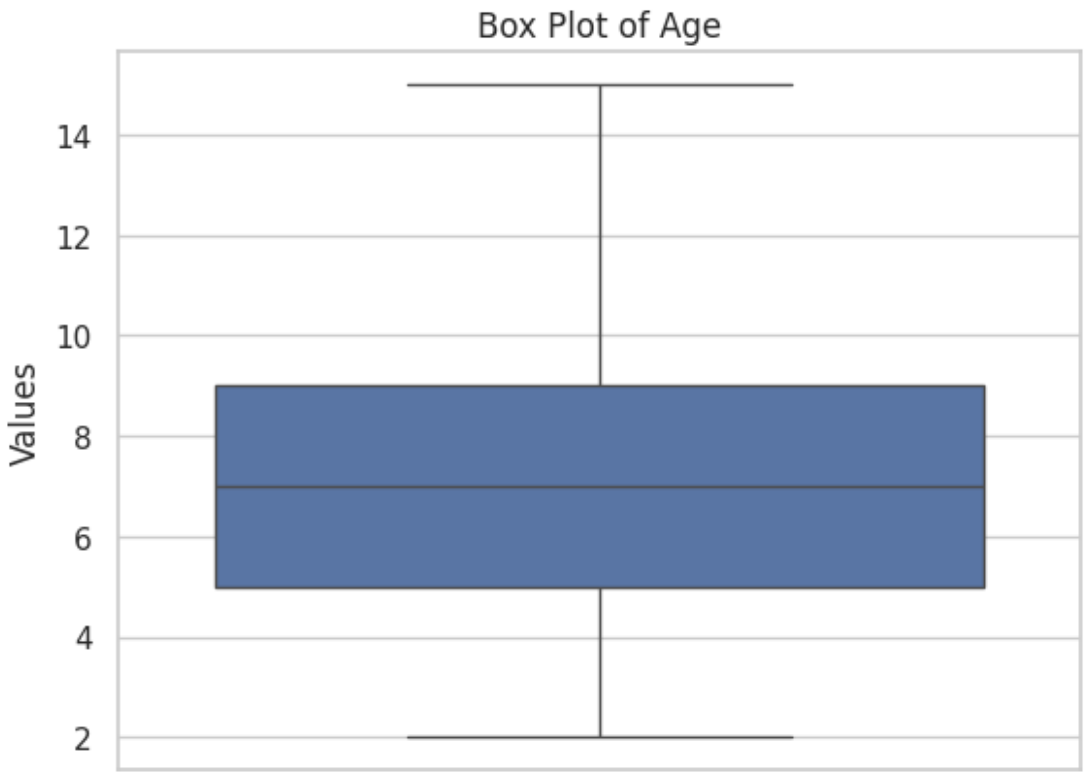
Outlier Handling

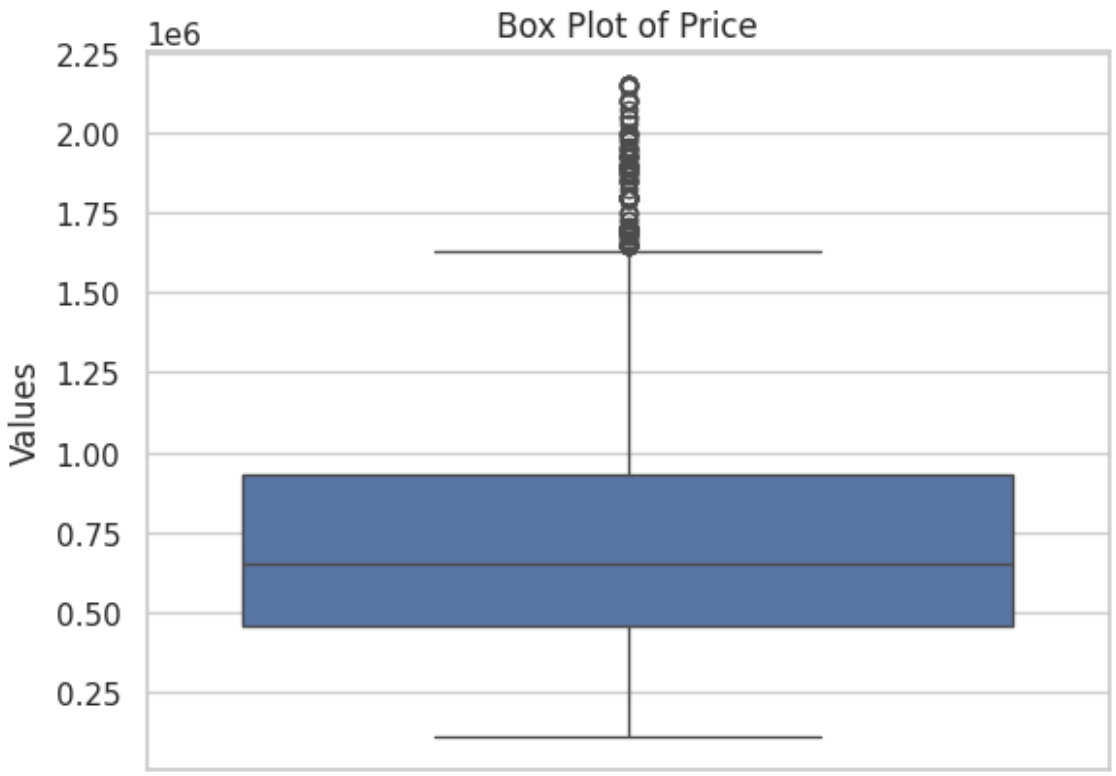
```
def remove_outliers_iqr(df, column, threshold=1.5):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - threshold * IQR
    upper_bound = Q3 + threshold * IQR
```

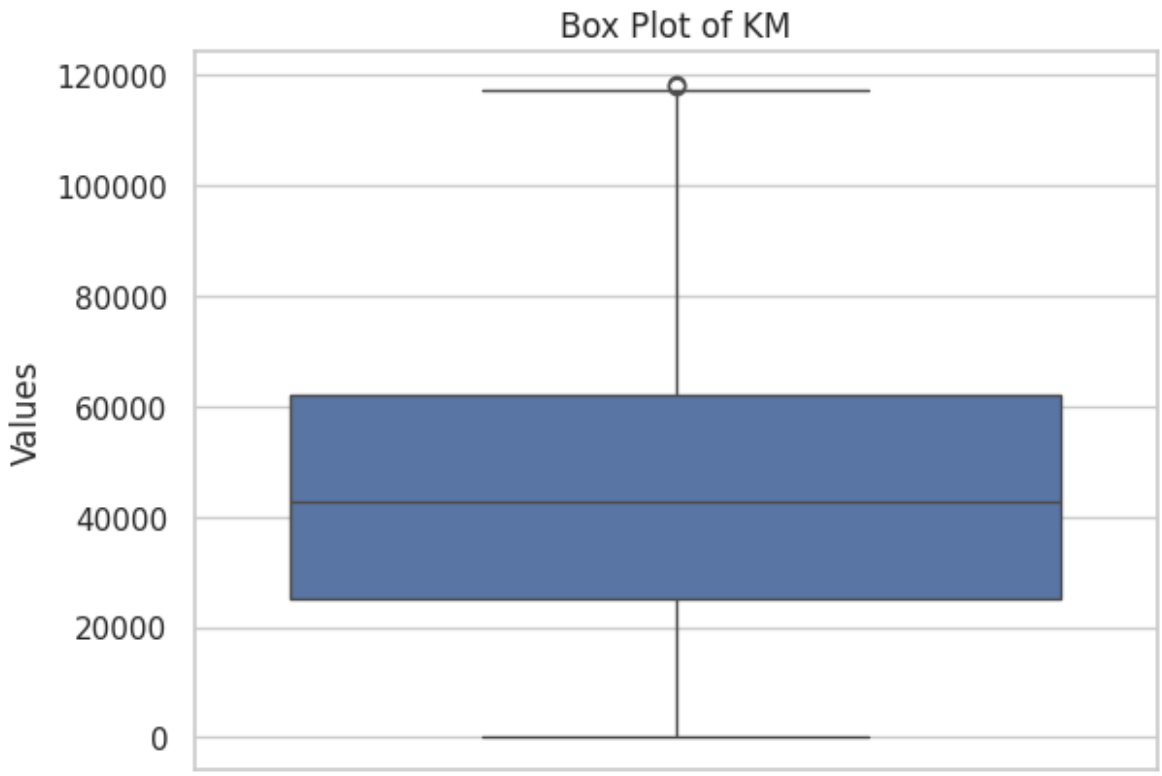
```
    outliers = (df[column] < lower_bound) | (df[column] > upper_bound)
    df = df[~outliers]

    return df

columns_to_check = df.columns.tolist()

for column in columns_to_check:
    df = remove_outliers_iqr(df, column)

sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))

columns_to_plot = ['fuel_type', 'city', 'Age', 'Price', 'KM']

for item in columns_to_plot:
  sns.boxplot(data=df[item])

  plt.xlabel("".format(item))
  plt.ylabel("Values")
  plt.title("Box Plot of {}".format(item))
  plt.show()
  print('\n')
```




