

2025

Home Project 2

Data pre-processing with Python

DANIEL DADZIE APPIAH | 156801227 | DDAPPIAH

2ND APRIL, 2025.

Table of Contents

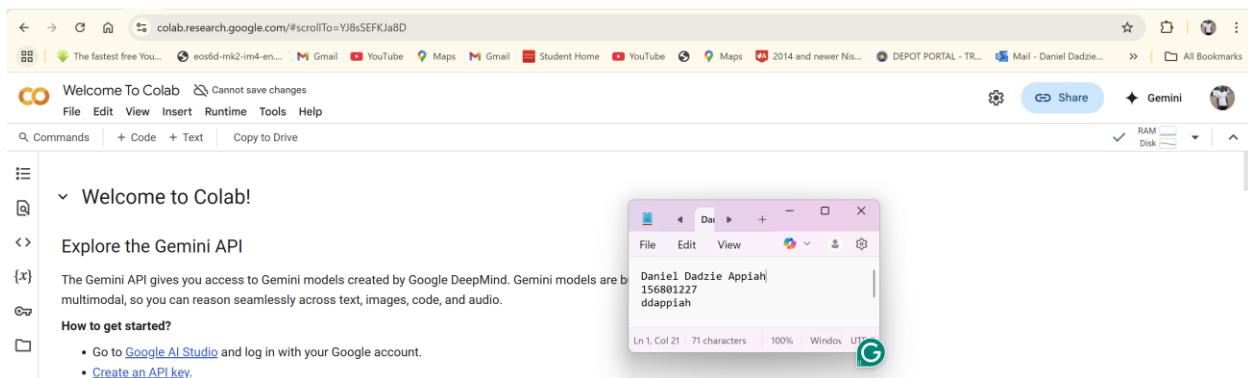
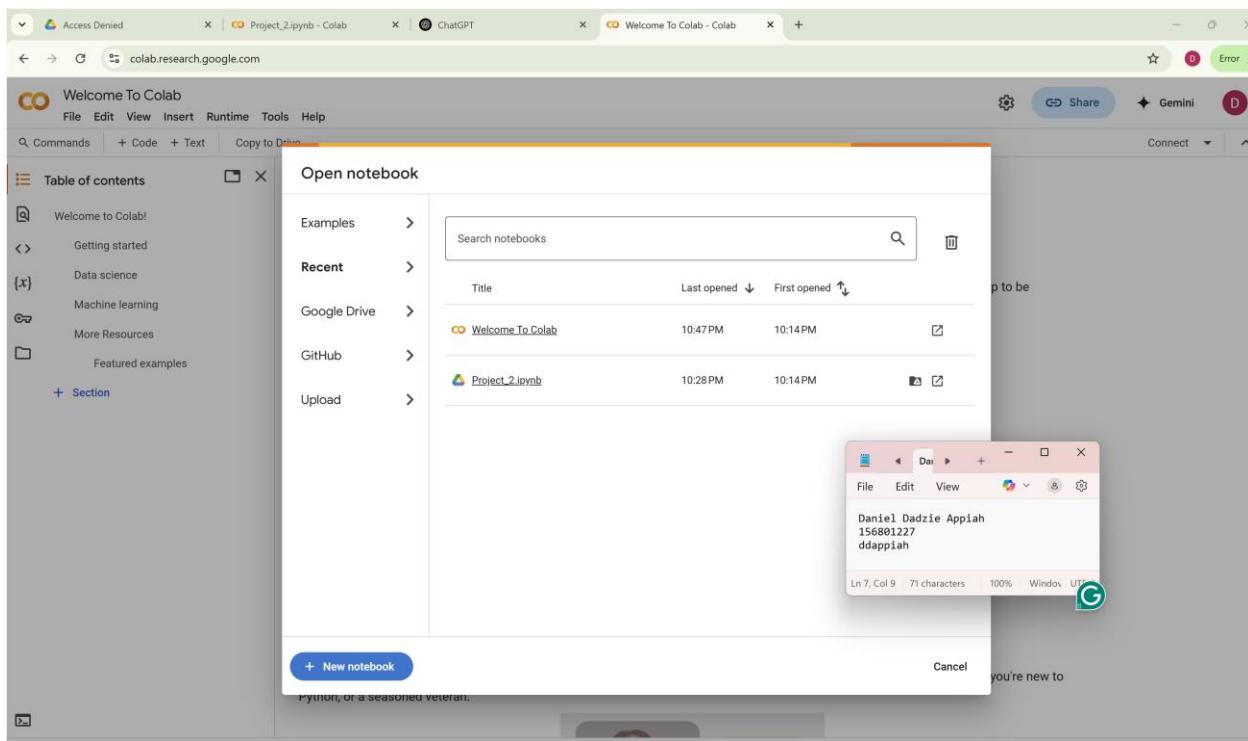
1	Introduction	2
2	0-Screen for Starting Project in Google Colab	2
3	Step 1: Load Data in Pandas.....	3
4	Display of headers.....	5
5	Explore Data: data format below	5
6	Check for Missing Values.....	6
7	Step 2: Drop Unnecessary Columns and Validating result after deleting the unnecessary columns.....	6
8	Step 2: Validating to see the deleted columns	7
9	Step 3: Drop Rows with Missing Values	7
10	Step 3: Drop Rows (handling) With Missing Values and check dataset size after dropping missing values	8
10.1	Explanation for Result	8
11	Step 4: Create Dummy Variables and Check Transformed Dataset	8
12	Step 5: Interpolate Missing Numeric Values and Confirm Missing Values Handled.....	9
12.1	Explanation of Results.....	9
13	Convert the Data Frame to NumPy and show the effect of conversion to NumPy	10
13.1	Summary & Explanation of Results.....	11
14	Step 7: Split Dataset into Training and Test Sets	12
14.1	Explanation of the Results	12

