SML312 Final Project Generative AI Citation

Below are citations from AI used for the SML312 Final Project.
The LLM used was ChatGPT. All prompts and full answers are listed below.

ChatGPT was utilized for examples of code in a similar manner to how websites such as StackOverflow would be used.

#AI CITATION_1

PROMPT: how to upload files to python colab and combine/clean from there

ChatGPT said:

To **upload files to Google Colab** and then **combine and clean them**, follow this step-by-step guide.

✅ 1. Upload Files to Colab

In a Colab notebook cell, run this code to upload files from your local computer:

```
from google.colab import files
uploaded = files.upload()
```

- This will open a file picker. You can select multiple files (e.g., CSVs).
- `uploaded` will be a dictionary of filenames and file content.

> 🔍 You can also mount Google Drive if your files are stored there (see bonus tip below).

✅ 2. Load Uploaded Files into Pandas DataFrames

If you're uploading multiple CSV files, you can load them like this:

```
import pandas as pd
```

```
# For a single file (example)
df = pd.read_csv('your_file.csv')

# Or for multiple files
dataframes = {}
for filename in uploaded.keys():
    df = pd.read_csv(filename)
    dataframes[filename] = df
```

Now `dataframes` is a dictionary where each key is a filename, and each value is a DataFrame.

✅ 3. Combine DataFrames

If they all have the same structure (i.e., same columns), you can concatenate them:

```
combined_df = pd.concat(dataframes.values(), ignore_index=True)
```

✅ 4. Clean Your Data

Here are some common cleaning steps:

Drop duplicates:

```
combined_df.drop_duplicates(inplace=True)
```

Handle missing values:

```
# Drop rows with any missing values
combined_df.dropna(inplace=True)

# Or fill missing values
combined_df.fillna('Unknown', inplace=True)
```

Convert columns to appropriate data types:

```
combined_df['date_column'] =
pd.to_datetime(combined_df['date_column'])
```

```
combined_df['numeric_column'] =
pd.to_numeric(combined_df['numeric_column'], errors='coerce')
```

Rename columns:

```
combined_df.rename(columns={'OldName': 'NewName'}, inplace=True)
```

✅ 5. Save and Download Cleaned Data (Optional)

```
combined_df.to_csv('cleaned_data.csv', index=False)
files.download('cleaned_data.csv')
```

🔁 Bonus: Upload from Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Then use file paths like:

```
df = pd.read_csv('/content/drive/MyDrive/your_folder/your_file.csv')
```

If you want, you can upload a few sample files and I'll help you combine/clean them right here.

#AI CITATION_2

ChatGPT said:

To **load files from Google Drive into Google Colab** (step 2) and use them with pandas, here's exactly what to do:

✅ Step-by-Step: Load CSVs from Google Drive into Pandas

🔹 Step 1: Mount Google Drive

Run this code in a Colab cell:

```
from google.colab import drive
drive.mount('/content/drive')
```

- This will prompt you to authorize Colab to access your Google Drive.
- After authorization, your Drive will be mounted under `/content/drive/MyDrive/`.

🔹 Step 2: List or Load Files from Drive

👉 Option A: If you know the exact file path(s)

Use `pandas.read_csv()` with the full path:

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/your_folder/your_file.csv')
```

You can repeat this for multiple files and then combine them.

👉 **Option B: If you want to load multiple files from a folder**

Use `glob` to grab all `.csv` files in a folder:

```
import pandas as pd
import glob

# Change this to the folder where your files are
folder_path = '/content/drive/MyDrive/your_folder/*.csv'

# List all CSV files
file_list = glob.glob(folder_path)

# Load and combine them
dfs = [pd.read_csv(file) for file in file_list]
combined_df = pd.concat(dfs, ignore_index=True)
```

✅ Example

Let’s say you have this structure in Google Drive:

```
MyDrive/
├── data/
│   ├── file1.csv
│   ├── file2.csv
```

Then your code would be:

```
import pandas as pd
import glob

folder_path = '/content/drive/MyDrive/data/*.csv'
file_list = glob.glob(folder_path)
```

```
dfs = [pd.read_csv(file) for file in file_list]
combined_df = pd.concat(dfs, ignore_index=True)
```

🧼 From here, you can clean `combined_df` as needed:

- `dropna()`
- `drop_duplicates()`
- `rename()`
- `astype()` / `to_datetime()`
- etc.

If you want, you can paste the exact folder path or filenames from your Drive, and I'll tailor the code for your specific setup.

#AI CITATION_3

PROMPT:
I have a column (dictionary of strings) that I want to separate into two new columns containing the relevant strings. can you give me an example of code?

Sure! Here's a clean, practical example in **pandas** showing how to split a column that contains **dictionary-like strings** into two new columns.

✅ Example: Split a column of dictionary strings into separate columns

📌 Your data might look like this:

```
import pandas as pd
```

```
df = pd.DataFrame({
    "info": [
        "{'first': 'Alice', 'last': 'Smith'}",
        "{'first': 'Bob', 'last': 'Jones'}"
    ]
})
```

This `info` column contains **strings that look like dictionaries**, not actual dict objects yet.

✅ Step 1 — Convert the string to a real dictionary

Use `ast.literal_eval` for safe parsing:

```
import ast

df["info_dict"] = df["info"].apply(ast.literal_eval)
```

✅ Step 2 — Expand the dict into new columns

```
df[["first", "last"]] = df["info_dict"].apply(pd.Series)
```

✅ Step 3 — Drop the temporary dict column (optional)

```
df = df.drop(columns=["info_dict"])
```

✅ Final DataFrame:

```
print(df)
```

Output:

```
                        info  first   last
0  {'first': 'Alice', 'last': 'Smith'} Alice   Smith
1  {'first': 'Bob', 'last': 'Jones'}   Bob     Jones
```

⭐ Alternate Case: If values are already dictionaries

If your column contains actual dictionaries (not strings), you can go straight to:

```
df[["first", "last"]] = df["info"].apply(pd.Series)
```

If you want, you can paste a sample of your actual data and I'll write the exact code for your case!

#AI CITATION_4

Prompt:
how do i create a new dataframe where it pulls rows with a certain answer in a column

ChatGPT said:

To create a new DataFrame that contains only the rows where a specific column has a certain value, you can filter the original DataFrame using a boolean condition.

✅ Basic example (Pandas)

```
import pandas as pd

# Example dataframe
```

```
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', 'Alice'],
    'Score': [85, 90, 78, 92]
})

# Filter rows where Name == 'Alice'
new_df = df[df['Name'] == 'Alice']

print(new_df)
```

✅ If you want to match multiple possible values

```
new_df = df[df['Name'].isin(['Alice', 'Bob'])]
```

✅ If you want rows where the column *does not* equal a value

```
new_df = df[df['Name'] != 'Alice']
```

✅ If the column contains text and you want partial matches

```
new_df = df[df['Name'].str.contains("Al", case=False, na=False)]
```

✅ If the condition is numeric

```
new_df = df[df['Score'] > 80]
```

#AI CITATION_5

Prompt:
how to convert na to 0 integers pandas

ChatGPT said:

To convert NaN values to **0** and make the column (or whole DataFrame) **integers**, you can do it in two steps:

✅ Option 1 — Replace NaN with 0, then convert to int

```
df = df.fillna(0).astype(int)
```

For a single column

```
df['col'] = df['col'].fillna(0).astype(int)
```

✅ Option 2 — Using Int64 nullable integer dtype

This is useful if you still want to allow missing values later.

```
df['col'] = df['col'].fillna(0).astype('Int64')
```

✅ If NaN came from calculations (floats), convert safely

```
df['col'] = df['col'].fillna(0).round().astype(int)
```

✅ If your DataFrame has mixed types and you want to convert only numeric cols

```
df = df.fillna(0)
df = df.apply(pd.to_numeric, errors='ignore').astype(int)
```

✅ Quick example

```
import pandas as pd
import numpy as np

df = pd.DataFrame({
    'A': [1, np.nan, 3],
    'B': [np.nan, 5, 6]
})

df = df.fillna(0).astype(int)
print(df)
```

Output

```
  A  B
0  1  0
1  0  5
2  3  6
```

If you'd like, tell me:

✅ column names
✅ whether all NaNs should become 0
✅ whether you need standard int or nullable Int64

…and I’ll tailor the exact code for your dataset.

#AI CITATION_6

PROMPT:
how do i convert coefficients into odds ratios in python?

ChatGPT said:

Converting logistic regression coefficients into **odds ratios** in Python is simple:
➡️ **Take the exponential of each coefficient.**

Here’s the concise explanation and example code.

✅ General Rule

If β is a logistic regression coefficient:

odds ratio=eβ\text{odds ratio} = e^{\beta}odds ratio=eβ

In Python:

```
import numpy as np
odds_ratio = np.exp(beta)
```

✅ Example: Using statsmodels

```
import statsmodels.api as sm
import numpy as np

model = sm.Logit(y, X).fit()
coefficients = model.params

odds_ratios = np.exp(coefficients)
print(odds_ratios)
```

✅ Example: Using scikit-learn

```
from sklearn.linear_model import LogisticRegression
import numpy as np

model = LogisticRegression().fit(X, y)

coefficients = model.coef_[0]     # log-odds
odds_ratios = np.exp(coefficients)
```

```
print(odds_ratros)
```

✅ With feature names

```
import pandas as pd

odds_ratio_df = pd.DataFrame({
    'feature': X.columns,
    'coefficient': coefficients,
    'odds_ratio': np.exp(coefficients)
})

print(odds_ratio_df)
```

📌 Interpretation

- **OR > 1** → predictor increases odds
- **OR < 1** → predictor decreases odds
- **OR = 1** → no effect

If you want, I can also show you how to:

- Get **odds ratio confidence intervals**
- Handle **standardized coefficients**
- Calculate ORs for categorical variables

Just tell me your next step!