CarVal (UAE used cars price prediction model)

January 28, 2025

[293]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('UAE_Used_Cars.csv')
```

[294]:
```
df.head()
```

[294]:

	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

[295]:
```
df = df[["Car Brand", "Car Model", "Production Year", "Mileage",  "Price","Specs"]]
df.head()
```

[295]:

	Car Brand	Car Model	Production Year	Mileage	Price	Specs
0	Nissan	Altima	2005	445,740 km	3,500	GCC Specs
1	Toyota	Camry	1999	200,000 km	5,500	GCC Specs
2	Ford	Focus	2006	366,135 km	5,500	GCC Specs
3	Toyota	Echo	2005	200,000 km	6,000	GCC Specs
4	Chevrolet	Epica	2009	250,000 km	6000	American Specs

[296]:
```
df = df[df["Price"].notnull()]
df.head()
```

[296]:

	Car Brand	Car Model	Production Year	Mileage	Price	Specs
0	Nissan	Altima	2005	445,740 km	3,500	GCC Specs
1	Toyota	Camry	1999	200,000 km	5,500	GCC Specs
2	Ford	Focus	2006	366,135 km	5,500	GCC Specs
3	Toyota	Echo	2005	200,000 km	6,000	GCC Specs
4	Chevrolet	Epica	2009	250,000 km	6000	American Specs

[297]:
```
rows_to_drop = df[df['Mileage'] == '445740 km']

print("Rows to drop:\n", rows_to_drop)
```

```
Rows to drop:
 Empty DataFrame
Columns: [Car Brand, Car Model, Production Year, Mileage, Price, Specs]
Index: []
```

[298]:
```
print(df.head())
```

```
   Car Brand Car Model  Production Year     Mileage  Price           Specs
0     Nissan    Altima             2005  445,740 km  3,500       GCC Specs
1     Toyota     Camry             1999  200,000 km  5,500       GCC Specs
2       Ford     Focus             2006  366,135 km  5,500       GCC Specs
3     Toyota      Echo             2005  200,000 km  6,000       GCC Specs
4  Chevrolet     Epica             2009  250,000 km   6000  American Specs
```

[299]:
```
df = df.drop(rows_to_drop.index)
```

[300]:
```
df['Price'] = df['Price'].astype(str)
df['Mileage'] = df['Mileage'].astype(str)

df['Price'] = df['Price'].str.replace(',', '').astype(float)
df['Mileage'] = df['Mileage'].str.replace(' km', '').str.replace(',', '').
 ↪astype(float)
```

[301]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8006 entries, 0 to 8005
Data columns (total 6 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   float64
```

```
 4   Price            8006 non-null   float64
 5   Specs            8006 non-null   object
dtypes: float64(2), int64(1), object(3)
memory usage: 375.4+ KB
```

[302]:
```
df = df.dropna()
df.isnull()
```

[302]:

	Car Brand	Car Model	Production Year	Mileage	Price	Specs
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
…	…	…	…	…	…	…
8001	False	False	False	False	False	False
8002	False	False	False	False	False	False
8003	False	False	False	False	False	False
8004	False	False	False	False	False	False
8005	False	False	False	False	False	False

[8006 rows x 6 columns]

[303]:
```
missing_values = df.isnull().sum()
print(missing_values)
```

```
Car Brand          0
Car Model          0
Production Year    0
Mileage            0
Price              0
Specs              0
dtype: int64
```

[304]:
```
df['Age'] = 2024 - df['Production Year']
```

[305]:
```
df['Mileage'].value_counts()
```

[305]:
```
Mileage
0.0         1375
200000.0      57
120000.0      51
10.0          48
130000.0      47
            …
22542.0        1
8182.0         1
```