Box Plot of city
Values
14
12
10
8
6
4
2
0


Box Plot of Age
Values
14
12
10
8
6
4
2

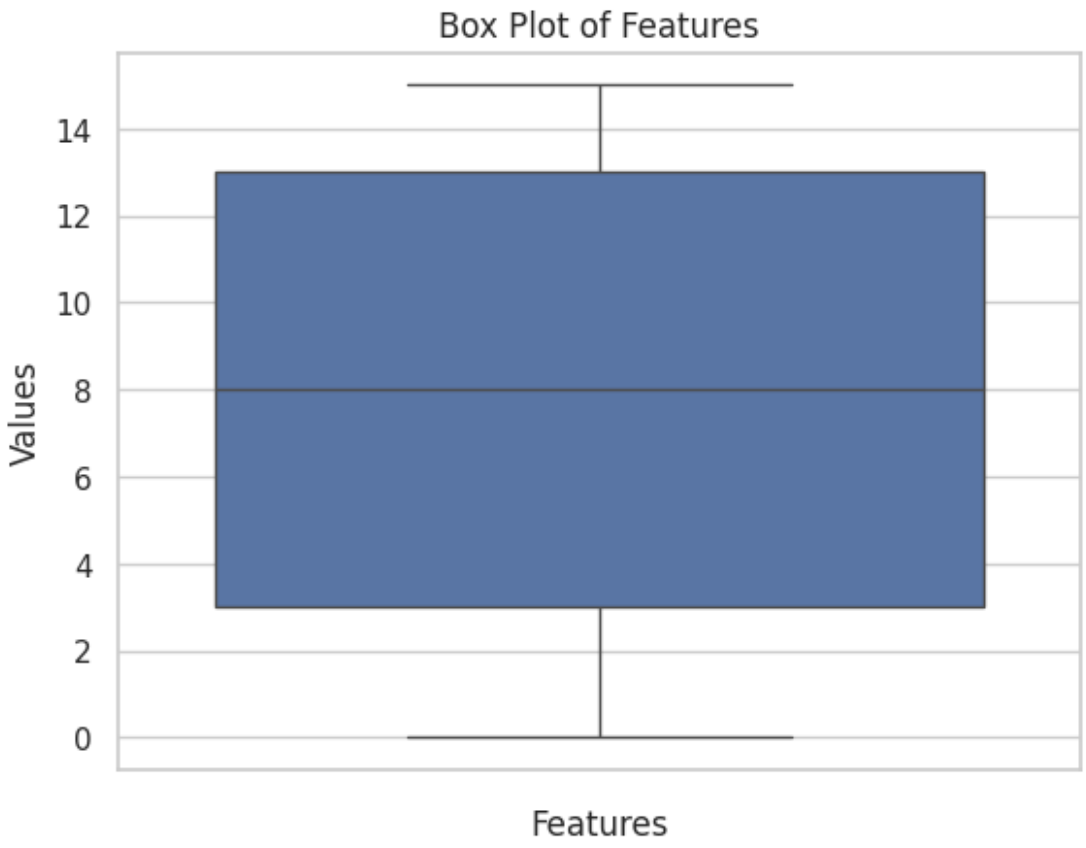

Box Plot of Price
1e6
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
Values



