Assignment-2 Results

Student: Snigdha Setty Chandaluri

ID:16352688

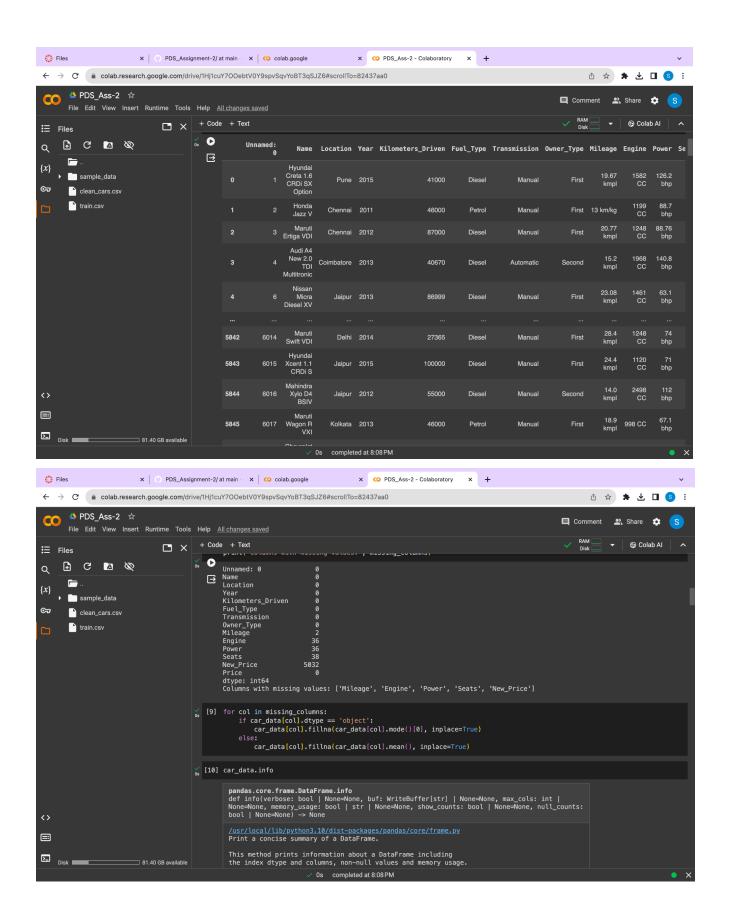
A)Look for the missing values in all the columns and either impute them (replace with mean, median, or mode) or drop them. Justify your action for this task.

1. **Verify Any Missing Data:**

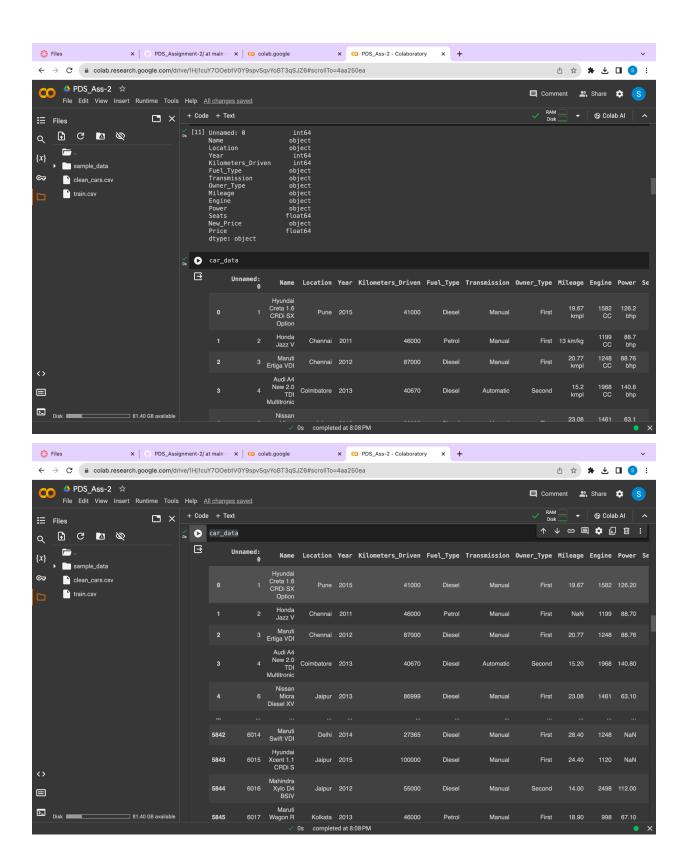
Each column's missing values are counted using the formula {missing_values = car_data.isnull().sum()}.