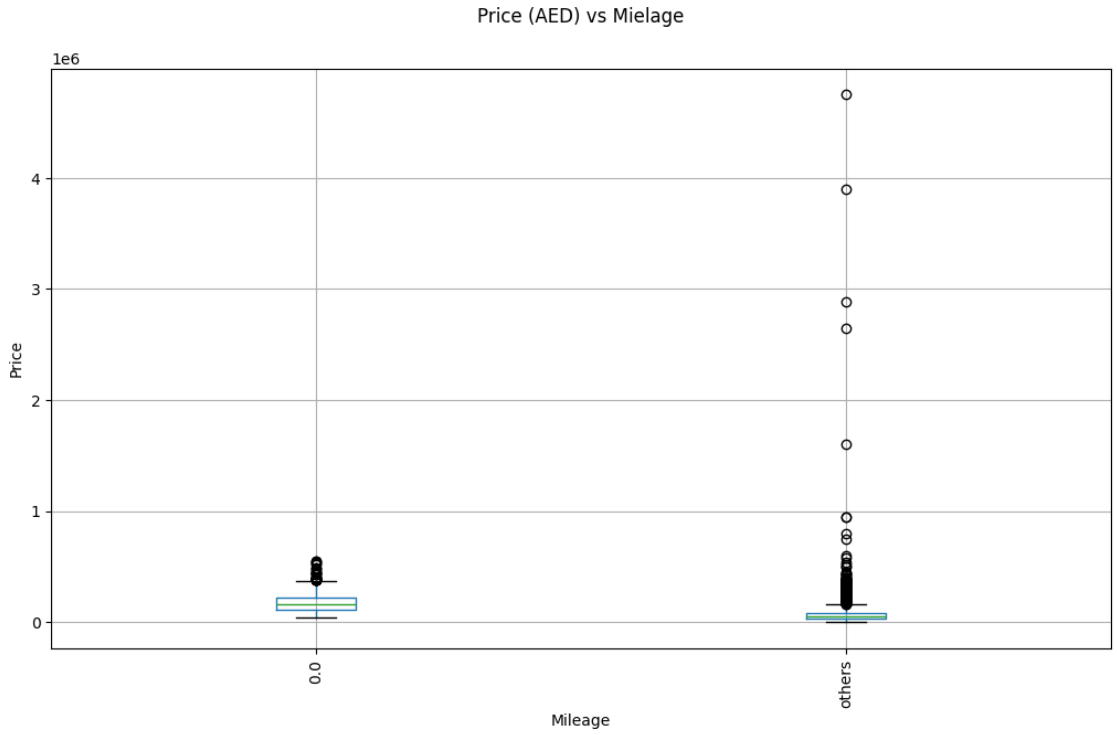
```
5413.0          1
2.0             1
18.0            1
Name: count, Length: 2472, dtype: int64
```

[306]:
```
def shorten_categories(categories, cutoff):
    categorical_map= {}
    for i in range(len(categories)):
        if categories.values[i] >= cutoff:
            categorical_map[categories.index[i]] = categories.index[i]
        else:
            categorical_map[categories.index[i]] = 'others'
    return categorical_map
```

[307]:
```
mileage = shorten_categories(df.Mileage.value_counts(), 400)
df['Mileage'] = df['Mileage'].map(mileage)
df.Mileage.value_counts()
```