1 Introduction

Data pre-processing is a crucial step in data analysis and machine learning, ensuring that raw data is cleaned, transformed, and structured for accurate and efficient modeling. Real-world data is often incomplete, inconsistent, and noisy, making pre-processing essential for improving data quality and optimizing model performance.

2 O-Screen for Starting Project in Google Colab

Starting up the command line as an administrator. Use of Windows Terminal.



3 Step 1: Load Data in Pandas

The screenshot shows a web browser window with the URL kaggle.com/datasets/willianoliveiragibin/electric-vehicle-population. The page title is "Electric Vehicle Population". On the left, there's a sidebar with navigation links like "Create", "Home", "Competitions", "Datasets", "Models", "Code", "Discussions", "Learn", and "More". The main content area displays the dataset details, including a "Data Card" section with links to "Code (9)", "Discussion (2)", and "Suggestions (0)". Below this is the "About Dataset" section, which contains a brief description of BEV sales growth and market leaders. To the right, there are sections for "Usability", "License (CC0: Public Domain)", and "Tags" (Automobiles and Vehicles, Electronics, Data Analytics, Data Visualization, Exploratory Data Analysis). A code editor window is overlaid on the page, showing the following Python code:

```
File Edit View
Daniel Dadzie Appiah
156801227
ddappiah

Ln 7, Col 9  71 characters  100% Windows UTF-8
```

The screenshot shows a Windows File Explorer window with the path "Semester_4 > DBS900_Assessing Big Data > project_2". The left sidebar shows icons for "Gallery", "Desktop", "Download", "Documents", "Pictures", "Music", "Videos", "term proj", "Assignments", "DBS900_A", and "project_2". The main pane lists files: "DBS900_FinalHomeProject_Fall2023.docx" (Microsoft Word Document, 19 KB), "DBS900_HomeProject2_Daniel_Dadzie_Appiah.docx" (Microsoft Word Document, 12,804 KB), and "Electric_Vehicle_Population_Data.csv" (Microsoft Excel Comma Separated Values, 55,969 KB). A code editor window is overlaid on the file list, showing the same Python code as the previous screenshot.