- The count of missing values for each column is shown by using {print(missing_values)}.
- 2. **Find Missing Values in Columns:**
 - The list "missing_columns" is generated empty.
 - It adds the missing values to the list by iteratively going over the columns.
- 3. **Impute Missing Values:** 'missing columns' is iterated through.
 - Uses the mode to fill in the missing values for object-type columns.
 - Uses the mean to fill in any missing values in non-object columns.
- 4. **Display Details about the Dataset:** Post-imputation dataset details (non-null counts, data types, memory utilization) are displayed using `car data.info()}.

Essentially, this function provides insights into the updated dataset and assures completeness by substituting acceptable measures for missing values.



- B)Remove the units from some of the attributes and only keep the numerical values (for example remove kmpl from "Mileage", CC from "Engine", bhp from "Power", and lakh from "New price").
- 1. **Exploration of Data Types**
 - Each column's data types are captured by `column data typ = car data.dtypes}.
 - The data types are displayed via `print(column_data_typ)}.
- 2. **Conversion of Data Type:** Using `.astype(str)}, the selected columns ('Mileage, Engine, Power, and New Price') are transformed to string type.
- 3. **Extract Numerical Values:** Using a regular expression, numerical values are taken out of the strings and transformed to float for the 'Mileage' column.
- 4. **Clean "Engine" Column:** "CC" is removed and "Engine" values are converted to numeric values. Erroneous inputs are forced to NaN.
- 5. **Handle Missing 'Engine' Values:** If a value is missing in the 'Engine' column, it is changed to integer type and supplied with a default value (0).
- 6. **Extract Numerical Values for 'Power', 'New_Price':** A regular expression is used to extract numerical values from the 'Power' and 'New_Price' columns, which are then converted to float.
- 7. **Final DataFrame:** The updated 'car_data' DataFrame is shown, displaying the value extractions and data type conversions that were made.



C)Change the categorical variables ("Fuel_Type" and "Transmission") into numerical one hot encoded value.

1. **Encoding of Labels:**

Makes use of `LabelEncoder()` to convert the 'Transmission' and 'Fuel_Type' columns into numerical labels.

- Inserts the encoded labels into the newly created columns "Transmission_Label" and "Fuel Type Label."

2. **Encoding One-Hot:**

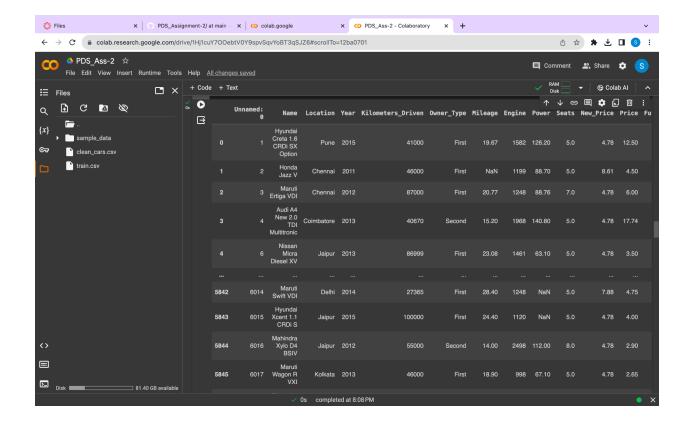
- Uses 'OneHotEncoder()' to create binary vectors from the label-encoded columns.