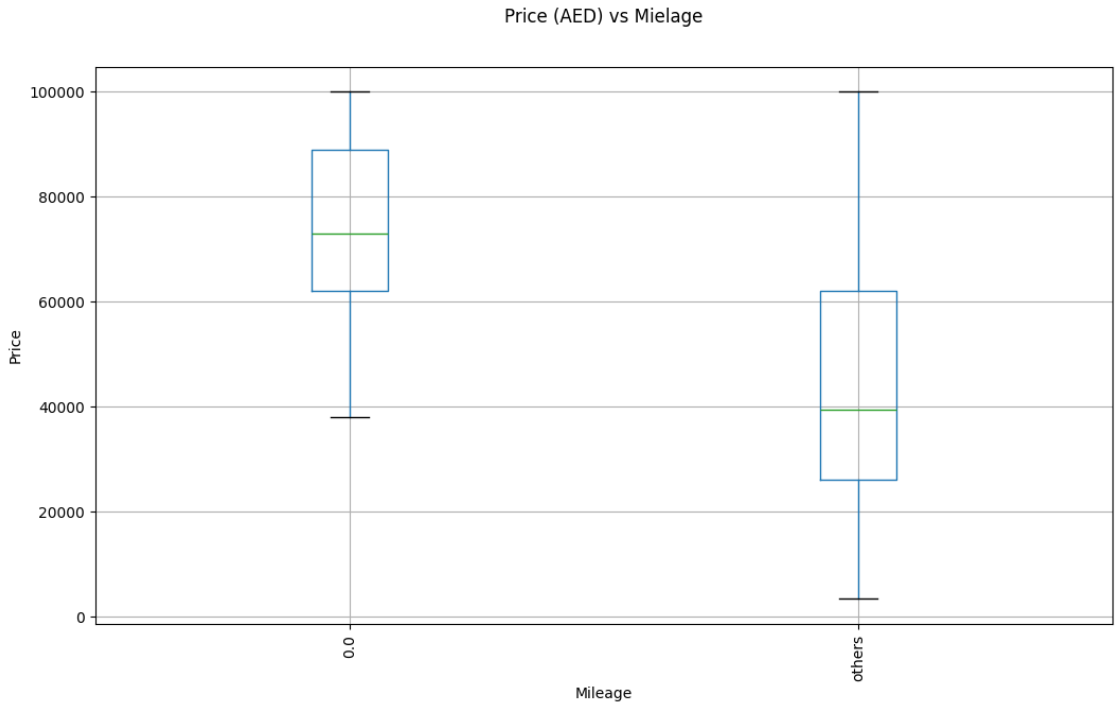
[307]:
```
Mileage
others    6631
0.0       1375
Name: count, dtype: int64
```

[308]:
```
fig, ax = plt.subplots(1,1, figsize = (12, 7))
df.boxplot('Price', 'Mileage', ax = ax)
plt.suptitle('Price (AED) vs Mielage')
plt.title('')
plt.ylabel('Price')
plt.xticks(rotation=90)
plt.show()
```



[309]:
```
df = df[df["Price"] <= 100000 ]
df = df[df["Price"] >= 1000]
```

[310]:
```
fig, ax = plt.subplots(1,1, figsize = (12, 7))
df.boxplot('Price', 'Mileage', ax = ax)
plt.suptitle('Price (AED) vs Mielage')
plt.title('')
plt.ylabel('Price')
plt.xticks(rotation=90)
plt.show()
```



[311]:
```
df["Specs"].unique()
```

[311]:
```
array(['GCC Specs', 'American Specs', 'Japanese Specs', 'Other',
       'Korean Specs', 'Canadian Specs', 'European Specs',
       'Chinese Specs'], dtype=object)
```

[312]:
```
def map_specs(spec):
    if spec == 'GCC Specs':
        return 'GCC'
    else:
        return 'Imported'
```

[313]:
```
df['Specs'] = df['Specs'].apply(map_specs)
```

[314]:
```
df["Specs"].unique()
```

[314]:
```
array(['GCC', 'Imported'], dtype=object)
```

[315]:
```
df["Car Brand"].unique()
```

[315]:
```
array(['Nissan', 'Toyota', 'Ford', 'Chevrolet', 'Honda', 'Kia', 'Hyundai'],
      dtype=object)
```

