

2025

# Home Project 2

## **Data pre-processing with Python**

DANIEL DADZIE APPIAH | 156801227 | DDAPPIAH

**2<sup>ND</sup> APRIL, 2025.**

## Table of Contents

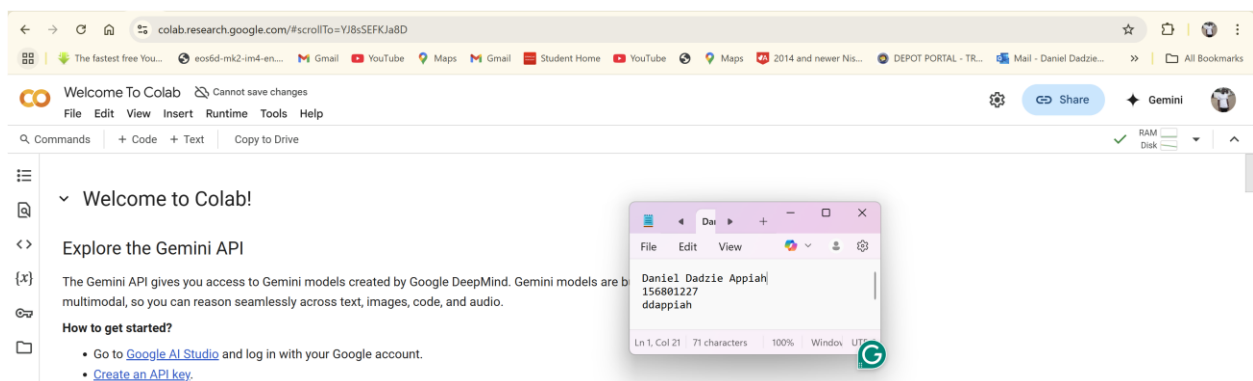
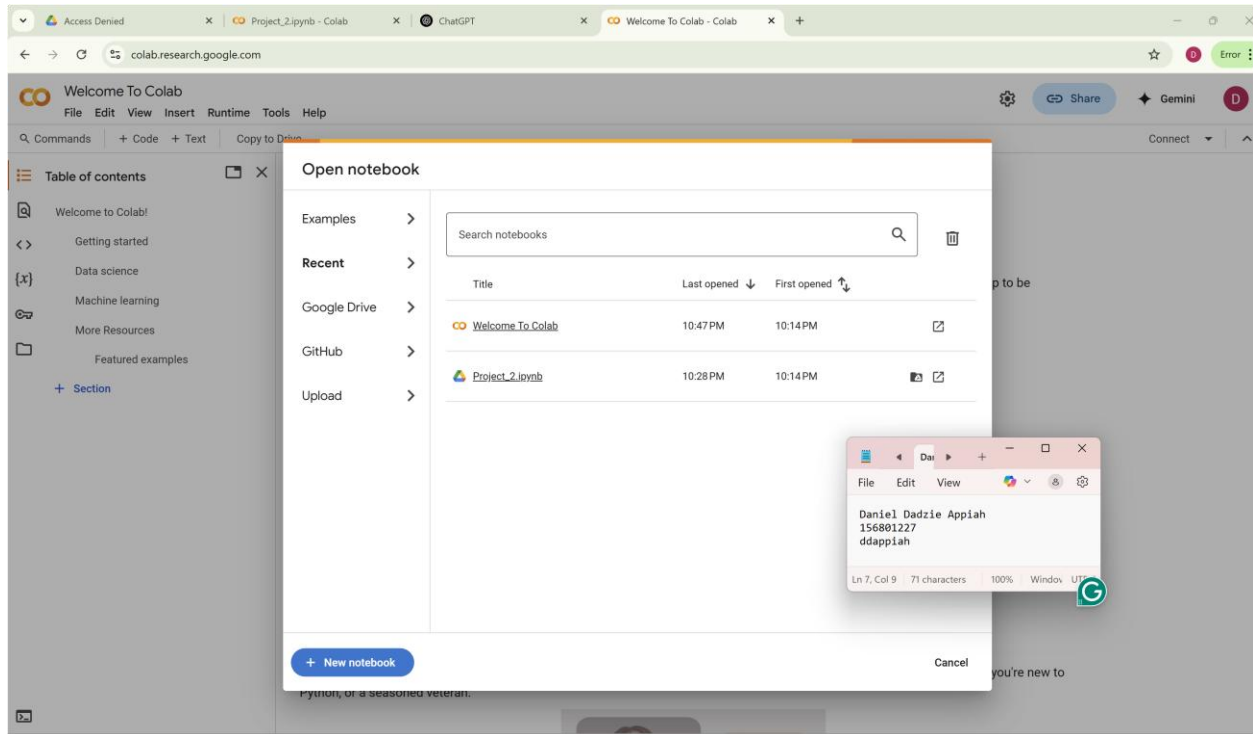
1	Introduction .....	2
2	0-Screen for Starting Project in Google Colab .....	2
3	Step 1: Load Data in Pandas.....	3
4	Display of headers.....	5
5	Explore Data: data format below .....	5
6	Check for Missing Values.....	6
7	Step 2: Drop Unnecessary Columns and Validating result after deleting the unnecessary 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

