

VENKATESH SOUNDARARAJAN

+1-(825) 962-9211 ● Calgary, AB ● venkatesh.balusoundar@gmail.com ● ca.linkedin.com/in/venkateshbalus
Portfolio: <https://venkateshbalu.streamlit.app/>

PROFILE

Business Analyst and Data Quality Engineer with over eight years of experience in the insurance industry, specializing in Guidewire Insurance Suite implementations and functional analysis across PolicyCenter, BillingCenter, and ClaimCenter modules. Proven ability to gather and translate complex business requirements into effective technical solutions, supporting process improvements and system enhancements. Currently advancing expertise in data science by completing a Master's degree in Data Science and Analytics at the University of Calgary, focusing on data quality, statistical modeling, and machine learning. Committed to leveraging a strong foundation in insurance domain knowledge and emerging data analytics skills to drive innovation and deliver value-added Guidewire business solutions.

CORE SKILLS

Functional Skills:

- Quality Assurance: Test Automation, Functional & Integration Testing, Defect Management
- Business Analysis & Requirements Management
- Guidewire Insurance Suite

Programming & Data Analysis:

- Java (Selenium WebDriver), Python, R, SQL
- Data Analysis Libraries: Pandas, NumPy, Matplotlib
- Machine Learning: Scikit-Learn

Data Visualization & Reporting:

- Tableau, Power BI, Excel, Streamlit Dashboard Development

Cloud & DevOps Tools:

- AWS (EC2, S3, DynamoDB)
- Version Control: Git, GitHub
- Agile Project Management: JIRA, Rally

EDUCATION

Masters in Data Science and Analytics

September 2024 – Present

University of Calgary, Alberta, Canada

Bachelor of Engineering

August 2009 – May-2013

Anna University, Chennai, India

EMPLOYMENT EXPERIENCE

Senior Consultant

October 2021 - August 2024

Deloitte Consulting India Private Limited, India

- Worked as a Functional Analyst for Personal Lines insurance projects, gathering and documenting business requirements with SMEs and product owners, translating them into detailed functional specs and JIRA user stories.
- Participated in 3 Amigos sessions (QA, BA, Dev) for new enhancements, helping define clear acceptance criteria and reduce requirement ambiguities.
- Supported client demos by walkthrough of implemented functionalities, enhancing client understanding and engagement.
- Executed integration and configuration functionalities aligned with evolving requirements, contributing to a 95% defect removal rate before release.
- Developed an automated status dashboard framework, reducing manual effort by 50% and improving project delivery efficiency.
- Prepared and shared Daily Status Reports with stakeholders to ensure clear communication of project progress and issue tracking
- Gained hands-on experience with AWS cloud services including EC2, DynamoDB, and S3, supporting cloud-based project components and enhancing technical proficiency.

Consultant**May 2018 - October 2021****Capgemini Technology Services India Private Limited, India**

- Conducted end-to-end testing of Guidewire-based Worker Compensation policies, validating Operational DataStore (ODS) data accuracy through ETL mapping verification from Customer Portal to Data Warehouse.
- Transitioned from QA to a hybrid BA/QA role, balancing responsibilities in requirements gathering, analysis, and test automation for insurance software projects.
- Created an automation tool that sent real-time failure alerts, improving response times to critical issues.
- Actively participated in agile forums within a team of eight members, reducing the sprint carryover rate by 15% and helping the team consistently meet sprint goals.
- Provided mentorship and leadership to the data QA team, fostering skill development and ensuring high standards of productivity and quality.

Associate**September 2013 - May 2018****Cognizant Technology Solutions India Private Limited, India**

- Joined as a fresher in the Mainframe Testing batch, specializing in DB2 database and batch processing testing within healthcare IT projects.
- Managed and tracked key project metrics for healthcare initiatives valued at over \$2M across all phases of the Software Testing Life Cycle (STLC), contributing to a 10% increase in process efficiency.
- Led test environment management for continuous testing across three healthcare projects, driving automation and reducing test environment setup time by 40%.
- Delivered 10+ knowledge transfer sessions to ensure smooth handover and retention of critical healthcare domain knowledge among team members and stakeholders.
- Supported healthcare production remediation processes including ad hoc data extraction, reporting, data patching, and document re-triggering to resolve critical live system issues.

PROFESSIONAL DEVELOPMENT

-
- Member of Data Science Club - University of Calgary -2024.
 - Member of Test Tribe - Calgary Chapter- 2024.

CERTIFICATIONS

-
- Insurance and Guidewire Suite Analyst 10.0 – Jasper –Guidewire Education -2024.
 - Karate DSL – Udemy -2023.
 - Rest API Automation - Test Leaf Software Solutions Private Limited – 2023.
 - Selenium WebDriver - Test Leaf Software Solutions Private Limited – 2022.
 - SQL for Data Science - Coursera -2020.
 - SDET- Capgemini – 2020.

ACCOMPLISHMENTS

-
- Awarded as “Spot Award” 2022 & 2023 for Insurcloud – Deloitte, Canada.
 - Awarded as “Best Contributor” 2018 for COMPASS Program – Hartford Insurance, USA.
 - Awarded as “QE & A Maestro” 2017 of Centene by Cognizant QE&A, USA.
 - Awarded as “Pride of the Quarter Q1”-April 2017 of Health Net by Cognizant QE&A, USA.
 - Awarded as the “Pillar of the Month “May2014” & “August-2015” of Health Net by Cognizant QE&A, USA.

INTERESTS

-
- Event Coordinator and active in Cognizant Outreach, India.
 - Volunteered in Cognizant Go Green Activities- Beach and Lake Cleaning, India.

CanadianQualityofLifeAnalysis

July 13, 2025

0.1 Introduction

Quality of life (QoL) refers to an individual's perception of their position in life, covering physical health, mental health, and social relationships, relative to their expectations and cultural context (World Health Organization, 1995). In health research, QoL is affected by many factors, including health drivers and health barriers. Health drivers such as physical activity and mental health consultations typically have a positive influence on QoL, while health barriers like smoking, alcohol consumption, and cannabis use are often linked to negative health outcomes (Haskell et al. 2007; Rehm et al., 2009).

This project aims to explore the relationship between these health drivers and barriers, focusing on their impact on physical and mental health outcomes. Using Visualizations and correlation statistics, we will analyze variables such as body mass index (BMI), self-reported health status, and stress to evaluate the influence of health barriers. Following this, we will perform subgroup analyses to determine how age, sex, region, and education are impacted by these health barriers. Depending on our findings we will either help identify the most vulnerable populations and suggest targeted interventions to improve their QoL, or try to suggest more appropriate ways to analyze this dataset.

0.2 Dataset and Guiding Questions

We chose the Canadian Community Health Survey as our dataset. Specifically, we will be using the 2019-2020 microdata file available at Statistics Canada (Statistics Canada, 2023).

The Canadian Community Health Survey is an annual survey that collects data from respondents across Canada and differentiates them by their province and respective health regions. The survey is voluntary response but is designed to have a large representative sample, all identifying data is removed prior to the release of the microdata file. The survey collects data on a wide variety of health indicators, potential health determinants, and plenty of demographic data to assist in analysis. The datafile comes in the form of a CSV file and comes with a data dictionary and a very accessible format given the sheer size of the CSV file.

The dataset has 691 individual columns including the respondent's record number with the columns representing the variables. For each of these 691 variables there are 108,252 individual responses as the rows for the data file. We plan to choose a specific number of variables and analyze them according to the questions listed below.

Our guiding questions can be listed into two categories:

- a. Health Drivers Vs. Health Barriers
 1. How does alcohol consumption affect mental health among different age groups?

2. How does cannabis use impact stress levels?
 3. What is the relationship between smoking and physical health outcomes?
 4. Which demographic groups (age, sex, region) are most affected by health barriers?
- b. Health Drivers Vs. Health Improvements
1. What is the relationship between regular exercise and self-reported health status?
 2. How does stress influence the maintenance or improvement of mental health?

```
[4]: #Make sure all appropriate libraries loaded
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
import plotly.express as px
from pydataset import data
import warnings
warnings.filterwarnings('ignore')
from pydataset import data
```

```
[7]: #Load dataset attached to submission
df=pd.read_csv('pumf_cchs.csv')
```

```
[15]: #This should filter all of the variables we want on the general level. You will
      ↵need to determine which variables you will use
#IMPORTANT I did not include all the CCC values for chronic disease here. I can
      ↵look and see what we need, and try to adjust shortly
#If you think anything is missing here, feel free to change
df_general=df.filter(items=[
    'ALC_015',
    'ALC_020',
    'GEN_010',
    'GEN_015',
    'GEN_020',
    'GEN_025',
    'CCC_195',
    'GEN_005',
    'HWTGISW',
    'CAN_015',
    'SMK_005',
    'SMK_060',
    'PAADVACV',
    'PAA_030',
    'PAA_060',
    'PAA_095',
    'SBE_005',
    'SBE_010',
```

```

'PAA_005',
'DHHGAGE',
'DHH_SEX',
'GEOGPRV',
'EHG2DVH3',
'HWT_050',
'PEX_005',
'ADM_RN01',
'GENDVHDI'
])
print(df_general.head())

```

	ALC_015	ALC_020	GEN_010	GEN_015	GEN_020	GEN_025	CCC_195	GEN_005	\
0	5.0	3.0	9.0	3.0	2.0	2.0	2.0	3.0	
1	1.0	1.0	4.0	3.0	3.0	6.0	1.0	3.0	
2	96.0	96.0	7.0	3.0	3.0	6.0	2.0	2.0	
3	96.0	96.0	8.0	3.0	3.0	6.0	2.0	3.0	
4	96.0	96.0	0.0	5.0	4.0	6.0	2.0	5.0	

	HWTDGISW	CAN_015	...	SBE_010	PAA_005	DHHGAGE	DHH_SEX	GEOGPRV	\
0	1.0	2.0	...	6.0	2.0	3.0	2.0	47.0	
1	2.0	2.0	...	6.0	2.0	5.0	1.0	47.0	
2	2.0	2.0	...	1.0	6.0	5.0	2.0	59.0	
3	2.0	2.0	...	6.0	6.0	5.0	1.0	13.0	
4	2.0	2.0	...	6.0	6.0	4.0	1.0	46.0	

	EHG2DVH3	HWT_050	PEX_005	ADM_RN01	GENDVHDI
0	3.0	3.0	96.0	1000	2.0
1	2.0	1.0	96.0	100005	2.0
2	1.0	1.0	96.0	100012	3.0
3	1.0	1.0	96.0	100015	2.0
4	3.0	3.0	96.0	100018	0.0

[5 rows x 27 columns]

0.3 Analysis

0.3.1 Health Drivers Vs. Health Barriers

Question 1 - How does alcohol consumption affect mental health among different age groups? To determine how Alcohol Consumption Affected Mental Health Among Different Age Groups, the data was first cleaned. Due to the size of the initial dataset, any data columns containing unnecessary variables were removed. Following this, it was made sure there were no missing values, and removed any values that were of no interest to my analysis, such as ‘valid skip’, do not know, and refuse to say. As they were not pertinent to the analysis, we felt it was safe to remove them.

```
[ ]: #Make a copy of our initial filtered data, and then filter for our question
df3=df_general.copy()
df_q3=df3.
    ↪filter(items=['ALC_015','ALC_020','GEN_010','GEN_015','GEN_020','GEN_025','CCC_195','DHHGAG'])

[ ]: #Some basic info on the dataset
print(df_q3.shape)
print(df_q3.info())

[ ]: #Ensure no null values
print(df_q3.isnull().sum())

[ ]: #Removed invalid values that reflected responses like unknown, did not answer, or valid skip
df_q3=df_q3[~df_q3['GEN_010'].isin(range(97,100))]
df_q3=df_q3[~df_q3['GEN_015'].isin(range(7,10))]
df_q3=df_q3[~df_q3['GEN_020'].isin(range(7,9))]
df_q3=df_q3[~df_q3['GEN_025'].isin(range(6,10))]
df_q3=df_q3[~df_q3['ALC_015'].isin(range(96,100))]
df_q3=df_q3[~df_q3['ALC_020'].isin(range(96,100))]
df_q3=df_q3[~df_q3['CCC_195'].isin(range(7,9))]
```

Upon inspection of the data dictionary, it was discovered that some likret scales went in the direction of negative to positive, while other went from positive to negative. To avoid confusion, the variable order was changed so that all progressed in a single logical direction.

```
[ ]: #Convert GEN_015,_020,_025 so they progress in a logical order with 1=more negative and 5=more positive
gen_015_new_order={1:5,2:4,3:3,4:2,5:1}
gen_020_new_order={1:5,2:4,3:3,4:2,5:1}
gen_025_new_order={1:5,2:4,3:3,4:2,5:1}

df_q3['GEN_015'] = df_q3['GEN_015'].map(gen_015_new_order)
df_q3['GEN_020'] = df_q3['GEN_020'].map(gen_015_new_order)
df_q3['GEN_025'] = df_q3['GEN_025'].map(gen_015_new_order)
```

```
[ ]: #Create labels that represent the values listed in the data dictionary
perceived_mental_health={1: 'Poor', 2: 'Fair', 3: 'Good', 4: 'Very good', 5:'Excellent'}
perceived_life_stress= {1: 'Extremely stressful',2: 'Quite a bit stressful',3:'A bit stressful',4: 'Not very stressful',5: 'Not at all stressful'}
perceived_work_stress={1: 'Extremely stressful',2: 'Quite a bit stressful',3:'A bit stressful',4: 'Not very stressful',5: 'Not at all stressful'}
alc_freq={
    1: 'Less than once/month',2: 'Once/month',3: '2-3 times/month',
    4: 'Once/week',5: '2-3 times/week',6: '4-6 times/week',7: 'Every day'}
```

```

alc_binge_freq= {1: 'Never',2: 'Less than once/month',3: 'Once/month',4: '2-3 times/month',
                 5: 'Once/week',6: 'More than once/week',}
age_group={1:"12 to 17 years",2:"18 to 34 years",3:"35 to 49 years",4:"50 to 64 years",5:"65 and older"}

```

```

[ ]: #Create new columns that list the previously created labels
df_q3["perceived_mental_health"] = df_q3[["GEN_015"]].copy().replace({"GEN_015": "perceived_mental_health"})
df_q3["perceived_life_stress"] = df_q3[["GEN_020"]].copy().replace({"GEN_020": "perceived_life_stress"})
df_q3["perceived_work_stress"] = df_q3[["GEN_025"]].copy().replace({"GEN_025": "perceived_work_stress"})
df_q3["alc_freq"] = df_q3[["ALC_015"]].copy().replace({"ALC_015": alc_freq})
df_q3["alc_binge_freq"] = df_q3[["ALC_020"]].copy().replace({"ALC_020": "alc_binge_freq"})
df_q3["age_group"] = df_q3[["DHHGAGE"]].copy().replace({"DHHGAGE": age_group})

```

```

[ ]: #Make a copy of the dataset, so if we make a mistake we don't need to start from the beginning
df_q3a=df_q3.copy()

```

```

[ ]: #Create a correlation amongst our variables of interest, and then plot them in a heatmap
labels = {
    'ALC_015': 'Alcohol Use Frequency',
    'ALC_020': 'Binge Drinking Frequency',
    'GEN_010': 'Life Satisfaction',
    'GEN_015': 'Self-Perceived Mental Health',
    'GEN_020': 'Perceived Life Stress',
    'CCC_195': 'Mood Disorders',
    'DHHGAGE': 'Age Group'
}
correlations = df_q3a[['ALC_015', 'ALC_020', 'GEN_010', 'GEN_015', 'GEN_020', 'CCC_195', 'DHHGAGE']].corr()
correlations.rename(columns=labels, index=labels, inplace=True)
print("Correlation Matrix:")
print(correlations)

#correlation heatmap
sns.heatmap(correlations, annot=True, cmap='coolwarm', center=0)
plt.title("Correlation Between Alcohol Use and Mental Health Variables")

plt.show()

```

Based on our correlation of all our variables of interest, we most see weak relationships, with a few moderate relationships, none of which were very surprising. The strongest relationships were seen

between alcohol consumption frequency and binge drinking frequency, and between life satisfaction and self-perceived mental health. We also see a weak correlation between any alcohol consumption variables and mental health variables.

```
[ ]: #Create multiple correlations based on age group
def correlation_by_age_group(df_q3a, age_groups):
    subset = df[df['DHHGAGE'] == age_groups]
    correlation_matrix = subset[['ALC_015', 'ALC_020', 'GEN_010', 'GEN_015', 'GEN_020', 'CCC_195']].corr()
    return correlation_matrix

correlation_age_2 = correlation_by_age_group(df_q3a, 2)
correlation_age_3 = correlation_by_age_group(df_q3a, 3)
correlation_age_4 = correlation_by_age_group(df_q3a, 4)
print(correlation_age_2, '\n')
print(correlation_age_3, '\n')
print(correlation_age_4, '\n')
```

```
[ ]: #Plot the different heatmap for each correlation based on age group
```

```
labels = {
    'ALC_015': 'Alcohol Use Frequency',
    'ALC_020': 'Binge Drinking Frequency',
    'GEN_010': 'Life Satisfaction',
    'GEN_015': 'Self-Perceived Mental Health',
    'GEN_020': 'Perceived Life Stress',
    'CCC_195': 'Mood Disorders',
    'DHHGAGE': 'Age Group'
}
```

```
correlation_age_2 = correlation_by_age_group(df_q3a, 2)
correlation_age_3 = correlation_by_age_group(df_q3a, 3)
correlation_age_4 = correlation_by_age_group(df_q3a, 4)

correlation_age_2.rename(columns=labels, index=labels, inplace=True)
correlation_age_3.rename(columns=labels, index=labels, inplace=True)
correlation_age_4.rename(columns=labels, index=labels, inplace=True)

#heatmaps for the 3 age groups
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

sns.heatmap(correlation_age_2, annot=True, cmap='coolwarm', center=0, ax=axes[0, 0])
axes[0, 0].set_title('Correlation Heatmap for Age Group 18-34')

sns.heatmap(correlation_age_3, annot=True, cmap='coolwarm', center=0, ax=axes[0, 1])
```

```

axes[0, 1].set_title('Correlation Heatmap for Age Group 35-49')

sns.heatmap(correlation_age_4, annot=True, cmap='coolwarm', center=0, □
    ↪ax=axes[1, 0])
axes[1, 0].set_title('Correlation Heatmap for Age Group 50-64')

fig.delaxes(axes[1, 1]) #rmv empty plot

plt.tight_layout()
plt.show()

