Project Milestone-3

Gunadheep Sakthivel

Introduction:

The milestone we set for ourselves is to conduct a detailed data analysis of the pricing distribution of used cars in the USA, drivers' preferences regarding the brands of cars, regional sales, and relationship between conditions of cars and their prices. We went with dynamic visualizations instead of static ones, this gave us more freedom to add interactive features and widgets that helped the data exploration process.

Interactive Visualization:

The missing values and abnormalities were among the first things we tackled so we began by importing the dataset into our environment and carried out the preprocessing. Tools such as Matplotlib, Seaborn, and Plotly were used for the initial exploratory data analysis to produce static chart. The subjects of our study were: price clustering, time periods for prices and discounts, brand preferences, cars under luxury segment, regional sales breakdowns, etc. At a later stage, we created ipywidgets which proffered an interactive approach to users where they interactively created and customized various plot types as per needs.

In [13]:
```
# Function to create and display the plot
def display_plot(plot_type, x_column, y_column):
    if plot_type == 'Scatter Plot':
        fig = px.scatter(car_data, x=x_column, y=y_column, title='Scatter Plot')
    elif plot_type == 'Line Plot':
        fig = px.line(car_data, x=x_column, y=y_column, title='Line Plot')
    elif plot_type == 'Bar Plot':
        fig = px.bar(car_data, x=x_column, y=y_column, title='Bar Plot')
    elif plot_type == 'Histogram':
        fig = px.histogram(car_data, x=x_column, title='Histogram')
    else:
        fig = None

    if fig:
        with plot_output:
            clear_output(wait=True)  # Clear the output area before displaying the new plot
            fig.show()

# Dropdown widgets for selecting plot type, x axis, and y axis
plot_type_dropdown = widgets.Dropdown(options=['Scatter Plot', 'Line Plot', 'Bar Plot', 'Histogram'], description='Plot Type:')
x_dropdown = widgets.Dropdown(options=car_data.columns, description='X Axis:', value='model')
y_dropdown = widgets.Dropdown(options=car_data.columns, description='Y Axis:', value='price')

# Button to generate the plot
plot_button = widgets.Button(description="Generate Plot")
plot_output = widgets.Output()

# Display the widgets
display(widgets.VBox([plot_type_dropdown, x_dropdown, y_dropdown, plot_button, plot_output]))

# Event handler for button click
def on_plot_button_clicked(b):
    with plot_output:
        display_plot(plot_type_dropdown.value, x_dropdown.value, y_dropdown.value)

# Bind the event handler to the button click
plot_button.on_click(on_plot_button_clicked)

# Display default plot
display_plot('Scatter Plot', 'model', 'price')
```