[316]:
```
df["Car Model"].unique()
```

[316]:
```
array(['Altima', 'Camry', 'Focus', 'Echo', 'Epica', 'Sunny', 'Accord',
       'Figo', 'Tiida', 'Fusion', 'Edge', 'Pathfinder', 'Versa', 'Pickup',
       'X-Trail', 'Fiesta', 'Murano', 'Rio', 'Yaris', 'Optima', 'Corolla',
       'Explorer', 'Aveo', 'Spark', 'Picanto', 'MR-V', 'Avalanche',
       'Micra', 'Tucson', 'Civic', 'Qashqai', 'XA', 'i10', 'Accent',
       'Mustang', 'Taurus', 'Malibu', 'Sonata', 'Maxima', 'Flex', 'Other',
       'Sienna', 'Coupe', 'Sportage', 'Azera', 'Sorento', 'Soul', 'Rav 4',
       'Grand i10', 'Santa Fe', 'CR-V', 'Mohave', 'Cadenza', 'Armada',
       'Cruze', 'Captiva', 'Elantra', 'Urvan', 'Sentra', 'H1', 'Carnival',
       'Escape', 'Ecosport', 'Trax', 'i20', 'Pilot', 'Jazz', 'Innova',
       'Veloster', 'Juke', 'Avanza', 'Hilux', 'Leaf', 'Expedition',
       'Navara', 'Cerato', 'Xterra', 'Avalon', 'Zelas', 'Odyssey',
       'Forte', 'IQ', 'Genesis', 'Prius', 'Avanti', 'Quest', 'i30',
       'Crosstour', 'Carens', 'City', 'Sedona', 'Rogue', 'Hiace',
       'Land Cruiser', 'Ranger', 'Prado', 'Pegas', 'Camaro', 'Tacoma',
       'Impala', 'Silverado', 'Fortuner', 'Previa', 'Crown', 'Cressida',
       'Aurion', 'Creta', 'F-Series Pickup', 'Kicks', 'Van', 'FJ Cruiser',
       'Ioniq', '370z', '86', '4Runner', 'Kona', 'Tahoe', 'K900', 'Venue',
       'Alphard', 'Tundra', 'Transit', 'Lumina', 'Patrol', 'Odyssey J',
       'Escort', 'Equinox', 'C-HR', 'K3', 'HR-V', 'Quoris', 'Rush', 'K5',
       'Seltos', 'Groove', 'Bongo', 'Caprice', 'Raize', 'Grandeur',
       'Stinger', 'Sequoia', 'Grand Santa Fe', 'Land Cruiser 70',
       'Traverse', 'Bronco', 'Highlander', '280ZX', 'Telluride', 'Sonet',
       'Menlo', 'Corolla Cross', 'ENS1', 'Titan', 'Palisade', 'Veloz',
       'Passport', 'Skyline', 'Urban Cruiser', 'Land Cruiser 76 series',
       'Santa Cruz', 'Venza', 'Thunderbird', 'Blazer', 'Staria'],
      dtype=object)
```

[317]:
```
from sklearn.preprocessing import LabelEncoder

le_Brand = LabelEncoder()
df['Car Brand'] = le_Brand.fit_transform(df['Car Brand'])
df['Car Brand'].unique()
```

[317]:
```
array([5, 6, 1, 0, 2, 4, 3])
```

[318]:
```
le_Model = LabelEncoder()
df['Car Model'] = le_Model.fit_transform(df['Car Model'])
df['Car Model'].unique()
```

[318]:
```
array([  7,  23,  55,  40,  44, 135,   5,  53, 141,  58,  42, 100, 155,
       104, 156,  52,  91, 115, 159,  96,  31,  49,  14, 131, 103,  85,
        10,  89, 146,  30, 109, 157, 161,   4,  92, 138,  86, 127,  87,
        54,  97, 124,  33, 132,  15, 129, 130, 114,  61, 119,  20,  90,
        21,   8,  38,  25,  43, 149, 122,  64,  27,  46,  41, 145, 162,
       105,  73,  71, 151,  74,  13,  68,  83,  48,  93,  28, 158,  11,
       160,  94,  56,  69,  59, 108,  12, 110, 163,  36,  26,  29, 120,
```

```
       116,  66,  80, 113, 106, 102,  22, 136,  70, 125,  57, 107,  37,
        34,   9,  35,  50,  78, 150,  51,  72,   1,   3,   2,  79, 137,
        77, 153,   6, 147, 143,  84, 101,  95,  47,  45,  19,  75,  65,
       111, 117,  76, 121,  63,  17,  24, 112,  62, 134, 123,  60,  81,
       144,  18,  67,   0, 139, 128,  88,  32,  39, 142,  98, 152,  99,
       126, 148,  82, 118, 154, 140,  16, 133])
```

