data_cleaning_and_preparation

July 13, 2025

1 Used Car Data Analysis Project

by Sam Buwalda | Portfolio Project, 2025

2 Used Car Data Analysis Project

By Sam Buwalda — Portfolio Project, 2025

This notebook covers the **data loading, inspection, and cleaning steps** for a dataset of 426,880 used car listings scraped from Craigslist. It is part of a larger project that analyzes pricing trends, brand depreciation, and feature importance based on realistic business questions.

2.1 Dataset Source

- Kaggle: Used Cars Dataset
- 426,880 rows before cleaning
- 26 columns before cleaning

2.2 Business Questions

1. Descriptive: What are the most common categorical traits of cars priced above $20,000?

2. Diagnostic: How does car age affect price?

3. Descriptive + Diagnostic: Which car brands retain their value best over time?

4. Diagnostic: How does fuel type and transmission affect car price?

5. Diagnostic + Predictive: What factors most influence the price of a used car (based on the available data)?

2.3 Cleaning Objectives

The cleaning decisions are guided by the analytical goals of the full project, which includes questions about price drivers, brand depreciation, and feature-value relationships.

- Dropping irrelevant features (based on the business questions) or high-missing-value columns
- Filtering out unrealistic prices and odometer readings
- Removing duplicate rows

- Dropping rows with missing values in key columns (like year, manufacturer, transmission, etc.)
- Keeping only features relevant to the business questions

2.4 Importing and loading CSV file

[1]:
```python
# Import pandas for data loading, inspection, and cleaning
import pandas as pd

# Load the CSV file and load into DataFrame 'df'

df = pd.read_csv("../Data Files/vehicles.csv", low_memory=False)

# Display general information about the dataset: column names, data types,␣
↪non-null counts, and memory usage.

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426880 entries, 0 to 426879
Data columns (total 26 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            426880 non-null  int64
 1   url           426880 non-null  object
 2   region        426880 non-null  object
 3   region_url    426880 non-null  object
 4   price         426880 non-null  int64
 5   year          425675 non-null  float64
 6   manufacturer  409234 non-null  object
 7   model         421603 non-null  object
 8   condition     252776 non-null  object
 9   cylinders     249202 non-null  object
 10  fuel          423867 non-null  object
 11  odometer      422480 non-null  float64
 12  title_status  418638 non-null  object
 13  transmission  424324 non-null  object
 14  VIN           265838 non-null  object
 15  drive         296313 non-null  object
 16  size          120519 non-null  object
 17  type          334022 non-null  object
 18  paint_color   296677 non-null  object
 19  image_url     426812 non-null  object
 20  description   426810 non-null  object
 21  county        0 non-null       float64
 22  state         426880 non-null  object
```

 ```
 23  lat           420331 non-null  float64
 24  long          420331 non-null  float64
 25  posting_date  426812 non-null  object
dtypes: float64(5), int64(2), object(19)
memory usage: 84.7+ MB
```

2.5 Data Cleaning & Preparation

In this section, we clean the dataset to prepare it for analysis. Steps include removing duplicates, handling missing values, filtering outliers, and dropping irrelevant columns.

2.5.1 Initial Inspection

[2]:
```
# Show the number of rows and columns in the dataset to understand its overall␣
↪size.

df.shape
```

[2]: (426880, 26)

[3]:
```
# Display the first 5 rows to visually inspect the structure and content of the␣
↪dataset.
# This helps confirm that the file was loaded correctly and gives an early look␣
↪at the values.

df.head()
```

[3]:
```
           id                                                 url  \
0  7222695916  https://prescott.craigslist.org/cto/d/prescott…
1  7218891961  https://fayar.craigslist.org/ctd/d/bentonville…
2  7221797935  https://keys.craigslist.org/cto/d/summerland-k…
3  7222270760  https://worcester.craigslist.org/cto/d/west-br…
4  7210384030  https://greensboro.craigslist.org/cto/d/trinit…

                  region                         region_url  price  year  \
0               prescott    https://prescott.craigslist.org   6000   NaN
1           fayetteville       https://fayar.craigslist.org  11900   NaN
2           florida keys        https://keys.craigslist.org  21000   NaN
3  worcester / central MA   https://worcester.craigslist.org   1500   NaN
4             greensboro  https://greensboro.craigslist.org   4900   NaN

  manufacturer model condition cylinders  … size  type paint_color  \
0          NaN   NaN       NaN       NaN  …  NaN   NaN         NaN
1          NaN   NaN       NaN       NaN  …  NaN   NaN         NaN
2          NaN   NaN       NaN       NaN  …  NaN   NaN         NaN
3          NaN   NaN       NaN       NaN  …  NaN   NaN         NaN
4          NaN   NaN       NaN       NaN  …  NaN   NaN         NaN
```

```
  image_url description county state lat long posting_date
0       NaN         NaN    NaN    az NaN  NaN          NaN
1       NaN         NaN    NaN    ar NaN  NaN          NaN
2       NaN         NaN    NaN    fl NaN  NaN          NaN
3       NaN         NaN    NaN    ma NaN  NaN          NaN
4       NaN         NaN    NaN    nc NaN  NaN          NaN

[5 rows x 26 columns]
```