Plot Type: Scatter Plot

X Axis: model

Y Axis: price

Generate Plot

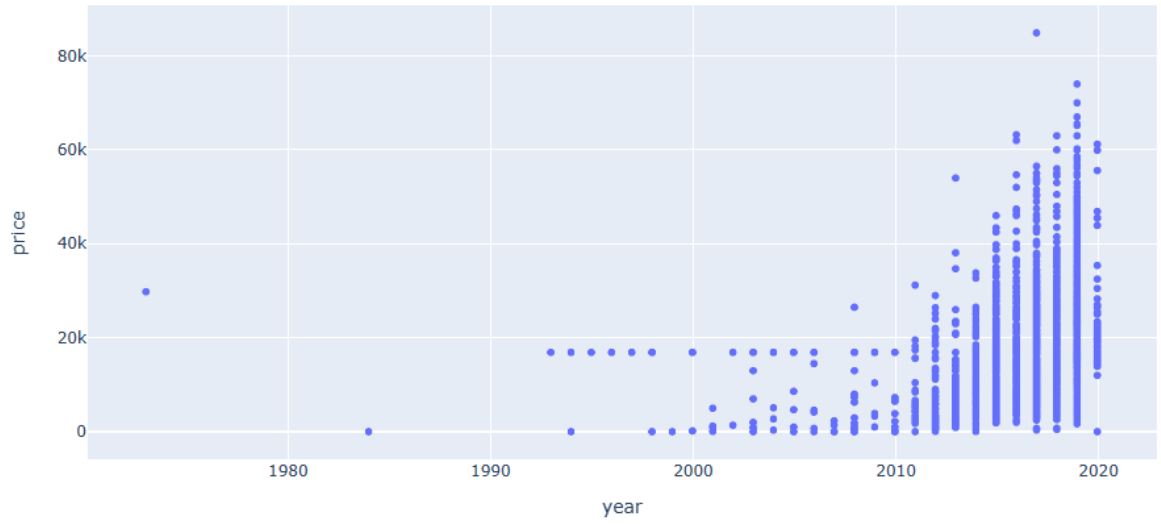
Plot Type: Scatter Plot

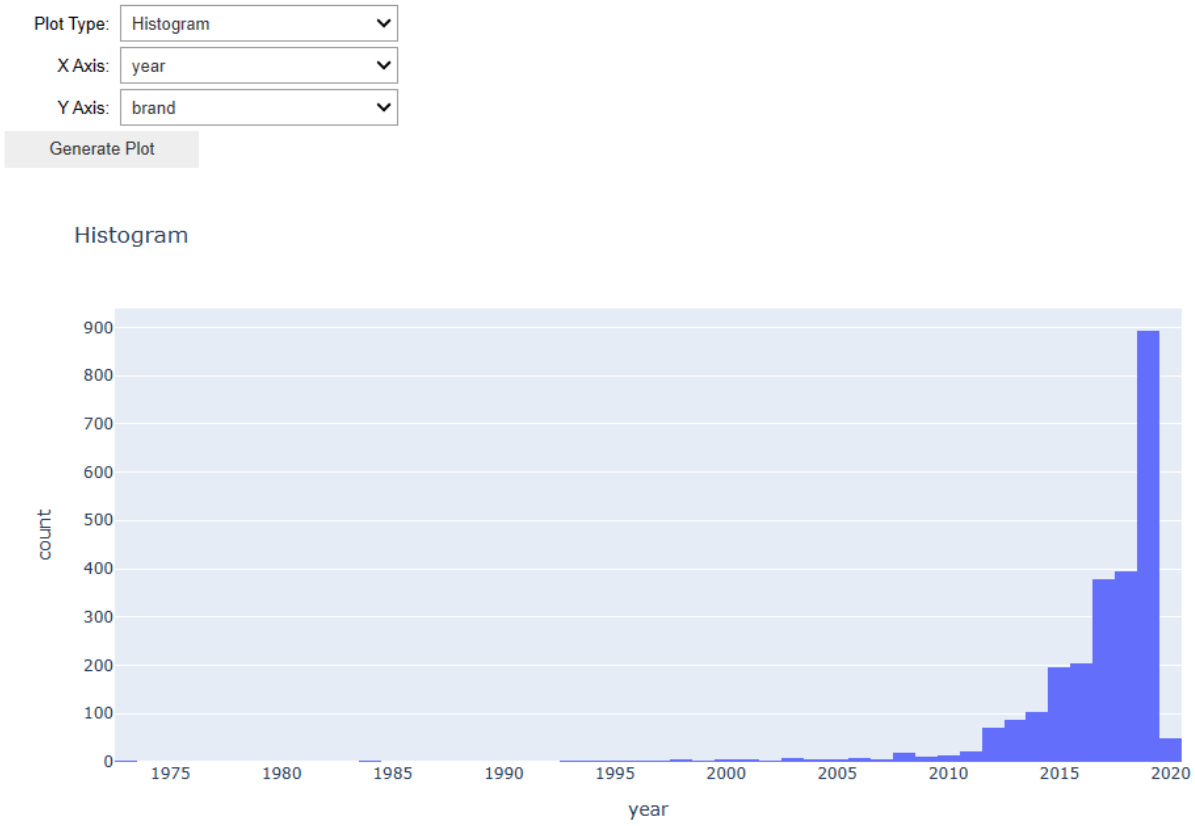
X Axis: year

Y Axis: price

Generate Plot

Scatter Plot



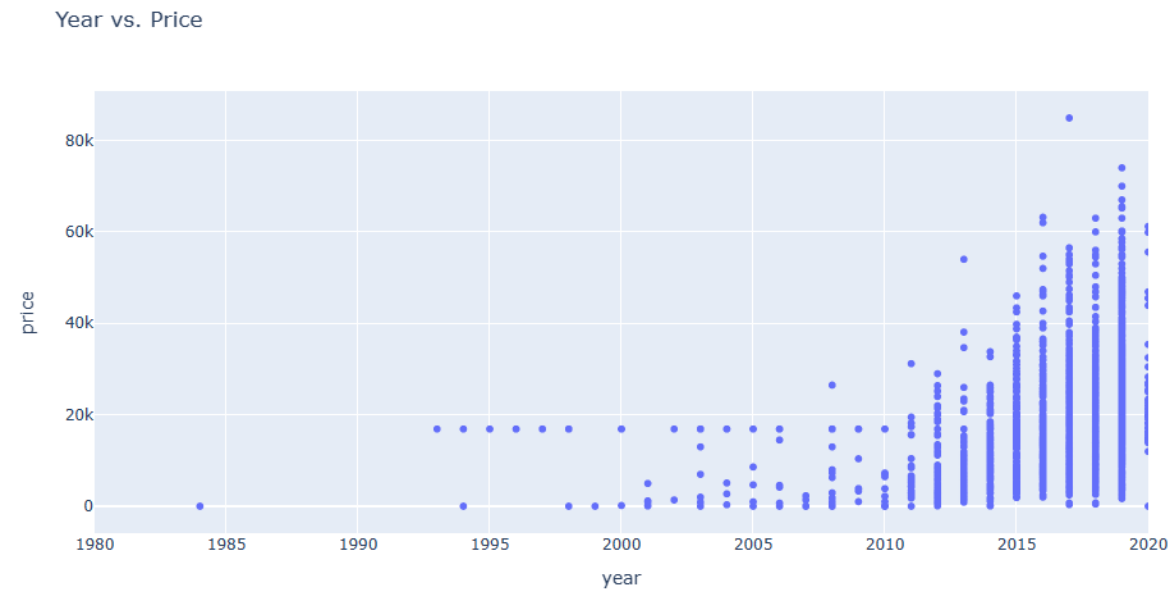


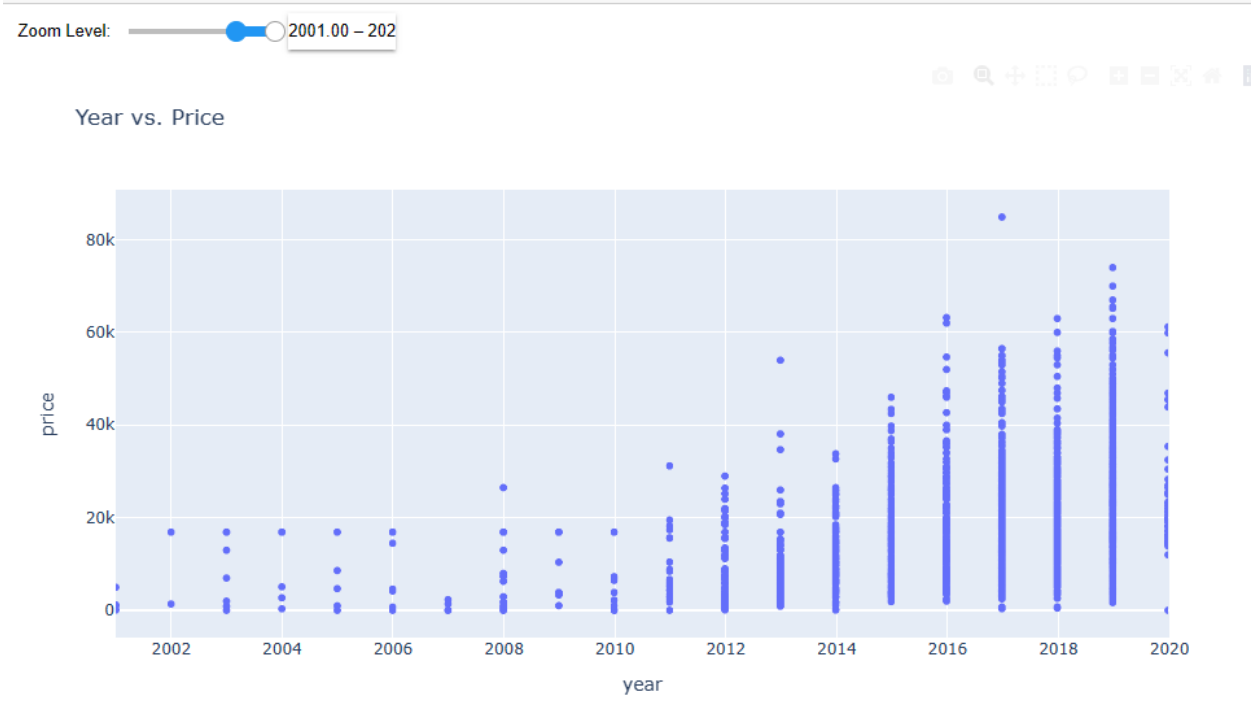
Our scatterplot zoom function ensures that the user can zoom in and out of the data plot and concentrate on data points. Users can alter and set the zoom level of the scatter plot with the zoom control slider widget and can use the free zoom option which will automatically zoom out when the user moves away result, keeping the plot extends. Through such customization, the user can hone down to the right range of years of cars on sale and compare the number plate year of registration to the actual price of a vehicle. This feature helps us in getting a deeper understanding of periodic price trends and increase precision in the research.

In [14]:
```
import plotly.express as px