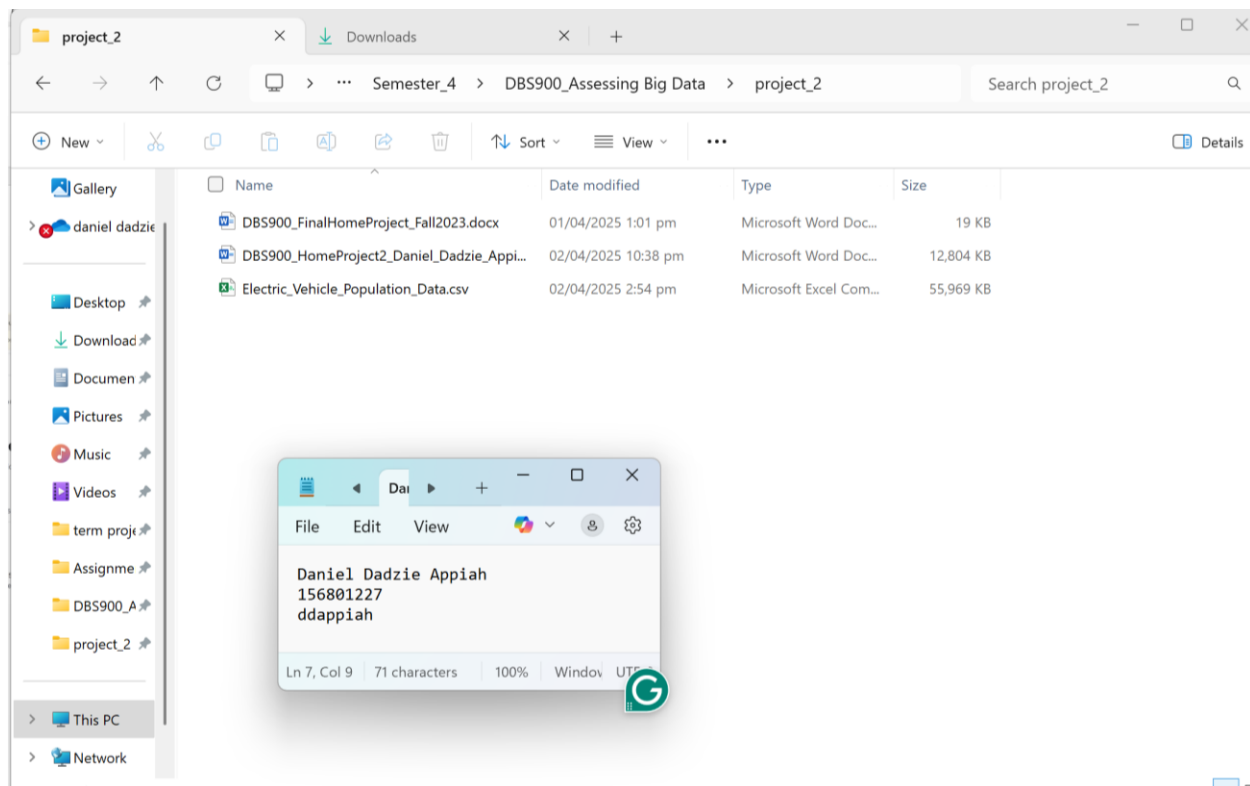
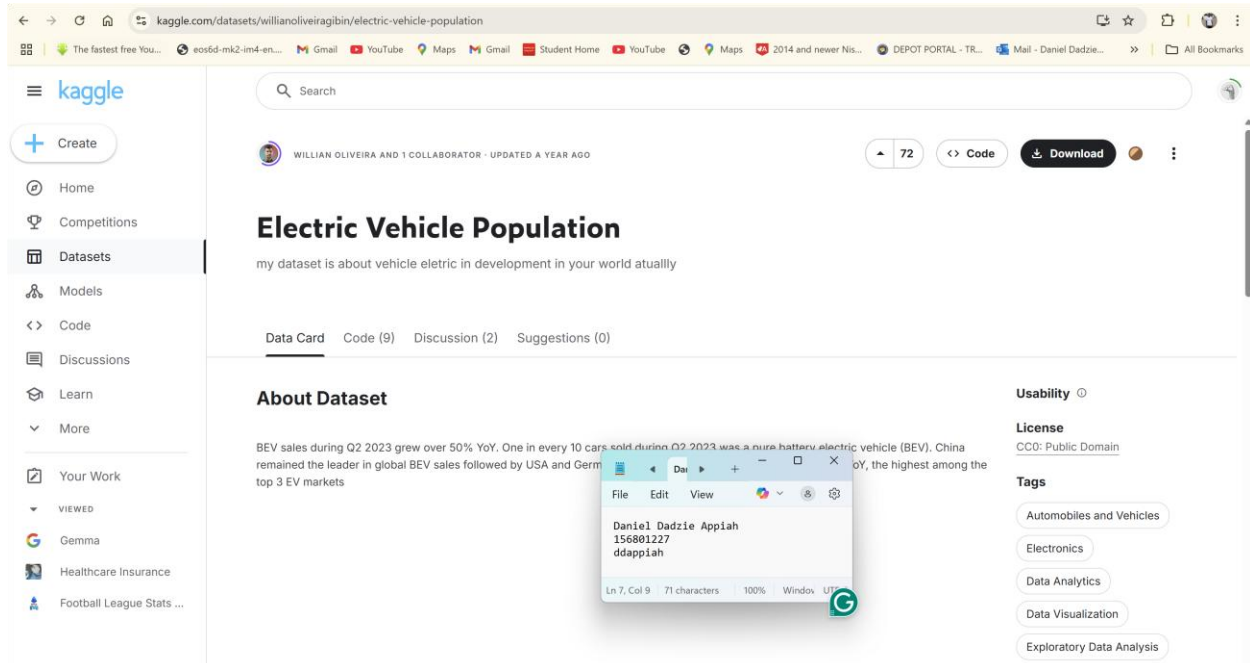
Data pre-processing is a crucial step in data analysis and machine learning, ensuring that raw data is cleaned, transformed, and structured for accurate and efficient modeling. Real-world data is often incomplete, inconsistent, and noisy, making pre-processing essential for improving data quality and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