[319]:
```
X = df.drop("Price", axis=1)
y = df["Price"]
```

[320]:
```
y.unique()
```

[320]:
```
array([  3500.,   5500.,   6000.,   7000.,   7500.,   8000.,   8500.,
         8700.,   8800.,   9000.,   9500.,   9850.,  10000.,  10500.,
        10900.,  10990.,  11000.,  11200.,  11300.,  11450.,  11500.,
        11850.,  11900.,  12000.,  12200.,  12250.,  12500.,  12700.,
        12800.,  12900.,  12950.,  12999.,  13000.,  13500.,  13850.,
        13900.,  14000.,  14200.,  14500.,  14600.,  14700.,  14800.,
        14900.,  14999.,  15000.,  15400.,  15500.,  15700.,  15800.,
        15900.,  15999.,  16000.,  16450.,  16499.,  16500.,  16800.,
        16900.,  17000.,  17200.,  17500.,  17700.,  17900.,  17999.,
        18000.,  18250.,  18300.,  18500.,  18700.,  18900.,  18999.,
        19000.,  19200.,  19250.,  19300.,  19500.,  19600.,  19700.,
        19800.,  19900.,  19999.,  20000.,  20500.,  20800.,  20900.,
        20991.,  20999.,  21000.,  21500.,  21600.,  21900.,  21999.,
        22000.,  22493.,  22500.,  22900.,  22975.,  22998.,  22999.,
        23000.,  23299.,  23400.,  23497.,  23500.,  23700.,  23800.,
        23900.,  23999.,  24000.,  24300.,  24400.,  24499.,  24500.,
        24800.,  24900.,  24950.,  24997.,  24999.,  25000.,  25400.,
        25500.,  25600.,  25900.,  25999.,  26000.,  26500.,  26600.,
        26700.,  26800.,  26999.,  27000.,  27250.,  27300.,  27500.,
        27800.,  27900.,  27999.,  28000.,  28350.,  28500.,  28700.,
        28800.,  28900.,  29000.,  29500.,  29800.,  29900.,  29990.,
        29999.,  30000.,  30500.,  30900.,  31000.,  31500.,  31900.,
        31999.,  32000.,  32100.,  32200.,  32400.,  32500.,  32600.,
        32700.,  32900.,  32999.,  33000.,  33400.,  33500.,  33700.,
        33750.,  33800.,  33900.,  33990.,  33999.,  34000.,  34400.,
        34500.,  34900.,  34950.,  34999.,  35000.,  35500.,  35800.,
        35900.,  35990.,  35998.,  35999.,  36000.,  36500.,  36700.,
        36800.,  36900.,  36990.,  37000.,  37500.,  37800.,  37900.,
        37999.,  38000.,  38500.,  38700.,  38800.,  38900.,  38999.,
        39000.,  39500.,  39700.,  39800.,  39900.,  39990.,  39999.,
        40000.,  40200.,  40400.,  40500.,  40900.,  40999.,  41000.,
        41111.,  41500.,  41999.,  42000.,  42500.,  42900.,  42999.,
        43000.,  43050.,  43500.,  43900.,  43999.,  44000.,  44100.,
        44300.,  44400.,  44500.,  44700.,  44900.,  44999.,  45000.,
