SENTIMENT ANALYSIS FOR MARKETING

PREPARED BY:MONISH.R

Step 1: Data Collection93

You can start by collecting a dataset containing customer reviews of competitor products. You can consider sources like e-commerce websites, social media platforms, or review websites. Websites like Amazon, Yelp, or Twitter can be good places to start. Ensure that the dataset includes text reviews and corresponding sentiment labels (positive, negative, neutral).

Step 2: Data Preprocessing

Before performing sentiment analysis, it's crucial to clean and preprocess the textual data:

Remove special characters, HTML tags, and punctuation.

Tokenize the text into words or subwords.

Convert text to lowercase.

Remove stop words (common words like "the," "and," "is" that don't carry much meaning).

Lemmatize or stem words to their base forms.

Step 3: Sentiment Analysis Techniques

You can employ various NLP techniques for sentiment analysis:

Bag of Words (BoW): Convert the preprocessed text into a matrix of word counts. Each document (review) is represented as a vector of word frequencies.

    - Word Embeddings (Word2Vec, GloVe): Use pre-trained word embeddings to represent words in a continuous vector space. Average or concatenate word embeddings for each document.

Transformer Models (e.g., BERT, GPT-3):

- Utilize pre-trained transformer-based models to capture contextual information in the text. Fine-tune the model on your sentiment analysis task.

Step 4: Feature Extraction

- Extract features and sentiments from the text data:
- Use the chosen technique (BoW, Word Embeddings, or Transformer) to represent the reviews as numerical features.
- Apply sentiment analysis algorithms or models to classify each review into positive, negative, or neutral sentiments.

Step 5: Visualization

- Create visualizations to depict sentiment distribution and analyze trends:
- Generate bar charts or pie charts to show the distribution of positive, negative, and neutral sentiments.
- Create word clouds to visualize the most frequently mentioned words in positive and negative reviews.
- Plot time series graphs to observe how sentiments change over time.

Step 6: Insights Generation

- Extract meaningful insights from the sentiment analysis results to guide business decisions and elaborate on the content:

Identify common themes or topics in positive and negative reviews.

- Analyse the sentiment trends over time to see if there are any seasonal patterns or product-specific events affecting sentiment.

Compare sentiment scores across different competitor products.

- Look for keywords or phrases that frequently appear in positive or negative reviews, which can provide actionable insights for product improvement.

- By following these steps, you can perform sentiment analysis on customer reviews of competitor products, gain valuable insights, and make informed business decisions based on customer feedback.

PROGRAM:

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read
_csv)
```

There is 1 csv file in the current version of the dataset:

In [2]:

```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/car data.csv
/kaggle/input/cardata.R
```

The next hidden code cells define functions for plotting data. Click on the "Code" button in the published kernel to reveal the hidden code.

unfold_lessHide code

In [3]:

```
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[c
ol] < 50]] # For displaying purposes, pick columns that have betw
een 1 and 50 unique values
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGra
phRow), dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)
):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```

unfold_lessHide code

In [4]:

```
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep
columns where there are more than 1 unique values
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-Na
N or constant columns ({df.shape[1]}) is less than 2')
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=8
0, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=
90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for {filename}', fontsize=15)
    plt.show()
```

unfold_lessHide code

In [5]:

```
# Scatter and density plots
def plotScatterMatrix(df, plotSize, textSize):
    df = df.select_dtypes(include =[np.number]) # keep only nume
rical columns
    # Remove rows and columns that would lead to df being singula
r
    df = df.dropna('columns')
    df = df[[col for col in df if df[col].nunique() > 1]] # keep
columns where there are more than 1 unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for
matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df = df[columnNames]
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='
kde')
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.
8, 0.2), xycoords='axes fraction', ha='center', va='center', siz
e=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

Now you're ready to read in the data and use the plotting functions to visualize the data.

Let's check 1st file: /kaggle/input/car data.csv

In [6]:

```
nRowsRead = 1000 # specify 'None' if want to read whole file
# car data.csv may have more rows in reality, but we are only loa
ding/previewing the first 1000 rows
df1 = pd.read_csv('/kaggle/input/car data.csv', delimiter=',', n
rows = nRowsRead)
```