## 2 0-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



Project\_2.ipynb

Saving failed since 10:32 PM

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text

RAM Disk

Files

Analyze your files with code written by Gemini Upload

project\_2

Electric\_Vehicle...

sample\_data

[1] import pandas as pd

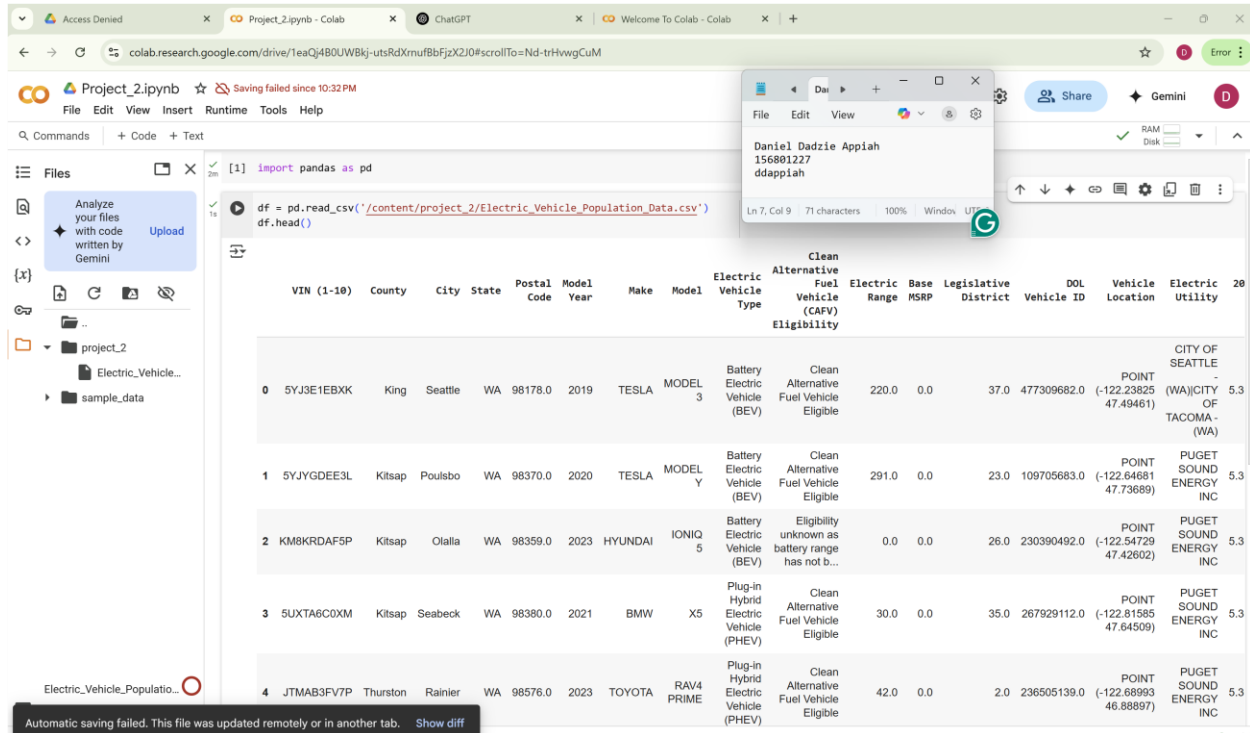
df = pd.read\_csv('/content/project\_2/Electric\_Vehicle\_Population\_Data.csv')

[ ] Start coding or generate with AI.