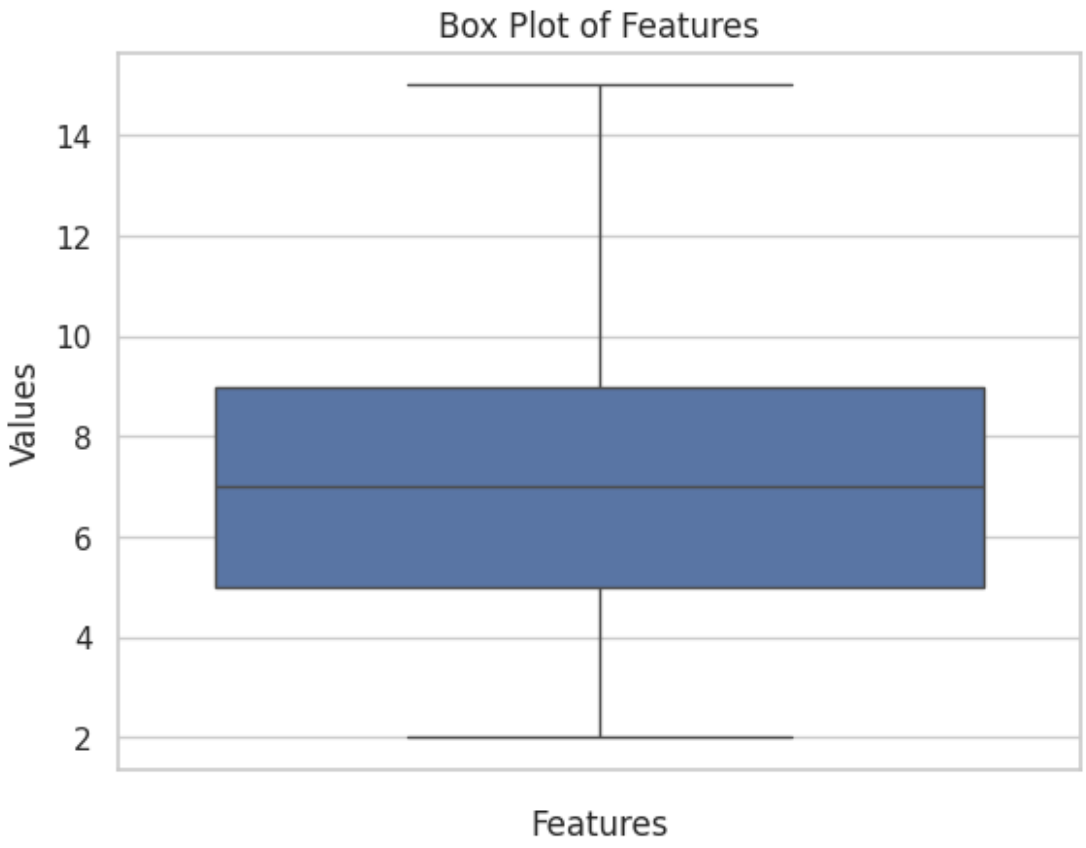

```
column = df.columns.tolist()
for item in column:
  Q1 = df[item].quantile(0.25)
  Q3 = df[item].quantile(0.75)
  IQR = Q3 - Q1
  whisker_width = 1.5
  lower_whisker = Q1 -(whisker_width*IQR)
  upper_whisker = Q3 + (whisker_width*IQR)

df[item]=np.where(df[item]>upper_whisker,upper_whisker,np.where(df[ite
m]<lower_whisker,lower_whisker,df[item]))

sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))

columns_to_plot = ['fuel_type', 'city', 'Age', 'Price', 'KM']

for item in columns_to_plot:
  sns.boxplot(data=df[item])

  plt.xlabel("Features")
  plt.ylabel("Values")
  plt.title("Box Plot of Features")
```

```
plt.show()
print("/n")
```

