

Secondary Market Automobile Sales Analysis

Project Introduction

The goal of this project will be to develop my technical skills using Python, GitHub, and Jupyter Notebook. During this project I will focus on my tasks in building and maintaining various Python environments, creating a web application, and uploading it to a cloud service provider to make it easily accessible to others who would want to use it.

Analysis Outline

To complete our goals, we are going to perform the following tasks:

- Create a new dataframe containing only columns that we will need for our web app
- Create a histogram of vehicle models sold by each manufacturer
- Create a histogram for the vehicle condition of each car model year
- Create a scatter plot to compare vehicle prices to their total odometer
- Create a scatter plot conveying how many days it takes to sell the vehicle based on its listed price
- Create a scatter plot displaying how long it takes to sell a vehicle based on its total odometer
- Create an interactive graphic to compare the prices between car manufacturers

Part 1: Importing the Data

The first part of the project will be importing all the libraries we will be using for this project

```
import streamlit as st
import pandas as pd
import plotly.express as px
```

To read our raw data file we will be using pandas

```
df = pd.read_csv('vehicles_us.csv')
```

Part 2: Pre-processing the Data

In this project, we will not be using every column in the raw data set for our analysis. Since we are not using some columns, we can temporarily remove them to keep everything more concise.

```
df.columns
```

```
Index(['price', 'model_year', 'model', 'condition', 'cylinders',
      'fuel',
      'odometer', 'transmission', 'type', 'paint_color', 'is_4wd',
      'date_posted', 'days_listed'],
      dtype='object')
```

Create a variable with a list of the columns we are not going to use. Then we will use this list to drop the columns we will not be using for this web app.

```
unused_columns = ['cylinders',
                  'fuel',
                  'transmission',
                  'paint_color',
                  'is_4wd',
                  'date_posted']
df = df.drop(unused_columns,
             axis='columns')

df.columns

Index(['price', 'model_year', 'model', 'condition', 'odometer',
      'type',
      'days_listed'],
      dtype='object')
```

This column contains multiple values that we can use in our analysis. To separate these variables, we can go to the 'model' column and split the data into a new "manufacturer" column.

```
df['manufacturer'] = df['model'].apply(lambda x: x.split()[0])
```

To check the new data set for the remaining columns we can use the head() function

```
df.head()
```

	price	model_year	model	condition	odometer	type	\
0	9400	2011.0	bmw x5	good	145000.0	SUV	
1	25500	NaN	ford f-150	good	88705.0	pickup	
2	5500	2013.0	hyundai sonata	like new	110000.0	sedan	
3	1500	2003.0	ford f-150	fair	NaN	pickup	
4	14900	2017.0	chrysler 200	excellent	80903.0	sedan	

	days_listed	manufacturer
0	19	bmw
1	50	ford
2	79	hyundai
3	9	ford
4	28	chrysler

Title for the cleaned data

```
st.header('Data Viewer')
```

2023-07-16 01:51:44.818

Warning: to view this Streamlit app on a browser, run it with the following command:

```
streamlit run c:\Users\jason\Desktop\python_apps\my_env\Lib\site-packages\ipykernel_launcher.py [ARGUMENTS]
```

```
DeltaGenerator()
```

Using streamlit to generate a visual for the cleaned data

```
st.dataframe(df)
```

```
DeltaGenerator()
```

Part 3: Creating Interactive Web Application Graphics

We are going to use the describe() function on the dataframe to get the general statistics on our current data

```
df.describe()
```

	price	model_year	odometer	days_listed
count	51525.000000	47906.000000	43633.000000	51525.000000
mean	12132.464920	2009.750470	115553.461738	39.55476
std	10040.803015	6.282065	65094.611341	28.20427
min	1.000000	1908.000000	0.000000	0.00000
25%	5000.000000	2006.000000	70000.000000	19.00000
50%	9000.000000	2011.000000	113000.000000	33.00000
75%	16839.000000	2014.000000	155000.000000	53.00000
max	375000.000000	2019.000000	990000.000000	271.00000

From the describe() function we can evaluate that, for this data set, the average pre-owned vehicle would be a 2009-2010 car that costs about \$12,132 with 115,553 total miles on it and it takes about 40 days to sell

Histogram of Vehicle Models Sold by each Manufacturer

To learn more about the vehicles sold we can create a histogram to show the volume of car models sold by every manufacturer

text header for "vehicle type manufacturer"

```
st.header('Vehicle Types by Manufacturer')
```

```
DeltaGenerator()
```

creating histogram graphic for "vehicle type manufacturer"

```
fig = px.histogram(df, x='manufacturer',  
                    color='type')
```

to display the histogram we will use streamlit

```
st.write(fig)  
df['manufacturer'].unique()  
array(['bmw', 'ford', 'hyundai', 'chrysler', 'toyota', 'honda', 'kia',  
       'chevrolet', 'ram', 'gmc', 'jeep', 'nissan', 'subaru', 'dodge',  
       'mercedes-benz', 'acura', 'cadillac', 'volkswagen', 'buick'],  
      dtype=object)
```

After creating this histogram it will give you a visual of how many cars are sold and specific models sold by these manufacturers

Histogram of Vehicle Condition of each Car Model Year

For this project we also want to view what is the car condition of each car. To do this we will create a histogram of vehicle conditions for each car model year.