The screenshot shows a Jupyter Notebook interface with the following details:

- File Menu:** File, Edit, View, Insert, Runtime, Tools, Help.
- Toolbar:** Share, Gemini, RAM Disk.
- Code Cell:** [1] `import pandas as pd`
[2] `df = pd.read_csv('/content/project_2/Electric_Vehicle_Population_Data.csv')`
- Text Cell:** [] Start coding or generate with AI.
- File Browser:** Shows a tree view of files and folders. The 'project_2' folder contains 'Electric_Vehicle_Population_Data.csv' and 'sample_data'. A tooltip says 'Analyze your files with code written by Gemini'.
- Code Editor:** A small window titled 'New Section' displays the following text:

```
File Edit View
Daniel Dadzie Appiah
15680127
ddappiah
```

4 Display of headers

The screenshot shows a Google Colab interface with a code cell containing the following Python code:

```
[1]: import pandas as pd  
df = pd.read_csv('/content/project_2/Electric_Vehicle_Population_Data.csv')  
df.head()
```

The output of the code is a Pandas DataFrame with 5 rows of data. The columns are:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAEV) Eligibility	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID	Vehicle Location	Electric Utility	2020 Census Tract
0	5YJ3E1EBXK	King	Seattle	WA	98178.0	2019	TESLA	MODEL 3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	220.0	0.0	37.0	477309682.0	POINT (-122.23825 47.49461)	CITY OF SEATTLE - (WA) CITY OF TACOMA - (WA)	5.3
1	5YJYGDEE3L	Kitsap	Poulsbo	WA	98370.0	2020	TESLA	MODEL Y	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	291.0	0.0	23.0	109705683.0	POINT (-122.64681 47.73689)	PUGET SOUND ENERGY INC	5.3
2	KM8KRDAF5P	Kitsap	Olalla	WA	98359.0	2023	HYUNDAI	IONIQ 5	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0.0	0.0	26.0	230390492.0	POINT (-122.54729 47.42602)	PUGET SOUND ENERGY INC	5.3
3	5UXTA6C0XM	Kitsap	Seabeck	WA	98380.0	2021	BMW	X5	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	30.0	0.0	35.0	267929112.0	POINT (-122.81585 47.64509)	PUGET SOUND ENERGY INC	5.3
4	JTMABSFV7P	Thurston	Rainier	WA	98576.0	2023	TOYOTA	RAV4 PRIME	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	42.0	0.0	2.0	236505139.0	POINT (-122.68993 46.88897)	PUGET SOUND ENERGY INC	5.3

Below the code cell, a message says "Automatic saving failed. This file was updated remotely or in another tab. Show diff".

5 Explore Data: data format below

The screenshot shows a Google Colab interface with a code cell containing the following Python code:

```
[1]: df.info()
```

The output of the code is a detailed summary of the DataFrame's structure:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232891 entries, 0 to 232890
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   VIN (1-10)      232888 non-null   object 
 1   County          232888 non-null   object 
 2   City            232888 non-null   object 
 3   State           232888 non-null   object 
 4   Postal Code     232888 non-null   float64
 5   Model Year     232891 non-null   int64  
 6   Make            232891 non-null   object 
 7   Model           232891 non-null   object 
 8   Electric Vehicle Type 232882 non-null   object 
 9   Clean Alternative Fuel Vehicle (CAEV) Eligibility 232890 non-null   object 
 10  Electric Range 232854 non-null   float64
 11  Base MSRP       232824 non-null   float64
 12  Legislative District 232438 non-null   float64
 13  DOL Vehicle ID 232898 non-null   float64
 14  Vehicle Location 232879 non-null   object 
 15  Electric Utility 232877 non-null   object 
 16  2020 Census Tract 232887 non-null   float64
dtypes: float64(6), int64(1), object(10)
memory usage: 30.2+ MB
```

6 Check for Missing Values

The screenshot shows a Jupyter Notebook interface. The code cell contains:

```
# Check for missing values
df.isnull().sum()
```

The output cell displays the count of missing values for each column:

	Count
VIN (1-10)	0
County	3
City	3
State	0
Postal Code	3
Model Year	0
Make	0
Model	0
Electric Vehicle Type	9
Clean Alternative Fuel Vehicle (CAFV) Eligibility	1
Electric Range	37
Base MSRP	37
Legislative District	453
DOL Vehicle ID	1
Vehicle Location	12
Electric Utility	14
2020 Census Tract	4

dtype: int64

7 Step 2: Drop Unnecessary Columns and Validating result after deleting the unnecessary columns

Issues the command twice. I too picture after it has been deleted

The screenshot shows a Google Colab interface. The code cell contains:

```
cols_to_drop = ['VIN (1-10)', 'Vehicle Location']
df = df.drop(columns=cols_to_drop)
print(df.info())
```

The output cell displays the DataFrame information:

	Non-Null Count	Dtype
0 County	235689	object
1 City	235689	object
2 State	235692	object
3 Postal Code	235681	float64
4 Model Year	235692	int64
5 Make	235685	object
6 Model	235693	object
7 Electric Vehicle Type	235684	object
8 Clean Alternative Fuel Vehicle (CAFV) Eligibility	235692	object
9 Electric Range	235656	float64
10 Base MSRP	235656	float64
11 Legislative District	235198	float64
12 DOL Vehicle ID	235692	int64
13 Electric Utility	235660	object
14 2020 Census Tract	235689	float64

dtypes: float64(5), int64(2), object(8)
memory usage: 27.0+ MB
None

8 Step 2: Validating to see the deleted columns

The screenshot shows a Google Colab interface. On the left, there's a sidebar with file navigation and analysis tools. The main area contains a code cell output showing the structure of a DataFrame named 'df'. The output includes:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232891 entries, 0 to 232890
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   County          232888 non-null   object  
 1   City            232888 non-null   object  
 2   State           232891 non-null   object  
 3   Postal Code     232888 non-null   float64 
 4   Model Year     232891 non-null   int64  
 5   Make            232891 non-null   object  
 6   Model           232891 non-null   object  
 7   Electric Vehicle Type 232882 non-null   object  
 8   Clean Alternative Fuel Vehicle (CAFV) Eligibility 232894 non-null   object  
 9   Electric Range  232854 non-null   float64 
 10  Base MSRP       232438 non-null   float64 
 11  Legislative District 232896 non-null   float64 
 12  DOL Vehicle ID 232877 non-null   object  
 13  Electric Utility 232887 non-null   float64 
 14  2020 Census Tract 232887 non-null   float64 
dtypes: float64(6), int64(1), object(8)
memory usage: 26.7+ MB
```

To the right of the code cell, a terminal window displays the command 'ddappiah'.

9 Step 3: Drop Rows with Missing Values

The screenshot shows a Google Colab interface. On the left, there's a sidebar with file navigation and analysis tools. The main area contains a code cell with the following content:

```
# Drop rows where critical categorical values are missing
df = df.dropna(subset=['County', 'City', 'Electric Vehicle Type'])

# Fill missing numeric values with median
num_cols = ['Postal Code', 'Electric Range', 'Base MSRP', 'Legislative District', 'DOL Vehicle ID', '2020 Census Tract']
df[num_cols] = df[num_cols].fillna(df[num_cols].median())

# Verify missing values
df.isnull().sum()
```

The code cell has a warning message at the top:

cipython-input-12-ac92dfcdac49>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

Below the code cell, a note about indexing is shown:

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df[num_cols] = df[num_cols].fillna(df[num_cols].median())

To the right of the code cell, a terminal window displays the command 'ddappiah'.