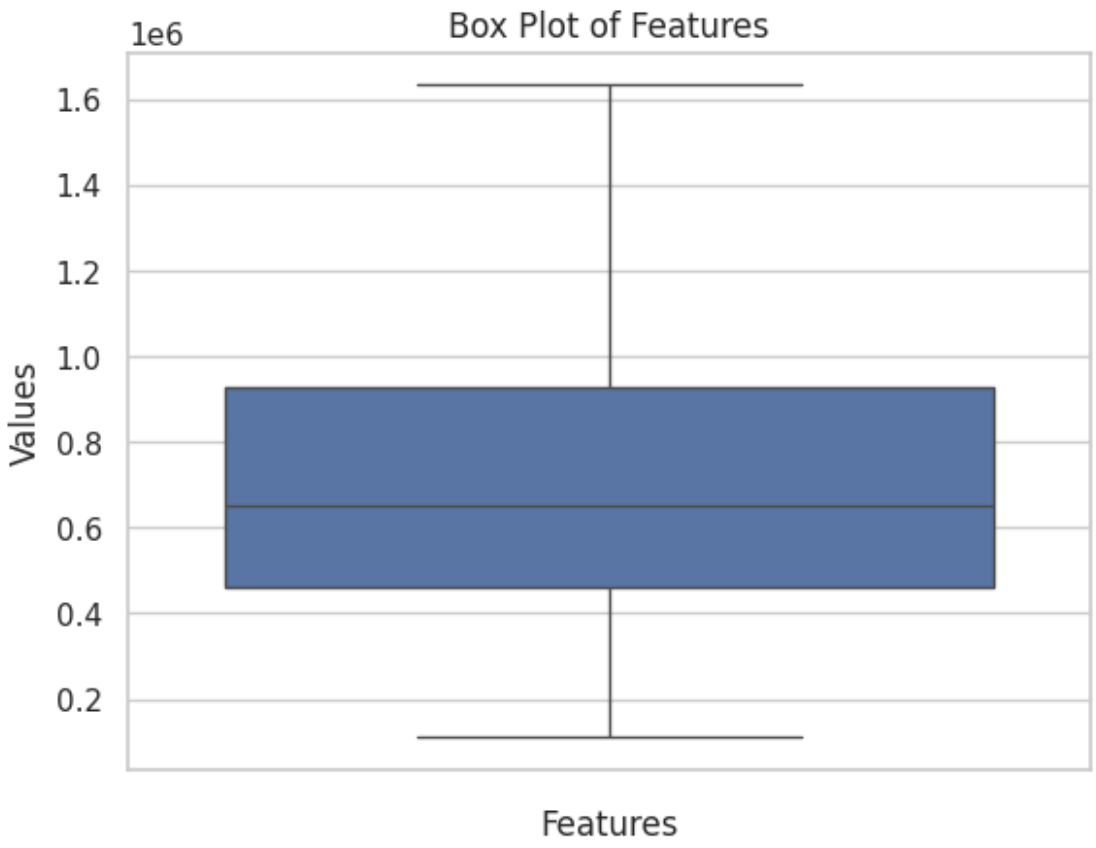

```
/n
```



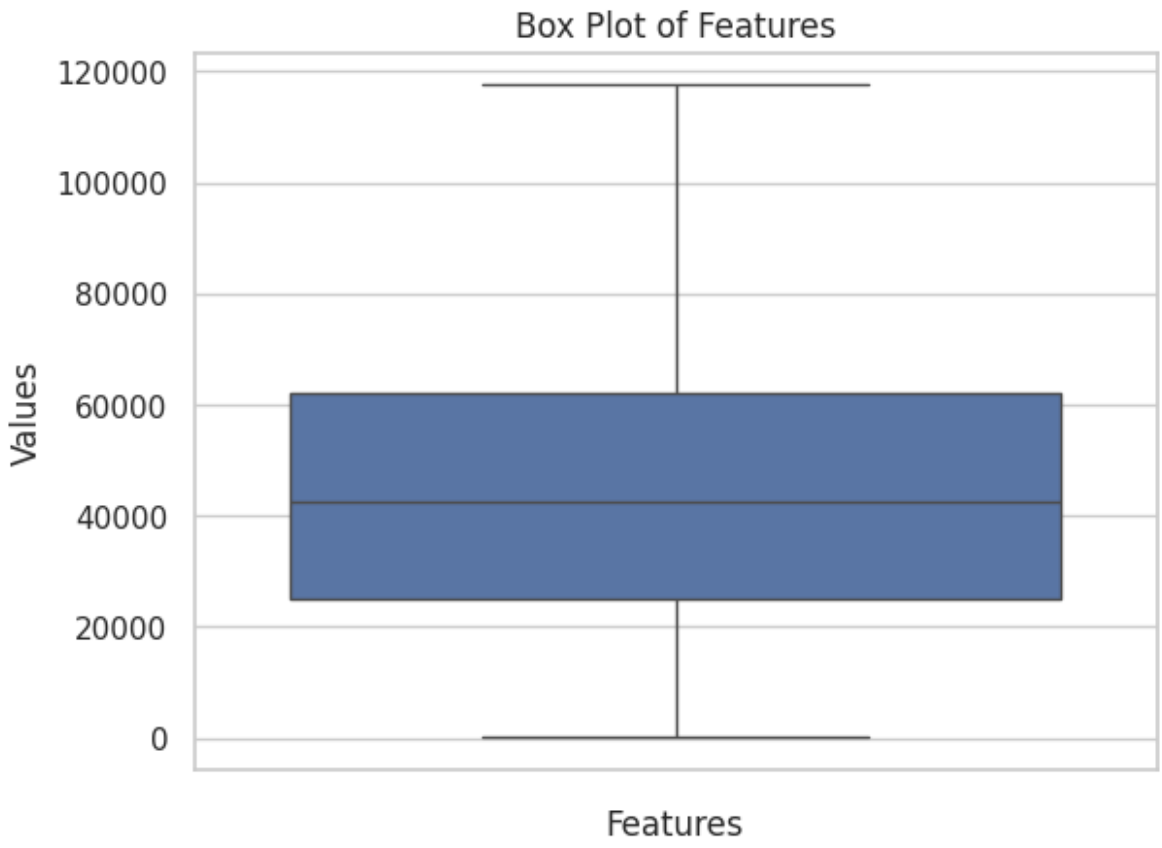

```
/n
```




```
/n
```


Box Plot of Features
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
Values
Features

```
/n
```




```
/n

df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1746 entries, 0 to 2104
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   fuel_type  1746 non-null   float64
 1   city       1746 non-null   float64
 2   Price      1746 non-null   float64
 3   Age        1746 non-null   float64
 4   KM         1746 non-null   float64
dtypes: float64(5)
memory usage: 81.8 KB
```

Scaling

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

```
scaler = StandardScaler()
df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1746 entries, 0 to 1745
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   fuel_type  1746 non-null   float64
 1   city       1746 non-null   float64
 2   Price      1746 non-null   float64
 3   Age        1746 non-null   float64
 4   KM         1746 non-null   float64
dtypes: float64(5)
memory usage: 68.3 KB
```

X & Y Split

```
X = df.drop(['Price'], axis=1)
y = df['Price']
```

Train - Test Split