```

From the above output, we can see the correlations of our variables of interest, segmented by age categories. By segmenting by age categories we are able to highlight how differing age groups experience the relationships between alcohol use and mental health differently. A few things to note are:

- The relationship between alcohol consumption frequency and binge drinking frequency is stronger for all age groups.
- The relationship between self-perceived mental health and life satisfaction is highest in the 18-34 age group.
- The relationship between self-perceived mental health and mood disorders has a negative correlation for all age groups, suggesting that individuals with mood disorders are will more frequently have lower self-perceived mental health
- The relationship between perceived life stress and mood disorders has a negative correlation for all age groups, suggesting that those with mood disorders are more likely to have higher perceived life stress

Despite this, similar to our previous correlation with all variables, we continue to see a weak correlation between alcohol consumption variables and mental health variables, indicating that there are additional variables that play a large role in affecting mental health.

```
[ ]: #Look at the mean value for the various age groups for each variable of interest
age_group_means = df_q3a.groupby('age_group')[['ALC_015', 'ALC_020', 'GEN_010', □
    ↪'GEN_015', 'GEN_020', 'GEN_025', 'CCC_195']].mean()

print(age_group_means)
```

```
[ ]: #Creates a dataframe to give the proportion of GEN_015, grouped by ALC_015
avg_counts_alc_freq=df_q3a.groupby('ALC_015', as_index=False)[['GEN_015']].
    ↪value_counts(normalize=True, sort=False)
avg_counts_alc_freq
```

```
[ ]: #Plotting alcohol consumption freq and perceived mental health proportions
#change the color palette
cmap = plt.get_cmap('coolwarm')
colors = cmap(np.linspace(0, 1, 5))

plt.figure().set_figwidth(16)
```

```

plt.bar(avg_counts_alc_freq['ALC_015'].unique()-0.2,
        avg_counts_alc_freq['proportion'][::5], width=0.1, label='Poor',
        color=colors[0])
plt.bar(avg_counts_alc_freq['ALC_015'].unique()-0.1,
        avg_counts_alc_freq['proportion'][1::5], width=0.1, label='Fair',
        color=colors[1])
plt.bar(avg_counts_alc_freq['ALC_015'].unique(),
        avg_counts_alc_freq['proportion'][2::5], width=0.1, label='Good',
        color=colors[2])
plt.bar(avg_counts_alc_freq['ALC_015'].unique()+0.1,
        avg_counts_alc_freq['proportion'][3::5], width=0.1, label='Very good',
        color=colors[3])
plt.bar(avg_counts_alc_freq['ALC_015'].unique()+0.2,
        avg_counts_alc_freq['proportion'][4::5], width=0.1, label='Excellent',
        color=colors[4])
plt.xticks(avg_counts_alc_freq['ALC_015'].unique()+0.1/2, ('Less than once/month',
    'Once/month', '2-3 times/month', 'Once/week', '2-3 times/week',
    '4-6 times/week', 'Every day'))
plt.xlabel('Perceived Mental Health Level by Alcohol Consumption Frequency')
plt.ylabel('Proportion')
plt.legend()
plt.show()

```

Based on output above, we can see that as alcohol consumption increase from less than once per month to 2-3 times per week, the proportion of individuals self perceived mental health improves slightly, with a larger proportion reporting excellent perceived mental health when consuming alcohol 2-3 times per week. However for individuals that drink 4-6 times per week, or every day, the proportion that report excellent mental health drops. This may indicate that those drinking more frequently may temporarily increase their perceived mental health, but increasing the alcohol consumption frequency by too much will lead to worse self perceived mental health.

```
[ ]: #Creates a dataframe to give the proportion of GEN_015, grouped by ALC_020
avg_counts_binge_freq=df_q3a.groupby('ALC_020', as_index=False)[['GEN_015']].
    value_counts(normalize=True, sort=False)
avg_counts_binge_freq
```

```
[ ]: #Plotting Binge freq and perceived mental health proportions
#change the color palette
cmap = plt.get_cmap('coolwarm')
colors = cmap(np.linspace(0, 1, 5))

plt.figure().set_figwidth(16)
plt.bar(avg_counts_binge_freq['ALC_020'].unique()-0.2,
        avg_counts_binge_freq['proportion'][::5], width=0.1,
        label='Poor', color=colors[0])
```

```

plt.bar(avg_counts_binge_freq['ALC_020'].unique()-0.
    ↪1,avg_counts_binge_freq['proportion'][1::5],width=0.1,✉
    ↪label='Fair',color=colors[1])
plt.bar(avg_counts_binge_freq['ALC_020'].
    ↪unique(),avg_counts_binge_freq['proportion'][2::5],width=0.1,✉
    ↪label='Good',color=colors[2])
plt.bar(avg_counts_binge_freq['ALC_020'].unique()+0.
    ↪1,avg_counts_binge_freq['proportion'][3::5],width=0.1, label='Very✉
    ↪good',color=colors[3])
plt.bar(avg_counts_binge_freq['ALC_020'].unique()+0.
    ↪2,avg_counts_binge_freq['proportion'][4::5],width=0.1,✉
    ↪label='Excellent',color=colors[4])
plt.xticks(avg_counts_binge_freq['ALC_020'].unique()+0.1/2,('Never','Less than✉
    ↪once/month','Once/month','2-3 times/month','Once/week',
                           'More than once/week'))
plt.xlabel('Perceived Mental Health Level by Binge Drinking Frequency')
plt.ylabel('Proportion')
plt.legend()
plt.show()

```

Based on output above, we can see that as binge drinking frequency increase, the proportion of individuals self perceived mental health steadily decreases, with individuals who binge drink more than once a week reporting the lowest proportions of very good and excellent mental health, while reporting the highest proportions of poor and fair mental health. This suggests a possible link between binge drinking and worse perceived mental health.

```

[ ]: #Plots average self-perceived mental health by alcohol consumption
avg_scores_num=df_q3a.groupby(
    ['ALC_015','ALC_020'])['GEN_015'].mean().reset_index()
alc_015_labels = {
    1: 'Less than once/month',
    2: 'Once/month',
    3: '2-3 times/month',
    4: 'Once/week',
    5: '2-3 times/week',
    6: '4-6 times/week',
    7: 'Every day'
}

# Mapping for ALC_020
alc_020_labels = {
    1: 'Never',
    2: 'Less than once/month',
    3: 'Once/month',
    4: '2-3 times/month',
    5: 'Once/week',
    6: 'More than once/week',
}

```

```

}

cmap = plt.get_cmap('coolwarm')
colors = cmap(np.linspace(0, 1, len(alc_020_labels)))
# Replace the numeric values with the corresponding labels
avg_scores_num['ALC_015'] = avg_scores_num['ALC_015'].map(alc_015_labels)
avg_scores_num['ALC_020'] = avg_scores_num['ALC_020'].map(alc_020_labels)

alc_020_order = list(alc_020_labels.values()) # Ensure correct order for
    ↪categories
color_map = dict(zip(alc_020_order, colors))
# Plotting
plt.figure(figsize=(12, 6))
sns.barplot(data=avg_scores_num, x='ALC_015', y='GEN_015', hue='ALC_020', □
    ↪errorbar='sd', palette=color_map)
plt.title('Average Self-Perceived Mental Health by Alcohol Consumption')
plt.xlabel('Frequency of Alcohol Use')
plt.ylabel('Average Self-Perceived Mental Health')
plt.legend(title='Binge Drinking Frequency', bbox_to_anchor=(1.05, 1), □
    ↪loc='upper left')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```

To visualize the relationship between alcohol consumption and mental health among different age group, a barchart was created to compare the average self-perceived mental health by both frequency of alcohol use and frequency of binge drinking. One of the main things noticed was the effect binge drinking had on mental health, with the avg self-perceived mental health of the most frequent binge drinkers dropping by almost 0.5 compared to those that did not

```
[ ]: #Plots average self-perceived mental health by age group and alcohol consumption
avg_scores2 = df_q3a.groupby(['DHHGAGE', 'ALC_015'])['GEN_015'].mean().
    ↪reset_index()
alc_015_labels = {
    1: 'Less than once/month',
    2: 'Once/month',
    3: '2-3 times/month',
    4: 'Once/week',
    5: '2-3 times/week',
    6: '4-6 times/week',
    7: 'Every day'
}
age_group={1:"12 to 17 years",2:"18 to 34 years",3:"35 to 49 years",4:"50 to 64
    ↪years",5:"65 and older"}
cmap = plt.get_cmap('coolwarm')
colors = cmap(np.linspace(0, 1, len(alc_015_labels)))
```

```

avg_scores2['ALC_015'] = avg_scores2['ALC_015'].map(alc_015_labels)
avg_scores2['DHHGAGE'] = avg_scores2['DHHGAGE'].map(age_group)

alc_015_order = list(alc_015_labels.values()) # Ensure correct order for
# categories
color_map = dict(zip(alc_015_order, colors))

# Create a bar plot

plt.figure(figsize=(12, 6))
sns.barplot(x='DHHGAGE', y='GEN_015', hue='ALC_015',
             data=avg_scores2, palette=color_map)
plt.title('Average Self-Perceived Mental Health by Age Group and Alcohol
Consumption')
plt.xlabel('Age Group')
plt.ylabel('Average Self-Perceived Mental Health')
plt.legend(title='Frequency of Alcohol Use', bbox_to_anchor=(1.05, 1),
           loc='upper left')
plt.show()

```

From the above figure, we can see that as frequency of alcohol consumption increase, there is a slight increase in the average perceived mental health, most notably in those 18-34 and 50-64. However, as individuals drink 4-6 times per week, or more, there is a decrease in average self perceived mental health.

```

[ ]: #Plots average self-perceived mental health by age group and binge frequency
avg_scores3 = df_q3a.groupby(['DHHGAGE', 'ALC_020'])['GEN_015'].mean().
reset_index()
alc_020_labels = {
    1: 'Never',
    2: 'Less than once/month',
    3: 'Once/month',
    4: '2-3 times/month',
    5: 'Once/week',
    6: 'More than once/week',
}
age_group={1:"12 to 17 years",2:"18 to 34 years",3:"35 to 49 years",4:"50 to 64
years",5:"65 and older"}
cmap = plt.get_cmap('coolwarm')
colors = cmap(np.linspace(0, 1, len(alc_020_labels)))

avg_scores3['ALC_020'] = avg_scores3['ALC_020'].map(alc_020_labels)
avg_scores3['DHHGAGE'] = avg_scores3['DHHGAGE'].map(age_group)

alc_020_order = list(alc_020_labels.values())
color_map = dict(zip(alc_020_order, colors))
# Create a bar plot

```

```

plt.figure(figsize=(12, 6))
sns.barplot(x='DHHGAGE', y='GEN_015', hue='ALC_020',  

             data=avg_scores3, palette=color_map)
plt.title('Average Self-Perceived Mental Health by Age Group and Frequency of  

           Binge Drinking')
plt.xlabel('Age Group')
plt.ylabel('Average Self-Perceived Mental Health')
plt.legend(title='Frequency of Binge Drinking', bbox_to_anchor=(1.05, 1),  

           loc='upper left')
plt.show()

```

Here we can see that as the frequency of binge drinking occurs, the average self perceived mental health of individuals decrease amongst all age groups.

Question 2 - How does cannabis use impact stress levels? To explore the impact of cannabis use on one's stress level, we want to extract the interested variables to this question that are cannabis use, perceived life stress and perceived work stress. Then these variables will be compared and analyzed to determine the impacts of cannabis use. From the initial survey data, values that are no interest to this question and may create problems to the analysis such as "valid skip", "not state", "don't know" and "refusal" were removed from the columns. All values from variables are converted into integer type as on the initial dataset, each survey code is represented by an integer.

```

[ ]: df_q4=df.copy() # make a copy of initial filtered data and filter for the  

      ↴variables relevant to this question.  

df_q4=df_q4.filter(items=['GEN_020','GEN_025','CAN_015'])  

print(df_q4.head())

```

```

[ ]: df_q4.to_csv('Question 4.csv',index=False) #Create a new csv for analysis for  

      ↴this guiding question.

```

```

[ ]: df_q4.info() #A peek to the dataset info for its data types and columns names.  

df_q4.describe()  

df_q4.dtypes

```

0.3.2 Data Wrangling

```
[ ]: df_q4.isnull().sum() #check for missing values
```

As we can see, after running the below code, all the values from interested variables have been converted to Int32 type.

```
[ ]: df_q4[['CAN_015','GEN_020','GEN_025']] = df_q4[['CAN_015','GEN_020','GEN_025']].  

      ↴astype(int) #Converts Cannabis Use CAN_015, GEN_020, GEN_025 into integer  

      ↴type.  

df_q4.info()
```

```
[ ]: #Removing irrelavent values from perceived life stress variable
df_q4=df_q4.drop(df_q4[df_q4['GEN_020']==7].index)
df_q4=df_q4.drop(df_q4[df_q4['GEN_020']==8].index)
```

```
[ ]: #Removing irrelavent values from perceived work stress variable
df_q4=df_q4.drop(df_q4[df_q4['GEN_025']==6].index)
df_q4=df_q4.drop(df_q4[df_q4['GEN_025']==7].index)
df_q4=df_q4.drop(df_q4[df_q4['GEN_025']==8].index)
df_q4=df_q4.drop(df_q4[df_q4['GEN_025']==9].index)
```

```
[ ]: #Removing irrelavent values from cannabis use variable
df_q4=df_q4.drop(df_q4[df_q4['CAN_015']==7].index)
df_q4=df_q4.drop(df_q4[df_q4['CAN_015']==8].index)
df_q4=df_q4.drop(df_q4[df_q4['CAN_015']==9].index)
```

After running the above code, we can see from below that the minimum value is 1 and maximum is 5. values of 6 ,7 ,8, 9 have been removed from dataset.

```
[ ]: df_q4.describe()
```

0.3.3 Data Analysis and Visualization

```
[ ]: # Group preceived life stress by cannabis use to prepare for the data analysis.
df4_stress=df_q4.groupby('GEN_020', as_index=False)[['CAN_015']].
    ↪value_counts(normalize=True,sort=False)
display(df4_stress)
```

```
[ ]: # Create a bar chart to visualize the impact of cannabis use to perceived life stress.
plt.figure(figsize=(12, 6))
plt.bar(df4_stress['GEN_020'].unique(),df4_stress['proportion'][::2],width=0.
    ↪3,label='Poeple Used Cannabis in Past 12 Months')
plt.bar(df4_stress['GEN_020'].unique()+0.3,df4_stress['proportion'][1::
    ↪2],width=0.3, label="People Didn't Use Cannabis in Past 12 Months")
plt.xticks(df4_stress['GEN_020'].unique()+0.3/2,('Not at all stressful','Not
    ↪very stressful','A bit stressful','Quite bit stressful','Extremely
    ↪stressful'))
plt.xlabel('Preceived Life Stress by Canabis Use')
plt.ylabel('Percent')
plt.legend()
plt.show()
```

Based on the visualization of the graph, we can see that people who used cannais and feeling stressful is slightly more than people who used cannabis feeling not stressful. This might show us that using cannabis can still increase one's life stress level. From the group of people who didn't use cannabis, the amount of people feeling not stressful is also more than the amount of people feeling stressful. This tells us that not using cannabis will result in a less stressful life. However,

the results can also be influenced by the factor that the number of people who didn't use cannabis is significantly more than the number of people who used cannabis in this dataset.

```
[ ]: # Group perceived work stress results by cannabis use.
df4_stress=df_q4.groupby('GEN_025', as_index=False)[['CAN_015']].
    ↪value_counts(normalize=True,sort=False)
display(df4_stress)

[ ]: # Plot a bar chart to visualize the relationship of cannabis use and perceived work stress.
plt.figure(figsize=(12, 6))
plt.bar(df4_stress['GEN_025'].unique(),df4_stress['proportion'][::2],width=0.
    ↪3,label='People Used Cannabis in Past 12 Months')
plt.bar(df4_stress['GEN_025'].unique()+0.3,df4_stress['proportion'][1::
    ↪2],width=0.3, label="People Didn't Use Cannabis in Past 12 Months")
plt.xticks(df4_stress['GEN_025'].unique()+0.3/2,('Not at all stressful','Not
    ↪very stressful','A bit stressful','Quite bit stressful','Extremely
    ↪stressful'))
plt.xlabel('Perceived Work Stress by Canabis Use')
plt.ylabel('Percent')
plt.legend()
plt.show()
```

From the visualization of this graph, we can see that the amount of people who used cannabis and feeling stressful is almost the same as the amount of people who used cannabis and feeling not stressful. This may reveal that the cannabis use does not make significant impact to one's work stress. As we can see people who didn't use cannabis can still feel stressful. We didn't see a tendency of increasing stress level in both groups. This may reveal that using cannabis does not make work much more stressful.

Question 3 - What is the relationship between smoking and physical health outcomes

Objective: When we start our analysis for this dataset, the most important barriers which we chose to focus is smoking and how it impacted the health levels. Also Smoking has complex associations with BMI and various physical health outcomes. Health effects of smoking can lead to conditions that may influence weight, such as chronic obstructive pulmonary disease (COPD), which can decrease physical activity levels.

Smoking can affect metabolism and body fat distribution, often leading to more visceral fat (fat around internal organs) compared to non-smokers, which is linked to higher health risks. While smokers may have a lower BMI, smoking itself is associated with numerous health risks, including cardiovascular disease, respiratory issues, and cancer. These risks can outweigh any benefits of having a lower BMI.

Many individuals experience weight gain after quitting smoking, as appetite increases and metabolism may normalize. This can lead to concerns about obesity, especially if individuals do not adopt healthier eating habits or increase physical activity.

Main Objective for this question is to analyze the relation between Smoking/BMI against Health level Indicators derived from the survey results. Analyze the health improvements whether Stopped Smoking and Physically Active Indicators impact and improve the health level of the people

```
[ ]: #dataset.head(5)
```

Data Cleaning and Transformation

- 1) Identifying the potential variables from the dataset and filtering those columns from the dataset and store it in separate data frame. Below are the list of variables which we focused to continue with our analysis. Focus Variable(Data Set Column) Survey Number(ADM_RNO1) Age(DHHGAGE) Sex(DHH_SEX) Perceived Health(GENDVHDI) Province Name(GEOGPRV) Smoking Level(SMK_005) BMI Level(HWTDGISW) Physical Activity Indicator(PAADVACV) Stopped Smoking Indicator(SMK_060)(SMK_060)

```
[ ]: print("Filtered out the required variables from the data set")
convdata=df[["ADM_RNO1","DHH_SEX","DHHGAGE","GENDVHDI","GEOGPRV","HWTDGISW",
            "PAADVACV","SMK_005","SMK_060"]]
print(convdata.dtypes)
```

- 2) Apply Transformation of the perceived health indicator variable to quantify it for further analysis. Original Data Set has the indicators starting from 0 which will not support our analysis during the average health calculation. Hence applying transformation to capture the indicators from (1-5)

```
[ ]: convdata["GENDVHDI"] = convdata["GENDVHDI"] + 1
```

- 3) Create python dictionary for the categorical variable with the values from the data dictionary

```
[ ]: agegroup={1:"12-17 years",2:"18-34 years",3:"35-49 years",4:"50-64 years",5:
        ↵"65<"}
sextype={1:"Male",2:"Female"}
perceivedhealthd={1:"Poor",2:"Fair",3:"Good",4:"Very good",5:"Excellent",10:
        ↵"Don't know"}
geo={10:"NEWFOUNDLAND AND LABRADOR",11:"PRINCE EDWARD ISLAND",12:"NOVA SCOTIA",
     13:"NEW BRUNSWICK",24:"QUEBEC",35:"ONTARIO",46:"MANITOBA",47:
        ↵"SASKATCHEWAN",
     48:"ALBERTA",59:"BRITISH COLUMBIA",60:"YUKON/NORTHWEST/NUNAVUT"}
bmi={1:"Normal weight",2:"Overweight/Obese",6:"Valid skip",9:"Not stated"}
SmokingFrequency={1:"Daily",2:"Occasionally",3:"Not at all",7:"Don't know",8:
        ↵"Refusal",9:"Not stated"}
physicallyactive={1:"Physically active above recommended level",2:"Physically
        ↵active below recommended level",3:"No physical activity",6:"Valid skip",9:
        ↵"Not stated"}
stoppedsmk={1:<1 yr ago",2:>1 year to <2 yrs ago",3:>2 year to <3 yrs ago",4:
        ↵">3 yrs ago",6:"Valid skip",7:"Don't know",8:"Refusal",9:"Not stated"}
```

- 4) Create new columns with the copy of the categorical variable and convert them using the dictionaries created below. Hence for the categorical variable we shall have two columns once with the numeric and other with the types which help in driving the analysis better in visualization.

