**PYTHON CASE STUDY**

**Name: Vikas Reddy Gorantla**

**Date: 17-11-2024**

**Step 1: Select a real-world dataset**

**Car Listings:** https://www.kaggle.com/datasets/yeohqiwei/car-listings?resource=download

**Step 2: Perform data preparation & cleaning**

1. **Load the dataset into a data frame using Pandas**

# Load the dataset into a data frame using Pandas

import pandas as pd

import numpy as np

file\_path = r"C:\Users\G.VIKAS REDDY\OneDrive\Documents\2023 September data.csv"

data = pd.read\_csv(file\_path)

print(data)

A screenshot of a computer

Description automatically generated

1. **Explore the number of rows & columns, ranges of values etc.**

# Check the number of rows and columns

print("Number of rows and columns:", data.shape)



# Get an overview of the DataFrame

print("\nDataFrame Info:")

print(data.info())

A screenshot of a computer code

Description automatically generated

# View basic statistical details

print("\nStatistical Summary:")

print(data.describe())

A table of numbers and symbols

Description automatically generated

# Check for missing values

print("Missing Values per Column:\n", data.isnull().sum())

A list of car parts

Description automatically generated with medium confidence

1. **Handle missing, incorrect and invalid data**

# Drop rows with any missing values

data.dropna()

A screenshot of a computer

Description automatically generated

# Check for duplicate values

duplicates = data.duplicated().sum()

print(f"Number of duplicate rows: {duplicates}")



# Drop duplicate rows

data.drop\_duplicates()

A screenshot of a computer

Description automatically generated

# Ensure that 'price' column is numeric and convert errors to NaN

df['Price'] = pd.to\_numeric(df['Price'], errors='coerce')

# Remove rows where price is negative or NaN

df = df.dropna(subset=['Price']) # Remove NaN rows in 'price'

df = df[df['Price'] >= 0] # Keep rows with non-negative price

# Print the first few rows to confirm the result

print(df.head())

A screenshot of a computer

Description automatically generated

1. **Perform any additional steps (parsing dates, creating additional columns, merging multiple dataset etc.)**

# Convert the 'DatePosted' column to datetime

data['DatePosted'] = pd.to\_datetime(data['DatePosted'])

# Extract the year and month from the 'DatePosted' for further analysis

data['YearPosted'] = data['DatePosted'].dt.year

data['MonthPosted'] = data['DatePosted'].dt.month

print(df.head())

A screenshot of a computer

Description automatically generated

# Check the data types of the columns

print(df.dtypes)

A screenshot of a computer screen

Description automatically generated

# Create a 'PriceCategory' column based on the price

data['PriceCategory'] = pd.cut(data['Price'], bins=[0, 100000, 200000, 300000],

labels=['Low', 'Medium', 'High'])

print(data)

A screenshot of a computer

Description automatically generated

import pandas as pd

# Load your datasets (replace with your file paths)

df1 = pd.read\_csv(r"C:\Users\G.VIKAS REDDY\OneDrive\Documents\2023 September data.csv")

df2 = pd.read\_csv(r"C:\Users\G.VIKAS REDDY\Downloads\annual-enterprise-survey-2023-financial-year-provisional.csv")

# Extract the year from the 'DatePosted' column in df1

df1['Year'] = pd.to\_datetime(df1['DatePosted'], errors='coerce').dt.year

# Now merge df1 and df2 on the 'Year' column

df\_merged = pd.merge(df1, df2, on='Year', how='inner') # Use 'inner' or another type of join as needed

# Display the merged DataFrame

print(df\_merged)

A screenshot of a computer

Description automatically generated

# First, check the 'Mileage' column to ensure the correct format

print(data['Mileage'].head())

# Split the 'Mileage' column to get the min and max values

data[['MileageMin', 'MileageMax']] = data['Mileage'].str.replace(' km', '').str.split(' - ', expand=True)

# Convert the new columns to numeric, removing commas and handling errors

data['MileageMin'] = pd.to\_numeric(data['MileageMin'].str.replace(',', ''), errors='coerce')

data['MileageMax'] = pd.to\_numeric(data['MileageMax'].str.replace(',', ''), errors='coerce')

# Verify the output

print(data[['Mileage', 'MileageMin', 'MileageMax']].head())

A screenshot of a computer

Description automatically generated

# Example: Plotting 'Mileage' vs 'Price'

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Mileage', y='Price', data=data)

plt.title('Mileage vs Price')

plt.xlabel('Mileage')

plt.ylabel('Price')

plt.show()

Mileage vs Price

A graph of blue dots

Description automatically generated with medium confidence

# Example: Violin plot for 'Price' with respect to 'Year'

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(10, 6))

sns.violinplot(x='DatePosted', y='Price', data=data)

plt.title('Price Distribution by Year')

plt.xlabel('DatePosted')

plt.ylabel('Price')

plt.show()