        45150.,  45499.,  45500.,  45900.,  45999.,  46000.,  46500.,
```

```
       46900.,  46999.,  47000.,  47500.,  47900.,  47950.,  47999.,
       48000.,  48500.,  48900.,  48999.,  49000.,  49500.,  49530.,
       49900.,  49990.,  49999.,  50000.,  50500.,  50900.,  51000.,
       51500.,  51675.,  51700.,  51900.,  52000.,  52500.,  52600.,
       52900.,  52999.,  53000.,  53500.,  53750.,  53900.,  54000.,
       54500.,  54900.,  54950.,  54997.,  54999.,  55000.,  55500.,
       55700.,  55900.,  55999.,  56000.,  56500.,  56600.,  56800.,
       56900.,  56999.,  57000.,  57100.,  57500.,  57600.,  57900.,
       57990.,  57999.,  58000.,  58500.,  58600.,  58900.,  58999.,
       59000.,  59500.,  59750.,  59800.,  59900.,  59950.,  59999.,
       60000.,  60001.,  60500.,  60900.,  61000.,  61500.,  61900.,
       61999.,  62000.,  62500.,  62800.,  62900.,  62990.,  62999.,
       63000.,  63500.,  63900.,  63999.,  64000.,  64500.,  64750.,
       64900.,  64999.,  65000.,  65400.,  65500.,  65999.,  66000.,
       66500.,  66900.,  66999.,  67000.,  67500.,  67750.,  67770.,
       67900.,  67999.,  68000.,  68500.,  68900.,  68999.,  69000.,
       69111.,  69499.,  69500.,  69900.,  69990.,  69999.,  70000.,
       70100.,  70900.,  70999.,  71000.,  71500.,  71800.,  71900.,
       71999.,  72000.,  72500.,  72900.,  72999.,  73000.,  73500.,
       73900.,  73999.,  74000.,  74500.,  74900.,  74999.,  75000.,
       75500.,  75900.,  75999.,  76000.,  76500.,  76999.,  77000.,
       77300.,  77500.,  77800.,  77900.,  77999.,  78000.,  78500.,
       78750.,  78900.,  79000.,  79500.,  79800.,  79900.,  79950.,
       79990.,  79999.,  80000.,  80500.,  81000.,  81500.,  81900.,
       81999.,  82000.,  82250.,  82500.,  82900.,  82999.,  83000.,
       83500.,  83999.,  84000.,  84500.,  84900.,  84999.,  85000.,
       85400.,  85500.,  85900.,  85999.,  86000.,  86500.,  86900.,
       86999.,  87000.,  87500.,  87900.,  87990.,  87999.,  88000.,
       88200.,  88500.,  89000.,  89500.,  89750.,  89900.,  89990.,
       89999.,  90000.,  90500.,  90900.,  91000.,  91500.,  91900.,
       91999.,  92000.,  92499.,  92500.,  92750.,  92900.,  92990.,
       93000.,  93500.,  93800.,  94000.,  94300.,  94498.,  94500.,
       94900.,  95000.,  95900.,  95999.,  96000.,  96500.,  96600.,
       96900.,  97000.,  97500.,  97900.,  98000.,  98500.,  99000.,
       99500.,  99750.,  99900.,  99990.,  99999., 100000.])
```