```
[ ]: convdata["sextypc"] = convdata[["DHH_SEX"]].copy().replace({"DHH_SEX": sextypc})
convdata["agegroup"] = convdata[["DHHGAGE"]].copy().replace({"DHHGAGE": agegroup})
convdata["perceivedhealthd"] = convdata[["GENDVHDI"]].copy().
    replace({"GENDVHDI": perceivedhealthd})
convdata["geo"] = convdata[["GEOGPRV"]].copy().replace({"GEOGPRV": geo})
convdata["bmi"] = convdata[["HWTGISW"]].copy().replace({"HWTGISW": bmi})
convdata["SmokingFrequency"] = convdata[["SMK_005"]].copy().replace({"SMK_005": SmokingFrequency})
convdata["physicallyactive"] = convdata[["PAADVACV"]].copy().
    replace({"PAADVACV": physicallyactive})
convdata["stoppedsmk"] = convdata[["SMK_060"]].copy().replace({"SMK_060": stoppedsmk})
```

```
[ ]: print(convdata.dtypes)
```

```
[ ]: heatmap=convdata.copy()
convdata=convdata[['ADM_RN01','agegroup','sextypc','geo','perceivedhealthd','bmi','SmokingFrecuency']]
```

```
[ ]: values1=['Daily','Occasionally','Not at all']
convdata= convdata[convdata['SmokingFrequency'].isin(values1)]
display(convdata['SmokingFrequency'].unique())
values2=['Normal weight','Overweight/Obese']
convdata= convdata[convdata['bmi'].isin(values2)]
values3=['Don't know']
convdata= convdata[~convdata['perceivedhealthd'].isin(values3)]
convdata
convdata.to_csv('C:/Users/Venkateshwaran/OneDrive/Desktop/Data601_Assignment/
    -mine.csv', index=False)
```

Data Summary: Initiated the Analysis by the Pie Plotting the Individual Charts to Understand the summary of the different Health levels, Smoking and BMI

```
[ ]: analysis1a = convdata.groupby('SmokingFrequency').size().reset_index().
    rename(columns={"index": "value", 0: "count"})
analysis1a= analysis1a[analysis1a['SmokingFrequency'].isin(values1)]
fig1 = px.pie(analysis1a, values='count', names='SmokingFrequency', hole=.
    5, width=500, height=500, color_discrete_sequence=["green", "red", "yellow"])
fig1.update_layout(title="Survey Population Density against Smoking
    Frequency", title_x=0.5)
fig1.show()
```

```
[ ]: analysis1c = convdata.groupby('perceivedhealthd').size().reset_index().
    rename(columns={"index": "value", 0: "count"})
#analysis1c= analysis1b[analysis1b['bmi'].isin(values2)]
fig3 = px.pie(analysis1c, values='count', names='perceivedhealthd', hole=.
    5, width=500, height=500, color_discrete_sequence=px.colors.qualitative.G10)
```

```
fig3.update_layout(title="Overall Health Status",title_x=0.5)
fig3.show()
```

```
[ ]: analysis1b = convdata.groupby('bmi').size().reset_index().
   ↪rename(columns={"index": "value", 0: "count"})
analysis1b= analysis1b[analysis1b['bmi'].isin(values2)]
fig2 = px.pie(analysis1b, values='count', names='bmi',hole=.
   ↪.5, width=500, height=500, color_discrete_sequence=["red", "green"])
fig2.update_layout(title="Survey Population Density against BMI", title_x=0.5)
fig2.show()
```

```
[ ]: #analysis1.to_csv('C:/Users/Venkateshwaran/OneDrive/Desktop/Data601_Assignment/
   ↪question1_data.csv', index=False)
```

Inference:

- 1) About 19.3% of people have reported their Health status as Excellent and 3.6% of Population as Poor. Our aim is to analyze and understand the health status of the people with the poor data reported
- 2) About 12.67% of people having smoking habits with 10.5% with Daily smokers
- 3) About 60% of people reported them as Obese which is a noticeable Data for suggestion.

Understanding the Correlation between the BMI and Smoking

```
[ ]: analysis1=convdata.
   ↪groupby(['bmi','SmokingFrequency'],as_index=False)[‘GENDVHDI’].mean().
   ↪reset_index()
analysis1['percentage'] = analysis1['GENDVHDI'] / analysis1.
   ↪groupby('bmi')[‘GENDVHDI’].transform('sum') * 100
analysis1['percentage'] = round(analysis1['percentage'],2)
```

```
[ ]: fig = px.bar(analysis1, x="bmi", ↴
   ↪y="percentage", color='SmokingFrequency', barmode='group', height=500, width=1000, text_auto=True
#fig.update_yaxes(range=[0, 100])
fig.update_layout(
    title="Body Mass Index Against Average Perceived Health Indicator wrt to
   ↪Smoking Frequency", title_x=0.5,
    xaxis_title="Body Mass Index (BMI)",
    yaxis_title="Perceived Health Indicator",
)
fig.show()
```

Inference:

- 1) People with smoking habits have less health satisfaction level than people with not smoke.
- 2) However, Occasional Smokers have improved health than Non- Smokers under Overweight Category and which signifies that having obese will ideally affect human life span
- 3) Overall Daily Smokers has Bad Average Health Indicator ,With Obese it even worse.

4) Overall Average Falls in the Good- Very Good Zone

Understanding the Correlation of the BMI & Smoking with Different Age groups

```
[ ]: analysis2=convdata.groupby(['SmokingFrequency', 'agegroup'])['GENDVHDI'].mean().reset_index()
      ↪reset_index()

fig = px.line(analysis2, x="agegroup", y="GENDVHDI", ↪
      ↪color="SmokingFrequency", hover_data=['agegroup'], color_discrete_sequence=['red', 'green', 'orange'],
      height=500, width=1000)
fig.update_yaxes(range=[0, 5])
fig.update_layout(
    title="Impact of Perceived Health VS smoking( Agewise)", title_x=0.5,
    xaxis_title="Age Category",
    yaxis_title="Perceived Health Indicator",
)
fig.show()
```

```
[ ]: analysis3=convdata.groupby(['bmi', 'agegroup'])['GENDVHDI'].mean().reset_index()
analysis3
fig = px.line(analysis3, x="agegroup", y="GENDVHDI", ↪
      ↪color="bmi", hover_data=['agegroup'], height=500, width=1000)
fig.update_yaxes(range=[0, 5])
fig.update_layout(
    title="Impact of Perceived Health Vs BMI (AgeWise) ", title_x=0.5,
    xaxis_title="Age Category",
    yaxis_title="Perceived Health Indicator",
)
fig.show()
```

Inference: Both Smoking habit and Increased BMI is gradually affecting the Indicator of the peoples when they grow older and older

Understanding the people Density of Smoking and BMI

```
[ ]: analysis4=convdata.groupby(['SmokingFrequency', 'bmi'])['GENDVHDI'].count().reset_index()
      ↪reset_index()

analysis4['percentage'] = analysis4['GENDVHDI'] / analysis4.
      ↪groupby('SmokingFrequency')['GENDVHDI'].transform('sum') * 100
analysis4['percentage'] = round(analysis4['percentage'],2)

[ ]: fig = px.bar(analysis4, x="SmokingFrequency", y="percentage", ↪
      ↪color="bmi", hover_data=['bmi'], barmode='group', height=500, width=800, text_auto=True)
fig.update_yaxes(range=[0, 70])
fig.show()
```

Inference

- 1) Bar Plot clearly tells us the health levels are being impacted when the frequency of smoking is increased and people with the higher frequency of smoking has more prone to obese with

higher BMI.

- 2) Interestingly people with no smoke has more obese which clearly stucked to our initial understanding of BMI and Smoking relation

Understanding the Health Level of People with Canada Geographical Province

```
[ ]: analysis5=convdata.groupby(['geo'])['GENDVHDI'].mean().reset_index()
analysis5["geo"] = analysis5["geo"].replace(["NEWFOUNDLAND AND LABRADOR"], "Newfoundland & Labrador")
analysis5["geo"] = analysis5["geo"].replace(["YUKON/NORTHWEST/NUNAVUT"], "YUKON TERRITORY")
analysis5['geo']= analysis5['geo'].str.title()

import json
canada_states= json.load(open("canada_provinces.geojson",'r'))
state_id_map={}
for feature in canada_states['features']:
    feature['id']= feature['properties']['CODE']
    state_id_map[feature['properties']['NAME']] = feature['id']
state_id_map
```

```
[ ]: analysis5['id']=analysis5['geo'].apply(lambda x: state_id_map[x])
analysis5.rename(columns={'GENDVHDI': 'HealthStatus(1-5)'}, inplace=True)
analysis5
```

```
[ ]: fig= px.
    ↪choropleth_mapbox(analysis5,locations='id',geojson=canada_states,color='HealthStatus(1-5)',
                         mapbox_style="carto-positron",center={'lat':62.
    ↪24,'lon':-96.78})
fig.update_layout(mapbox_zoom=2)
fig.update_layout(title='Geographical wise Health Status',title_x=0.5)
fig.show()
```

Inference: Average Health levels of people in Quebec is decent and the health level of people in New Brunswick is comparatively poor

```
[ ]: analysissmokingmap=convdata.groupby('geo')['SmokingFrequency'].count().
    ↪reset_index()
analysissmokingmap["geo"] = analysissmokingmap["geo"].replace(["NEWFOUNDLAND AND LABRADOR"], "Newfoundland & Labrador")
analysissmokingmap["geo"] = analysissmokingmap["geo"].replace(["YUKON/NORTHWEST/NUNAVUT"], "YUKON TERRITORY")
analysissmokingmap['geo']= analysissmokingmap['geo'].str.title()
analysissmokingmap['geo']
analysissmokingmap['id']=analysissmokingmap['geo'].apply(lambda x: state_id_map[x])
analysissmokingmap['percentage'] = analysissmokingmap['SmokingFrequency'] / analysissmokingmap['SmokingFrequency'].sum() * 100
```

```

fig= px.
    ↪choropleth_mapbox(analysissmokingmap,locations='id',geojson=canada_states,color='percentage'
                        mapbox_style="carto-positron",center={'lat':62.
    ↪24,'lon':-96.78})
fig.update_layout(mapbox_zoom=2)
fig.update_layout(title='Geographical wise Smoking Status',title_x=0.5)
fig.show()

```

Inference:

- 1) Smoking Frequency of Yukon reported higher all across the canada, however their health status is average
- 2) We infer that because of the climatic conditions people tends to smoke comparitively more than other province
- 3) Peoples in Ontario has lesser smoking frequency and their health status is reported average

```

[ ]: analysisbmimap=convdata.copy()
analysisbmimap = analysisbmimap[analysisbmimap['bmi'] == 'Overweight/Obese']
analysisbmimap=analysisbmimap.groupby('geo')['bmi'].count().reset_index()
analysisbmimap["geo"] = analysisbmimap["geo"].replace(["NEWFOUNDLAND AND
    ↪LABRADOR"], "Newfoundland & Labrador")
analysisbmimap["geo"] = analysisbmimap["geo"].replace(["YUKON/NORTHWEST/
    ↪NUNAVUT"], "YUKON TERRITORY")
analysisbmimap['geo']= analysisbmimap['geo'].str.title()
analysisbmimap['id']=analysisbmimap['geo'].apply(lambda x: state_id_map[x])
analysisbmimap['percentage'] = analysisbmimap['bmi'] / analysisbmimap['bmi'] .
    ↪sum() * 100
fig2= px.
    ↪choropleth_mapbox(analysisbmimap,locations='id',geojson=canada_states,color='percentage',ho
                        mapbox_style="carto-positron",center={'lat':62.
    ↪24,'lon':-96.78})
fig2.update_layout(mapbox_zoom=2)
fig2.update_layout(title='Geographical wise Obese level',title_x=0.5)
fig2.show()

```

Inference:

- 1) Ontario Reported people with overweight its because of the population density with the average mean health reported
- 2) People in Yukon reported Lower BMi, with the smoking habit which clearly tell us the smoking is not directly related to the BMI component on peoples'health.However smoking ideally affects the health levels in other plots.

Understanding the Health Level of People with Stopped Smoking This Analysis is to understand the average health status of the people who have stopped smoking and people with still having smoking habits

```
[ ]: analysis5a = convdata[convdata['stoppedsmk'].isin(['<1 yr ago', '>1 year to <2 yrs ago', '>2 year to <3 yrs ago', '>3 yrs ago'])]
analysis5a= analysis5a.groupby(['stoppedsmk'])['GENDVHDI'].mean().reset_index()
analysis5a['Ind']='Yes'
analysis5b = convdata[~convdata['stoppedsmk'].isin(['<1 yr ago', '>1 year to <2 yrs ago', '>2 year to <3 yrs ago', '>3 yrs ago'])]
analysis5b= analysis5b.groupby(['stoppedsmk'])['GENDVHDI'].mean().reset_index()
analysis5b['Ind']='No'
analysis5c=pd.concat([analysis5a, analysis5b], axis=0)
analysis5c.groupby(['Ind'])['GENDVHDI'].mean().reset_index()
```

```
[ ]: fig = px.bar(analysis5c.groupby(['Ind'])['GENDVHDI'].mean().reset_index(), x="Ind", y="GENDVHDI", color='Ind', text_auto=True, height=500, width=500)
fig.update_yaxes(range=[0, 5])
fig.update_layout(
    title="Average Health Improvement upon stopping the smoking",
    xaxis_title="Stopped Smoking Indicator",
    yaxis_title="Perceived Health Indicator", title_x=0.5
)
fig.show()
```

Inference: It is clear that the people who have stopped smoking have better health indicator than the people with smoking habits. Hence its clear that Smoking has the direct relationship with the people health

Understanding the Health Level of People with Physical Activity This Analysis is to understand the health levels of the people with their levels of physical activities

```
[ ]: analysis6=convdata[['physicallyactive', 'bmi', 'GENDVHDI']]
analysis6['physicallyactive']= analysis6['physicallyactive'].apply(lambda x:'No physical activity' if x in ('Not stated') else x)
analysis6= analysis6[analysis6['physicallyactive']!='Valid skip']
analysis6=analysis6.groupby(['physicallyactive', 'bmi'])['GENDVHDI'].mean().reset_index()
analysis6['percentage'] = analysis6['GENDVHDI'] / analysis6.groupby('bmi')['GENDVHDI'].transform('sum') * 100
analysis6.sort_values(by = 'percentage', ascending=True)
```

```
[ ]: fig = px.bar(analysis6, x="bmi", y="percentage", color='physicallyactive', barmode='group', height=500, width=1000, text_auto=True)
fig.update_yaxes(range=[0, 50])
fig.update_layout(
    title="Physically Active Against Average Perceived Health Indicator wrt to BMI", title_x=0.5,
    xaxis_title="Body Mass Index (BMI)", yaxis_title="Perceived Health Indicator",
```

```
)
fig.show()
```

Inference: We infer that the people with the physical activity shows greater health level than people with no activity. It seems like linear based on the activity progression health levels may tend to improve a lot.

Understanding the Health Level of People with BMI (Gender Type- Comparison)
This analysis is to understand the health levels of the people against BMI with Gender Level Comparisons

```
[ ]: import plotly.express as px
values1=['Daily','Occasionally','Not at all']
heatmap= heatmap[heatmap['SmokingFrequency'].isin(values1)]
display(heatmap['SmokingFrequency'].unique())
values2=['Normal weight','Overweight/Obese']
heatmap= heatmap[heatmap['bmi'].isin(values2)]
values3=['Don't know']
heatmap= heatmap[~heatmap['perceivedhealthd'].isin(values3)]
heatmap
fig = px.density_heatmap(heatmap, x="SmokingFrequency", y="bmi", z="GENDVHDI", facet_col="sextyle", histfunc="avg", text_auto=True,
                           labels=dict(x="Smoking Frequency", y="bmi", z= "average health Indicator"))
fig.update_layout(title='Gender Wise- BMI level for different smoking categories', title_x=0.5)
fig.show()
```

Inference As observed in the other charts, it's clear that the health levels are relating to BMI and it explains that people with daily smoking have very low health levels, with the higher obese status.

Understanding the Health Level of People with their Gender Wise- Physical Active level This analysis is to understand the health levels of the people who are physically active.