import ipywidgets as widgets
from IPython.display import clear_output

# Assuming you have a DataFrame named car_data

# Define function to display scatter plot with zoom feature
def display_scatterplot(size, zoom_range):
    fig = px.scatter(car_data, x='year', y='price', title='Year vs. Price', size_max=size)
    fig.update_xaxes(range=zoom_range)  # Adjust x-axis range
    fig.show()

# Create slider widget for zoom control
zoom_slider = widgets.FloatRangeSlider(value=[1980, 2020], min=1950, max=2020, step=1, description='Zoom Level:')
output = widgets.Output()

def on_zoom_slider_change(change):
    with output:
        clear_output(wait=True)
        display_scatterplot(zoom_slider.value, zoom_slider.value)

# Register event handler for slider change
zoom_slider.observe(on_zoom_slider_change, names='value')

# Display the widgets
display(widgets.VBox([zoom_slider, output]))

# Display default scatter plot
display_scatterplot(10, [1980, 2020])
```

Zoom Level: 1980.00 – 202

Zoom Level: 1980.00 – 202





Self-assessment:
Changing over for interactive visualization made the process of exploratory data analysis much more efficient. People got more elbowroom to exercise their data exploration skills; they did it through interactive widgets, which let the users create charts according to their preferences. As a result of giving zoom functionality to the scatter plot, any specific data subset can be focused on, and this made analysis so much easier and effective.

Conclusion:

Thus, this milestone can be considered as one of the significant stages that aided us in extracting the knowledge regarding US auto market dataset. Using interactive visualization techniques, we have developed data exploration that has become much more interesting and educational for the target audience. The diagnosis of the US auto market can be treated better, and the strategic planning can be improved by applying the data obtained from this analysis, the aim of which is better decision-making concerning the automotive industry.