```
df1.dataframeName = 'car data.csv'
nRow, nCol = df1.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
There are 301 rows and 10 columns
```

Let's take a quick look at what the data looks like:

In [7]:

```
df1.head(5)
```

Out[7]:

	Car_Name	company	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Ritz	maruti suzuki	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	maruti suzuki	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	Ciaz	maruti suzuki	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	maruti suzuki	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	Swift	maruti suzuki	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

Distribution graphs (histogram/bar graph) of sampled columns:

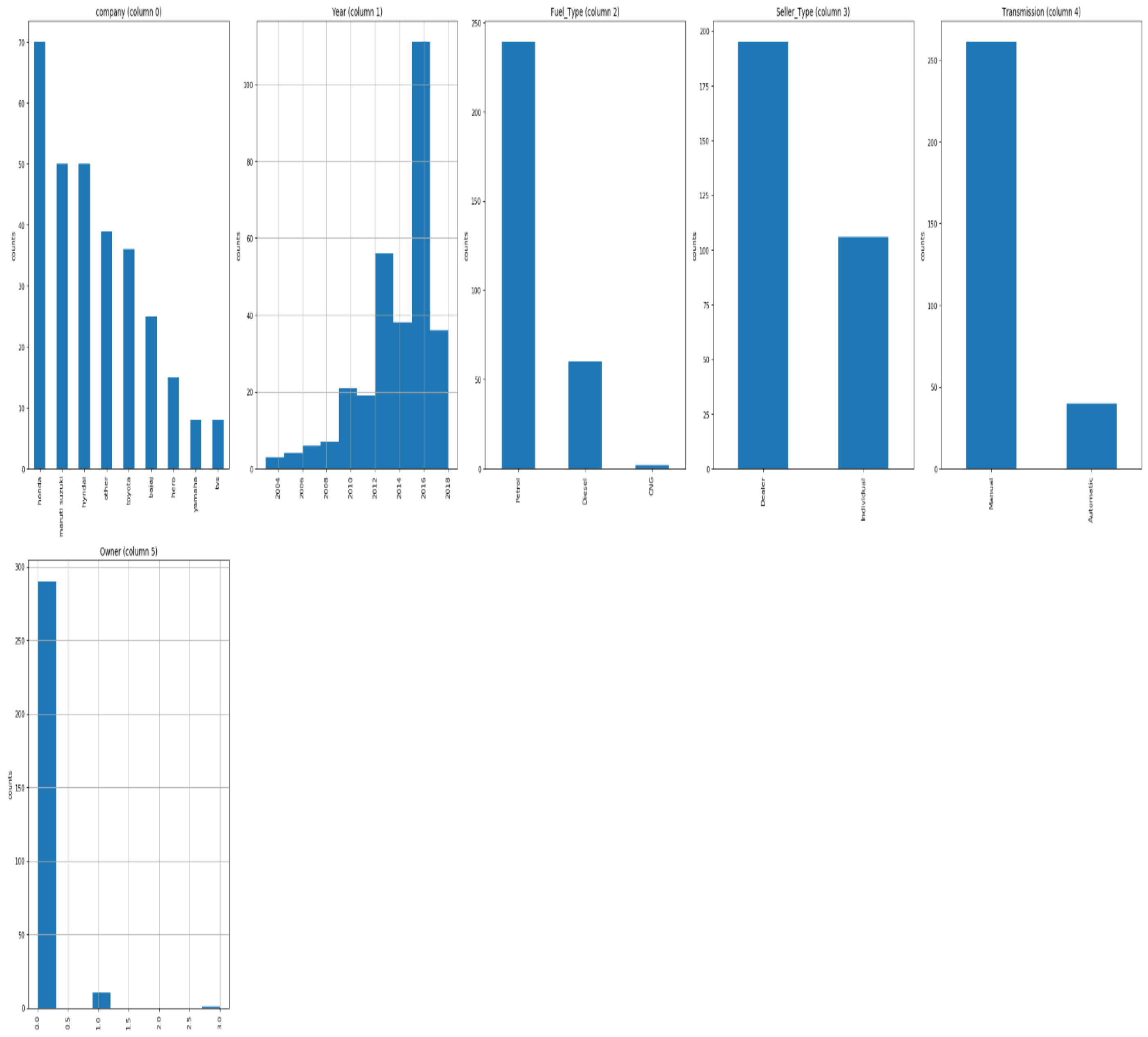
In [8]:

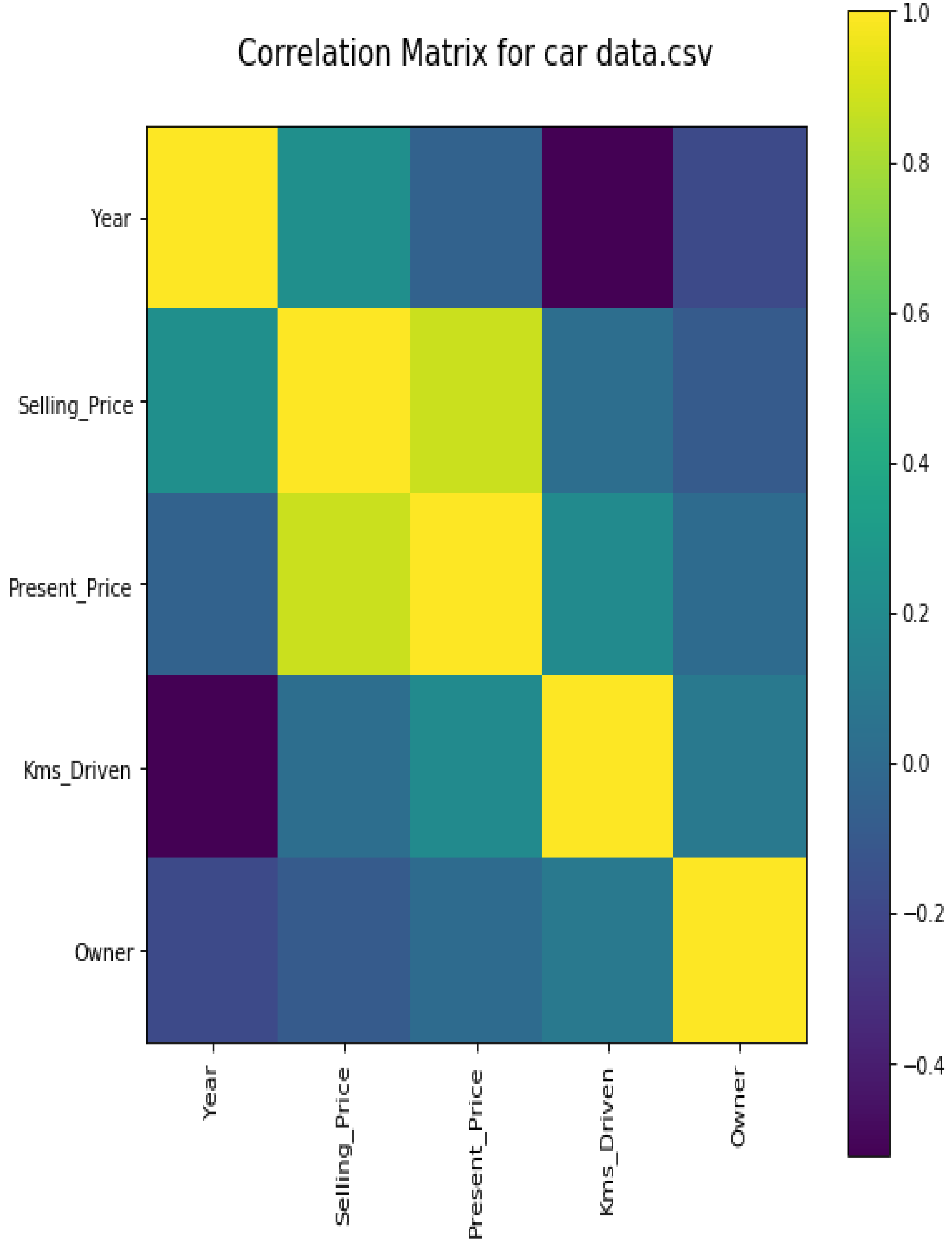
```
plotPerColumnDistribution(df1, 10, 5)
```

Correlation matrix:

```
plotCorrelationMatrix(df1, 8)
```

In [9]:

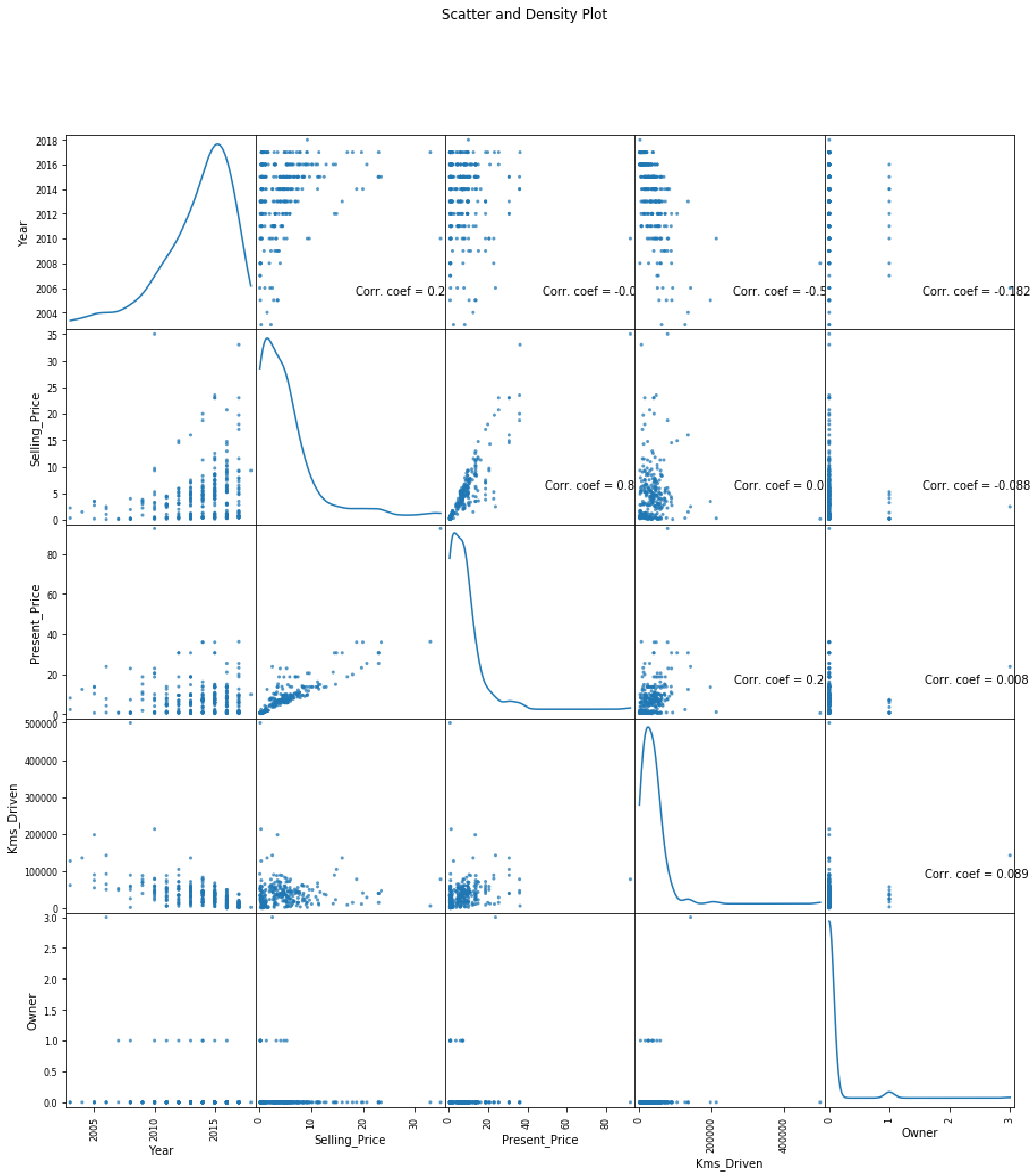




Scatter and density plots:

In [10]:

```
plotScatterMatrix(df1, 15, 10)
```



Conclusion:

- ✓ This concludes your starter analysis! To go forward from here, click the blue "Fork Notebook" button at the top of this kernel.
- ✓ This will create a copy of the code and environment for you to edit. Delete, modify, and add code as you please.