```
[ ]: heatmap2= heatmap[heatmap['PAADVACV'].isin([1,2,3])]
fig = px.density_heatmap(heatmap2, x="physicallyactive", y="bmi", z="GENDVHDI", facet_col="sextyle", histfunc="avg", text_auto=True,
                           labels=dict(x="physicallyactive", y="bmi", z= "average health Indicator"))
fig.update_layout(title='Gender Wise- Physical Active level', title_x=0.5)
fig.show()
```

Inference We infer that the people with physically active shows improved health than people with no activity.

Understanding the Health Level of People with their Stopped Smoking Status This Analysis is to understand the health levels of the people against their Stopped Smoking Status in years.

```
[ ]: fig = px.density_heatmap(heatmap[~heatmap['stoppedsmk'].isin(["Not\u2014\u2014stated","Don't know","Valid skip","Refusal"])], x="stoppedsmk", y="sextpe",\u2014\u2014z="GENDVHDI", histfunc="avg",text_auto=True,\u2014\u2014\n      labels=dict(x="stoppedsmk", y="SmokingFrequency", z=\u2014\u2014\n      \u2014\u2014"avge health Indicator"))\nfig.update_layout(title='Genderwise Status of People Stopped Smoking',title_x=0.\u2014\u20145)\nfig.show()\n\nfig = px.density_heatmap(heatmap[heatmap['stoppedsmk'].isin(["Not\u2014\u2014stated","Don't know","Valid skip","Refusal"])], x="stoppedsmk", y="sextpe",\u2014\u2014z="GENDVHDI", histfunc="avg",text_auto=True,\u2014\u2014\n      labels=dict(x="stoppedsmk", y="SmokingFrequency", z=\u2014\u2014\n      \u2014\u2014"avge health Indicator"))\nfig.update_layout(title='Genderwise Status of People dont smoke',title_x=0.5)\nfig.show()
```

Inference

- 1) We Infer that the people with stopped smoking has shown greater improvement in health and people without smoking habit has close to excellent health status

Conclusion Overall we infer that the smoking has major impact on physical health upon our analysis against geography & gender, However overweight/obese doesn't have much directly related to smoking. People who stopped smoking shows greater improvements in health levels. Also we infer that the BMI is greatly improved for the people with physically active and its declining for people with no physically active

Hence we conclude that Smoking is directly related to Physical health in converse to BMI and suggest that to improve physical health it's strongly advised to stop smoking and increase fitness level.

Question 4 - Which demographic groups (age, sex, region) are most affected by health barriers? This is a more comprehensive question that involved different demographic of people. The health barriers refer to cannabis use, smoking frequency and alcohol consumption. In the initial dataset, the demographic groups includes people of different gender, age, region and education level. This analysis will explore the impact of each health barriers to each demographic group one by one. It's crucial for our analysis to further consider people from different demographic groups as the health barriers usually affects people differently to their background. To begin data analysis, there are several steps to conduct the data cleaning. The data cleaning process removes the values that are undesired to answering this guiding questions such as "don't know", "refusal", "not stated" and "valid skip". Then it will convert the values of all the desired variables into integer type as the survey response received are all in format of integer.

```
[ ]: df_q5=df.copy() # make a copy of initial filtered data and filter for the
      ↪variables relevant to this question.
df_q5=df_q5.
      ↪filter(items=['CAN_015','SMK_005','SMMK_060','ALC_015','ALC_020','DHHGAGE','DHH_SEX','GEOGP
print(df_q5.head())

[ ]: df_q5.to_csv('Question 5.csv',index=False) #Create a new csv for analysis for
      ↪this guiding question.

[ ]: #A peek to the dataset info for its data types and columns names.
df_q5.info()
df_q5.describe()
df_q5.dtypes

[ ]: df_q5.isnull().sum() #check for missing values
```

0.3.4 Data Cleaning:

```
[ ]: #Removing undesired values from cannabis use variable.
df_q5=df_q5.drop(df_q5[df_q5['CAN_015']==7].index)
df_q5=df_q5.drop(df_q5[df_q5['CAN_015']==8].index)
df_q5=df_q5.drop(df_q5[df_q5['CAN_015']==9].index)

[ ]: #Removing undesired values from smoking frequency variable.
df_q5=df_q5.drop(df_q5[df_q5['SMK_005']==7].index)
df_q5=df_q5.drop(df_q5[df_q5['SMK_005']==8].index)
df_q5=df_q5.drop(df_q5[df_q5['SMK_005']==9].index)

[ ]: #Removing undesired values from alcohol consumption variable.
df_q5=df_q5.drop(df_q5[df_q5['ALC_015']==96].index)
df_q5=df_q5.drop(df_q5[df_q5['ALC_015']==97].index)
df_q5=df_q5.drop(df_q5[df_q5['ALC_015']==98].index)
df_q5=df_q5.drop(df_q5[df_q5['ALC_015']==99].index)

[ ]: #Removing undesired values from alcohol consumption (drinking frequency) variable.
df_q5=df_q5.drop(df_q5[df_q5['ALC_020']==96].index)
df_q5=df_q5.drop(df_q5[df_q5['ALC_020']==97].index)
df_q5=df_q5.drop(df_q5[df_q5['ALC_020']==98].index)
df_q5=df_q5.drop(df_q5[df_q5['ALC_020']==99].index)

[ ]: #Removing undesired values from household education level variable.
df_q5=df_q5.drop(df_q5[df_q5['EHG2DVH3']==9].index)

[ ]: # Converted interested variables into integer type
df_q5['CAN_015'] = df_q5['CAN_015'].astype(int)
df_q5['GEOGP
      ↪RV'].dtypes
```

```

df_q5['SMK_005'] = df_q5['SMK_005'].astype(int)
df_q5['SMK_005'].dtypes
df_q5['ALC_015'] = df_q5['ALC_015'].astype(int)
df_q5['ALC_015'].dtypes
df_q5['ALC_020'] = df_q5['ALC_020'].astype(int)
df_q5['ALC_020'].dtypes
df_q5['DHH_SEX'] = df_q5['DHH_SEX'].astype(int)
df_q5['DHH_SEX'].dtypes
df_q5['DHHGAGE'] = df_q5['DHHGAGE'].astype(int)
df_q5['DHHGAGE'].dtypes
df_q5['GEOGPRV'] = df_q5['GEOGPRV'].astype(int)
df_q5['GEOGPRV'].dtypes
df_q5['EHG2DVH3'] = df_q5['EHG2DVH3'].astype(int)
df_q5['EHG2DVH3'].dtypes

```

After running the above code, we can see from below that the maximum values shows the undesired values have been removed from all the interested varibels, and all the values have been converted to integer type.

```
[ ]: df_q5.describe()
```

0.3.5 Data Analysis and Visualization

```

[ ]: #Group cannabis use results based on sex type.
df5_cansex=df_q5.groupby('DHH_SEX', as_index=False)[['CAN_015']].
    ↪value_counts(normalize=True,sort=False)
display(df5_cansex)

[ ]: # Create a bar plot to show the cannabis use infromation based on each gender
    ↪type.
plt.figure(figsize=(10,7))
plt.bar(df5_cansex['DHH_SEX'].unique(),df5_cansex['proportion'][::2],width=0.
    ↪3,label='Poole People Used Cannabis in Past 12 Months')
plt.bar(df5_cansex['DHH_SEX'].unique()+0.3,df5_cansex['proportion'][1::2],width=0.3,
    ↪label="People Didn't Use Cannabis in Past 12 Months")
plt.xticks(df5_cansex['DHH_SEX'].unique()+0.3/2,('Male','Female'))
plt.xlabel('Gender Impacted by Canabis')
plt.ylabel('Percent')
plt.legend()
plt.show()

```

From the above bar plot, we can find that females uses canabis slightly more than males.

```

[ ]: #Create a pie chart to visualize the proportaion of male and female using
    ↪cannabis.
cannabis_sex_data=df_q5[['CAN_015', 'DHH_SEX']].dropna()
cannabis_sex=cannabis_sex_data.groupby('DHH_SEX')[['CAN_015']].sum()
DHH_SEX_labels=

```

```

1: "Male",
2: "Female"
}
cannabis_sex.index=cannabis_sex.index.map(DHH_SEX_labels)

plt.figure(figsize=(5,5))
cannabis_sex.plot(kind='pie', autopct='%.1f%%', colors=['blue', 'orange'], startangle=90, textprops={'fontsize':12})
plt.title('Cannabis Use by Sex')
plt.ylabel('')
plt.show()

```

A supplementary pie chart is created to better visualize the proportion of male and female using cannabis. We can see that 8% more of females use cannabis compared to males.

```

[ ]: #Create a bar plot to explore the smoking frequency by each gender.
smoke_sex=df_q5[['SMK_005', 'DHH_SEX']].dropna()
smoke_sex=smoke_sex.groupby('DHH_SEX')['SMK_005'].mean()
DHH_SEX_labels={
    1: "Male",
    2: "Female"
}
smoke_sex.index=smoke_sex.index.map(DHH_SEX_labels)

plt.figure(figsize=(8,6))
plt.ylim(0,3)
smoke_sex.plot(kind='bar', color=['skyblue', 'pink'])
plt.title('Average Smoke Frequency by Gender')
plt.xlabel('Sex')
plt.ylabel('Average Smoke Frequency')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```

Based on the bar chart, we can interpret that females smoke slightly more often than males.

```

[ ]: #Create a pie chart to visualize the proportion of male and female for their smoking frequency.
smoking_sex=df_q5[['SMK_005', 'DHH_SEX']].dropna()
smoking_bysex=smoking_sex.groupby('DHH_SEX')['SMK_005'].sum()
DHH_SEX_labels={
    1: "Male",
    2: "Female"
}
smoking_bysex.index=smoking_bysex.index.map(DHH_SEX_labels)

plt.figure(figsize=(5, 5))

```

```

smoking_bysex.plot(kind='pie', autopct='%.1f%%', colors=['skyblue', 'pink'],  

    startangle=90, textprops={'fontsize': 12})  

plt.title('Smoke Frequency by Sex')  

plt.ylabel('')
plt.show()

```

A supplementary pie chart is created to better visualize the proportion of male and female for their smoking frequency. We can see that females smoke 8% more often than males.

```

[ ]: # Create bar chart for alcohol consumptions based on two gender types.  

alcohol_sex=df_q5[['ALC_015', 'ALC_020', 'DHH_SEX']].dropna()  

alc_015_bysex=alcohol_sex.groupby('DHH_SEX')['ALC_015'].mean()  

alc_020_bysex=alcohol_sex.groupby('DHH_SEX')['ALC_020'].mean()  

DHH_SEX_labels={  

    1: "Male",  

    2: "Female"  

}  

alc_015_bysex.index=alc_015_bysex.index.map(DHH_SEX_labels)  

alc_020_bysex.index=alc_020_bysex.index.map(DHH_SEX_labels)  
  

plt.figure(figsize=(14,6))  
  

plt.subplot(1,2,1)  

alc_015_bysex.plot(kind='bar',color=['blue','orange'])  

plt.title('Average Alcohol Consumption Frequency by Sex')  

plt.xlabel('Sex')  

plt.ylabel('Average Alcohol Consumption Frequency')  

plt.xticks(rotation=0)  

plt.grid(axis='y',linestyle='--',alpha=0.7)  
  

plt.subplot(1,2,2)  

alc_020_bysex.plot(kind='bar',color=['blue','orange'])  

plt.title('Average Alcohol Consumption (Drink 4+/5+ One Occasion) by Sex')  

plt.xlabel('Sex')  

plt.ylabel('Average Alcohol Consumption (Drink 4+/5+ One Occasion)')  

plt.xticks(rotation=0)  

plt.grid(axis='y',linestyle='--',alpha=0.7)

```

From the above bar plots, we found that males generally consumes more alcohols than females in terms of both alcohol consumption frequency and drinking 4+/5+ cans on one occasion.

```

[ ]: # Plot a bar chart to investigate the cannabis use by different age group.  

data_filtered=df_q5[['CAN_015', 'DHHGAGE']].dropna()  

cannabis_use_different_age=data_filtered.groupby('DHHGAGE')['CAN_015'].mean()  

DHHGAGE_labels={  

    1: '12 to 17 years',  

    2: '18 to 34 years',  

    3: '35 to 49 years',
}

```

```

4: '50 to 64 years',
5: '65 years and older'
}
cannabis_use_different_age.index=cannabis_use_different_age.index.
↪map(DHHGAGE_labels)

plt.figure(figsize=(10,6))
cannabis_use_different_age.plot(kind='bar',color='olive')
plt.title('Average Cannabis Use by Different Age Group')
plt.xlabel('Age Group')
plt.ylabel('Average Cannabis Use')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)
plt.show()

```

Based on the above graph, we can tell that people who are at 12 to 17 years and 65 years and older use cannabis most often, with age group 50 years to 64 years and 30 years to 49 years following closely behind. People of 18 years to 34 years use cannabis least often.

```

[ ]: # Plot a bar chart to investigate the smoking freqnecy by differnt age group.
smoke_data_filtered=df_q5[['SMK_005', 'DHHGAGE']].dropna()
smoke_byage=smoke_data_filtered.groupby('DHHGAGE')['SMK_005'].mean()
DHHGAGE_labels={
    1: "12 to 17 years",
    2: "18 to 34 years",
    3: "35 to 49 years",
    4: "50 to 64 years",
    5: "65 years and older"
}
smoke_byage.index=smoke_byage.index.map(DHHGAGE_labels)

plt.figure(figsize=(10,6))
smoke_byage.plot(kind='bar',color='purple')
plt.title('Average Smoke Frequency by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Average Smoke Frequency')
plt.xticks(rotation=45,ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)
plt.show()

```

From the bar plot, we found that people people who are at 12 to 17 years and 65 years and older smoke most often. However, the smoking frequency for the rest of age groups are approximately the same.

```

[ ]: # Create two bar charts to analyze the affect of alcohol consumption to each
↪age group.
alcohol_data_filtered=df_q5[['ALC_015', 'ALC_020', 'DHHGAGE']].dropna()
alc_015_by_age=alcohol_data_filtered.groupby('DHHGAGE')['ALC_015'].mean()

```

```

alc_020_by_age=alcohol_data_filtered.groupby('DHHGAGE')['ALC_020'].mean()
DHHGAGE_labels={
    1: "12 to 17 years",
    2: "18 to 34 years",
    3: "35 to 49 years",
    4: "50 to 64 years",
    5: "65 years and older"
}
alc_015_by_age.index=alc_015_by_age.index.map(DHHGAGE_labels)
alc_020_by_age.index=alc_020_by_age.index.map(DHHGAGE_labels)

plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
alc_015_by_age.plot(kind='bar',color='coral')
plt.title('Average Alcohol Consumption Frequency in Past 12 Months')
plt.xlabel('Age Group')
plt.ylabel('Average Alcohol Consumption Frequency')
plt.xticks(rotation=45,ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)

plt.subplot(1,2,2)
plt.ylim(0,2.5)
alc_020_by_age.plot(kind='bar',color='mediumblue')
plt.title('Average Alcohol Consumption (Drink 4+/5+ One Occasion)')
plt.xlabel('Age Group')
plt.ylabel('Average Alcohol Consumption (Drink 4+/5+ One Occasion)')
plt.xticks(rotation=45,ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)

```

Based on the above graphs, people at the age of 12 to 17 years drink least often. This is because the law forbids minors to drink alcohol. From the age of 18 years to 34 years, people start to drink more often, and then reach the maximum alcohol consumption frequency at the age of 50 years to 64 years. However, people at younger age (18 years to 34 years) tend to drink most of alcohol at once. After the age of 35 years, people start to drink less and less alcohol at one occasion. This might be the result of physically aging that the body is less able to handle alcohols over time.

```
[ ]: # Create a bar chart to analyze cannabis use by region.
cannabis_region=df_q5[['CAN_015', 'GEOGPRV']].dropna()
GEOGPRV_labels={
    10: "NEWFOUNDLAND AND LABRADOR",
    11: "PRINCE EDWARD ISLAND",
    12: "NOVA SCOTIA",
    13: "NEW BRUNSWICK",
    24: "QUEBEC",
    35: "ONTARIO",
    46: "MANITOBA",
    47: "SASKATCHEWAN",
}
```

```

48: "ALBERTA",
59: "BRITISH COLUMBIA",
60: "YUKON/NORTHWEST/NUNAVUT TERRITORIES"
}
cannabis_region['GEOGPRV']=cannabis_region['GEOGPRV'].map(GEOGPRV_labels)
cannabis_byregion=cannabis_region.groupby('GEOGPRV')['CAN_015'].mean()

plt.figure(figsize=(10,7))
plt.ylim(0,2)
cannabis_byregion.plot(kind='bar', color='olive')
plt.title('Average Cannabis Use by Region')
plt.xlabel('Region')
plt.ylabel('Average Cannabis Use')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

```

The above plot reveals that people among different region use cannabis approximately same frequently except for the YUKON/NORTHWEST/NUNAVUT territories. This may be due to the less population living in these territories.

```

[ ]: # Create bar plot for smoking frequency for people at each region.
smoke_region=df_q5[['SMK_005', 'GEOGPRV']].dropna()
GEOGPRV_labels={
    10: "NEWFOUNDLAND AND LABRADOR",
    11: "PRINCE EDWARD ISLAND",
    12: "NOVA SCOTIA",
    13: "NEW BRUNSWICK",
    24: "QUEBEC",
    35: "ONTARIO",
    46: "MANITOBA",
    47: "SASKATCHEWAN",
    48: "ALBERTA",
    59: "BRITISH COLUMBIA",
    60: "YUKON/NORTHWEST/NUNAVUT TERRITORIES"
}
smoke_region['GEOGPRV']=smoke_region['GEOGPRV'].map(GEOGPRV_labels)
smoke_byregion=smoke_region.groupby('GEOGPRV')['SMK_005'].mean()

plt.figure(figsize=(10,7))
smoke_byregion.plot(kind='bar', color='purple')
plt.title('Average Smoke Frequency by Region')
plt.xlabel('Region')
plt.ylabel('Average Smoke Frequency')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)

```

```

plt.tight_layout()
plt.show()