10 Step 3: Drop Rows (handling) With Missing Values and check dataset size after dropping missing values

The screenshot shows a Google Colab notebook interface. A code cell contains the following Python code:

```
# Step 3: Handle Missing Values
df = df.dropna()
print(df.info()) # Check dataset size after dropping NaNs
```

The output of the code shows the DataFrame structure:

```
<class 'pandas.core.frame.DataFrame'>
Index: 235111 entries, 0 to 235691
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   County          235111 non-null   object  
 1   City            235111 non-null   object  
 2   State           235111 non-null   object  
 3   Postal Code    235111 non-null   float64 
 4   Model Year     235111 non-null   int64  
 5   Make            235111 non-null   object  
 6   Model           235111 non-null   object  
 7   Electric Vehicle Type 235111 non-null   object  
 8   Clean Alternative Fuel Vehicle (CAVF) Eligibility 235111 non-null   object  
 9   Electric Range  235111 non-null   float64 
 10  Base MSRP       235111 non-null   float64 
 11  Legislative District 235111 non-null   float64 
 12  DOL Vehicle ID 235111 non-null   int64  
 13  Electric Utility 235111 non-null   object  
 14  2020 Census Tract 235111 non-null   float64 
dtypes: float64(5), int64(2), object(8)
memory usage: 28.7+ MB
None
```

A preview window on the right shows the first few rows of the DataFrame:

	Column	Non-Null Count	Dtype
0	County	235111	non-null object
1	City	235111	non-null object
2	State	235111	non-null object
3	Postal Code	235111	non-null float64
4	Model Year	235111	non-null int64
5	Make	235111	non-null object
6	Model	235111	non-null object
7	Electric Vehicle Type	235111	non-null object
8	Clean Alternative Fuel Vehicle (CAVF) Eligibility	235111	non-null object
9	Electric Range	235111	non-null float64
10	Base MSRP	235111	non-null float64
11	Legislative District	235111	non-null float64
12	DOL Vehicle ID	235111	non-null int64
13	Electric Utility	235111	non-null object
14	2020 Census Tract	235111	non-null float64

10.1 Explanation for Result

This line uses Pandas' `dropna()` method to remove any row in the DataFrame `df` that contains at least one missing value (`NaN`).

This prints an updated summary of the DataFrame after dropping rows with missing values.

11 Step 4: Create Dummy Variables and Check Transformed Dataset

The screenshot shows a Google Colab notebook interface. A code cell contains the following Python code:

```
# Step 5: Interpolate Missing Numeric Values
numeric_cols = ['Postal Code', 'Electric Range', 'Base MSRP', 'Legislative District', '2020 Census Tract']
for col in numeric_cols:
    if df[col].isnull().sum() > 0:
        df[col] = df[col].interpolate()
print(df.info()) # Confirm missing values handled
```