[4]:
```python
# Identify the number of missing values in each column to assess data quality and decide on cleaning strategy.

df.isnull().sum()
```

[4]:
```
id                   0
url                  0
region               0
region_url           0
price                0
year              1205
manufacturer     17646
model             5277
condition       174104
cylinders       177678
fuel              3013
odometer          4400
title_status      8242
transmission      2556
VIN             161042
drive           130567
size            306361
type             92858
paint_color     130203
image_url           68
description         70
county          426880
state                0
lat               6549
long              6549
posting_date        68
dtype: int64
```

[5]:
```python
# Generate summary statistics for numerical columns to detect potential outliers and understand value distributions.

df.describe()
```

```
[5]:                  id         price           year      odometer  county  \
count  4.268800e+05  4.268800e+05  425675.000000  4.224800e+05     0.0
mean   7.311487e+09  7.519903e+04    2011.235191  9.804333e+04     NaN
std    4.473170e+06  1.218228e+07       9.452120  2.138815e+05     NaN
min    7.207408e+09  0.000000e+00    1900.000000  0.000000e+00     NaN
25%    7.308143e+09  5.900000e+03    2008.000000  3.770400e+04     NaN
50%    7.312621e+09  1.395000e+04    2013.000000  8.554800e+04     NaN
75%    7.315254e+09  2.648575e+04    2017.000000  1.335425e+05     NaN
max    7.317101e+09  3.736929e+09    2022.000000  1.000000e+07     NaN

                 lat           long
count  420331.000000  420331.000000
mean       38.493940     -94.748599
std         5.841533      18.365462
min       -84.122245    -159.827728
25%        34.601900    -111.939847
50%        39.150100     -88.432600
75%        42.398900     -80.832039
max        82.390818     173.885502
```

2.5.2 Remove Duplicates

```python
[6]: # Check for and remove any duplicate rows to avoid skewing the analysis.

print("Number of duplicates:", df.duplicated().sum())
df = df.drop_duplicates()

# Confirm new shape after dropping

print("New shape:", df.shape)
```

```
Number of duplicates: 0
New shape: (426880, 26)
```

2.5.3 Drop (Irrelevant) Columns with High Missing Values

```python
[7]: # Drop columns that are mostly missing ('county' and 'size')
# 'errors="ignore"' ensures the code doesn't break if a column is already␣
↪dropped earlier.

df = df.drop(columns=['county', 'size'], errors='ignore')
```

```python
[8]: # Confirm new shape after dropping 'county' column

print("New shape:", df.shape)
```

```
New shape: (426880, 24)
```

[9]:
```python
# Drop unnecessary or low-value columns that do not contribute meaningfully to␣
↪the current analysis goals

df = df.drop(columns=[

    # 'url' and 'region_url' are only useful for linking externally to the␣
↪original listings;
    # they have no analytical value for understanding vehicle pricing or␣
↪features.
    'url',
    'region_url',

    # 'image_url' is purely visual and not relevant for data-driven analysis.
    'image_url',

    # 'VIN' is a unique identifier per vehicle. It is not informative for␣
↪analysis unless checking for duplicates,
    # which we've already handled. It adds no predictive or explanatory power.
    'VIN',

    # 'condition' is around 33% missing and subjective in nature (e.g., "good"␣
↪vs "like new").
    # Removing it avoids potential inconsistencies and row loss.
    'condition',

    # 'cylinders' is missing in ~36% of the data. Including it would require␣
↪dropping over a third of the dataset.
    # It's not part of our final questions, so we drop it to retain data␣
↪integrity.
    'cylinders',

    # 'drive' (e.g., AWD, FWD, RWD) is missing in ~30% of the data. While it␣
↪might influence price in certain use cases,
    # it is not part of our business questions and is excluded to preserve row␣
↪count and clarity.
    'drive',

    # 'paint_color' is missing in ~28% of rows and is a mostly aesthetic␣
↪feature.
    # Since our goal is to analyze core pricing factors (like year, mileage,␣
↪brand), it's excluded.
    'paint_color'

], errors='ignore')  # 'errors=ignore' ensures no crash if a column was already␣
↪removed earlier
```