```

Based on the plot, we found that the smoking frequency for people at each provinces are approximately the same except for the three territories and Alberta. The reason behind why Albertans smoke less frequently compared to people at other provinces needs to be further investigated.

```

[ ]: # Create bar plot for alcohol consumption frequency across each province/
      ↵region.

alcohol_data_filtered=df_q5[['ALC_015', 'ALC_020', 'GEOGPRV']].dropna()
alc_015_region=alcohol_data_filtered.groupby('GEOGPRV')['ALC_015'].mean()
alc_020_region=alcohol_data_filtered.groupby('GEOGPRV')['ALC_020'].mean()
GEOGPRV_labels={

10: "NEWFOUNDLAND AND LABRADOR",
11: "PRINCE EDWARD ISLAND",
12: "NOVA SCOTIA",
13: "NEW BRUNSWICK",
24: "QUEBEC",
35: "ONTARIO",
46: "MANITOBA",
47: "SASKATCHEWAN",
48: "ALBERTA",
59: "BRITISH COLUMBIA",
60: "YUKON/NORTHWEST/NUNAVUT TERRITORIES"
}
alc_015_region.index=alc_015_region.index.map(GEOGPRV_labels)
alc_020_region.index=alc_020_region.index.map(GEOGPRV_labels)

plt.figure(figsize=(14,6))

plt.subplot(1,2,1)
alc_015_region.plot(kind='bar',color='coral')
plt.title('Average Alcohol Consumption Frequency (Past 12 Months) By Region')
plt.xlabel('Region Group')
plt.ylabel('Average Alcohol Consumption Frequency')
plt.xticks(rotation=45,ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)

plt.subplot(1,2,2)
alc_020_region.plot(kind='bar',color='mediumblue')
plt.title('Average Alcohol Consumption (Drink 4+/5+ One Occasion) By Region')
plt.xlabel('Region Group')
plt.ylabel('Average Alcohol Consumption (Drink 4+/5+ One Occasion)')
plt.xticks(rotation=45,ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)

```

From the bar charts, we found that people in Quebec and British Columbia drink alcohol most frequently, and people in New Brunswick have least alcohol consumption frequency. However, people

in the YUKON/NORTHWEST/NUNAVUT territories consumes more alcohol at one occasion at a significantly high frequency compared to other provinces.

```
[ ]: # Plot bar chart to visualize the cannabis use circumstance for households at each educational level.
cannabis_education=df_q5[['CAN_015', 'EHG2DVH3']].dropna()
EHG2DVH3_labels={
    1: "Less than secondary school graduation",
    2: "Secondary school graduation, no post-secondary education",
    3: "Post-secondary certificate/diploma/university degree"
}
cannabis_education['EHG2DVH3']=cannabis_education['EHG2DVH3'].map(EHG2DVH3_labels)
cannabis_education=cannabis_education.groupby('EHG2DVH3')['CAN_015'].mean()

plt.figure(figsize=(8.5,7))
cannabis_education.plot(kind='bar',color='olive',width=0.25)
plt.title('Average Cannabis Use by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Average Cannabis Use')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)
plt.tight_layout()
plt.show()
```

Based on the bar chart, we see that households with less than secondary graduation use cannabis most often, however, the difference is not significant among other two groups.

```
[ ]: # Create bar plot to visualize smoking frequency among each education level
smoke_education=df_q5[['SMK_005', 'EHG2DVH3']].dropna()
EHG2DVH3_labels={
    1: "Less than secondary school graduation",
    2: "Secondary school graduation, no post-secondary education",
    3: "Post-secondary certificate/diploma/university degree"
}
smoke_education['EHG2DVH3']=smoke_education['EHG2DVH3'].map(EHG2DVH3_labels)
smoke_education=smoke_education.groupby('EHG2DVH3')['SMK_005'].mean()

plt.figure(figsize=(8, 7))
plt.ylim(0,3)
smoke_education.plot(kind='bar', color='purple', width=0.3)
plt.title('Average Smoke Frequency by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Average Smoke Frequency')
plt.xticks(rotation=45,ha='right')
plt.grid(axis='y',linestyle='--',alpha=0.7)
plt.tight_layout()
```

```
plt.show()
```

For smoking frequency, we see that the post secondary group has slightly higher smoking frequency but not significantly high among other groups. The smoking freuqency for less than secondary group and secondary graduation group is about the same.

```
[ ]: #Plot bar chart for alcohol consumption for each edcuation level group.  
alcohol_education=df_q5[['ALC_015', 'ALC_020', 'EHG2DVH3']].dropna()  
alc_015_education=alcohol_education.groupby('EHG2DVH3')['ALC_015'].mean()  
alc_020_education=alcohol_education.groupby('EHG2DVH3')['ALC_020'].mean()  
  
EHG2DVH3_labels={  
    1: "Less than secondary school graduation",  
    2: "Secondary school graduation, no post-secondary education",  
    3: "Post-secondary certificate/diploma/university degree"  
}  
alc_015_education.index=alc_015_education.index.map(EHG2DVH3_labels)  
alc_020_education.index=alc_020_education.index.map(EHG2DVH3_labels)  
  
plt.figure(figsize=(14,8))  
  
plt.subplot(1,2,1)  
alc_015_education.plot(kind='bar',color='coral')  
plt.title('Average Alcohol Consumption by Education Level')  
plt.xlabel('Education Level')  
plt.ylabel('Average Alcohol Consumption')  
plt.xticks(rotation=45, ha='right')  
plt.grid(axis='y',linestyle='--',alpha=0.7)  
  
plt.subplot(1,2,2)  
alc_020_education.plot(kind='bar',color='mediumblue')  
plt.title('Average Alcohol Consumption by Education Level')  
plt.xlabel('Education Level')  
plt.ylabel('Average Alcohol Consumption')  
plt.xticks(rotation=45,ha='right')  
plt.grid(axis='y',linestyle='--',alpha=0.7)  
plt.tight_layout()  
plt.show()
```

Based on the graph, we found that people with post-secondary education drink most frequently, with people with secondary education following closely behind. We also found that people wth post-secondary or secondary education consume more alcohol at once compared to people with no secondary education.

```
[ ]:
```

0.3.6 Health Drivers Vs. Health Improvements

Question 5 - What is the relationship between regular exercise and self-reported health status? In order to answer our 5th question, I'll start by charting reported health by reported physical activity against each other. Afterwards I'll break down how different demographics and other relevant variables provided in the dataset affect how the two variables of interest affect their results. To begin I'll give some details on reported health and reported physical activity. Reported health is broken down into 5 responses ranging from "Poor" to "Excellent" and reported physical health is broken down into if the respondent's activity is "Above" or "Below" recommended activity guidelines.

```
[ ]: df_question6=df_general.copy()
df_question6=df_question6.filter(items=[
    'GEN_005',
    'HWTGISW',
    'PAADVACV',
    'DHHGAGE',
    'DHH_SEX',
    'GEOGPRV',
    'EHG2DVH3'
])
#print(df_question6.head())
#setting some print and summary functions to comments to remove clutter
```

Concerning data cleaning I have cleaned the data to select the variables I will be working with and then removed rows with response that are not useful to my analysis. Responses such as "valid skip", "refused", and "don't know" are not of interest to my analysis and have been removed from the dataset I will be working with. Finally, I run code to check for missing values in my dataset.

```
[ ]: df_question6=df_question6.drop(df_question6[df_question6['GEN_005']==8].index)
df_question6=df_question6.drop(df_question6[df_question6['GEN_005']==7].index)
df_question6=df_question6.drop(df_question6[df_question6['HWTGISW']==6].index)
df_question6=df_question6.drop(df_question6[df_question6['HWTGISW']==9].index)
df_question6=df_question6.drop(df_question6[df_question6['PAADVACV']==9].index)
df_question6=df_question6.drop(df_question6[df_question6['PAADVACV']==6].index)
df_question6=df_question6.drop(df_question6[df_question6['PAADVACV']==3].index)
df_question6=df_question6.drop(df_question6[df_question6['EHG2DVH3']==9].index)
df_question6.isnull().sum()
```

I added a correlation heat map but there are no strong correlations between any of the variables included in my analysis.

```
[ ]: cor6 = df_question6.corr()

labels6=['General Health', 'BMI', 'Physical Activity', 'Age', 'Sex', 'Location', ↴'Education']
sns.heatmap(cor6, annot=True, cmap='coolwarm', center=0, xticklabels=labels6, ↴
            yticklabels=labels6)
plt.title("Correlation Between Physical Activity and Health Variables")
```

```
plt.show()
```

To start I've visualized the initial two variables physical activity and reported health. To do this I grouped physical activity by the reported health categories and visualised them in a bar chart. The initial plot shows a clear trend between the reported health and how much of that population is above the recommended activity guideline. As the reported health decreases the difference between the proportion of people above and below the activity guideline shows downwards and upward trends respectively.

```
[ ]: df6_health=df_question6.groupby('GEN_005', as_index=False)[['PAADVACV']].  
      ↪value_counts(normalize=True, sort=False)  
plt.figure().set_figwidth(10)  
plt.bar(df6_health['GEN_005'].unique(), df6_health['proportion'][::2], width=0.  
      ↪3, label='Percent Above Activity Guideline')  
plt.bar(df6_health['GEN_005'].unique()+0.3, df6_health['proportion'][1::  
      ↪2], width=0.3, label='Percent Below Activity Guideline')  
plt.xticks(df6_health['GEN_005'].unique()+0.3/2, ('Excellent', 'Very  
      ↪Good', 'Good', 'Fair', 'Poor'))  
plt.xlabel('Physical Activity By Health Status')  
plt.ylabel('Percent')  
plt.legend()  
plt.show()
```

I created subplots showing the first relationship but divided up by age groups. As can be seen in the graphs for the younger age groups there is no discernible trend between physical activity and reported health. This plot indicates that the younger age group are overall more active than older age groups but there are other factors that are more important to the younger age groups reported health. The age groups over 50 years old do show the trend in the initial plot. From this it could be reasonably concluded that as age increases physical activity becomes more important to ones reported health.

```
[ ]: df_question6_age1=df_question6.drop(df_question6[df_question6['DHHGAGE']!=2].  
      ↪index)  
df6_health_age1=df_question6_age1.groupby('GEN_005',  
      ↪as_index=False)[['PAADVACV']].value_counts(normalize=True, sort=False)  
df_question6_age2=df_question6.drop(df_question6[df_question6['DHHGAGE']!=3].  
      ↪index)  
df6_health_age2=df_question6_age2.groupby('GEN_005',  
      ↪as_index=False)[['PAADVACV']].value_counts(normalize=True, sort=False)  
df_question6_age3=df_question6.drop(df_question6[df_question6['DHHGAGE']!=4].  
      ↪index)  
df6_health_age3=df_question6_age3.groupby('GEN_005',  
      ↪as_index=False)[['PAADVACV']].value_counts(normalize=True, sort=False)  
df_question6_age4=df_question6.drop(df_question6[df_question6['DHHGAGE']!=5].  
      ↪index)  
df6_health_age4=df_question6_age4.groupby('GEN_005',  
      ↪as_index=False)[['PAADVACV']].value_counts(normalize=True, sort=False)
```

```

fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes[0,0].bar(df6_health_age1['GEN_005'].
    ↪unique(),df6_health_age1['proportion'][::2],width=0.3,label='Percent Above
    ↪Activity Guideline')
axes[0,0].bar(df6_health_age1['GEN_005'].unique()+0.
    ↪3,df6_health_age1['proportion'][1::2],width=0.3, label='Percent Below
    ↪Activity Guideline')
axes[0,0].set_xticks(df6_health_age1['GEN_005'].unique()+0.3/
    ↪2,('Excellent','Very Good','Good','Fair','Poor'))
axes[0,0].set_xlabel('Physical Activity By Health Status 18-34')
axes[0,0].set_ylabel('Percent')
axes[0,0].legend()

axes[0,1].bar(df6_health_age2['GEN_005'].
    ↪unique(),df6_health_age2['proportion'][::2],width=0.3,label='Percent Above
    ↪Activity Guideline')
axes[0,1].bar(df6_health_age2['GEN_005'].unique()+0.
    ↪3,df6_health_age2['proportion'][1::2],width=0.3, label='Percent Below
    ↪Activity Guideline')
axes[0,1].set_xticks(df6_health_age2['GEN_005'].unique()+0.3/
    ↪2,('Excellent','Very Good','Good','Fair','Poor'))
axes[0,1].set_xlabel('Physical Activity By Health Status 35-49')
axes[0,1].set_ylabel('Percent')
axes[0,1].legend()

axes[1,0].bar(df6_health_age3['GEN_005'].
    ↪unique(),df6_health_age3['proportion'][::2],width=0.3,label='Percent Above
    ↪Activity Guideline')
axes[1,0].bar(df6_health_age3['GEN_005'].unique()+0.
    ↪3,df6_health_age3['proportion'][1::2],width=0.3, label='Percent Below
    ↪Activity Guideline')
axes[1,0].set_xticks(df6_health_age3['GEN_005'].unique()+0.3/
    ↪2,('Excellent','Very Good','Good','Fair','Poor'))
axes[1,0].set_xlabel('Physical Activity By Health Status 50-64')
axes[1,0].set_ylabel('Percent')
axes[1,0].legend()

axes[1,1].bar(df6_health_age4['GEN_005'].
    ↪unique(),df6_health_age4['proportion'][::2],width=0.3,label='Percent Above
    ↪Activity Guideline')
axes[1,1].bar(df6_health_age4['GEN_005'].unique()+0.
    ↪3,df6_health_age4['proportion'][1::2],width=0.3, label='Percent Below
    ↪Activity Guideline')

```

```

axes[1,1].set_xticks(df6_health_age4['GEN_005'].unique()+0.3/
    ↵2,('Excellent','Very Good','Good','Fair','Poor'))
axes[1,1].set_xlabel('Physical Activity By Health Status 65 plus')
axes[1,1].set_ylabel('Percent')
axes[1,1].legend()

plt.show()

```

Another breakdown was the two main variables of interest by the respondent's level of education. The plots show that if a respondent did not complete high school, they are less likely to be active than respondents who completed high school or have a post secondary degree. There were minimal differences between those that completed high school or hold a post secondary degree. All plots follow the trend shown in the first graph.

```

[ ]: df_question6edu1=df_question6.drop(df_question6[df_question6['EHG2DVH3']!=1] .
    ↵index)
df6_edu1=df_question6edu1.groupby('GEN_005', as_index=False)[['PAADVACV']].
    ↵value_counts(normalize=True,sort=False)
df_question6edu2=df_question6.drop(df_question6[df_question6['EHG2DVH3']!=2] .
    ↵index)
df6_edu2=df_question6edu2.groupby('GEN_005', as_index=False)[['PAADVACV']].
    ↵value_counts(normalize=True,sort=False)
df_question6edu2=df_question6.drop(df_question6[df_question6['EHG2DVH3']!=3] .
    ↵index)
df6_edu3=df_question6edu2.groupby('GEN_005', as_index=False)[['PAADVACV']].
    ↵value_counts(normalize=True,sort=False)

fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes[0,0].bar(df6_edu1['GEN_005'].unique(),df6_edu1['proportion'][::2],width=0.
    ↵3,label='Percent Above Activity Guideline')
axes[0,0].bar(df6_edu1['GEN_005'].unique()+0.3,df6_edu1['proportion'][1::2],
    ↵width=0.3, label='Percent Below Activity Guideline')
axes[0,0].set_xticks(df6_edu1['GEN_005'].unique()+0.3/2,('Excellent','Very
    ↵Good','Good','Fair','Poor'))
axes[0,0].set_xlabel('Physical Activity By Health Status No HS Diploma')
axes[0,0].set_ylabel('Percent')
axes[0,0].legend()

axes[0,1].bar(df6_edu2['GEN_005'].unique(),df6_edu2['proportion'][::2],width=0.
    ↵3,label='Percent Above Activity Guideline')
axes[0,1].bar(df6_edu2['GEN_005'].unique()+0.3,df6_edu2['proportion'][1::2],
    ↵width=0.3, label='Percent Below Activity Guideline')
axes[0,1].set_xticks(df6_edu2['GEN_005'].unique()+0.3/2,('Excellent','Very
    ↵Good','Good','Fair','Poor'))
axes[0,1].set_xlabel('Physical Activity By Health Status HS Diploma')
axes[0,1].set_ylabel('Percent')
axes[0,1].legend()

```

```

axes[1,0].bar(df6_edu3['GEN_005'].unique(),df6_edu3['proportion'][::2],width=0.3,label='Percent Above Activity Guideline')
axes[1,0].bar(df6_edu3['GEN_005'].unique()+0.3,df6_edu3['proportion'][1::2],width=0.3, label='Percent Below Activity Guideline')
axes[1,0].set_xticks(df6_edu3['GEN_005'].unique()+0.3/2,['Excellent','Very Good','Good','Fair','Poor'])
axes[1,0].set_xlabel('Physical Activity By Health Status Uni Degree')
axes[1,0].set_ylabel('Percent')
axes[1,0].legend()
fig.delaxes(axes[1, 1])
plt.show()

```

Question 6 - How does stress influence the maintenance or improvement of mental health? For our 6th question I started by charting reported mental health by the respondent's stress level. After that ill break down the relationship by relevant demographic variables in the dataset along with some other visualizations of the data. Much like question 6 reported mental health is broken down into categories ranging from "poor" to "excellent". Reported stress level is broken down into 5 categories ranging from "not at all stressful" to "extremely stressful".

```

[ ]: df_question7=df_general.copy()
df_question7=df_question7.filter(items=[
    'GEN_015',
    'GEN_020',
    'HWTGISW',
    'PAADVACV',
    'DHHGAGE',
    'DHH_SEX',
    'GEOGPRV',
    'EHG2DVH3',
    'HWT_050'
])
#print(df_question7.head())

```

Like question 5 for data cleaning, I selected variables of interest and put them into a new data set for me to perform my analysis. I cleaned unwanted variables like "don't know" and "refused" as they are not of interest to my analysis. Lastly, I checked the dataset for missing values. For the initial graph plotting stress level by reported mental health status. As can be seen in the graph there is a distinct trend shown as respondents mental health decreases the percentage of the mental health level reporting higher stress levels grows. This trend shows the relationship that one could expect that stress does seem have be a determining factor or mental health.

```

[ ]: df_question7=df_question7.drop(df_question7[df_question7['GEN_015']==9].index)
df_question7=df_question7.drop(df_question7[df_question7['GEN_015']==8].index)
df_question7=df_question7.drop(df_question7[df_question7['GEN_015']==7].index)
df_question7=df_question7.drop(df_question7[df_question7['GEN_020']==8].index)
df_question7=df_question7.drop(df_question7[df_question7['GEN_020']==7].index)