Makes two distinct DataFrames ('transmission_encoded_df' and 'fuel_type_encoded_df') for every new binary vector.

3. **Concatenation:** - Merges the one-hot encoded DataFrames with the original 'car_data' DataFrame.

4. Eliminate Superfluous Columns:

- Eliminates the label-encoded columns ('Fuel_Type_Label', 'Transmission_Label') and the original categorical columns ('Fuel_Type', 'Transmission').
- 5. **Final DataFrame:** Shows the updated "car_data" DataFrame with one-hot encoded representations of "Transmission" and "Fuel Type."

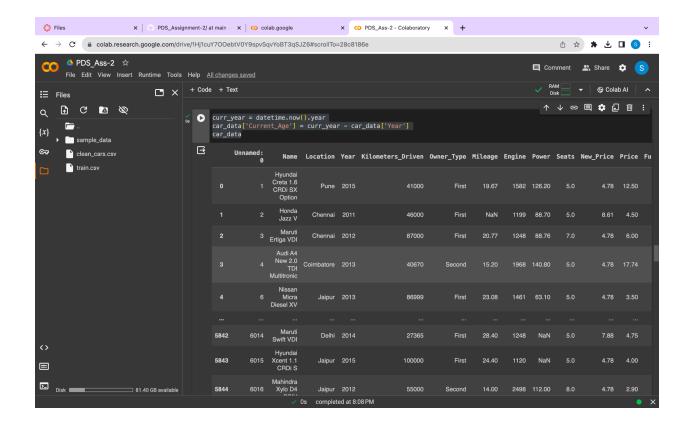


D)Create one more feature and add this column to the dataset (you can use mutate function in R for this). For example, you can calculate the current age of the car by subtracting "Year" value from the current year.

- 1. **Current Year Calculation:** Using the `datetime` module, `curr_year = datetime.now().year} retrieves the current year.
- 2. **Determine Age of Car:**

Each car's age is calculated by deducting its production year (found in the 'Year' column) from the current year using the formula {car_data['Current_Age'] = curr_year - car_data['Year']}.

3. **Result:** - The age of each automobile is now displayed in a new column called "Current_Age" in the updated "car_data" DataFrame.



E)Perform select, filter, rename, mutate, arrange and summarize with group by operations (or their equivalent operations in python) on this dataset.

1. **Choosing a Column:**

- {car_data = select_data[['Year, 'Name,' Price, 'Engine,' Location]]Name, Year, Price, Engine, and Location are the columns that are chosen, and they are then saved in a new DataFrame by {.

Data Filtering: - The dataset is filtered to include only rows where the 'Location' is 'Chennai' using the formula {filt_data = car_data[car_data['Location'] == 'Chennai']}.

3. **Renaming of Columns:**

- The 'New Price' column is renamed to 'NewPriceCAR' by running the command `car_data = car_data.rename(columns={'New_Price': 'NewPriceCAR'})}.

- {car_data['NewIncreseprice'] = car_data['NewPriceCAR'] 4 **Column Operation:**Column 'NewPriceCAR' is multiplied by three to create a new column 'NewIncreseprice' using the formula *3 {.

5. **Data Sorting:**

- The dataset is sorted in ascending order using the 'Location' column by sort_car_data = car_data.sort_values(by='Location', ascending=True)}.

6. **Aggregation and Grouping:**

- {grouped_summary = car_data.groupby('Year').The data is grouped by 'Year' and the average of the 'Mileage' for each group is computed using the formula agg(Average_MPG=('Mileage', 'mean'))}.

7. **CSV Saving:**

The adjusted 'car_data' DataFrame is saved to a CSV file called 'clean_cars.csv' with the command `car_data.to_csv('clean_cars.csv', index=False)}.