New Section

Daniel Dadzie Appiah  
156801227  
ddappiah

## 4 Display of headers



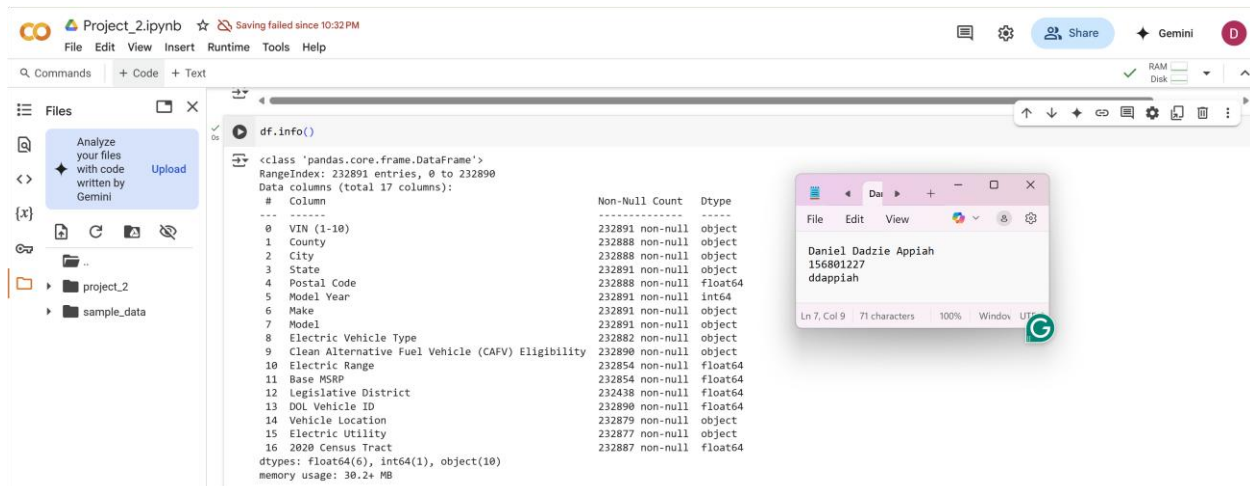
The screenshot shows a Google Colab notebook titled "Project\_2.ipynb". The code cell contains the following Python code:

```
[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 showing the first 5 rows of the "Electric\_Vehicle\_Population\_Data.csv" file. The DataFrame has 17 columns: VIN (1-10), County, City, State, Postal Code, Year, Make, Model, Electric Vehicle Type, Clean Alternative Fuel Vehicle (CAFV) Eligibility, Electric Range, Base MSRP, Legislative District, Vehicle ID, Vehicle Location, Electric Utility, and 2020 Census Tract. The first row shows a Tesla Model 3 with VIN 5YJ3E1EBXX, located in King County, Seattle, WA, with a postal code of 98178.0 and a year of 2019. The second row shows a Tesla Model Y with VIN 5YJYGDEE3L, located in Kitsap County, Poulsbo, WA, with a postal code of 98370.0 and a year of 2020. The third row shows a Hyundai Ioniq 5 with VIN KM8KRDAF5P, located in Kitsap County, Olalla, WA, with a postal code of 98359.0 and a year of 2023. The fourth row shows a BMW X5 with VIN 5UXTA6C0XM, located in Kitsap County, Seabeck, WA, with a postal code of 98380.0 and a year of 2021. The fifth row shows a Toyota RAV4 Prime with VIN JTMAB3FV7P, located in Thurston County, Rainier, WA, with a postal code of 98576.0 and a year of 2023.

## 5 Explore Data: data format below



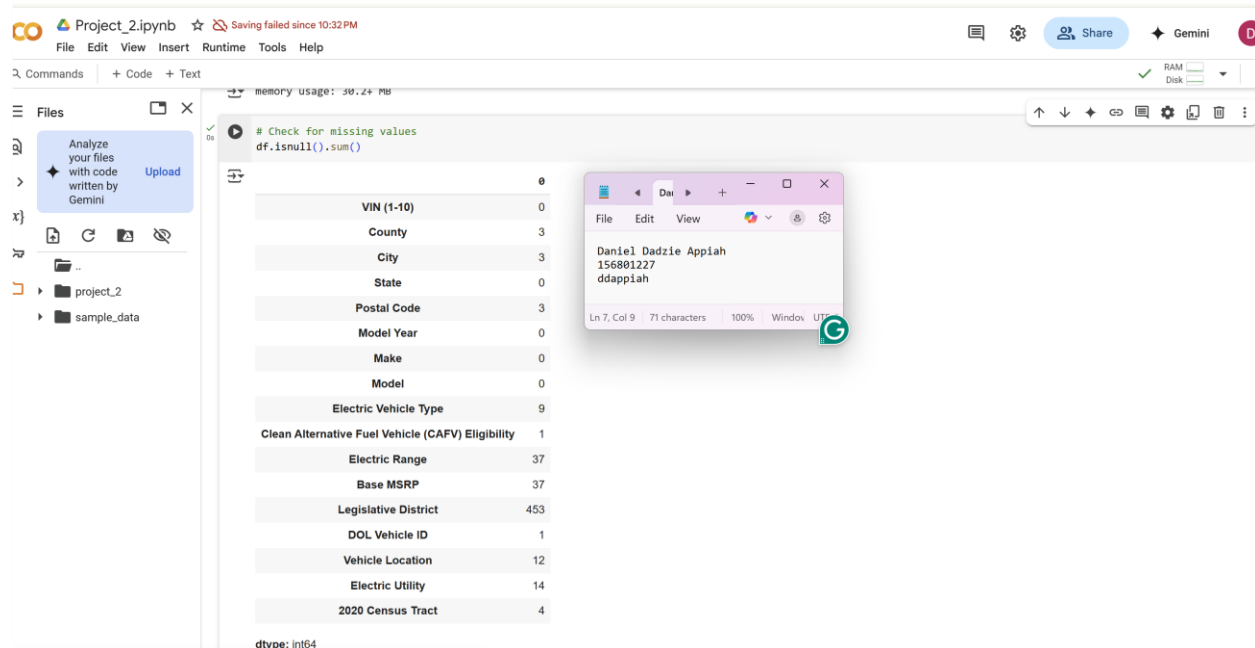
The screenshot shows a Google Colab notebook titled "Project\_2.ipynb". The code cell contains the following Python code:

```
df.info()
```

The output of the code is a summary of the DataFrame's structure, including the number of entries, the number of columns, the data types of each column, and the non-null counts for each column. The output is as follows:

```
<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)                             232891 non-null object  
 1   County                                232888 non-null object  
 2   City                                  232888 non-null object  
 3   State                                 232891 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 (CAFV) Eligibility 232890 non-null object  
10   Electric Range                         232854 non-null float64  
11   Base MSRP                             232854 non-null float64  
12   Legislative District                   232438 non-null float64  
13   DOL Vehicle ID                        232890 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 with a file explorer on the left containing 'project\_2' and 'sample\_data'. The code cell contains the following Python code:

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

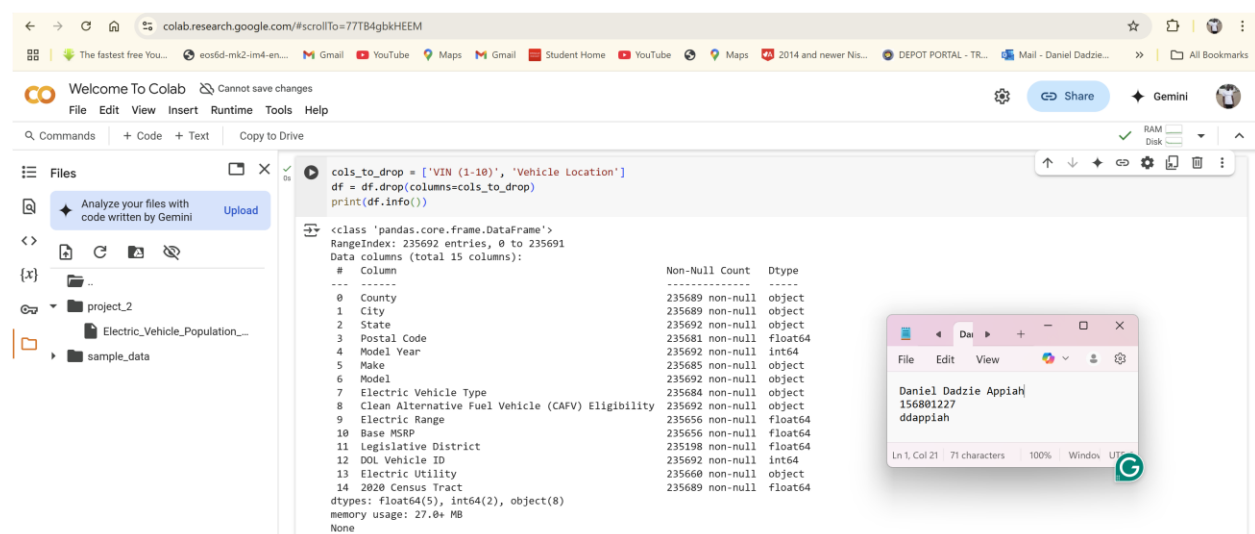
The output is a series of counts for each column:

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

The dtype is int64. A small window titled 'Dai' is open, showing the text: 'Daniel Dadzie Appiah', '156801227', and 'ddappiah'.

## 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 Jupyter Notebook interface with a file explorer on the left containing 'project\_2' and 'sample\_data'. The code cell contains the following Python code:

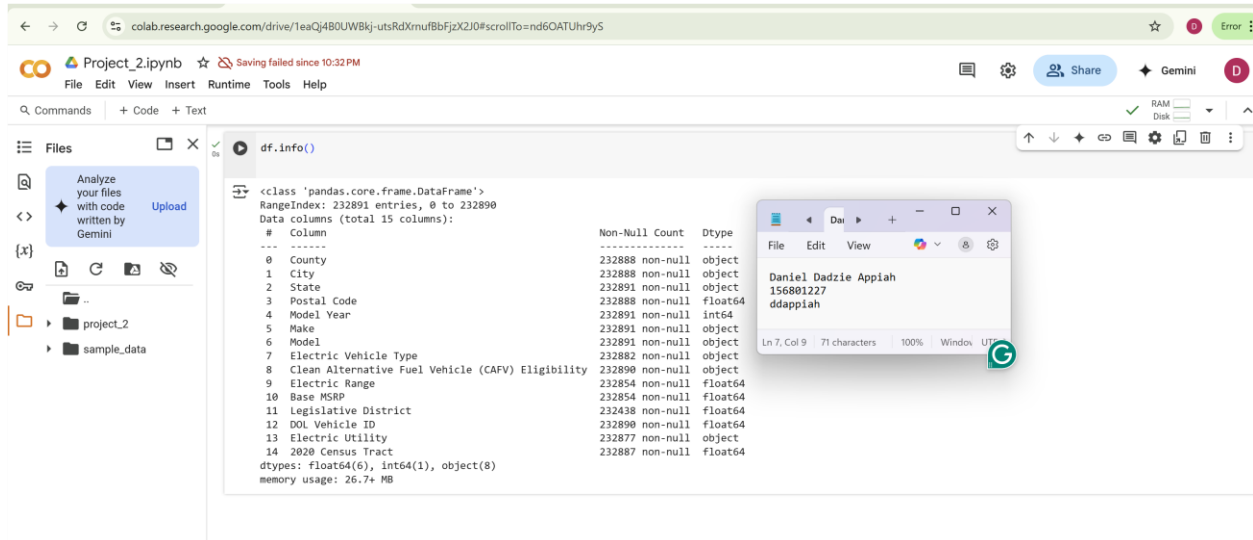
```
cols_to_drop = ['VIN (1-10)', 'Vehicle Location']
df = df.drop(columns=cols_to_drop)
print(df.info())
```

The output is a DataFrame information summary:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235692 entries, 0 to 235691
Data columns (total 15 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   County                                   235689 non-null object
1   City                                    235689 non-null object
2   State                                   235689 non-null object
3   Postal Code                             235681 non-null float64
4   Model Year                             235692 non-null int64
5   Make                                    235685 non-null object
6   Model                                   235692 non-null object
7   Electric Vehicle Type                   235684 non-null object
8   Clean Alternative Fuel Vehicle (CAFV) Eligibility 235692 non-null object
9   Electric Range                         235656 non-null float64
10  Base MSRP                             235656 non-null float64
11  Legislative District                   235198 non-null float64
12  DOL Vehicle ID                       235692 non-null int64
13  Electric Utility                      235660 non-null object
14  2020 Census Tract                     235689 non-null float64
dtypes: float64(5), int64(2), object(8)
memory usage: 27.0+ MB
None
```

A small window titled 'Dai' is open, showing the text: 'Daniel Dadzie Appiah', '156801227', and 'ddappiah'.

## 8 Step 2: Validating to see the deleted columns



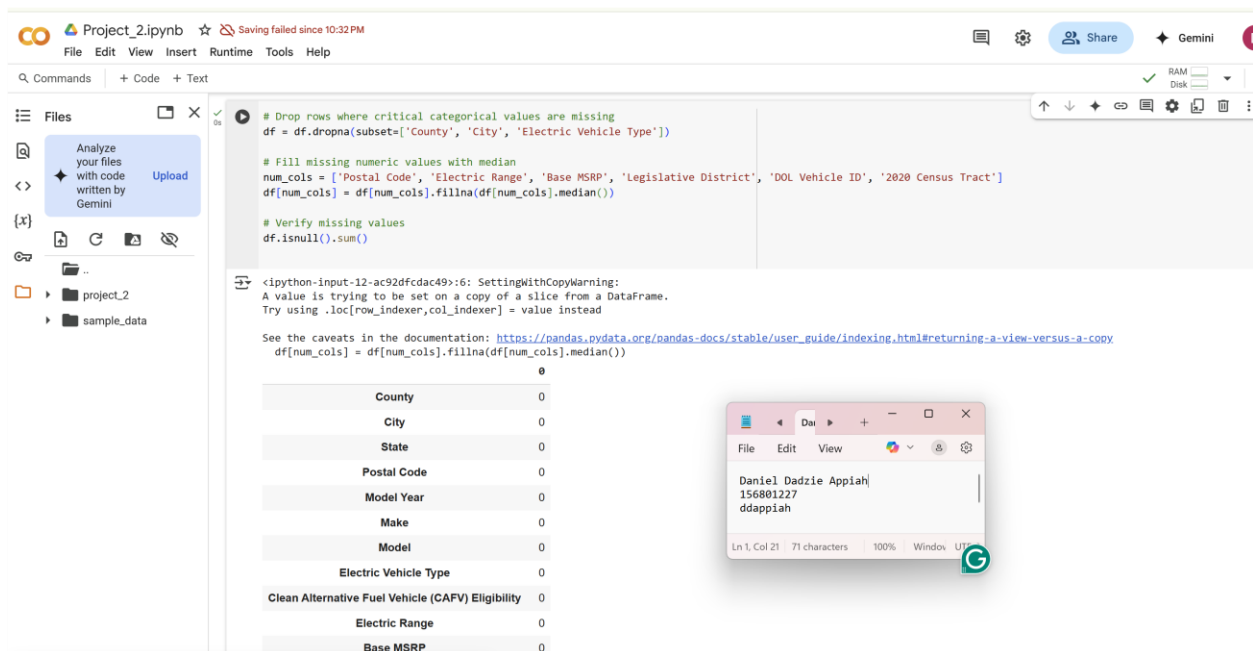
The screenshot shows a Google Colab notebook titled "Project\_2.ipynb". The code cell contains `df.info()`, which outputs the following information:

```
<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 232890 non-null object
9   Electric Range                           232854 non-null float64
10  Base MSRP                               232854 non-null float64
11  Legislative District                    232438 non-null float64
12  DOL Vehicle ID                         232890 non-null float64
13  Electric Utility                        232877 non-null object
14  2020 Census Tract                      232887 non-null float64
dtypes: float64(6), int64(1), object(8)
memory usage: 26.7+ MB
```

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

	County	City	State	Postal Code	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID	Electric Utility	2020 Census Tract
0	Daniel Dadzie Appiah	156801227	ddappiah												

## 9 Step 3: Drop Rows with Missing Values



The screenshot shows a Google Colab notebook titled "Project\_2.ipynb". The code cell contains the following code:

```
# 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 output shows the result of the `df.isnull().sum()` operation:

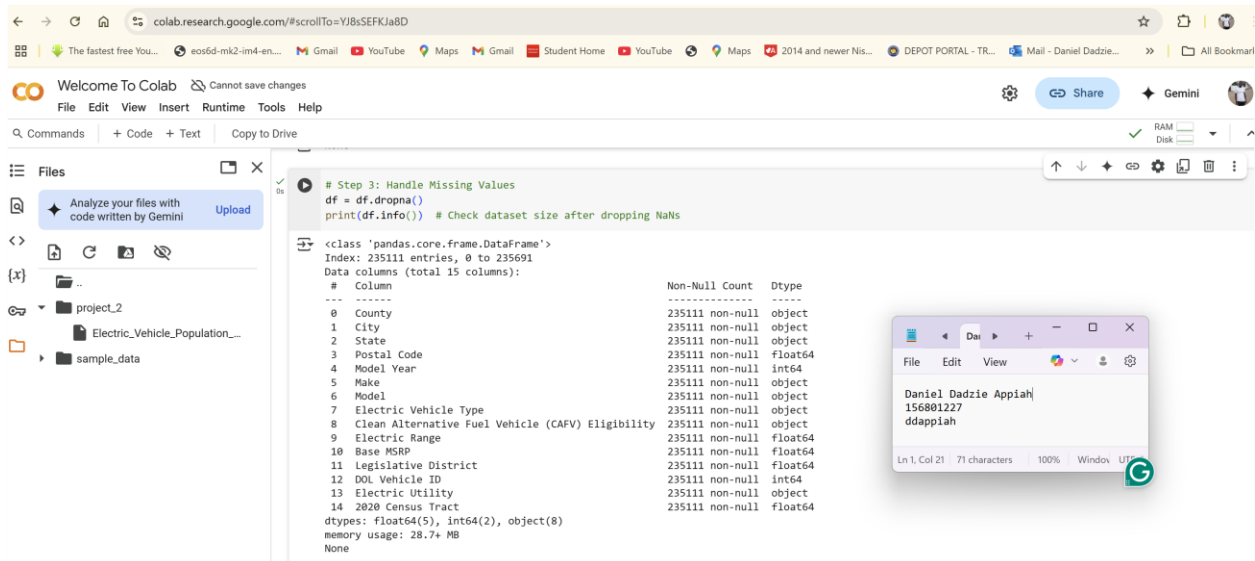
	County	City	State	Postal Code	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP
0	0	0	0	0	0	0	0	0	0	0	0

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

	County	City	State	Postal Code	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP
0	Daniel Dadzie Appiah	156801227	ddappiah								



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



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

<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 (CAFV) 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
```

### 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.

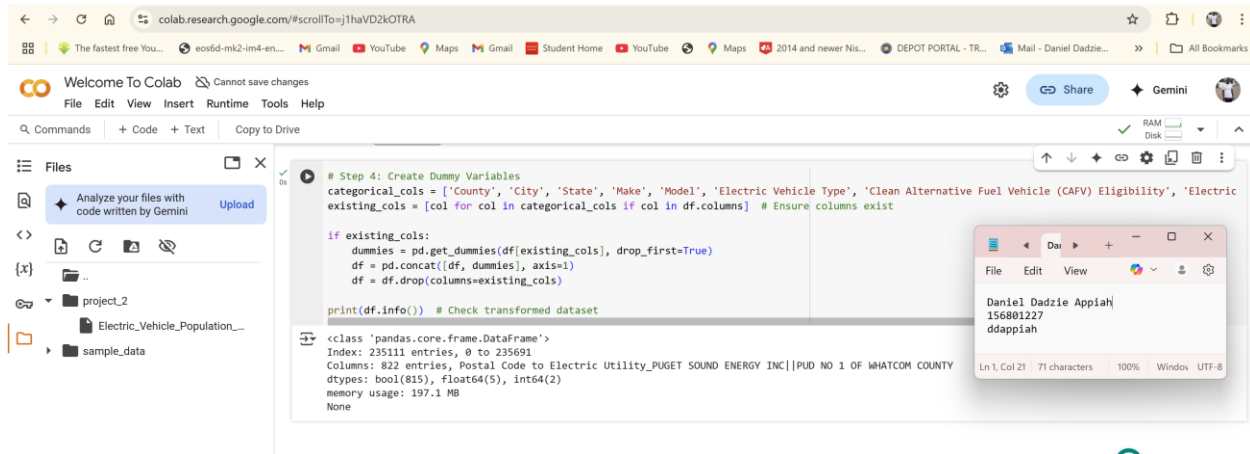
## 11 Step 4: Create Dummy Variables and Check Transformed Dataset



```
# 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

<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
```

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



```
# Step 4: Create Dummy Variables
categorical_cols = ['County', 'City', 'State', 'Make', 'Model', 'Electric Vehicle Type', 'Clean Alternative Fuel Vehicle (CAFE) Eligibility', 'Electric Vehicle Type']
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

<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
```

### 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

```
[9]
# 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("\nNumPy X shape:", X.shape)
print("\nNumPy 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\_Asotin County\_Benton

	DOL Vehicle ID	2020 Census Tract	County_Asotin	County_Benton
0	477309682	5.303301e+10	False	False
1	109705683	5.303509e+10	False	False
2	230398492	5.303509e+10	False	False
3	267929112	5.303509e+10	False	False
4	236505139	5.306701e+10	False	False

County\_Chelan ... Electric Utility\_PORTLAND GENERAL ELECTRIC CO

	County_Chelan	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 CHELAN COUNTY

	Electric Utility_PUD NO 1 OF CHELAN COUNTY
0	False
1	False
2	False
3	False
4	False

```
# Show effect of conversion to NumPy
print("\nDataFrame Head:\n", df.head())
print("\nNumPy X shape:", X.shape)
print("\nNumPy 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])
```

Electric Utility\_PUD NO 1 OF CHELAN COUNTY

	Electric Utility_PUD NO 1 OF CHELAN COUNTY
0	False
1	False
2	False
3	False
4	False

Electric Utility\_PUD NO 1 OF DOUGLAS COUNTY

	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

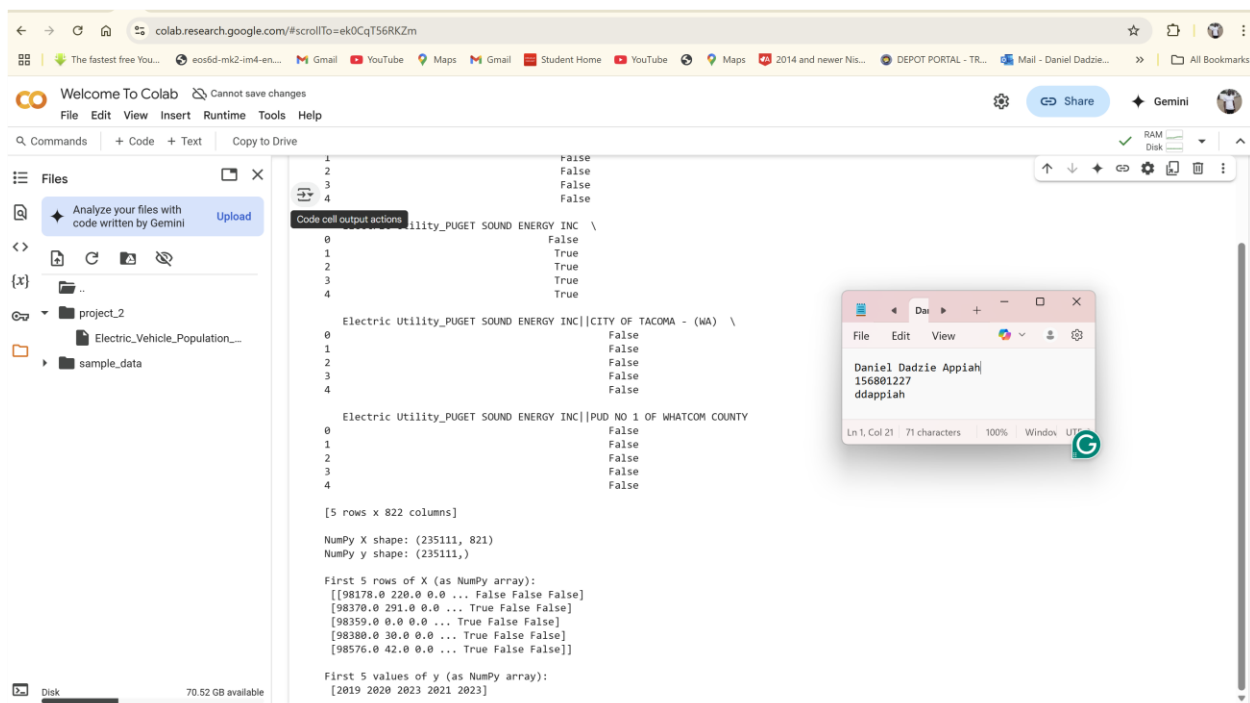
	Electric Utility_PUD NO 1 OF OKANOGAN COUNTY
0	False
1	False
2	False
3	False
4	False

Electric Utility\_PUD NO 1 OF PEND OREILLE COUNTY

	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

	Electric Utility_PUD NO 1 OF WHATCOM COUNTY
0	False
1	False
2	False



## 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:

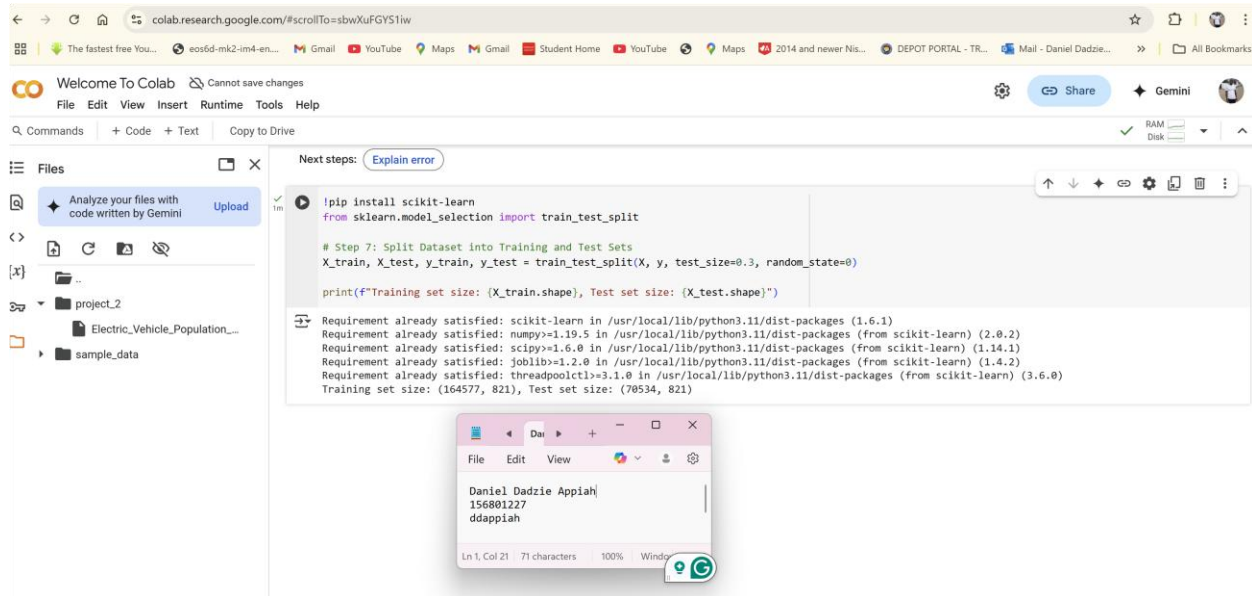
The first 5 rows of x show numerical and categorical (dummy) variables converted to True/False values. The first 5 values of y confirm that the Model Year is correctly extracted.

### 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



### 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.