```

```

df_question7=df_question7.drop(df_question7[df_question7['HWTGISW']==6].index)
df_question7=df_question7.drop(df_question7[df_question7['HWTGISW']==9].index)
df_question7=df_question7.drop(df_question7[df_question7['PAADVACV']==9].index)
df_question7=df_question7.drop(df_question7[df_question7['PAADVACV']==6].index)
df_question7=df_question7.drop(df_question7[df_question7['PAADVACV']==3].index)
df_question7=df_question7.drop(df_question7[df_question7['HWT_050']==6].index)
df_question7=df_question7.drop(df_question7[df_question7['HWT_050']==7].index)
df_question7=df_question7.drop(df_question7[df_question7['HWT_050']==8].index)
df_question7=df_question7.drop(df_question7[df_question7['HWT_050']==9].index)
df_question7=df_question7.drop(df_question7[df_question7['EHG2DVH3']==9].index)
df_question7.isnull().sum()

```

The correlation matrix with a heat map is included but it does not show any particular results of interest.

```

[ ]: cor7 = df_question7.corr()

labels7=['Mental Health', 'Stress Level', 'BMI', 'Physical Activity', 'Age', 'Sex', 'Location', 'Education', 'Self Perceived Weight']
sns.heatmap(cor7, annot=True, cmap='coolwarm', center=0, xticklabels=labels7, yticklabels=labels7)
plt.title("Correlation Between Stress and Mental Health Variables")
plt.show()

```

For the first breakdown by demographics, I have broken down the first graph by the age groups reported in the dataset. In each graph the general trend of higher stress levels being seen at the lower reported mental health states. However, as age increases the number of respondents in the extremes of the reported stress levels differs significantly with older age groups responding more in the extremes.

```

[ ]: df_healthstress=df_question7.groupby('GEN_015', as_index=False)[['GEN_020']].value_counts(normalize=True, sort=False)
plt.figure().set_figwidth(10)
plt.bar(df_healthstress['GEN_015'].unique()-0.2,df_healthstress['proportion'][::5],width=0.1,label='Not at all stressful')
plt.bar(df_healthstress['GEN_015'].unique()-0.1,df_healthstress['proportion'][1::5],width=0.1, label='Not very stressful')
plt.bar(df_healthstress['GEN_015'].unique(),df_healthstress['proportion'][2::5],width=0.1, label='A bit stressful')
plt.bar(df_healthstress['GEN_015'].unique()+0.1,df_healthstress['proportion'][3::5],width=0.1, label='Quite a bit stressful')
plt.bar(df_healthstress['GEN_015'].unique()+0.2,df_healthstress['proportion'][4::5],width=0.1, label='Extremely stressful')
plt.xticks(df_healthstress['GEN_015'].unique(),('Excellent','Very Good','Good','Fair','Poor'))
plt.xlabel('Stress Level by Mental Health Status')
plt.ylabel('Percent')

```

```

plt.legend()
plt.show()

```

For my last visualisation I plotted reported mental health by self perceived weight to see how one's self-perceived weight influences their mental health at all. In the graph there is a trend shown, as the respondents mental health decreases the percentage of the population in each mental health category reporting they view themselves as "just right" decreases. Likewise, the amount of each category's population reporting that they view themselves as "underweight" or "overweight" increases as reported mental health decreases.

```

[ ]: df_question7_age1=df_question7.drop(df_question7[df_question7['DHHGAGE']!=2].
                                         ↴index)
df7_health_age1=df_question7_age1.groupby('GEN_015', as_index=False)[['GEN_020']].
    ↴value_counts(normalize=True,sort=False)
df_question7_age2=df_question7.drop(df_question7[df_question7['DHHGAGE']!=3].
                                         ↴index)
df7_health_age2=df_question7_age2.groupby('GEN_015', as_index=False)[['GEN_020']].
    ↴value_counts(normalize=True,sort=False)
df_question7_age3=df_question7.drop(df_question7[df_question7['DHHGAGE']!=4].
                                         ↴index)
df7_health_age3=df_question7_age3.groupby('GEN_015', as_index=False)[['GEN_020']].
    ↴value_counts(normalize=True,sort=False)
df_question7_age4=df_question7.drop(df_question7[df_question7['DHHGAGE']!=5].
                                         ↴index)
df7_health_age4=df_question7_age4.groupby('GEN_015', as_index=False)[['GEN_020']].
    ↴value_counts(normalize=True,sort=False)

temp=pd.DataFrame({'GEN_015':5,'GEN_015':1,'proportion':0},index=[20])
df7_health_age1=pd.concat([df7_health_age1.iloc[:20], temp, df7_health_age1.
    ↴iloc[20:]],ignore_index=True)
df7_health_age1.iloc[20,0]=5
df7_health_age1.iloc[20,1]=1
df7_health_age1.iloc[20,2]=0

temp=pd.DataFrame({'GEN_015':5,'GEN_015':1,'proportion':0},index=[20])
df7_health_age2=pd.concat([df7_health_age2.iloc[:20], temp, df7_health_age2.
    ↴iloc[20:]],ignore_index=True)
df7_health_age2.iloc[20,0]=5
df7_health_age2.iloc[20,1]=1
df7_health_age2.iloc[20,2]=0

temp=pd.DataFrame({'GEN_015':5,'GEN_015':1,'proportion':0},index=[20])
df7_health_age4=pd.concat([df7_health_age4.iloc[:20], temp, df7_health_age4.
    ↴iloc[20:]],ignore_index=True)
df7_health_age4.iloc[20,0]=5
df7_health_age4.iloc[20,1]=1
df7_health_age4.iloc[20,2]=0

```

```

fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes[0,0].bar(df7_health_age1['GEN_015'].unique()-0.
    ↪2,df7_health_age1['proportion'][::5],width=0.1,label='Not at all stressful')
axes[0,0].bar(df7_health_age1['GEN_015'].unique()-0.
    ↪1,df7_health_age1['proportion'][1::5],width=0.1, label='Not very stressful')
axes[0,0].bar(df7_health_age1['GEN_015'].
    ↪unique(),df7_health_age1['proportion'][2::5],width=0.1, label='A bit ↪
    ↪stressful')
axes[0,0].bar(df7_health_age1['GEN_015'].unique()+0.
    ↪1,df7_health_age1['proportion'][3::5],width=0.1, label='Quite a bit ↪
    ↪stressful')
axes[0,0].bar(df7_health_age1['GEN_015'].unique()+0.
    ↪2,df7_health_age1['proportion'][4::5],width=0.1, label='Extremely stressful')
axes[0,0].set_xticks(df7_health_age1['GEN_015'].unique(),('Excellent','Very ↪
    ↪Good','Good','Fair','Poor'))
axes[0,0].set_xlabel('Stress Level by Mental Health Status 18-34')
axes[0,0].set_ylabel('Percent')
axes[0,0].legend()

axes[0,1].bar(df7_health_age2['GEN_015'].unique()-0.
    ↪2,df7_health_age2['proportion'][::5],width=0.1,label='Not at all stressful')
axes[0,1].bar(df7_health_age2['GEN_015'].unique()-0.
    ↪1,df7_health_age2['proportion'][1::5],width=0.1, label='Not very stressful')
axes[0,1].bar(df7_health_age2['GEN_015'].
    ↪unique(),df7_health_age2['proportion'][2::5],width=0.1, label='A bit ↪
    ↪stressful')
axes[0,1].bar(df7_health_age2['GEN_015'].unique()+0.
    ↪1,df7_health_age2['proportion'][3::5],width=0.1, label='Quite a bit ↪
    ↪stressful')
axes[0,1].bar(df7_health_age2['GEN_015'].unique()+0.
    ↪2,df7_health_age2['proportion'][4::5],width=0.1, label='Extremely stressful')
axes[0,1].set_xticks(df7_health_age2['GEN_015'].unique(),('Excellent','Very ↪
    ↪Good','Good','Fair','Poor'))
axes[0,1].set_xlabel('Stress Level by Mental Health Status 34-49')
axes[0,1].set_ylabel('Percent')
axes[0,1].legend()

axes[1,0].bar(df7_health_age3['GEN_015'].unique()-0.
    ↪2,df7_health_age3['proportion'][::5],width=0.1,label='Not at all stressful')
axes[1,0].bar(df7_health_age3['GEN_015'].unique()-0.
    ↪1,df7_health_age3['proportion'][1::5],width=0.1, label='Not very stressful')
axes[1,0].bar(df7_health_age3['GEN_015'].
    ↪unique(),df7_health_age3['proportion'][2::5],width=0.1, label='A bit ↪
    ↪stressful')

```

```

axes[1,0].bar(df7_health_age3['GEN_015'].unique()+0.
    ↪1,df7_health_age3['proportion'][3::5],width=0.1, label='Quite a bit ↪
    ↪stressful')
axes[1,0].bar(df7_health_age3['GEN_015'].unique()+0.
    ↪2,df7_health_age3['proportion'][4::5],width=0.1, label='Extremely stressful')
axes[1,0].set_xticks(df7_health_age3['GEN_015'].unique(),('Excellent','Very ↪
    ↪Good','Good','Fair','Poor'))
axes[1,0].set_xlabel('Stress Level by Mental Health Status 50-64')
axes[1,0].set_ylabel('Percent')
axes[1,0].legend()

axes[1,1].bar(df7_health_age4['GEN_015'].unique()-0.
    ↪2,df7_health_age4['proportion'][0::5],width=0.1,label='Not at all stressful')
axes[1,1].bar(df7_health_age4['GEN_015'].unique()-0.
    ↪1,df7_health_age4['proportion'][1::5],width=0.1, label='Not very stressful')
axes[1,1].bar(df7_health_age4['GEN_015'].
    ↪unique(),df7_health_age4['proportion'][2::5],width=0.1, label='A bit ↪
    ↪stressful')
axes[1,1].bar(df7_health_age4['GEN_015'].unique()+0.
    ↪1,df7_health_age4['proportion'][3::5],width=0.1, label='Quite a bit ↪
    ↪stressful')
axes[1,1].bar(df7_health_age4['GEN_015'].unique()+0.
    ↪2,df7_health_age4['proportion'][4::5],width=0.1, label='Extremely stressful')
axes[1,1].set_xticks(df7_health_age4['GEN_015'].unique(),('Excellent','Very ↪
    ↪Good','Good','Fair','Poor'))
axes[1,1].set_xlabel('Stress Level by Mental Health Status 65 plus')
axes[1,1].set_ylabel('Percent')
axes[1,1].legend()

plt.show()

```

0.4 Discussion

From our analysis of how alcohol consumption affected mental health among different age groups, we saw that individuals who moderately consume alcohol may report a better self-perceived mental health and life satisfaction. However, if alcohol consumption becomes excessive, especially in the case of frequent binge drinking, we can see a noticeable decrease in self-perceived mental health and life satisfaction. This trend is noticeable across all age groups to varying degrees. It should be noted that weak correlation between alcohol consumption and mental health variables indicates that there are additional variables that play a greater role in affecting these mental health variables.

Based on our investigation of how cannabis use impact people's perceived stress level, we found a weak trend that using cannabis will slightly increase perceived life stress and work stress. Due to the medicinal nature of cannabis, it can affect one's nerve system in some degree. It is difficult to say how strong that effect is, as we see people who used cannabis and feeling stressful or not at all stressful are evenly spread out on the graph. Therefore we conclude that using cannabis can slightly increase the stress level in both life and work aspect, but the increase is not obvious.

Overall we infer that smoking has a major impact on physical health upon our analysis against geography & gender, however overweight/obese doesn't have as strong of a relationship to smoking. People who stopped smoking show greater improvements in health levels. We can also infer that the BMI is greatly improved for the people with higher physical activity, and it declines for people with no physical activity. Hence we conclude that Smoking is directly related to Physical health in converse to BMI.

From our exploration of which demographic groups are mostly affected health barriers, we found that females tend to use cannabis more and smoke more frequently by approximately 8% compared to males. However, males consumed alcohol more often than females and binge drank more compared to females. When broken down by age, we found that a lot more teenagers (12 to 17 years) use cannabis than middle aged groups, then the trend shows an immediate decrease on 18 to 34 years group. After that we see a increasing trend on each age group until 65 years and older reaches the maximum cannabis use. This might be because some senior people need cannabis for medical treatment. In terms of smoking frequency, both 12 to 17 years and 65 years and older smoke most frequently, and other age groups smoke slightly less frequently than the former two age groups. When looking at different regions, there are almost equal amounts of cannabis use and smoking frequency across each province except territories. This may be due to a lower population in these territories. In terms of alcohol consumptions, our analysis revealed that people in Quebec and British Columbia consume alcohol most often, while people in territories consume more alcohol on one occasion compared to the other provinces.

When broken down to household education level, people with no secondary school education use more cannabis than secondary and post-secondary educated people, but the differences between these groups are not obvious. We found that people with post-secondary education smoke slightly more often than the other two groups. Our analysis revealed that people with post-secondary education drink alcohol most frequently, with the secondary education and no-secondary education group following closely behind. When looking at binge drinking, people with secondary and post-secondary education binge drank more compared to no-secondary education group.

From our investigation of how physical activity affects reported health we saw that there is a noticeable trend that people who reported they had better health were more likely to be above the recommended physical activity guideline. When broken down by age group we could see that that trend was stronger in older populations than younger populations. Similarly, our analysis of how stress affects reported mental health showed that once again respondents who reported mental health were more likely to report they had lower stress levels. While the trend continued to show ill all age groups older age groups appeared to be more likely to select the 2 extremes to not at all stressed or extremely stressed. Additionally, a respondent's self perception of their weight did seem to relate to their mental health. To conclude these two questions, we have shown that there is at least a relationship between these factors and how a respondent reported their general and mental health state.

Some limitations of our results are that they only utilized visual inspection and correlation statistics. While correlation statistics are useful at identifying a relationship between variables, we cannot guarantee causation. The combination of visual inspection with correlation statistics should give us a better idea of this relationship, we still cannot guarantee that to be the case. To better investigate these relationships, future testing could see the introduction of regression models, most likely of the ordinal variety, as well as interaction terms.

0.5 Conclusion

Based on the results of our analysis we saw that alcohol consumption and smoking both resulted in worsening some key variables related to quality of life, those being mental and physical health. Although the strengths of these relationships varied, we can still suggest that reducing the frequency of drinking and smoking is likely to lead to an improved quality of life.

We found that cannabis use showed a weak trend indicating that cannabis slightly increases perceived life and work stress, potentially due to its effects on the nervous system.

While females tend to use cannabis and smoke more frequently than males, males more frequently consume alcohol and binge drink. Teenagers exhibit higher cannabis use, declining in older age groups, except for seniors (65+), where we see an increase, possibly due to medical use. Smoking and alcohol consumption also varied by region, with Quebec and British Columbia leading in alcohol consumption frequency, while territories showed a higher frequency of binge drinking.

We can also see that regular physical activity is related to improved self-reported physical and mental health, especially in an older population. Therefore, encouraging regular exercise, particularly in an older population should lead to an improved quality of life.

To summarize, although the exact strength of our relationships is unknown, it is likely that promoting reductions in the frequency of alcohol and smoking, in addition to encouraging physical activity, may contribute to an increase in mental and physical health, and therefore improve ones quality of life.

0.6 Participant Contributions

Participants contributed in the following way:

- Aaron Gelfand: in charge of answering question 1, led team meetings, developed templates for submissions, and collaborated members work into final document.
- David Griffin: in charge of answering questions 5 and 6, provided essential code for certain visualizations.
- David Li: in charge of answering question 2 and 4, provided valuable opinions during meetings on guiding questions.
- Venkateshwaran Balu Soundararajan: in charge of answering question 3, provided insight on multiple fields during team meetings, including data wrangling and question development.

0.7 References

1. Canadian Community Health Survey – Annual Component (CCHS). Government of Canada, Statistics Canada. (2023, December 29). <https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=1531795>
2. Canadian Community Health Survey: Public Use Microdata File. Government of Canada, Statistics Canada (2024, September 22). <https://www150.statcan.gc.ca/n1/en/catalogue/82M0013x>
3. Haskell, W. L., Lee, I. M., Pate, R. R., Powell, K. E., Benjamin, G. A., & Flegal, K. M. (2007). Physical activity and public health: Updated recommendation for adults from the

- American College of Sports Medicine and the American Heart Association. Circulation, 116(9), 1081-1093.
4. Rehm, J., Taylor, B., & Room, R. (2006). Global burden of disease from alcohol, illicit drugs and tobacco. Drug and Alcohol Review, 25(6), 503-513.

R Notebook

Introduction

Wildfires in Canada are a significant threat to ecosystems, human safety, and property, necessitating a comprehensive understanding of the factors that influence their behavior. Although the prevalence of forest fires varies year to year, there has been an observed trend over the past years indicating an overall increase in the occurrence of forest fires. There is also evidence of an increase in severe forest fires in recent years, compared to the last few decades¹.

Part of our motivation for our questions is aimed towards looking at predicting the size and spread rate of fires. The 2023 and 2024 fire seasons have been some of the worst on record in terms of area burned. Being able to predict wildfire sizes may help in allocations of resources to fight fires. As instances such as Fort McMurray and Jasper become more common with populated areas finding themselves in the way of these fires understanding spread rates of fires may be important to inform evacuation decisions and keep more people safe.

The main objective of our project is to examine multiple variables related to forest fires, to determine if they reduce or increase the severity of forest fires. Based on this information we can make suggestions to help decrease the increasing threats of forest fires.

Objective Questions

To answer our main objective, we examine four main questions:

1. Explore the relationship between wind speed and fire spread rate. Understanding this relationship can inform firefighting strategies and preparedness efforts.
2. Examine the relationship between temperature and fire spread rate. Analyzing this relationship will provide insights into how varying temperature levels affect wildfire behavior.
3. Examine the relationship between temperature and fire spread rate. Analyzing this relationship will provide insights into how varying temperature levels affect wildfire behavior.
4. Examine whether fire size relates to the weather conditions when the fire starts.