[10]:
```
# Check current columns to confirm unnecessary ones were successfully dropped

print("  Remaining columns in the DataFrame:")
print(df.columns.tolist())
```

```
 Remaining columns in the DataFrame:
['id', 'region', 'price', 'year', 'manufacturer', 'model', 'fuel', 'odometer',
'title_status', 'transmission', 'type', 'description', 'state', 'lat', 'long',
'posting_date']
```

2.5.4 Filter Unrealistic Price & Odometer Readings

[11]:
```
# Filter out vehicle listings with unrealistic prices.
# Prices below $500 or above $120,000 are likely data entry errors or rare edge
↪cases.

df = df[(df['price'] >= 500) & (df['price'] <= 120000)]

# Remove vehicle listings with unrealistic (used car) odometer readings of 0 or
↪over 300,000 miles.
# Odometer values of 0 likely represent missing data (since this is about used
↪cars), and over 300,000 miles are extremely rare.

df = df[(df['odometer'] > 0) & (df['odometer'] < 300000)]
```

[12]:
```
# Confirm dataset statistics after filtering to ensure the changes were applied
↪correctly and data looks reasonable.

df.describe()
```

[12]:
```
                 id          price           year       odometer  \
count  3.782430e+05  378243.000000  377161.000000  378243.000000
mean   7.311470e+09   19261.528972    2011.071548   91917.520419
std    4.390466e+06   14545.539541       9.434529   61678.501355
min    7.301583e+09     500.000000    1900.000000       1.000000
25%    7.308073e+09    7900.000000    2008.000000   38335.000000
50%    7.312575e+09   15987.000000    2013.000000   87155.000000
75%    7.315245e+09   27990.000000    2017.000000  135000.000000
max    7.317101e+09  120000.000000    2022.000000  299999.000000

                 lat           long
count  374895.000000  374895.000000
mean       38.523599     -94.255949
std         5.845846      18.083272
min       -81.838232    -159.719900
25%        34.720000    -110.890427
50%        39.254962     -87.971900
```

```
75%         42.364188     -80.820900
max         82.390818     173.885502
```

2.5.5 Final Cleaning: Drop Rows with & filling Missing Info

[13]:
```
# Drop rows with missing values in key columns needed for analysis

df = df.dropna(subset=[

    # 'year' is essential for calculating vehicle age, which directly affects␣
 ↪price and depreciation.
    'year',

    # 'manufacturer' is required to analyze brand-level trends and pricing.
    'manufacturer',

    # 'model' provides necessary detail within each brand for comparing␣
 ↪specific vehicles.
    'model',

    # 'fuel' type impacts price significantly (e.g., electric vs gas vs diesel).
    'fuel',

    # 'title_status' affects vehicle value and trustworthiness - a salvaged␣
 ↪title lowers price.
    'title_status',

    # 'transmission' (auto/manual) influences price and is important for buyer␣
 ↪preference analysis.
    'transmission',

    # 'lat' and 'long' allow for location-based analysis, regional trends, and␣
 ↪mapping.
    'lat', 'long'

])
```

[14]:
```
# Fill missing descriptions with a placeholder string

df['description'] = df['description'].fillna('No description provided')
```

2.5.6 Confirm Final Dataset Structure

[15]:
```
# Final confirmation after cleaning

# Check the final shape of the dataset
```

```python
print("  Final dataset shape:", df.shape)

# Confirm that there are no missing values remaining

print("\n  Missing values per column (should all be 0, except for 'type'):")
print(df.isnull().sum())
```

```
 Final dataset shape: (347305, 16)

 Missing values per column (should all be 0, except for 'type'):
id                  0
region              0
price               0
year                0
manufacturer        0
model               0
fuel                0
odometer            0
title_status        0
transmission        0
type            73849
description         0
state               0
lat                 0
long                0
posting_date        0
dtype: int64
```

2.6 Export to CSV in Data Files folder

[16]:
```python
df.to_csv("../Data Files/vehicles_cleaned.csv", index=False)
```

[ ]:
```python

```