A graph showing a number of lines

Description automatically generated with medium confidence

DatePosted

1. **Write the panda and Numpy queries on the EDA Of Data**

# Select only numeric columns for the groupby operation

numeric\_columns = df.select\_dtypes(include=['number'])

# Group by 'Price' and calculate the mean for numeric columns only

grouped = numeric\_columns.groupby(df['Price']).mean()

# Group by 'Price' and calculate the sum for numeric columns only

category\_sales = numeric\_columns.groupby(df['Price']).sum()

# Check the results

print(grouped.head())

print(category\_sales.head())

A table of numbers with numbers

Description automatically generated with medium confidence

# Sorting Values

df\_sorted = df.sort\_values(by='DatePosted', ascending=False)

print(df\_sorted.head())

A screenshot of a car sales report

Description automatically generated

# Counting

data = pd.read\_csv(r"C:\Users\G.VIKAS REDDY\OneDrive\Documents\2023 September data.csv")

print(data.count())

A list of cars with numbers

Description automatically generated

**Summarize your inferences & write a conclusion**

**Write a summary of what you've learned from the analysis**

**Data Loading & Exploration:**

* The dataset was successfully loaded using pandas from the provided CSV file.
* Basic exploration of the dataset revealed its dimensions, data types, and statistical details.
  + The dataset has multiple columns, and key columns like Price, Mileage, and DatePosted were identified for analysis.
  + There are missing values in some columns, especially in numeric ones, which were handled using appropriate methods such as replacing missing values with the mean and removing rows with negative or invalid values in the Price column.

**Handling Missing & Invalid Data:**

* Missing values were addressed using the fillna() method and by dropping rows with invalid data.
* Negative values in the Price column were filtered out, ensuring that only valid rows remained.
* For the Mileage column, values were split into MileageMin and MileageMax for further analysis and converted to numeric values for consistency.

**Data Transformation & Feature Engineering:**

* The DatePosted column was converted to a datetime format, and additional features like YearPosted and MonthPosted were created for temporal analysis.
* A new feature, PriceCategory, was created based on price ranges to classify the data into low, medium, and high categories.

**Data Merging:**

* The df1 dataset was merged with another dataset df2 based on the common column Year. This allowed for enhanced analysis by combining additional information from the second dataset, such as financial data from the "annual-enterprise-survey-2023-financial-year-provisional.csv" file.

**Visualization:**

* Various plots were created to visualize the relationship between key variables:
  + **Mileage vs. Price:** A scatter plot was created to visualize how the mileage of vehicles relates to their price.
  + **Price Distribution by Year:** A violin plot was used to show the distribution of prices for each year, providing insight into how vehicle prices have varied over time.

**Data Aggregation:**

* Grouped data was analyzed by the Price column, calculating the mean and sum for numeric columns. This helped in understanding the overall trends of different price categories.
* The dataset was also sorted based on DatePosted in descending order to explore the latest entries.

**Duplicates:**

* Duplicate rows were identified and removed, ensuring the data quality for further analysis.

**Final Data Summary:**

* The cleaned and transformed dataset is now ready for further exploration and analysis. We confirmed that the Price and Mileage columns are in appropriate formats for subsequent analysis.

**Include interesting insights and graphs from previous sections**

**Insights:**

1. **Price Distribution:**
   * The PriceCategory feature gives us a clear division of the vehicles into low, medium, and high price ranges, making it easier to analyze the distribution of vehicles across these categories.
   * The violin plot shows that prices for vehicles tend to be higher in certain years, indicating trends in pricing over time.
2. **Mileage and Price Correlation:**
   * There is an inverse relationship between mileage and price, where higher-mileage vehicles tend to have lower prices, as expected.
3. **Data Cleanliness:**
   * After addressing missing values and removing duplicates, the data quality has significantly improved. The dataset is now more reliable for analysis.

**Conclusion:**

This analysis helped clean and transform a complex dataset of vehicle information. Key columns such as Price, Mileage, and DatePosted were handled appropriately, and several additional features were created for more detailed insights. The merging of additional datasets provided deeper insights, and the visualizations highlighted key trends and relationships within the data.

**Share ideas for future work on the same topic using other relevant datasets**

**Future Work:**

* **More In-Depth Analysis:** Analyze the correlation between Mileage, Price, and YearPosted further to identify other patterns.
* **Prediction Models:** Build regression models to predict the price of vehicles based on features like mileage, year, and price category.
* **Time Series Analysis:** Perform a time series analysis on vehicle prices over the years to predict future price trends.
* **Geographical Analysis:** If location data is available, perform a geographical analysis to check how vehicle prices vary by region.

**Share links to resources you found useful during your analysis**

**Here are the direct links to the official resources**

1. **Pandas Documentation:**
   * https://pandas.pydata.org/pandas-docs/stable/
2. **Matplotlib Documentation:**
   * https://matplotlib.org/stable/users/index.html
3. **Seaborn Documentation:**
   * https://seaborn.pydata.org/