Before any analyses, we want to make sure to load all appropriate libraries, as well as the data set.

```
library(mosaic)

## Registered S3 method overwritten by 'mosaic':
##   method           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features. The original behavior of these functions should not be affected by this.
```

```

## 
## Attaching package: 'mosaic'

## The following objects are masked from 'package:dplyr':
## 
##     count, do, tally

## The following object is masked from 'package:Matrix':
## 
##     mean

## The following object is masked from 'package:ggplot2':
## 
##     stat

## The following objects are masked from 'package:stats':
## 
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
## 
##     max, mean, min, prod, range, sample, sum

library(dplyr)
library(ggplot2)
library(e1071)
library(leaflet)

```

```

fire_data<-read.csv("https://raw.githubusercontent.com/aaronngelf/data602-data/refs/heads/main/fp-historical-wildfire-data.csv")
#We have also include the dataset in our submission, in case there is an error accessing the URL.

```

Question 1 - What is the Relationship Between Wind Speed and Fire Spread Rate

To examine the relationship between wind speed and fire spread rate, we can perform a few visual functions, to get an idea of the dataset itself. It is also important to note that data dictionary provided should be used to help interpret the columns based on their names, as well as what the values represent. The data dictionary can be found at <https://open.alberta.ca/dataset/a221e7a0-4f46-4be7-9c5a-e29de9a3447e/resource/1b635b8b-a937-4be4-857e-8aeeff77365d2/download/fp-historical-wildfire-data-dictionary-2006-2023.pdf>.

```

head(fire_data)

##   fire_year fire_number fire_name current_size size_class
## 1      2006       PWF001      <NA>        0.10         A
## 2      2006       EWF002      <NA>        0.20         B
## 3      2006       EWF001      <NA>        0.50         B
## 4      2006       EWF003      <NA>        0.01         A
## 5      2006       PWF002      <NA>        0.10         A
## 6      2006       CWF001      <NA>        0.20         B
##   fire_location_latitude fire_location_longitude   fire_origin

```

```

## 1      56.24996      -117.1820      Private Land
## 2      53.60637      -115.9157      Provincial Land
## 3      53.61093      -115.5943      Provincial Land
## 4      53.60887      -115.6095      Provincial Land
## 5      56.24996      -117.0502      Provincial Land
## 6      51.15293      -115.0346      Indian Reservation
##   general_cause_desc industry_identifier_desc      responsible_group_desc
## 1      Resident          <NA>             Resident
## 2      Incendiary        <NA>            Others (explain in remarks)
## 3      Incendiary        <NA>            Others (explain in remarks)
## 4      Incendiary        <NA>            Others (explain in remarks)
## 5      Other Industry    Waste Disposal     Employees
## 6      Resident          <NA>             Resident
##   activity_class      true_cause  fire_start_date det_agent_type det_agent
## 1      Grass   Permit Related 2006-04-02 12:00      UNP      310
## 2      Lighting Fires Arson Suspected 2006-04-03 12:10      UNP      310
## 3      Lighting Fires Arson Suspected 2006-04-03 12:15      UNP      310
## 4      Lighting Fires Arson Suspected 2006-04-03 12:10      UNP      PUB
## 5      Refuse   Permit Related 2006-04-03 17:00      UNP      LFS
## 6      Unclassified      Unsafe Fire 2006-04-02 14:25      UNP      310
##   discovered_date discovered_size      reported_date dispatched_resource
## 1          <NA>           NA 2006-04-02 20:46      FPD Staff
## 2          <NA>           NA 2006-04-03 12:27      FPD Staff
## 3          <NA>           NA 2006-04-03 12:36      FPD Staff
## 4          <NA>           NA 2006-04-03 13:23      FPD Staff
## 5 2006-04-03 19:11          NA 2006-04-03 19:12      FPD Staff
## 6 2006-04-02 14:27          NA 2006-04-02 14:30      FPD Staff
##   dispatch_date start_for_fire_date assessment_resource assessment_datetime
## 1 2006-04-02 21:10      2006-04-02 21:20      IA Forces 2006-04-02 22:00
## 2 2006-04-03 12:33      2006-04-03 12:35      IA Forces 2006-04-03 13:20
## 3 2006-04-03 12:36      2006-04-03 12:42      IA Forces 2006-04-03 13:23
## 4 2006-04-03 13:50      2006-04-03 13:50      IA Forces 2006-04-03 14:08
## 5 2006-04-03 19:19      2006-04-03 19:22      Other   2006-04-03 19:57
## 6 2006-04-02 14:40      2006-04-02 14:45      IA Forces 2006-04-02 16:00
##   assessment_hectares fire_spread_rate fire_type fire_position_on_slope
## 1          0.01          0.0 Surface      Flat
## 2          0.20          0.0 Surface      Lower 1/3
## 3          0.50          0.0 Surface      Bottom
## 4          0.01          0.0 Surface      Flat
## 5          0.10          0.1 Surface      Flat
## 6          0.20          0.0 Surface      Flat
##   weather_conditions_over_fire temperature relative_humidity wind_direction
## 1          Clear          18          10      SW
## 2          Clear          12          22      SW
## 3          Clear          12          22      SW
## 4          Clear          12          22      SW
## 5          Clear          6           37      SW
## 6          Clear          11          32      S
##   wind_speed fuel_type initial_action_by ia_arrival_at_fire_date ia_access
## 1          2       01a Land Owner      <NA>      <NA>
## 2          10      01a Fire Department  <NA>      <NA>
## 3          10      01a Fire Department  <NA>      <NA>
## 4          10      01b Industry      <NA>      <NA>
## 5          2       <NA> Fire Department  <NA>      <NA>

```

```

## 6      20      01b   Fire Department          <NA>      <NA>
##   fire_fighting_start_date fire_fighting_start_size bucketing_on_fire
## 1             <NA>           NA           <NA>
## 2             <NA>           NA           <NA>
## 3             <NA>           NA           <NA>
## 4             <NA>           NA           <NA>
## 5             <NA>           NA           <NA>
## 6             <NA>           NA           <NA>
##   distance_from_water_source first_bucket_drop_date     bh_fs_date
## 1                   NA           <NA> 2006-04-02 22:00
## 2                   NA           <NA> 2006-04-03 13:20
## 3                   NA           <NA> 2006-04-03 13:23
## 4                   NA           <NA> 2006-04-03 14:08
## 5                   NA           <NA> 2006-04-03 19:57
## 6                   NA           <NA> 2006-04-02 16:00
##   bh_hectares      uc_fs_date uc_hectares      to_fs_date to_hectares
## 1    0.01 2006-04-02 22:00    0.01        <NA>       NA
## 2    0.20 2006-04-03 13:20    0.20        <NA>       NA
## 3    0.50 2006-04-03 13:23    0.50        <NA>       NA
## 4    0.01 2006-04-03 14:08    0.01        <NA>       NA
## 5    0.10 2006-04-03 20:19    0.10 2006-04-03 20:20    0.1
## 6    0.20 2006-04-02 16:00    0.20        <NA>       NA
##   ex_fs_date ex_hectares
## 1 2006-04-03 10:20    0.10
## 2 2006-04-03 14:00    0.20
## 3 2006-04-03 15:00    0.50
## 4 2006-04-03 15:05    0.01
## 5 2006-04-05 10:18    0.10
## 6 2006-04-03 18:00    0.20

```

```
colnames(fire_data)
```

## [1] "fire_year"	"fire_number"
## [3] "fire_name"	"current_size"
## [5] "size_class"	"fire_location_latitude"
## [7] "fire_location_longitude"	"fire_origin"
## [9] "general_cause_desc"	"industry_identifier_desc"
## [11] "responsible_group_desc"	"activity_class"
## [13] "true_cause"	"fire_start_date"
## [15] "det_agent_type"	"det_agent"
## [17] "discovered_date"	"discovered_size"
## [19] "reported_date"	"dispatched_resource"
## [21] "dispatch_date"	"start_for_fire_date"
## [23] "assessment_resource"	"assessment_datetime"
## [25] "assessment_hectares"	"fire_spread_rate"
## [27] "fire_type"	"fire_position_on_slope"
## [29] "weather_conditions_over_fire"	"temperature"
## [31] "relative_humidity"	"wind_direction"
## [33] "wind_speed"	"fuel_type"
## [35] "initial_action_by"	"ia_arrival_at_fire_date"
## [37] "ia_access"	"fire_fighting_start_date"
## [39] "fire_fighting_start_size"	"bucketing_on_fire"
## [41] "distance_from_water_source"	"first_bucket_drop_date"
## [43] "bh_fs_date"	"bh_hectares"

```

## [45] "uc_fs_date"                      "uc_hectares"
## [47] "to_fs_date"                       "to_hectares"
## [49] "ex_fs_date"                       "ex_hectares"

```

Based on initial inspection, we are only interested in a few columns, therefore we will create a new data frame, focusing on variables that will help examine the relationship between wind speed and fire spread rate.

```

fire_data_1=fire_data %>%
  select(wind_speed,fire_spread_rate)
sum(is.na(fire_data_1))

```

```

## [1] 5575

```

```

sum(sapply(fire_data_1[c("fire_spread_rate","wind_speed")],is.na))

```

```

## [1] 5575

```

#Remove rows with NA values. We won't include fuel_type for this part, as we are not initially concerned with fuel type.

```

fire_clean=na.omit(fire_data_1[,c("wind_speed","fire_spread_rate")])
sum(is.na(fire_clean))

```

```

## [1] 0

```

```

summary(fire_clean)

```

```

##      wind_speed    fire_spread_rate
##  Min.   : 0.000   Min.   :-1.0000
##  1st Qu.: 3.000   1st Qu.: 0.0000
##  Median : 6.000   Median : 0.0000
##  Mean   : 8.813   Mean   : 0.8962
##  3rd Qu.:12.000   3rd Qu.: 1.0000
##  Max.   :90.000   Max.   :100.0000

```

Based on our summary statistics, we can see that the minimum value for fire_spread_rate is -1. This seems peculiar, and warrants a bit of further investigation, therefore we will go back to the original data set and inspect any rows where fire_spread_rate is -1.

```

negative_fire_spread_data <- fire_data[!is.na(fire_data$fire_spread_rate) & fire_data$fire_spread_rate == -1]
head(negative_fire_spread_data)

```

	fire_year	fire_number	fire_name	current_size	size_class	fire_location_latitude	fire_location_longitude	fire_origin
##	12962	2014	RWF022	<NA>	0.20	B		
##	13717	2015	HWF100	<NA>	0.02	A		
##	18213	2017	MWF091	<NA>	0.10	A		
##	20474	2020	CWF038	<NA>	0.01	A		
##	20778	2019	SWF063	<NA>	0.01	A		
##	21355	2019	SWF092	<NA>	0.04	A		
##								
##								
##								

```

## 18213      56.83563      -111.7322 Provincial Land
## 20474      51.10135      -115.3091 Provincial Land
## 20778      55.94107      -113.7807 Indian Reservation
## 21355      56.79307      -114.7125 Provincial Land
## general_cause_desc industry_identifier_desc responsible_group_desc
## 12962      Undetermined <NA> <NA>
## 13717      Incendiary <NA> <NA>
## 18213      Recreation <NA> Others (explain in remarks)
## 20474      Recreation <NA> Campers
## 20778      Incendiary <NA> <NA>
## 21355      Lightning <NA> <NA>
## activity_class true_cause fire_start_date det_agent_type
## 12962      <NA> <NA> 2014-05-24 4:00 LKT
## 13717      Unclassified <NA> 2015-05-18 10:48 LKT
## 18213      OHV Operation Burning Substance 2017-09-16 16:38 AIR
## 20474 Cooking and Warming Unsafe Fire 2020-06-27 18:00 UNP
## 20778      Arson <NA> 2019-05-22 7:00 UNP
## 21355      <NA> <NA> 2019-06-02 15:40 UNP
## det_agent discovered_date discovered_size reported_date
## 12962      RA 2014-05-24 7:28 NA 2014-05-24 7:32
## 13717      FG 2015-05-18 10:48 NA 2015-05-18 10:50
## 18213      HAC 2017-09-16 16:45 NA 2017-09-16 16:45
## 20474      310 <NA> NA 2020-06-28 10:09
## 20778      LFS <NA> NA 2019-05-22 7:14
## 21355      PUB 2019-06-02 15:50 NA 2019-06-02 15:50
## dispatched_resource dispatch_date start_for_fire_date
## 12962      HAC 2014-05-24 8:14 2014-05-24 8:29
## 13717      FPD Staff 2015-05-18 11:00 2015-05-18 11:00
## 18213      HAC 2017-09-16 16:45 2017-09-16 16:45
## 20474      HAC 2020-06-28 14:30 2020-06-28 15:15
## 20778      HAC 2019-05-22 9:27 2019-05-22 9:49
## 21355      HAC 2019-06-02 15:59 2019-06-02 16:00
## assessment_resource assessment_datetime assessment_hectares
## 12962      IA Forces 2014-05-24 10:35 0.20
## 13717      Other 2015-05-18 11:16 0.02
## 18213      IA Forces 2017-09-16 16:45 0.10
## 20474      IA Forces 2020-06-28 18:00 0.01
## 20778      IA Forces 2019-05-22 10:06 0.01
## 21355      IA Forces 2019-06-02 16:05 0.04
## fire_spread_rate fire_type fire_position_on_slope
## 12962      -1 Ground Flat
## 13717      -1 Surface Flat
## 18213      -1 Surface Bottom
## 20474      -1 Surface Bottom
## 20778      -1 Ground Flat
## 21355      -1 Surface Flat
## weather_conditions_over_fire temperature relative_humidity wind_direction
## 12962      Rainshowers 10.5 73 N
## 13717      Clear 20.0 22 SE
## 18213      Clear 15.6 32 S
## 20474      Cloudy 11.0 75 W
## 20778      Clear 17.0 30 E
## 21355      Cloudy 18.0 41 W
## wind_speed fuel_type initial_action_by ia_arrival_at_fire_date

```

```

## 12962      1      S1      Industry          <NA>
## 13717     10     D1      FPD Staff  2015-05-18 11:14
## 18213     20     01b      HAC   2017-09-16 16:51
## 20474      2     <NA>      HAC   2020-06-28 17:53
## 20778      5      M2      HAC   2019-05-22 10:06
## 21355     15     C2      HAC   2019-06-02 16:05
##           ia_access fire_fighting_start_date fire_fighting_start_size
## 12962            <NA>                  <NA>             NA
## 13717            <NA>    2015-05-18 11:16          0.02
## 18213 Conventional R/W 2017-09-16 16:57          0.10
## 20474      Ground 2020-06-28 18:00          0.01
## 20778      Ground 2019-05-22 10:13          0.01
## 21355 Conventional R/W 2019-06-02 16:10          1.00
##           bucketing_on_fire distance_from_water_source first_bucket_drop_date
## 12962            <NA>                      NA          <NA>
## 13717            Y                      0.2 2015-05-18 11:16
## 18213            N                      NA          <NA>
## 20474            N                      NA          <NA>
## 20778            N                      NA          <NA>
## 21355            Y                      0.1 2019-06-02 16:25
##           bh_fs_date bh_hectares      uc_fs_date uc_hectares to_fs_date
## 12962 2014-05-24 10:35        0.20 2014-05-24 11:40        0.20      <NA>
## 13717 2015-05-18 11:16        0.02 2015-05-18 11:50        0.02      <NA>
## 18213 2017-09-16 17:09        0.10 2017-09-16 17:55        0.20      <NA>
## 20474 2020-06-28 18:00        0.01 2020-06-28 18:00        0.01      <NA>
## 20778 2019-05-22 10:06        0.01 2019-05-22 10:06        0.01      <NA>
## 21355 2019-06-02 16:05        0.04 2019-06-02 19:21        0.04      <NA>
##           to_hectares ex_fs_date ex_hectares
## 12962            NA 2014-05-24 14:31        0.20
## 13717            NA 2015-05-18 13:00        0.02
## 18213            NA 2017-09-17 10:50        0.10
## 20474            NA 2020-06-28 19:56        0.01
## 20778            NA 2019-05-22 10:25        0.01
## 21355            NA 2019-06-02 19:50        0.04

```

Upon visual inspection there does not seem to be any pattern related to the fire_spread_rate being -1. Based on this, and the definition provided in the data dictionary, with fire_spread_rate being ‘The rate of spread of the wildfire at the time of initial assessment, capture in metres per minute’, we felt it was safe to remove these rows, as this is most likely an error with these entries. For the fire to have a negative spread rate, would mean that the fire is retreating instead of spreading, and given that this rate of spread is a measure of how fast the fire moves from a point of origin, this seems counter intuitive to how forest fires work. Given more time, we could reach out to the providers of the data, to try to clarify this area, but for the time being, and since there are only 6 data points, we will remove them.