[321]:
```
X = pd.get_dummies(X)
print(X.dtypes)
```

```
Car Brand          int64
Car Model          int64
Production Year    int64
Age                int64
Mileage_0.0         bool
Mileage_others      bool
Specs_GCC           bool
Specs_Imported      bool
```

```
dtype: object
```

[322]:
```
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()
linear_reg.fit(X, y)
```

[322]:
```
LinearRegression()
```

[323]:
```
y_pred = linear_reg.predict(X)
```

[324]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
error = np.sqrt(mean_squared_error(y, y_pred))
```

[325]:
```
error
```

[325]:
```
np.float64(20301.358503457435)
```

[326]:
```
from sklearn.tree import DecisionTreeRegressor
dec_tree_reg = DecisionTreeRegressor(random_state=0)
dec_tree_reg.fit(X, y.values)
```

[326]:
```
DecisionTreeRegressor(random_state=0)
```

[327]:
```
y_pred = dec_tree_reg.predict(X)
```

[328]:
```
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

```
$6,168.18
```

[329]:
```
from sklearn.ensemble import RandomForestRegressor
random_forest_reg = RandomForestRegressor(random_state=0)
random_forest_reg.fit(X, y.values)
```

[329]:
```
RandomForestRegressor(random_state=0)
```

[330]:
```
y_pred = random_forest_reg.predict(X)
```

[331]:
```
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

```
$6,441.89
```

[332]:
```
from sklearn.model_selection import GridSearchCV

max_depth = [None, 2,4,6,8,10,12]
parameters = {"max_depth": max_depth}
```

```
regressor = DecisionTreeRegressor(random_state=0)
gs = GridSearchCV(regressor, parameters, scoring='neg_mean_squared_error')
gs.fit(X, y.values)
```

[332]:
```
GridSearchCV(estimator=DecisionTreeRegressor(random_state=0),
             param_grid={'max_depth': [None, 2, 4, 6, 8, 10, 12]},
             scoring='neg_mean_squared_error')
```

[333]:
```
regressor = gs.best_estimator_

regressor.fit(X, y.values)
y_pred = regressor.predict(X)
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

```
$6,168.18
```

[334]:
```
X
```

[334]:

	Car Brand	Car Model	Production Year	Age	Mileage_0.0	Mileage_others	Specs_GCC	Specs_Imported
0	5	7	2005	19	False	True	True	False
1	6	23	1999	25	False	True	True	False
2	1	55	2006	18	False	True	True	False
3	6	40	2005	19	False	True	True	False
4	0	44	2009	15	False	True	False	True
…	…	…	…	…	…	…	…	…
5734	1	92	2017	7	False	True	True	False
5735	1	92	2007	17	False	True	False	True
5736	1	92	2014	10	False	True	True	False
5737	1	92	2020	4	False	True	False	True
5738	5	156	2023	1	False	True	True	False

[5739 rows x 8 columns]

[335]:
```
X = np.array([["Ford", 'Mustang', '2009', 15, '48000', 0, 1, 0 ]])
X
```

[335]:
```
array([['Ford', 'Mustang', '2009', '15', '48000', '0', '1', '0']],
      dtype='<U21')
```

[336]:
```
X[:, 0] = le_Brand.transform(X[:,0])
X[:, 1] = le_Model.transform(X[:,1])
X = X.astype(float)
X
```

[336]:
```
array([[1.000e+00, 9.200e+01, 2.009e+03, 1.500e+01, 4.800e+04, 0.000e+00,
        1.000e+00, 0.000e+00]])
```

[337]:
```
y_pred = dec_tree_reg.predict(X)
y_pred
```

```
C:\Users\lab\anaconda3\envs\ml\Lib\site-packages\sklearn\base.py:493:
UserWarning: X does not have valid feature names, but DecisionTreeRegressor was
fitted with feature names
  warnings.warn(
```

[337]:
```
array([25000.])
```

[338]:
```
import pickle
```

[339]:
```
data = {"model": dec_tree_reg, "Car Brand": le_Brand, "Car Model": le_Model}
with open('saved_steps.pkl', 'wb') as file:
    pickle.dump(data, file)
```

[340]:
```
with open('saved_steps.pkl', 'rb') as file:
    data = pickle.load(file)

regressor_loaded = data["model"]
le_Brand = data["Car Brand"]
le_Model = data["Car Model"]
```

[341]:
```
y_pred = regressor_loaded.predict(X)
y_pred
```

```
C:\Users\lab\anaconda3\envs\ml\Lib\site-packages\sklearn\base.py:493:
UserWarning: X does not have valid feature names, but DecisionTreeRegressor was
fitted with feature names
  warnings.warn(
```

[341]:
```
array([25000.])
```

[ ]:

[ ]: