Demo Part-1

CMPT -3830

Machine Learning Work Integrated Project 1

Agenda

Introduction

Dataset Overview

Problem Statement

Understanding Exploratory Data Analysis (EDA)

EDA on Go Auto Dataset

Results & Insights

Conclusion

Open floor for any questions or clarifications

AUTO CODERS

Auto Coder Team

Aashish Arora

Model evaluator

Manveen Kaur

Project Manager
Scrum Master

Sahil Chand

Deadline Manager
Code Developer

Getachew

Lead Developer

Love Maan

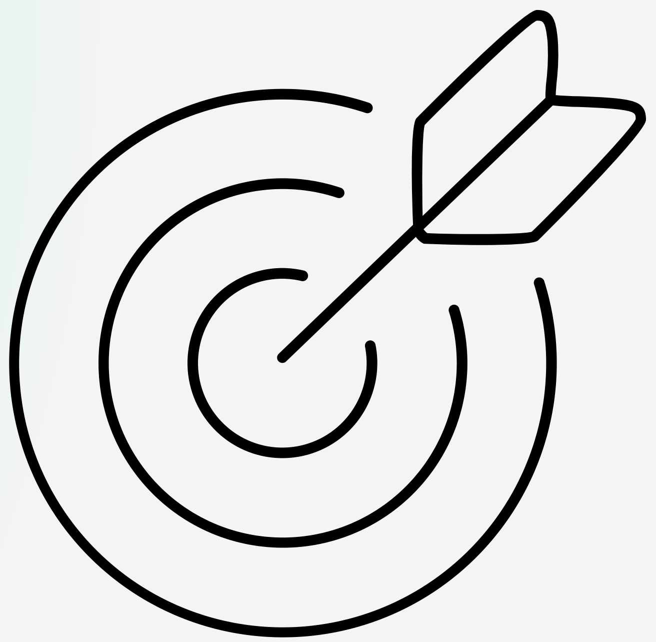
Interpretator

Tip: Collaboration makes teamwork easier! Click "Share" and invite your teammates to fill this up. Use this whiteboard page for bulletins, brainstorms, and other fun team ideas!

Go Auto-Project Goal

AUTO CODERS

Analyze vehicle makes using clustering techniques to identify patterns based on factors such as price, mileage, and age. The goal is to provide insights into similarities and differences among various makes, offering dealerships strategic recommendations for cross-selling similar vehicles from different brands.

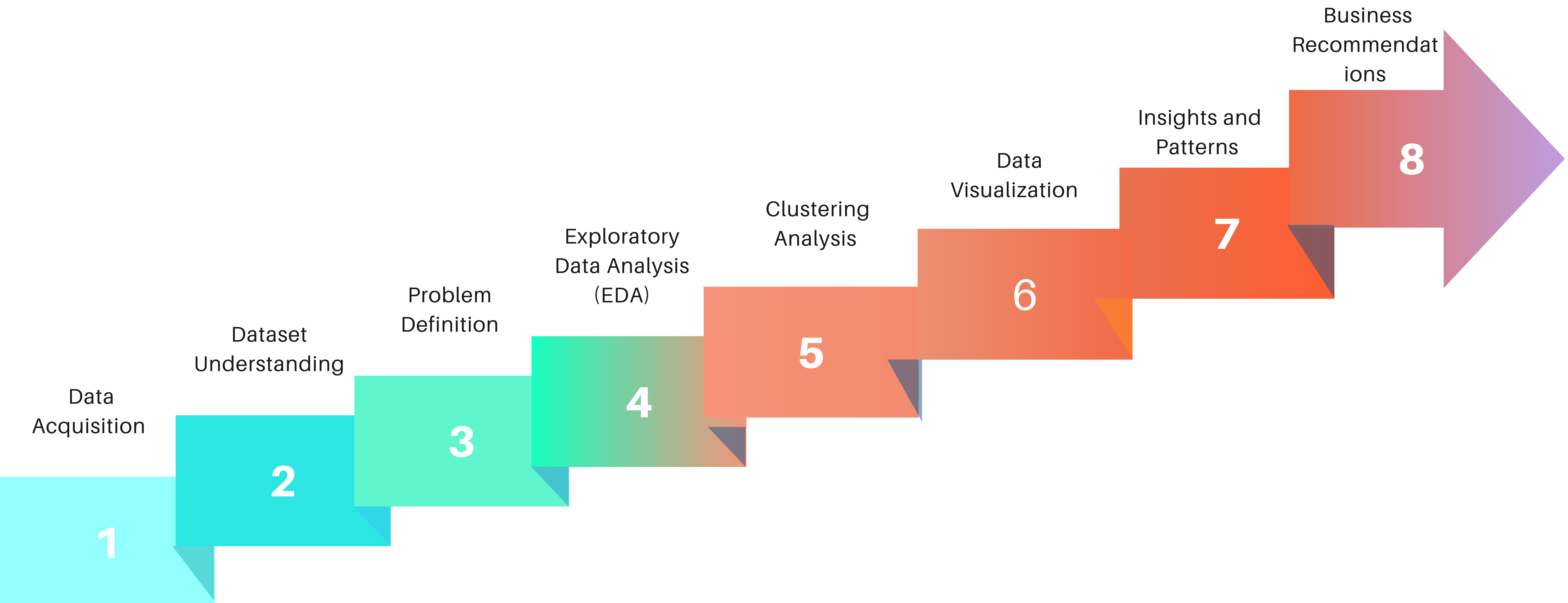


Business Impact

Unlock Cross-Selling	Boost sales by strategically offering complementary vehicle brands based on clustering insights
Smarter Pricing	Adjust pricing within clusters (premium, mid-range, budget) to stay competitive and drive profits
Optimized Inventory	Stock the right vehicles based on demand, turning data into smarter inventory decisions

AUTO CODERS

PROJECT OVERVIEW



AUTO CODERS

Dataset Description

- **Dataset Size:** 145,114 vehicle listings
- **Source:** Compiled by Go Auto's Business Intelligence Team using APIs from the Canadian Black Book (CBB)
- **Timeframe:** Contains both active and sold vehicle listings.
- **Geography:** Data from dealerships, mainly in Edmonton, Alberta

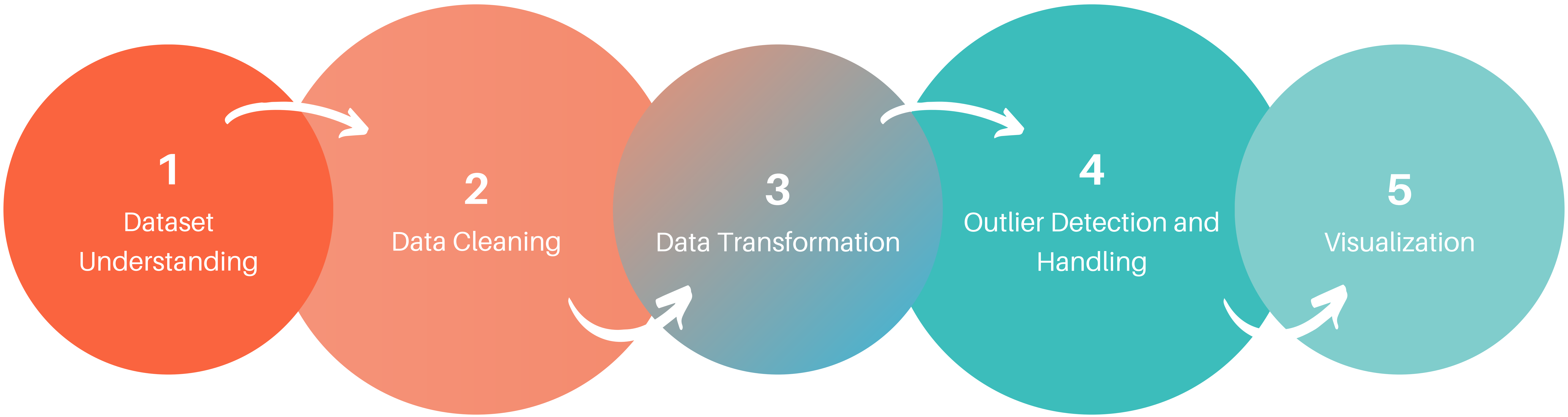
AUTO CODERS

Key Features

Vehicle Information	Dealership Information	Vehicle Listing Information	VIN-Based Attributes	Additional Data Points
• Make, Model, Year, Mileage, Price, MSRP (Manufacturer's Suggested Retail Price) • Vehicle Identification Number (VIN) and Style • Certified Status, Leather, Navigation, and Exterior Colo	• Dealer ID, Dealer Name, Location (City, Province, Postal Code) • Dealer Type and Stock Type (e.g., New, Used)	• Listing ID, URL, First Date on Market, Days on Market • Number of Price Changes and Price History	• Information extracted from VIN, such as Engine Type, Transmission, Drivetrain, and Fuel Type	• Distance to Dealer, Listing Dropoff Date, and Location Scor

AUTO CODERS

Data Analysis for Optimized Dealership Strategies



DATA SET OVERVIEW

AUTO CODERS

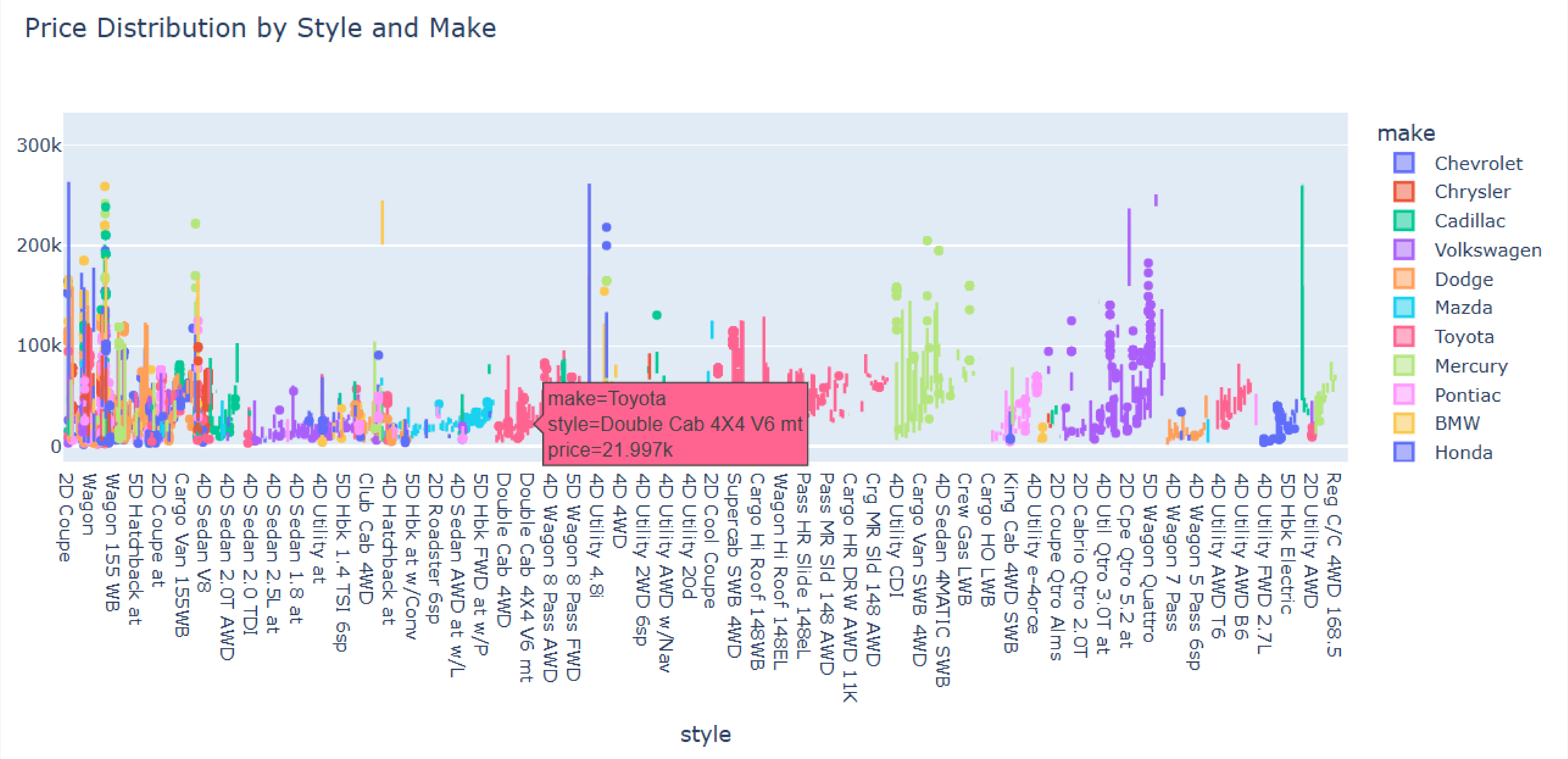
Column we are working with

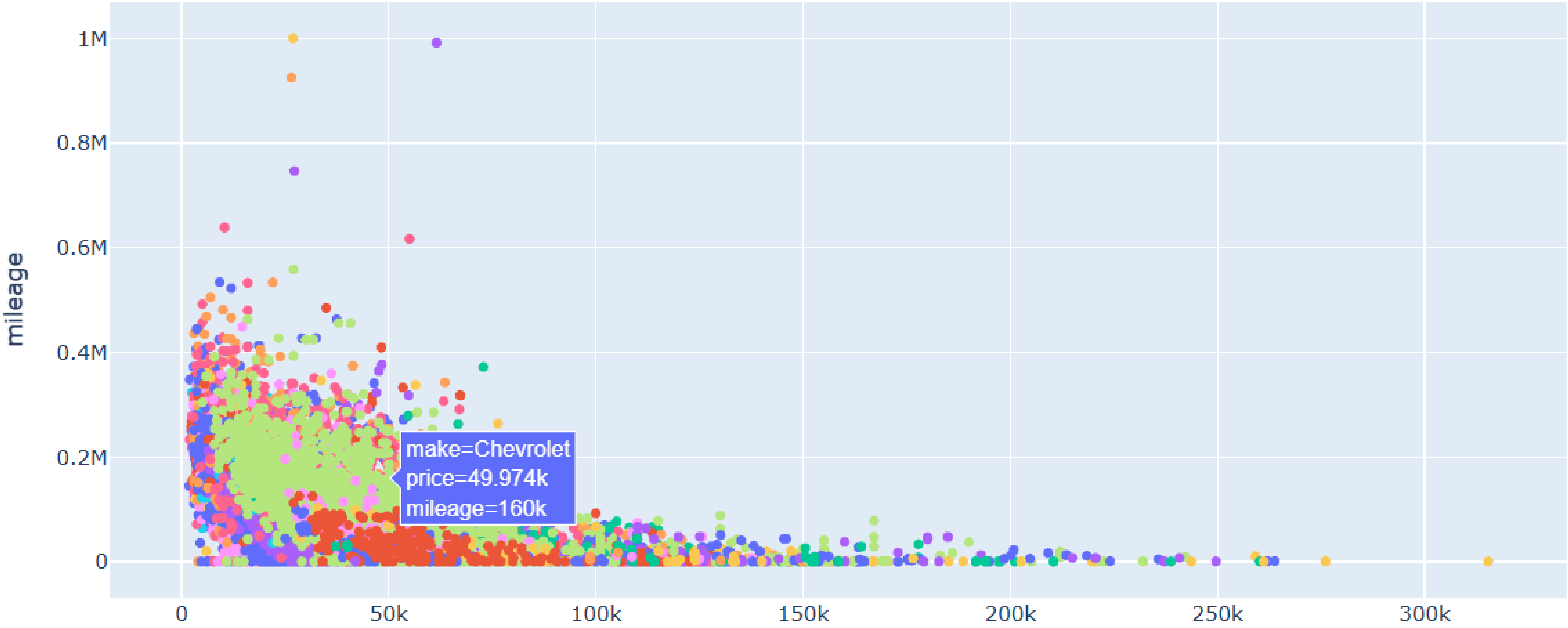
1. listing_heading
2. listing_type
3. dealer_name
4. dealer_city
5. dealer_province
6. dealer_postal_code
7. dealer_phone
8. stock_type
9. Mileage
10. Price
11. MSRP
12. Model Year
13. Make
14. Model Style

Column we are dropping

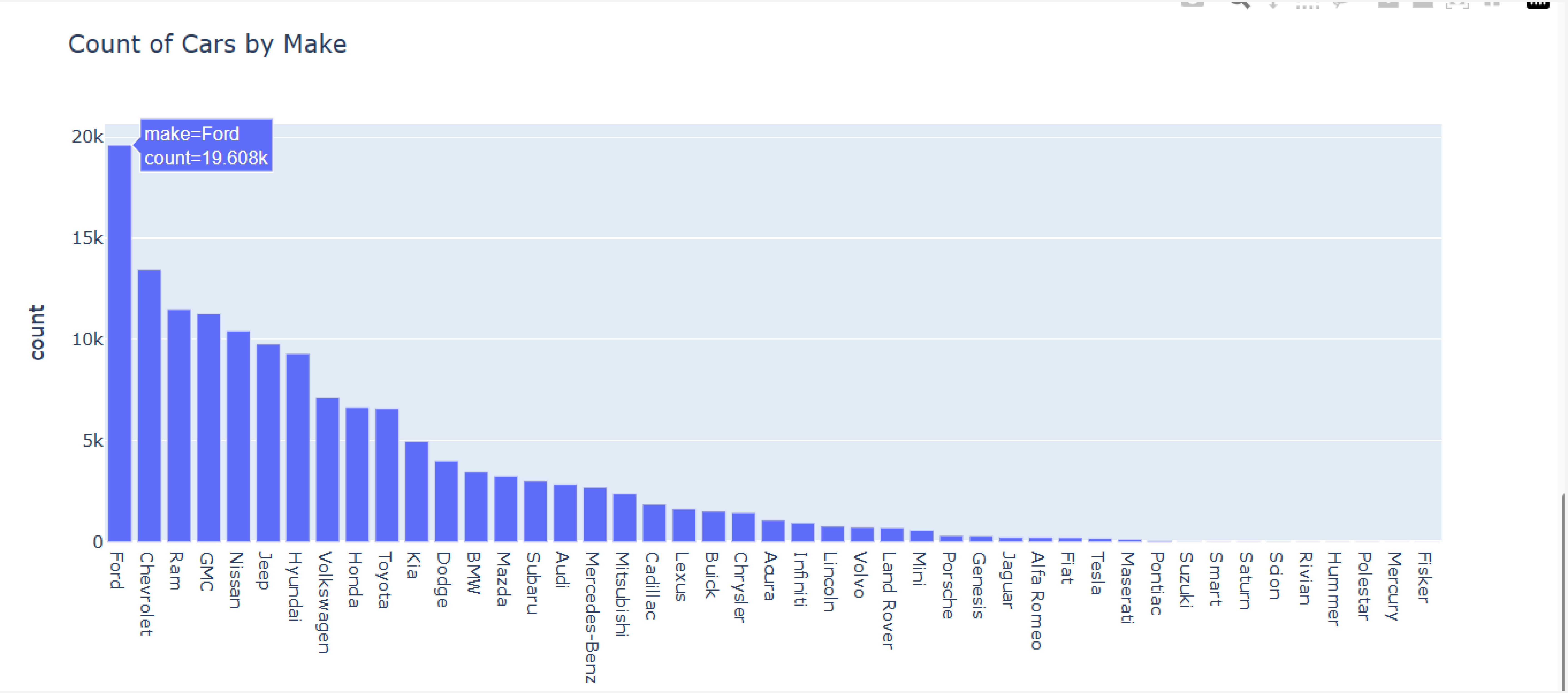
1. 'listing_id', listing_url
2. 'listing_first_date',
3. 'days_on_market'
4. ,'dealer_type''
5. ,'dealer_street'
6. ,'series',
7. 'dealer_url'
8. ,'exterior_color
9. ,'interior_color
10. ,'wheelbase_from_vin'
11. ,'listing_dropoff_date'
12. ,'price_History_delimited
13. vin
14. UVC

Data Visualization

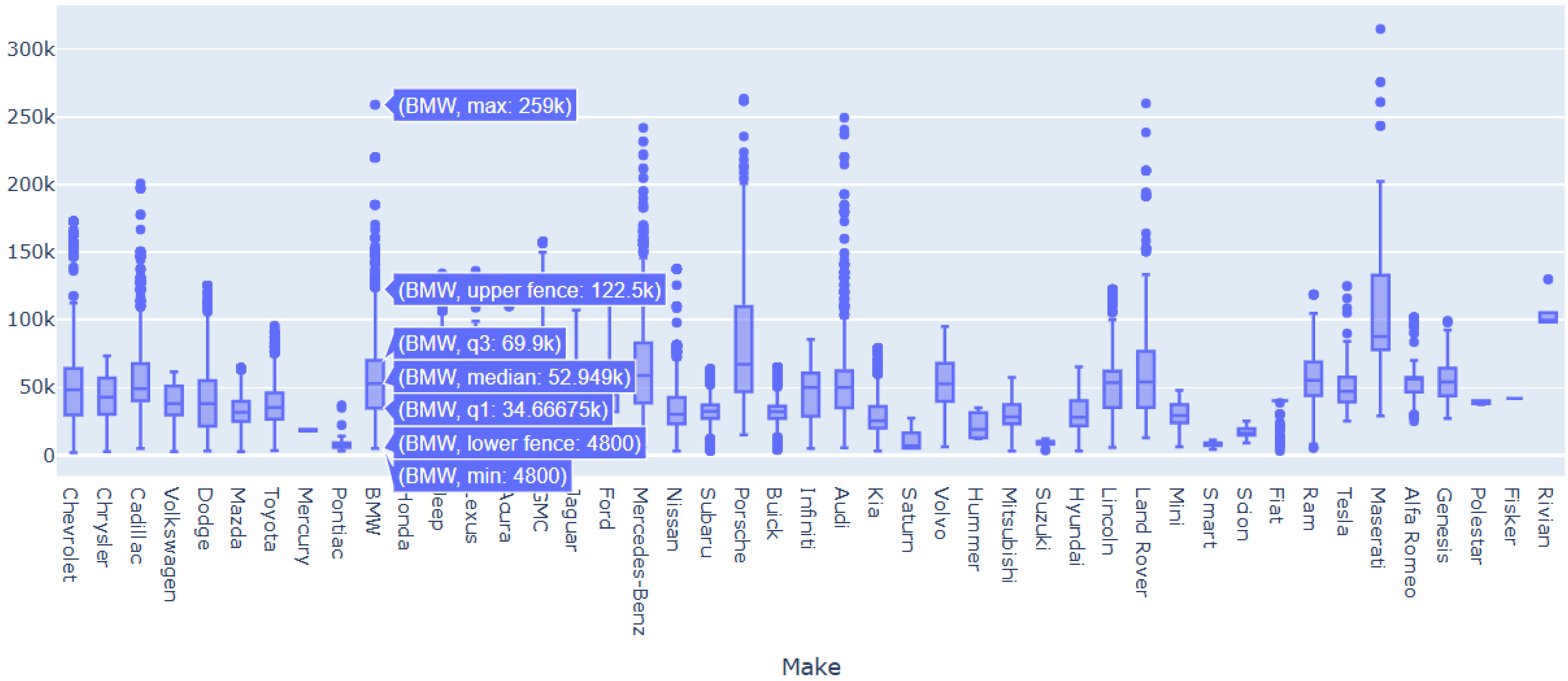




mileage
1M
0.8M
0.6M
0.4M
0.2M
0
0
50k
100k
150k
200k
250k
300k
make=Chevrolet
price=49.974k
mileage=160k


Count of Cars by Make
make=Ford
count=19.608k
20k
15k
10k
5k
0
count
Ford
Chevrolet
Ram
GMC
Nissan
Jeep
Hyundai
Volkswagen
Honda
Toyota
Kia
Dodge
BMW
Mazda
Subaru
Audi
Mercedes-Benz
Mitsubishi
Cadillac
Lexus
Buick
Chrysler
Acura
Infiniti
Lincoln
Volvo
Land Rover
Mini
Porsche
Genesis
Jaguar
Alfa Romeo
Fiat
Tesla
Maserati
Pontiac
Suzuki
Smart
Saturn
Scion
Rivian
Hummer
Polestar
Mercury
Fisker

Price Distribution by Make



AUTO CODERS

Data Interpretation

- Missing values in categorical features (e.g., interior_color, exterior_color) were filled using the mode (most frequent value).
- Duplicate Removal: No duplicate data
- Uninformative Feature Removal: Columns like listing_heading, msrp, and dealer_url were removed as they didn't contribute to the analysis.

```
import pandas as pd

for column in df2.columns:
    # Calculate mode for the column
    mode_values = df2[column].mode()

    # Check if mode is not empty
    if not mode_values.empty:
        # Use the first mode value
        mode_value = mode_values[0]

        # Fill NaN values with the mode
        df2[column] = df2[column].fillna(mode_value)
```

```
mode_value = df2['interior_color_category'].mode()[0]
df2['interior_color_category'] = df2['interior_color_category'].fillna(mode_value)
```

```
mode_value = df['exterior_color_category'].mode()[0]
df2['exterior_color_category'] = df2['exterior_color_category'].fillna(mode_value)
df2.head()
```

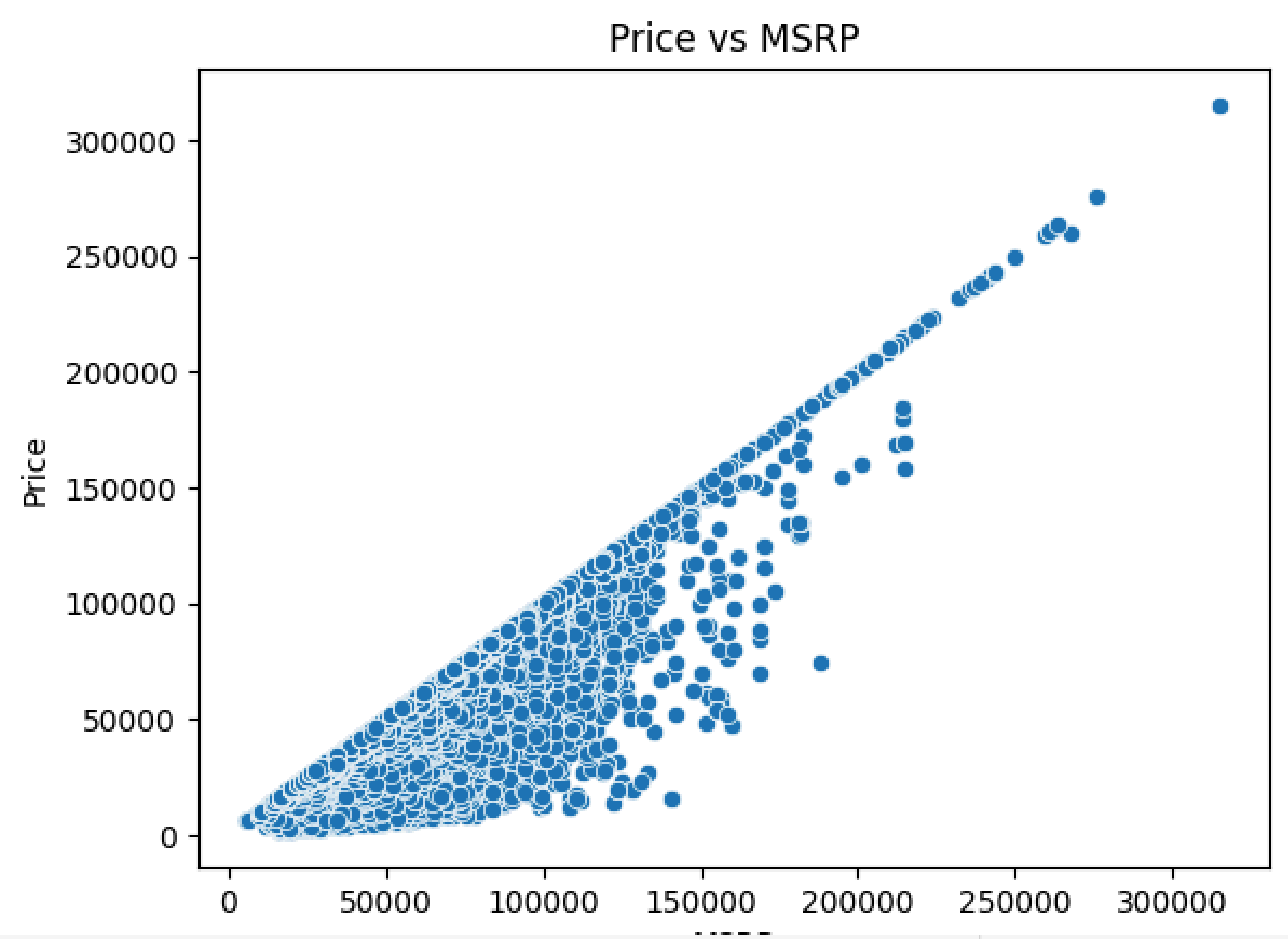
```python
# Fixing Transmission
df2['transmission_from_vin'].unique()
# The mapping for the transmission values
transmission_mapping = {
    '7': 'M',  # Convert '7' to 'M'
    '6': 'A',  # Convert '6' to 'A'
}

# Replacing the values in the transmission column
df2['transmission_from_vin'] = df2['transmission_from_vin'].replace(transmission_mapping)

# check unique values after conversion
print("Unique values in 'transmission' column after conversion:")
print(df2['transmission_from_vin'].unique())
```

Working with data

- Null values in the price column were further updated with the median of the vehicle having a similar MSRP

- Feature “Age” was added by calculating the difference between the current year and model year.

- The transmission column was fixed through mapping where the unique values “6” & “7” were change to automatic nad manual respectively




```
# Fixing Price
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Replace MSRP values of 0 with corresponding price values
df2['msrp'] = np.where(df2['msrp'] == 0, df2['price'], df2['msrp'])

# Step 2: Cap prices greater than MSRP to MSRP
df2['price'] = np.where(df2['price'] > df2['msrp'], df2['msrp'], df2['price'])

# Step 3: Optionally, replace very low prices with the median price based on MSRP group
def fix_low_prices(row):
    if row['price'] < 0.1 * row['msrp']:
        median_price = df2[(df2['msrp'] > row['msrp'] * 0.9) & (df2['msrp'] < row['msrp'] * 1.1)]['price'].median()
        return median_price if not pd.isnull(median_price) else row['price']
    else:
        return row['price']

df2['price'] = df2.apply(fix_low_prices, axis=1)
```

Data Transformation

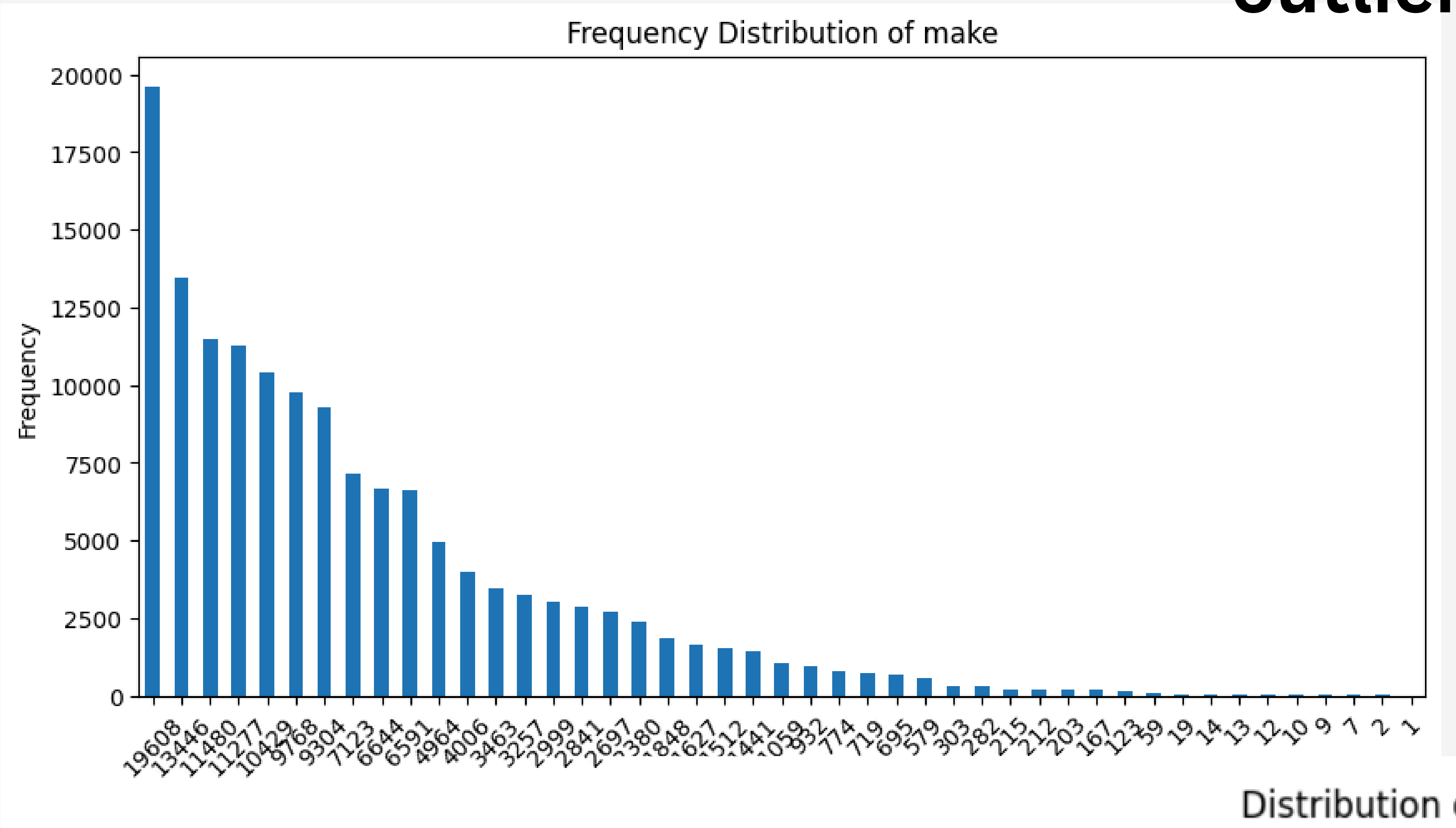
AUTO CODERS

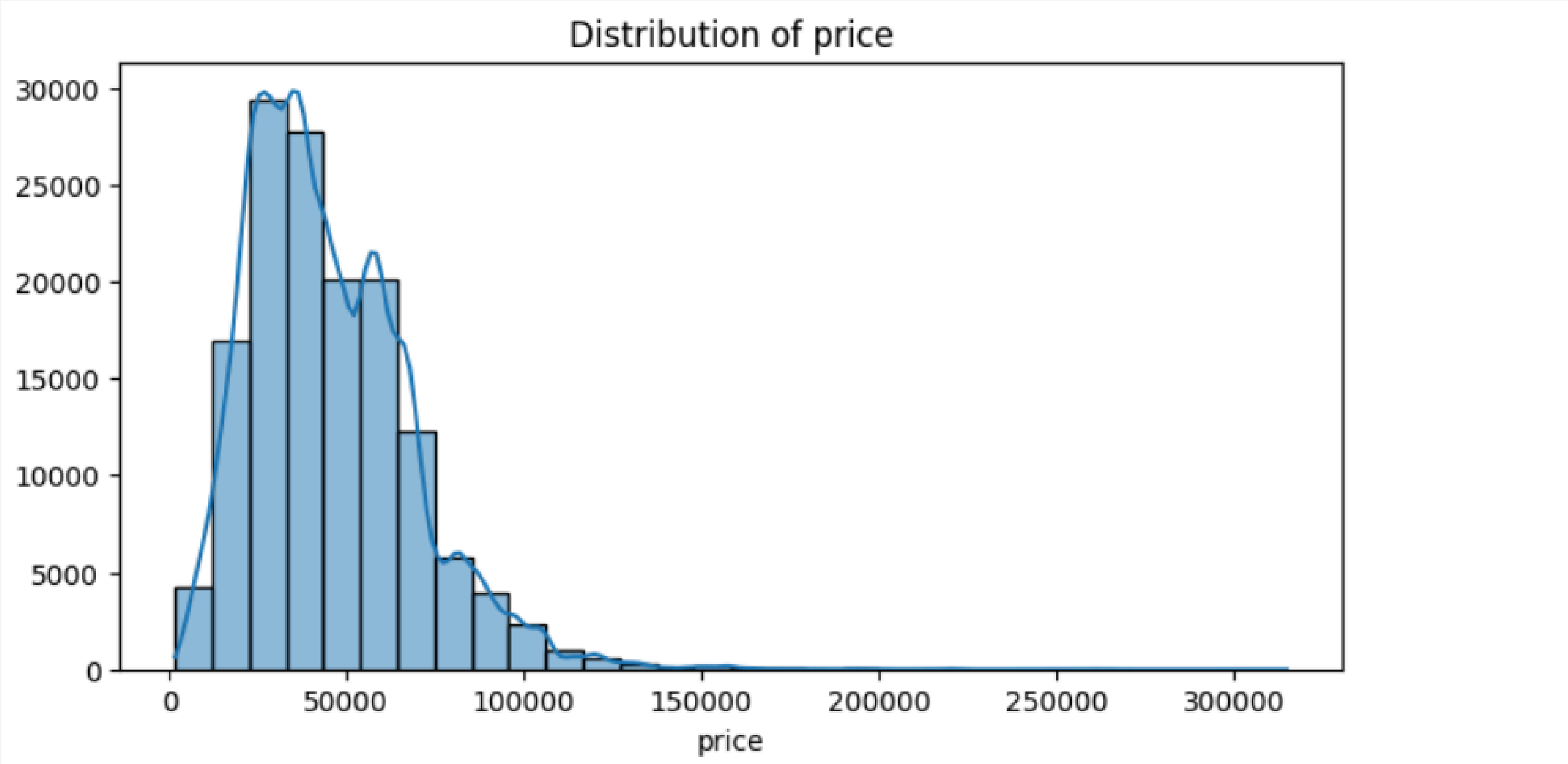
- One-hot encoding was used for features with lesser unique values like stock type , dealer type.
- Frequency Encoding was used for categorical values with high unique values like make, dealer postal code

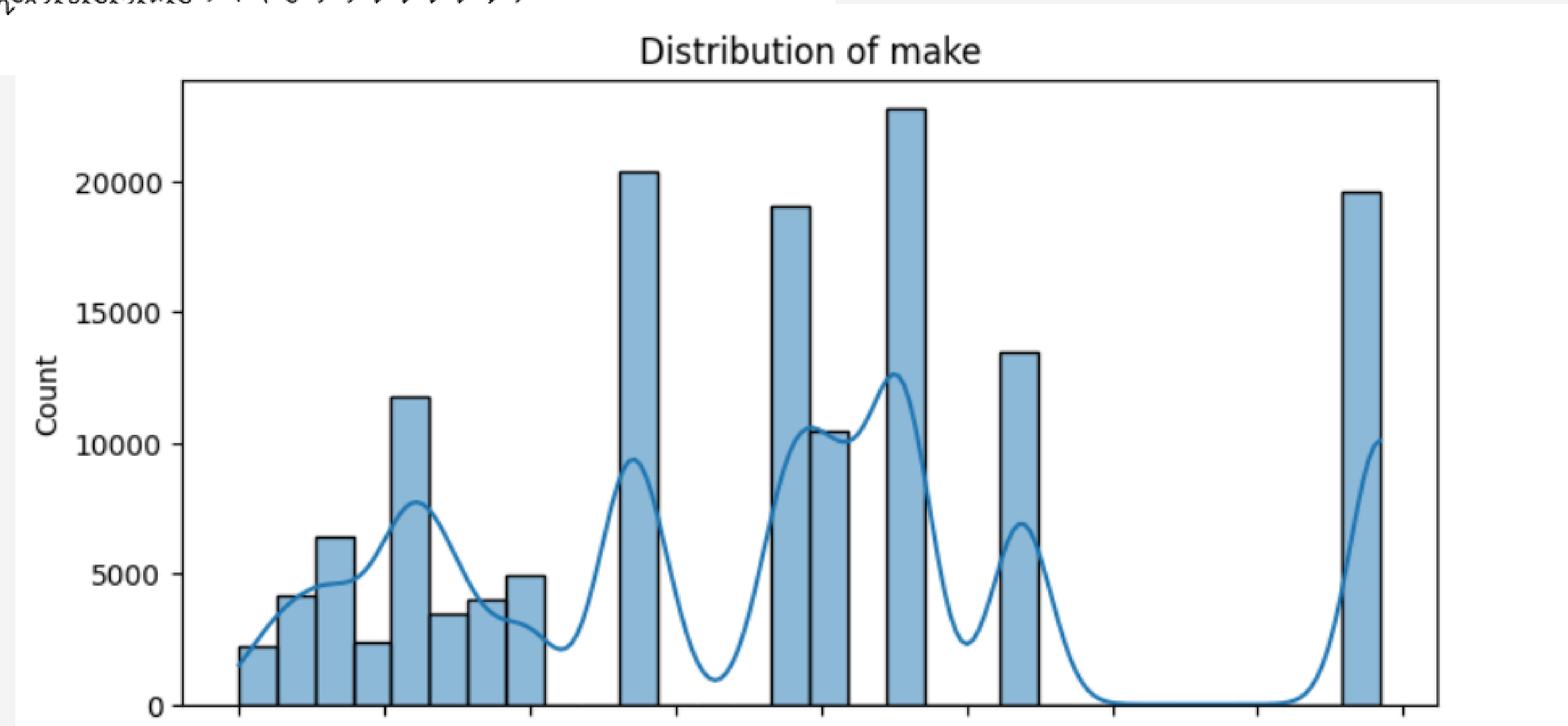
```
#FREQUENCY ENCODING
# List of columns to frequency encode
columns_to_encode = ['listing_heading', 'dealer_name', 'make', 'model',
                     'style', 'exterior_color_category',
                     'interior_color_category', 'engine_from_vin','dealer_postal_code'

# Frequency Encoding
for column in columns_to_encode:
    freq_encoding = df_final[column].value_counts()
    df_final[column] = df_final[column].map(freq_encoding)

# Display the updated DataFrame
print(df_final.head())
```

```
# Specify the columns to encode
columns_to_encode = [
    'listing_type',
    'stock_type',
    'drivetrain_from_vin',
    'fuel_type_from_vin',
    'dealer_city',
    'dealer_province',
    'transmission_from_vin'
]

# Initialize the OneHotEncoder
encoder = OneHotEncoder(drop='first', sparse_output=False)  # drop='first' to avoid

# Fit and transform the specified columns
encoded_array = encoder.fit_transform(df2[columns_to_encode])

# Create a DataFrame from the encoded array with proper column names
encoded_df = pd.DataFrame(encoded_array, columns=encoder.get_feature_names_out(colu

# Concatenate the encoded DataFrame with the original DataFrame (excluding the colu
df_final = pd.concat([df2.drop(columns=columns_to_encode).reset_index(drop=True), e
```

Visuals after Encoding potraying skewness of data for outlier removal







AUTO CODERS

OUTLIER REMOVAL PROCESS

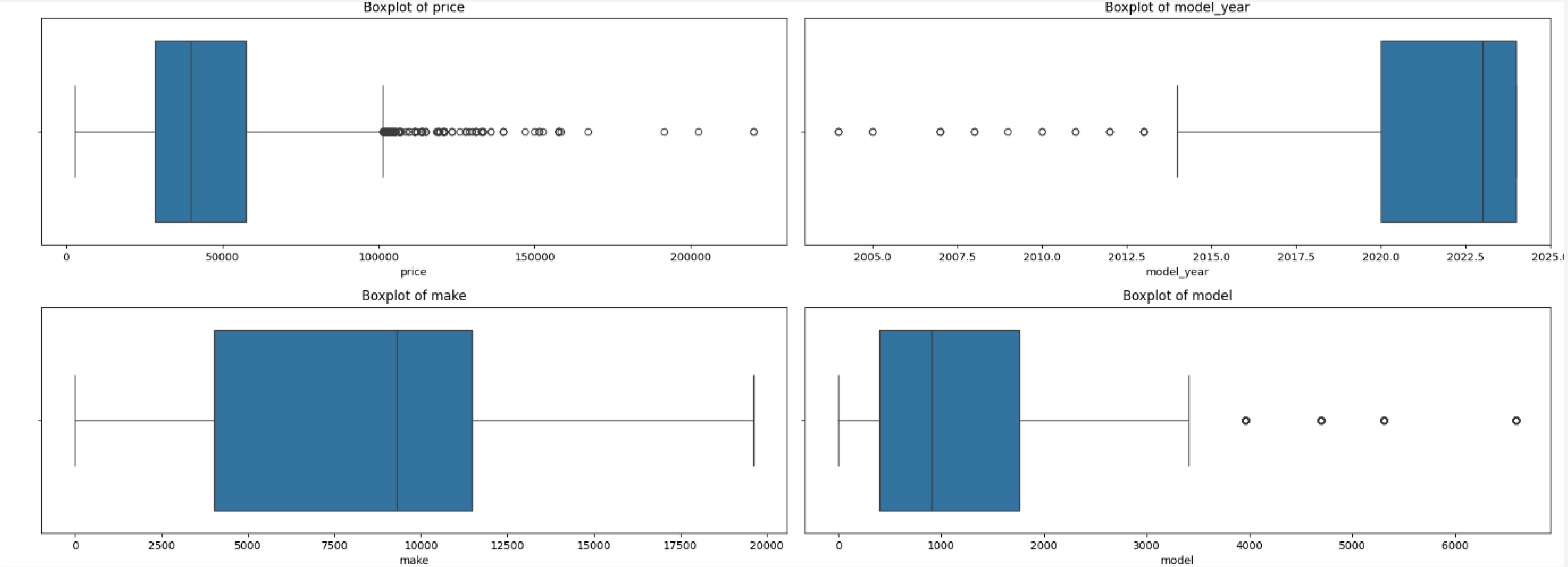
Outlier Handling: The Quantile Transformation effectively manages outliers by spreading out the frequent values and compressing the range of extreme values.

Z-Score was used to manages the attributes with normal distribution

Improved Data Distribution: The Quartile method was used on skewed data (where the mean was higher than the median) and Z-Score Method was used with data having normal distribution

AUTO CODERS

OUTLIER REMOVAL AND STANDARDIZATION OF DATA



The boxplots above potray data after outlier removals, the reason for some outliers still visible are because they are not actually true outliers.

AUTO CODERS

OUTLIER REMOVAL AND STANDARDIZATION OF DATA

```python
# Function to detect outliers using IQR method
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]

# Identify columns to treat differently
zscore_columns = ['model_year', 'make', 'style']  # Normal distribution columns according to the data
iqr_columns = ['price', 'mileage', 'model', 'distance_to_dealer', 'engine_from_vin', 'Age']  # Skewed columns (mean i higher than the median)

# Detect outliers using Z-score method for normally distributed columns
zscore_outliers = detect_outliers_zscore(df_final, zscore_columns)

# Detect outliers using IQR method for skewed columns
iqr_outliers = pd.DataFrame()
for col in iqr_columns:
    iqr_outliers = pd.concat([iqr_outliers, detect_outliers_iqr(df_final, col)])

# Combine all detected outliers
all_outliers = pd.concat([zscore_outliers, iqr_outliers]).drop_duplicates()

# Remove the outliers from the dataset
df_clean = df_final.drop(all_outliers.index)
print(f"Number of outliers removed: {len(all_outliers)}")
print(f"Number of rows after removing outliers: {len(df_clean)}")
```

```
Number of outliers removed: 35725
Number of rows after removing outliers: 109389
```

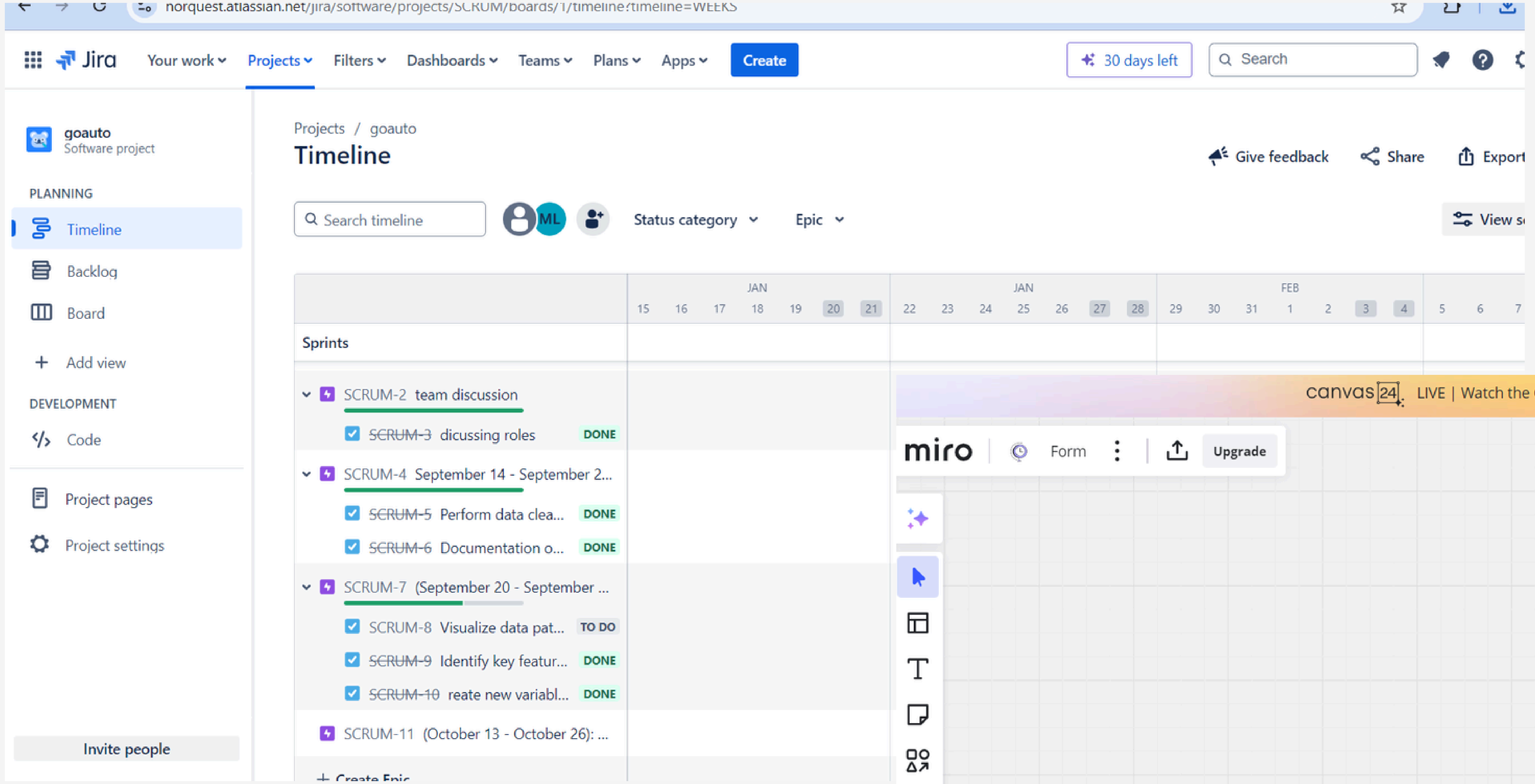
Code Snippet Using Standard Scaler

```python
#STANDARDIZATION
# Create a StandardScaler object
scaler = StandardScaler()

# Select numerical columns to standardize
columns = df_clean.columns
df_clean_original = df_clean.copy()
# Standardize the entire DataFrame
df_clean[columns] = scaler.fit_transform(df_clean[columns])

# Display the first few rows of the standardized DataFrame
print(df_clean.head())
```

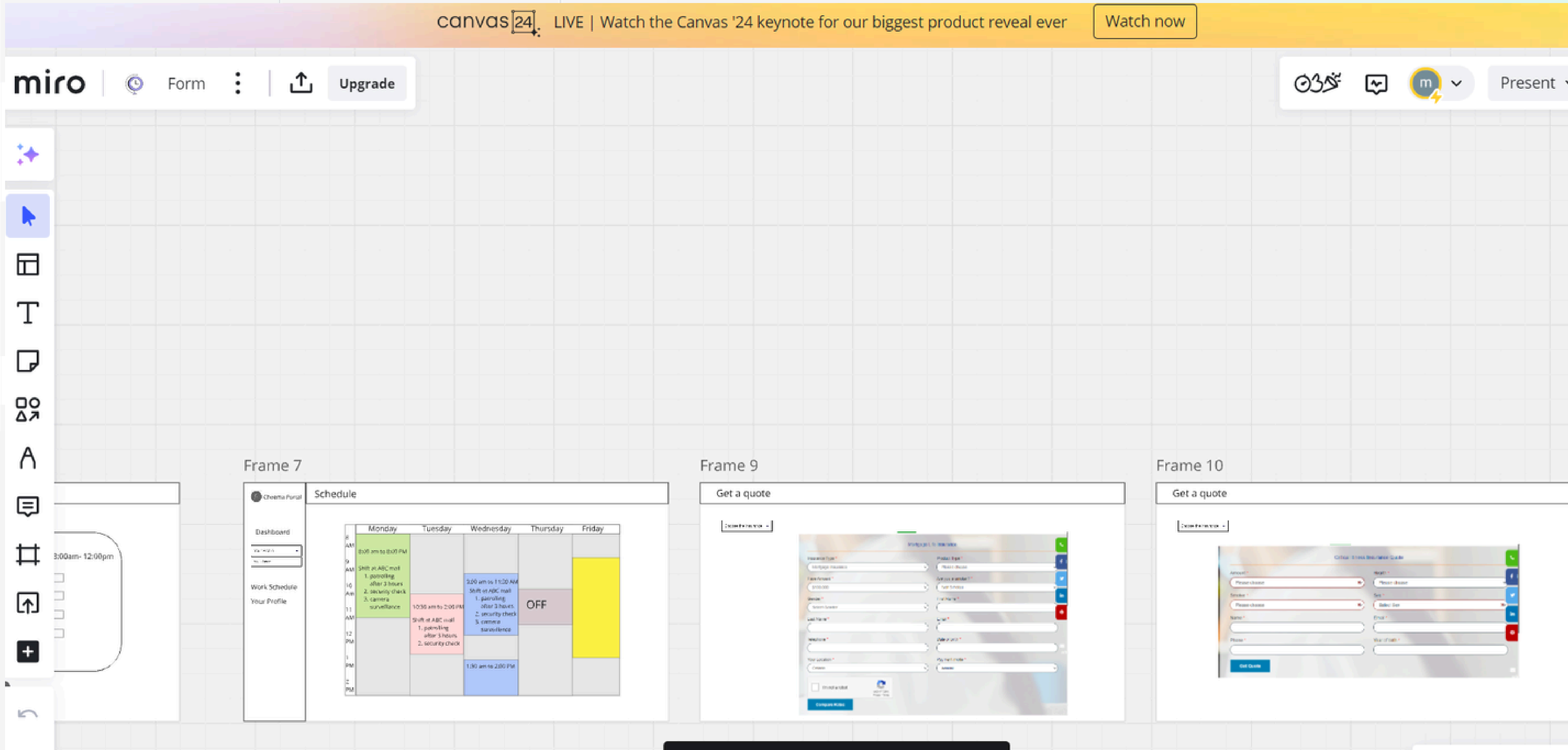
```
    listing_heading  dealer_name  dealer_postal_code   mileage     price  \
244       -0.188152    -0.826482           -0.866805  5.852189 -1.777564
271       -0.188152    -0.826482           -0.866805  5.628576  1.111486
286       -0.188152    -0.826482           -0.866805  4.266079 -1.538899
393       -0.188152    -0.826482           -0.866805  3.781113 -1.685230
451       -0.188152    -0.826482           -0.866805  4.421003 -1.831560

    model_year      make     model     style  has_leather  has_navigation  \
244  -6.646508  2.067699 -1.017073 -0.005456          0.0             0.0
271  -6.646508 -1.558654 -0.915438 -0.083956          0.0             0.0
286  -6.646508  0.300347 -0.708259 -1.132468          0.0             0.0
393  -6.272997  0.173077  0.895231 -0.005456          0.0             0.0
451  -6.272997 -0.438632 -0.344717 -0.827504          0.0             0.0

    exterior_color_category  interior_color_category  engine_from_vin  \
244                0.980884                 0.353983        -0.928841
```

Scrum Reports

AUTO CODERS



Jira



Miro

Resources

1. Data Resources
    - Go Auto's database
2. Software Tools:
    - Python: For data analysis, modeling, and visualization (with libraries like Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn).
    - Jupyter Notebook: For interactive coding, documenting the code, and presenting visualizations.
    - Tableau or Power BI: For building dashboards to visualize the clustering results and business insights.
    - Database Systems:
    - SQL Database: For storing and querying data efficiently
3. Communication and Collaboration Tools
    - Slack/Microsoft Teams: For team communication and collaboration.
    - Jira: For project management and task tracking.
    - Google Drive/SharePoint: For sharing documents, reports, and project files among team members.

THANK YOU SO MUCH!

Q/A