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR

from sklearn.metrics import mean_squared_error, r2_score

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Model Training

```
model1 = RandomForestRegressor()
model1.fit(X_train, y_train)
y_pred1 = model1.predict(X_test)
```

```
model2 = LinearRegression()
model2.fit(X_train, y_train)
y_pred2 = model2.predict(X_test)

model3 = DecisionTreeRegressor()
model3.fit(X_train, y_train)
y_pred3 = model3.predict(X_test)

model4 = KNeighborsRegressor()
model4.fit(X_train, y_train)
y_pred4 = model4.predict(X_test)

model5 = SVR()
model5.fit(X_train, y_train)
y_pred5 = model5.predict(X_test)
```

Model Evaluation

```
print("Random Forest",r2_score(y_test, y_pred1))
```

```
Random Forest 0.37559431558600653
```

```
print("Linear Regression",r2_score(y_test, y_pred2))
```

```
Linear Regression 0.2909252047817328
```

```
print("Decision Trees Regression",r2_score(y_test, y_pred3))
```

```
Decision Trees Regression -0.03886027911600354
```

```
print('KNN',r2_score(y_test, y_pred4))
```

```
KNN 0.2986720725784908
```

```
print("SVR",r2_score(y_test, y_pred5))
```

```
SVR 0.380695077156769
```