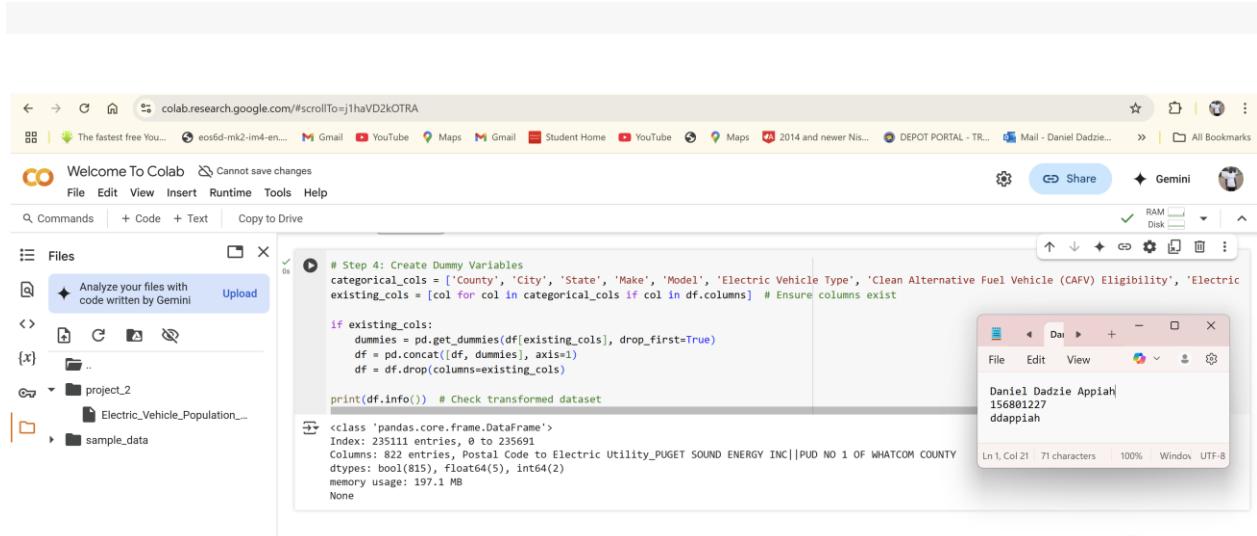
The output of the code shows the DataFrame structure:

```
<class 'pandas.core.frame.DataFrame'>
Index: 235111 entries, 0 to 235691
Columns: 822 entries, Postal Code to Electric Utility_PUGET SOUND ENERGY INC||PUD NO 1 OF WHATCOM COUNTY
dtypes: bool(815), float64(5), int64(2)
memory usage: 197.1 MB
None
```

A preview window on the right shows the first few rows of the DataFrame:

	Column	Non-Null Count	Dtype
0	County	235111	non-null object
1	City	235111	non-null object
2	State	235111	non-null object
3	Postal Code	235111	non-null float64
4	Model Year	235111	non-null int64
5	Make	235111	non-null object
6	Model	235111	non-null object
7	Electric Vehicle Type	235111	non-null object
8	Clean Alternative Fuel Vehicle (CAVF) Eligibility	235111	non-null object
9	Electric Range	235111	non-null float64
10	Base MSRP	235111	non-null float64
11	Legislative District	235111	non-null float64
12	DOL Vehicle ID	235111	non-null int64
13	Electric Utility	235111	non-null object
14	2020 Census Tract	235111	non-null float64

12 Step 5: Interpolate Missing Numeric Values and Confirm Missing Values Handled



The screenshot shows a Google Colab notebook interface. The left sidebar displays a file tree with 'project_2' containing 'Electric_Vehicle_Population...' and 'sample_data'. The main workspace contains Python code for creating dummy variables:

```
# Step 4: Create Dummy Variables
categorical_cols = ['County', 'City', 'State', 'Make', 'Model', 'Electric Vehicle Type', 'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Electric existing_cols = [col for col in categorical_cols if col in df.columns] # Ensure columns exist

if existing_cols:
    dummies = pd.get_dummies(df[existing_cols], drop_first=True)
    df = pd.concat([df, dummies], axis=1)
    df = df.drop(columns=existing_cols)

print(df.info()) # Check transformed dataset
```

The output pane shows the result of the `df.info()` command:

```
<class 'pandas.core.frame.DataFrame'>
Index: 235111 entries, 0 to 235691
Columns: 822 entries, Postal Code to Electric Utility_PUGET SOUND ENERGY INC||PUD NO 1 OF WHATCOM COUNTY
dtypes: bool(815), float64(5), int64(2)
memory usage: 197.1 MB
None
```

A small terminal window is also visible in the bottom right corner.

12.1 Explanation of Results

The dataset contains 235,111 entries. After one-hot encoding categorical variables, the total number of columns increased to 822. The dataset now has:

- 815 boolean columns (dummy variables for categorical features).
- 5 float64 columns (numeric data, including interpolated missing values).
- 2 int64 columns (discrete numerical values).
- The memory usage is 197.1 MB, indicating a significant increase due to the expansion of dummy variables.