text header for "vehicle condition to the model year"

```
st.header('Histogram of `condition` vs `model_year`')  
DeltaGenerator()
```

creating histogram graphic for vehicle condition to the model year"

```
fig = px.histogram(df,  
                    x='model_year',  
                    color='condition')
```

to display the histogram we will use streamlit

```
st.write(fig)  
df['model_year'].unique()  
array([2011., nan, 2013., 2003., 2017., 2014., 2015., 2012., 2008.,  
       2018., 2009., 2010., 2007., 2004., 2005., 2001., 2006., 1966.,  
       1994., 2019., 2000., 2016., 1993., 1999., 1997., 2002., 1981.,  
       1995., 1996., 1975., 1998., 1985., 1977., 1987., 1974., 1990.,  
       1992., 1991., 1972., 1967., 1988., 1969., 1989., 1978., 1965.,  
       1979., 1968., 1986., 1980., 1964., 1963., 1984., 1982., 1973.,  
       1970., 1955., 1971., 1976., 1983., 1954., 1962., 1948., 1960.,  
       1908., 1961., 1936., 1949., 1958., 1929.] )
```

```
df['condition'].unique()  
array(['good', 'like new', 'fair', 'excellent', 'salvage', 'new'],  
      dtype=object)
```

After creating this histogram it will show you the conditions of vehicles made from the 1900s to the 2010s

Scatter Plot of Vehicle Price VS Odometer

A major component of a vehicle's value is its total mileage. To find the highest price a vehicle could sell for would be to process the past data into a scatter plot to find the range where anyone could value their vehicle based on their vehicle odometer.

text header for "vehicle price to odometer"

```
st.header('Scatter Plot of `price` vs `odometer`')  
DeltaGenerator()
```

creating scatter plot graphic for "vehicle price to odometer"

```
fig = px.scatter(df,  
                 x='odometer',  
                 y='price')
```

display the scatter plot with streamlit

```
st.write(fig)  
df['odometer'].mean()  
115553.4617376756  
df['price'].mean()  
12132.464919941776
```

On average a vehicle will have about 115,553 miles on the odometer which will cost on average about \$12,132

Scatter Plot of Listed Days to Sell VS Vehicle Price

Another important metric for secondary market vehicle sellers is their timeframe for selling their cars. This scatter plot will show a range of how fast or long it takes to sell their car based on their estimated sell price.

text header for "vehicle days listed to price"

```
st.header('Scatter Plot of `days_listed` vs `price`')
```

```
DeltaGenerator()
```

creating scatter plot graphic for "vehicle days listed to price"

```
fig = px.scatter(df,  
                 x='days_listed',  
                 y='price')
```

display the scatter plot with streamlit

```
st.write(fig)  
df['days_listed'].mean()  
39.55475982532751  
df['price'].mean()  
12132.464919941776
```

On average a car will price at \$12,132 and will take about 40 days to sell

Scatter Plot of Vehicle Odometer VS Listed Days to Sell

Another method to see how fast it takes to sell a vehicle is by the odometer. In this scatter plot it will show a range of how many days it would take to sell a vehicle based on it's odometer.

text header for "vehicle odometer to days listed"

```
st.header('Scatter Plot of `odometer` vs `days_listed`')  
DeltaGenerator()
```

creating scatter plot graphic for "vehicle odometer to days listed"

```
fig = px.scatter(df,  
                 x='days_listed',  
                 y='odometer')
```

display the scatter plot with streamlit

```
st.write(fig)  
df['odometer'].mean()  
115553.4617376756  
df['days_listed'].mean()  
39.55475982532751
```

On average a car will have about 115,553 miles on the odometer and it will take about 40 days to sell

Interactive Graphic to Compare Price Between Car Manufacturers

text header for "compare price distribution between manufacturers"

```
st.header('Compare Price Distribution Between Manufacturers')  
DeltaGenerator()
```

get a list of car manufacturers

```
manufac_list = sorted(df['manufacturer'].unique())
```

get user's inputs from a dropdown menu

```
manufacturer_1 = st.selectbox(label='Select manufacturer 1', # title  
                             of the select box  
                             options=manufac_list, # options listed  
                             in the select box  
                             index=manufac_list.index('chevrolet')) #  
                             default pre-selected option
```

repeat for the second dropdown menu

```
manufacturer_2 = st.selectbox(label='Select manufacturer 2',  
                              options=manufac_list,  
                              index=manufac_list.index('hyundai'))
```

filter the dataframe

```
mask_filter = (df['manufacturer'] == manufacturer_1) |  
(df['manufacturer'] == manufacturer_2)  
df_filtered = df[mask_filter]
```

add a checkbox for if a user wants to normalize the histogram

```
normalize = st.checkbox('Normalize histogram', value=True)  
if normalize:  
    histnorm = 'percent'  
else:  
    histnorm = None
```

create a plotly histogram figure

```
fig = px.histogram(df_filtered,  
                  x='price',
```

```
nbins=30,  
color='manufacturer',  
histnorm=histnorm,  
barmode='overlay')
```

display the figure with streamlit

```
st.write(fig)
```

Conclusion

Within this web application we can view several different metrics on the data of automobile sales on the secondary market. By manipulating the data in the web application any end user can find the value of the car of interest and how long will it take to sell based on the car model and odometer milage.