This transformation ensures that the dataset is fully numerical and ready for machine learning.

13 Convert the Data Frame to NumPy and show the effect of conversion to NumPy

The screenshot shows a Google Colab notebook interface. In the code cell, a script is run to demonstrate DataFrame conversion to NumPy arrays. The terminal window shows the resulting output.

```
# Step 6: Convert DataFrame to NumPy Arrays
X = df.drop(columns=['Model Year']).values # Example target variable
y = df[['Model Year']].values

# Show effect of conversion to NumPy
print("\nDataFrame Head:\n", df.head())
print("NumPy X shape:", X.shape)
print("NumPy y shape:", y.shape)
print("\nFirst 5 rows of X (as NumPy array):\n", X[:5])
print("\nFirst 5 values of y (as NumPy array):\n", y[:5])

DataFrame Head:
   Postal Code Model Year  Electric Range  Base MSRP  Legislative District \
0    98178.0      2019       220.0        0.0          37.0
1    98370.0      2020       291.0        0.0          23.0
2    98359.0      2023        0.0        0.0          26.0
3    98380.0      2021       30.0        0.0          35.0
4    98576.0      2023       42.0        0.0          2.0

   DOL Vehicle ID 2020 Census Tract County_Astotin County_Benton \
0    4773095682  5.303301e+10     False    False
1    109705683  5.303509e+10     False    False
2    2303960492  5.303509e+10     False    False
3    267929112  5.303509e+10     False    False
4    236505139  5.306701e+10     False    False

   County_Chehal ... Electric Utility_PORTLAND GENERAL ELECTRIC CO \
0      False ...                                False
1      False ...                                False
2      False ...                                False
3      False ...                                False
4      False ...                                False

   Electric Utility_PUD NO 1 OF CHERLAN COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF DOUGLAS COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF OKANOGAN COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF PEND OREILLE COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF WHATCOM COUNTY \
0           False
1           False
2           False
3           False
4           False
```

Detailed description: This screenshot shows a Google Colab session. The code cell contains Python code to convert a DataFrame to NumPy arrays and print their shapes and first few rows. The terminal window shows the execution results, including the DataFrame head, NumPy arrays for X and y, and the first five rows of each. The environment includes a sidebar with files and a status bar indicating 70.52 GB available.

The screenshot shows a Google Colab notebook interface. In the code cell, a script is run to demonstrate DataFrame conversion to NumPy arrays. The terminal window shows the resulting output.

```
# Show effect of conversion to NumPy
print("\nDataFrame Head:\n", df.head())
print("NumPy X shape:", X.shape)
print("NumPy y shape:", y.shape)
print("\nFirst 5 rows of X (as NumPy array):\n", X[:5])
print("\nFirst 5 values of y (as NumPy array):\n", y[:5])

DataFrame Head:
   Postal Code Model Year  Electric Range  Base MSRP  Legislative District \
0    98178.0      2019       220.0        0.0          37.0
1    98370.0      2020       291.0        0.0          23.0
2    98359.0      2023        0.0        0.0          26.0
3    98380.0      2021       30.0        0.0          35.0
4    98576.0      2023       42.0        0.0          2.0

   DOL Vehicle ID 2020 Census Tract County_Astotin County_Benton \
0    4773095682  5.303301e+10     False    False
1    109705683  5.303509e+10     False    False
2    2303960492  5.303509e+10     False    False
3    267929112  5.303509e+10     False    False
4    236505139  5.306701e+10     False    False

   County_Chehal ... Electric Utility_PORTLAND GENERAL ELECTRIC CO \
0      False ...                                False
1      False ...                                False
2      False ...                                False
3      False ...                                False
4      False ...                                False

   Electric Utility_PUD NO 1 OF CHERLAN COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF DOUGLAS COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF OKANOGAN COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF PEND OREILLE COUNTY \
0           False
1           False
2           False
3           False
4           False

   Electric Utility_PUD NO 1 OF WHATCOM COUNTY \
0           False
1           False
2           False
3           False
4           False
```

Detailed description: This screenshot shows a Google Colab session. The code cell contains Python code to convert a DataFrame to NumPy arrays and print their shapes and first few rows. The terminal window shows the execution results, including the DataFrame head, NumPy arrays for X and y, and the first five rows of each. The environment includes a sidebar with files and a status bar indicating 70.52 GB available.

```

1      False
2      False
3      False
4      False
      utility_PUGET SOUND ENERGY INC \
0      False
1      True
2      True
3      True
4      True
      Electric Utility_PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA) \
0      False
1      False
2      False
3      False
4      False
      Electric Utility_PUGET SOUND ENERGY INC||PUD NO 1 OF WHATCOM COUNTY
0      False
1      False
2      False
3      False
4      False
[5 rows x 822 columns]
NumPy X shape: (235111, 821)
NumPy y shape: (235111,)

First 5 rows of X (as NumPy array):
[[98178.0 220.0 0.0 ... False False False]
 [98370.0 291.0 0.0 ... True False False]
 [98359.0 0.0 0.0 ... True False False]
 [98380.0 30.0 0.0 ... True False False]
 [98576.0 42.0 0.0 ... True False False]]

First 5 values of y (as NumPy array):
[2019 2020 2023 2021 2023]

```

Daniel Dadzie Appiah
156801227
ddappiah

13.1 Summary & Explanation of Results

DataFrame Overview:

The dataset initially contained multiple columns, including Postal Code, Model Year, Electric Range, Base MSRP and many more. Unnecessary columns (DOL Vehicle ID) were removed.

Missing values were dropped or interpolated for numerical columns.

Effect of Converting to NumPy Arrays:

x (features) and y (target variable: Model Year) were extracted from the DataFrame.

The conversion to NumPy arrays results in:

x shape: (235,111 rows, 821 features), y shape: (235,111 rows, 1 target variable)

Verifying the Conversion:

The first 5 rows of x show numerical and categorical (dummy) variables converted to True/False values. The first 5 values of y confirm that the Model Year is correctly extracted.

Splitting Data into Training & Test Sets:

The dataset was split into:

Training set: 70% of the data (164,577 rows, 821 features). Test set: 30% of the data (70,534 rows, 821 features). This ensures a balanced dataset for machine learning models.

14 Step 7: Split Dataset into Training and Test Sets

The screenshot shows a Google Colab notebook interface. On the left, there's a sidebar with 'Files' containing a 'project_2' folder with 'Electric_Vehicle_Population.csv' and 'sample_data'. The main area has a code cell with the following content:

```
!pip install scikit-learn
from sklearn.model_selection import train_test_split

# Step 7: Split Dataset into Training and Test Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print(f"Training set size: {X_train.shape}, Test set size: {X_test.shape}")

Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Training set size: (164577, 821), Test set size: (70534, 821)
```

Below the code cell is a terminal window showing the output of the command:

```
Daniel Dadzie Appiah
156801227
ddappiah
```

14.1 Explanation of the Results

Data Preprocessing Steps:

The dataset was loaded from Google Drive and checked for missing values.

Irrelevant columns (DOL Vehicle ID) were dropped. Rows with missing values were removed.

Categorical variables (County, City, State) were converted into dummy variables.

Missing numeric values (Postal Code, Electric Range) were interpolated.

Conversion to NumPy Arrays:

The df DataFrame was converted into two NumPy arrays: X (features), which contains all columns except Model Year. y (target variable) contains only the Model Year. The shapes of X and Y confirm a successful conversion.

Dataset Splitting for Machine Learning:

The dataset was split into:

Training set : 70% of the data (X_train and y_train) : 164,577 samples with 821 features

Test set : 30% of the data (X_test and y_test) : 70,534 samples with 821 features

The split ensures the model has enough data for training and evaluation.