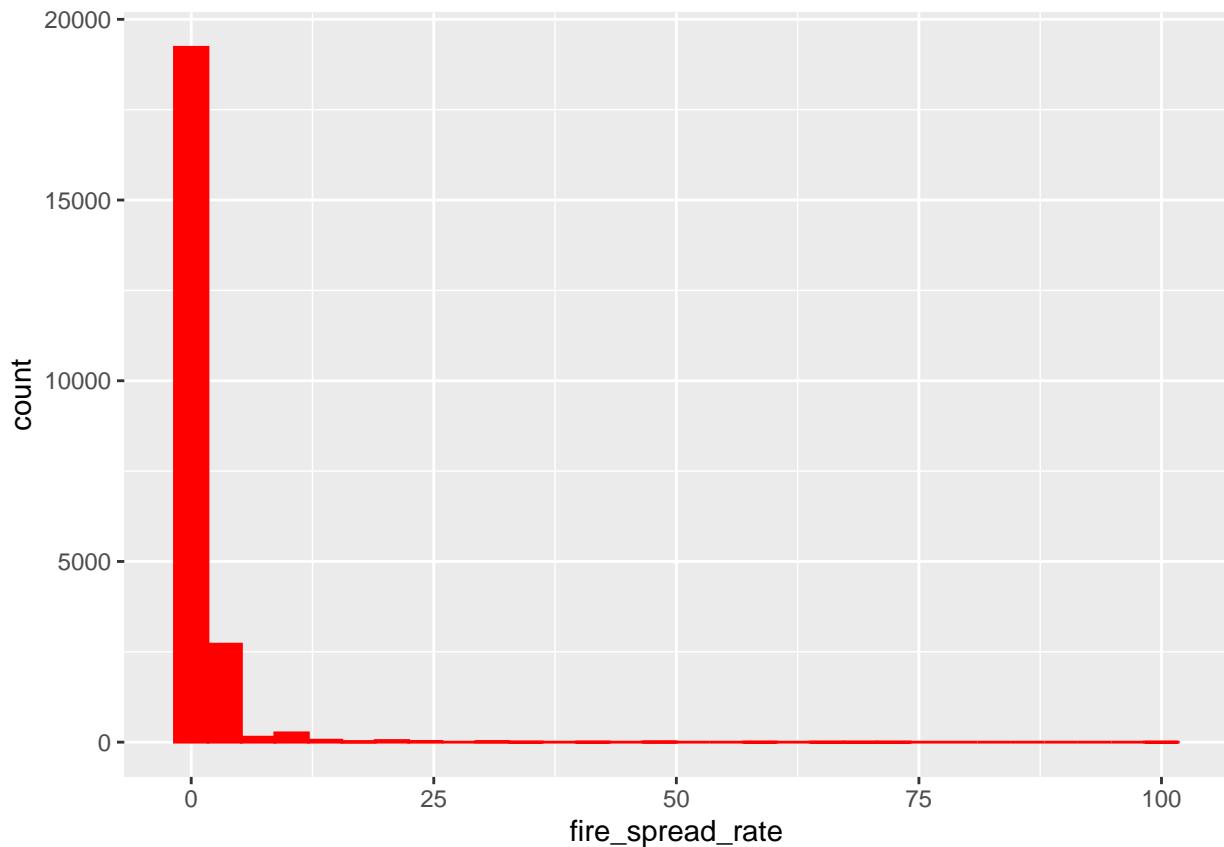
After we remove the rows with a fire spread rate of -1, we can plot the data for a preliminary visualization.

```

fire_clean_no_neg=fire_clean[fire_clean['fire_spread_rate']>=0,]
ggplot(fire_clean_no_neg, aes(x = fire_spread_rate)) +
  geom_histogram(color='red',fill='red')

```

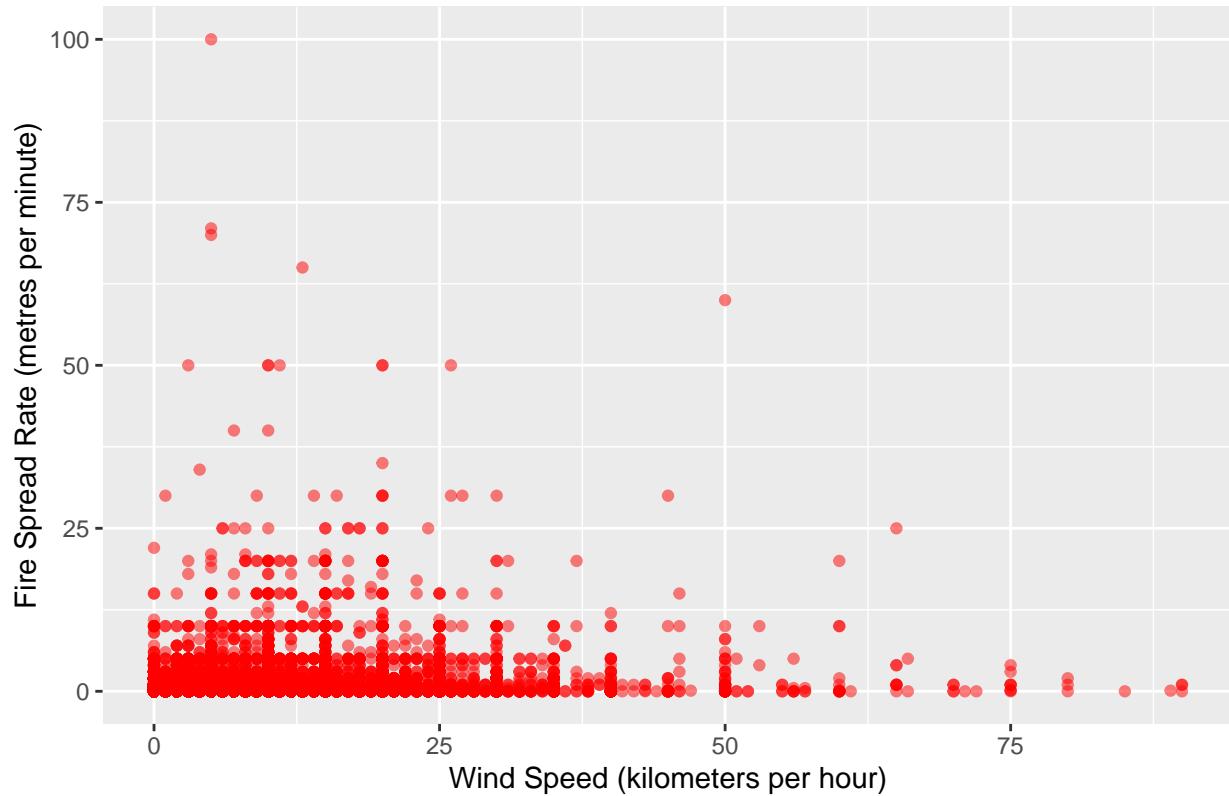
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Based on our histogram, we can see that the data for fire_spread_rate is heavily skewed.

```
fire_clean_no_neg=fire_clean[fire_clean['fire_spread_rate']>=0,]
ggplot(fire_clean_no_neg, aes(x = wind_speed, y = fire_spread_rate)) +
  geom_point(color='red',alpha = 0.5) +
  labs(title = "Relationship between Fire Spread Rate and Wind Speed",
       x = "Wind Speed (kilometers per hour)", # Replace with actual units if known
       y = "Fire Spread Rate (metres per minute)")
```

Relationship between Fire Spread Rate and Wind Speed



Initial inspection of the scatterplot is difficult to arrive to any meaningful conclusion without further analysis.

Additionally, we will look at the correlation coefficient between fire_spread_rate and wind_speed.

```
fire_corr=cor(fire_clean_no_neg,use="pairwise.complete.obs")
print(fire_corr)
```

```
##                  wind_speed fire_spread_rate
## wind_speed      1.0000000   0.1346716
## fire_spread_rate 0.1346716   1.0000000
```

Based on our output we can see a very weak positive relationship between fire spread rate and wind speed.

```
fire_no_neg_model=lm(fire_spread_rate ~ wind_speed, data = fire_clean_no_neg)

summary(fire_no_neg_model)
```

```
##
## Call:
## lm(formula = fire_spread_rate ~ wind_speed, data = fire_clean_no_neg)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.263 -0.863 -0.597  0.054 99.261
##
```

```

## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.531266  0.024815 21.41   <2e-16 ***
## wind_speed  0.041468  0.002035 20.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.571 on 22478 degrees of freedom
## Multiple R-squared:  0.01814,    Adjusted R-squared:  0.01809
## F-statistic: 415.2 on 1 and 22478 DF,  p-value: < 2.2e-16

```

Intercept: The expected value of fire_spread_rate when wind_speed is zero is 0.53130. As our p-value is <0.05 , this indicates that we can reject the H_0 that $\beta_0=0$. Therefore, we accept the H_1 that $\beta_0 \neq 0$ and conclude that the intercept is statistically significant.

Slope: For each additional kilometer per hour in wind_speed, the fire_spread_rate is expected to increase by approximately 0.04150 meters per minute. As our p-value is <0.05 , this indicates that we can reject the H_0 that $\beta_1=0$. Therefore, we accept the H_1 that $\beta_1 \neq 0$ and conclude that there is a significant relationship between wind_speed and fire_spread_rate.

Based on our output table, the equation for our model can be written out as, $fire_spread_rate = 0.53130 + (0.04150 * wind_speed)$

Our R-squared value indicates that approximately 1.81% of the variance in fire_spread_rate is explained by wind_speed. This low value suggests that there are other factors affecting fire spread that are not included in our model.

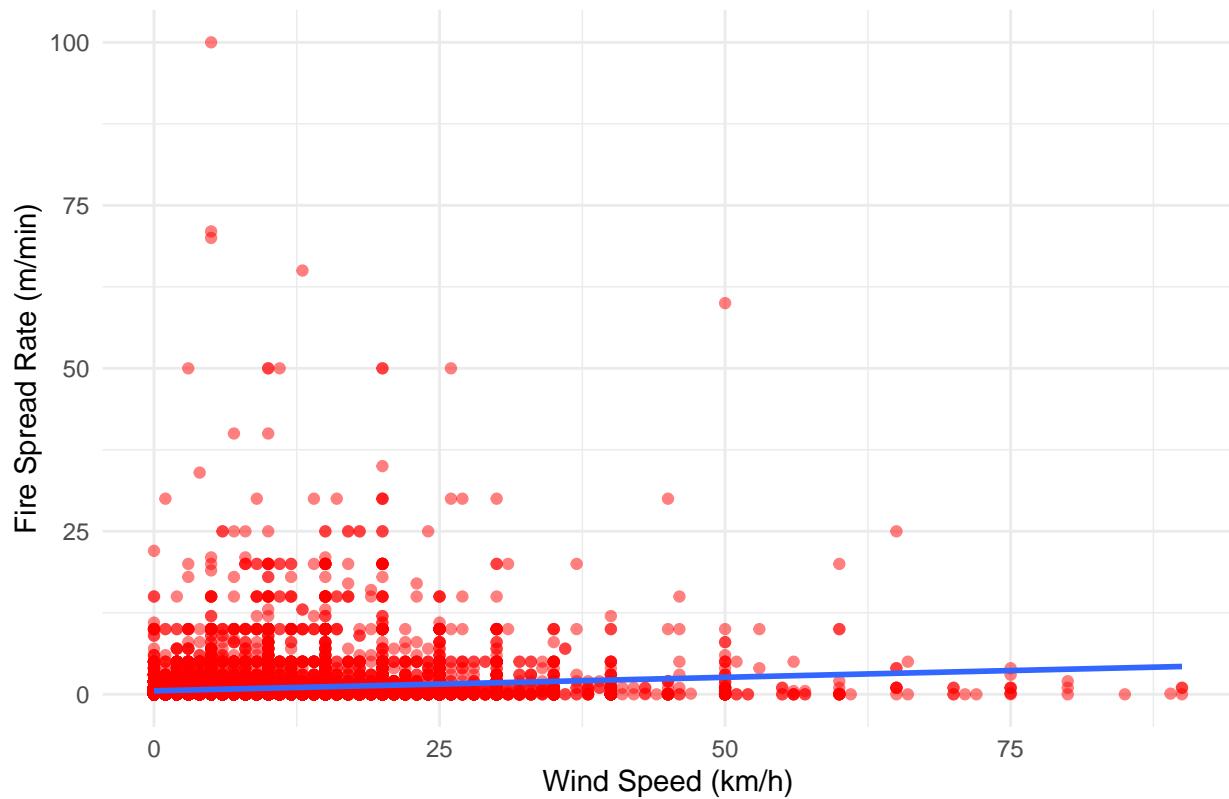
We can plot this model using the following code:

```

ggplot(fire_clean_no_neg, aes(x = wind_speed, y = fire_spread_rate)) +
  geom_point(alpha = 0.5,color='red') +
  stat_smooth(method = "lm", formula = y~x) + # Add regression line
  labs(title = "Relationship between Fire Spread Rate and Wind Speed",
       x = "Wind Speed (km/h)",
       y = "Fire Spread Rate (m/min)") +
  theme_minimal()

```

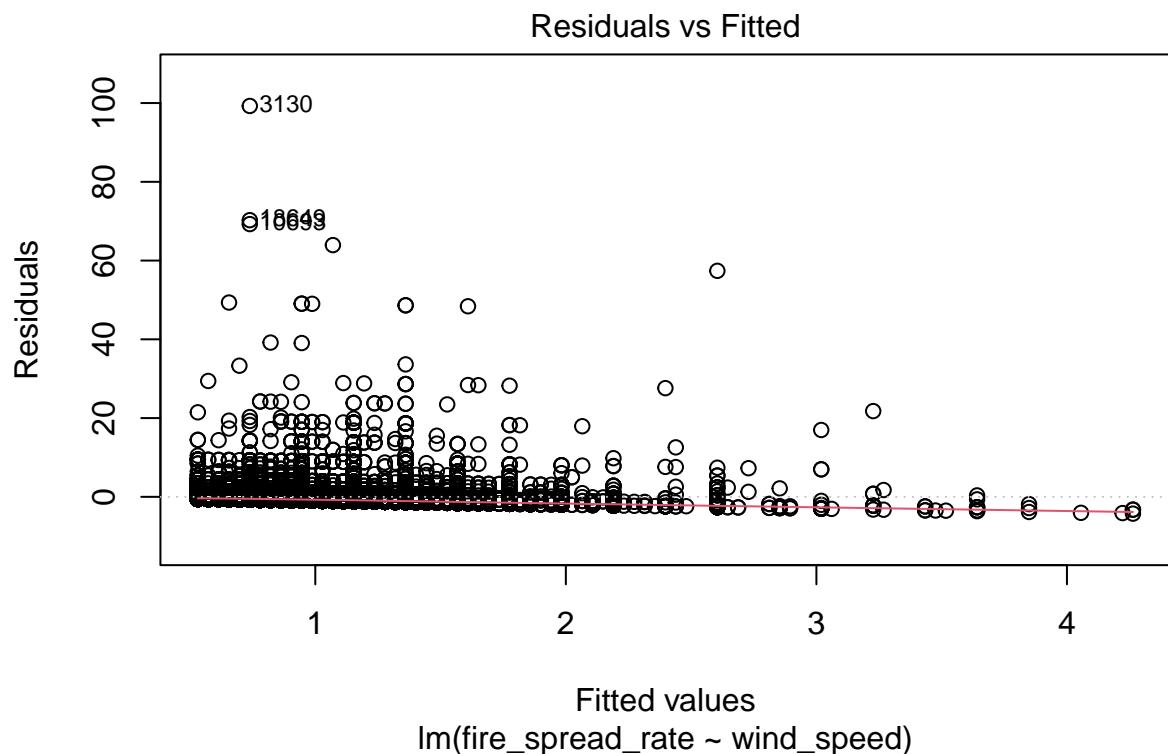
Relationship between Fire Spread Rate and Wind Speed



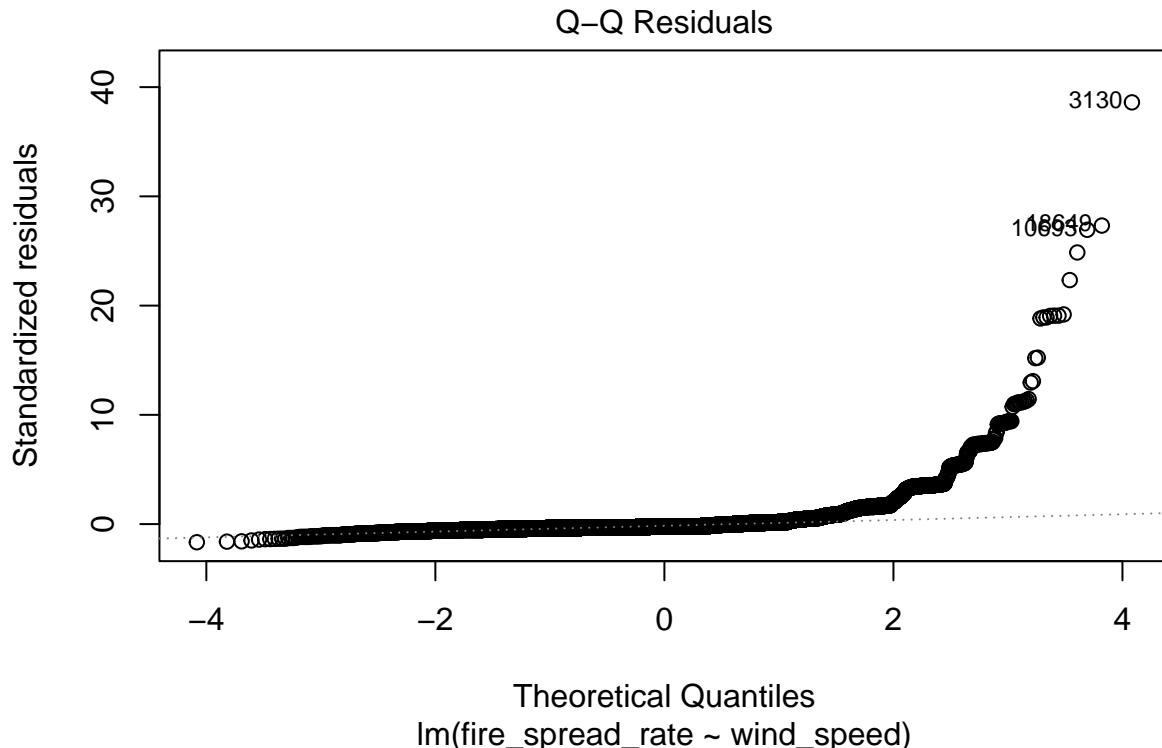
From our output, we can see a weak relationship between wind_speed and fire_spread_rate. as suggested by our correlation statistics.

To determine whether the assumptions of independency and normality are met, we can plot the residual and QQ plots.

```
#Residual plot  
plot(fire_no_neg_model, which =1)
```



```
#QQ-plot
plot(fire_no_neg_model, which =2)
```



For our residual plot we see a large cluster of points on the upper left side of the figure, as well as some outliers. For our QQ plot, we see that the points stray off far from the line towards the right side of the figure. Based on these observation, we can suggest that both assumptions of independency and normality of residuals fails.

By failing both of these assumptions, it suggests issues in our model that can lead to unreliable results. Some potential solutions are to transform the data, or to include interaction terms.

Question 2 - What is the Relationship Between Temperature and Fire Spread Rate

In this project, we explored the relationship between temperature and fire spread rate in Canada. We visualized the distribution of fire spread rate and temperature, calculated the correlation coefficient, performed a linear regression analysis, and conducted a hypothesis test to determine whether the observed relationship is statistically significant. Additionally, we created a geospatial representation of fire spread rate along with temperature to better understand the spatial patterns and relationships between these variables.

The results of the hypothesis test indicate whether there is a statistically significant relationship between temperature and fire spread rate.

```
data=fire_data

# Filter out rows where Fire Spread Rate is negative
data<- data %>%
  filter(fire_spread_rate >= 0)
# Create a scatter plot
ggplot(data, aes(x = temperature, y = fire_spread_rate)) +
  geom_point(color = "red", size = 2) +
```

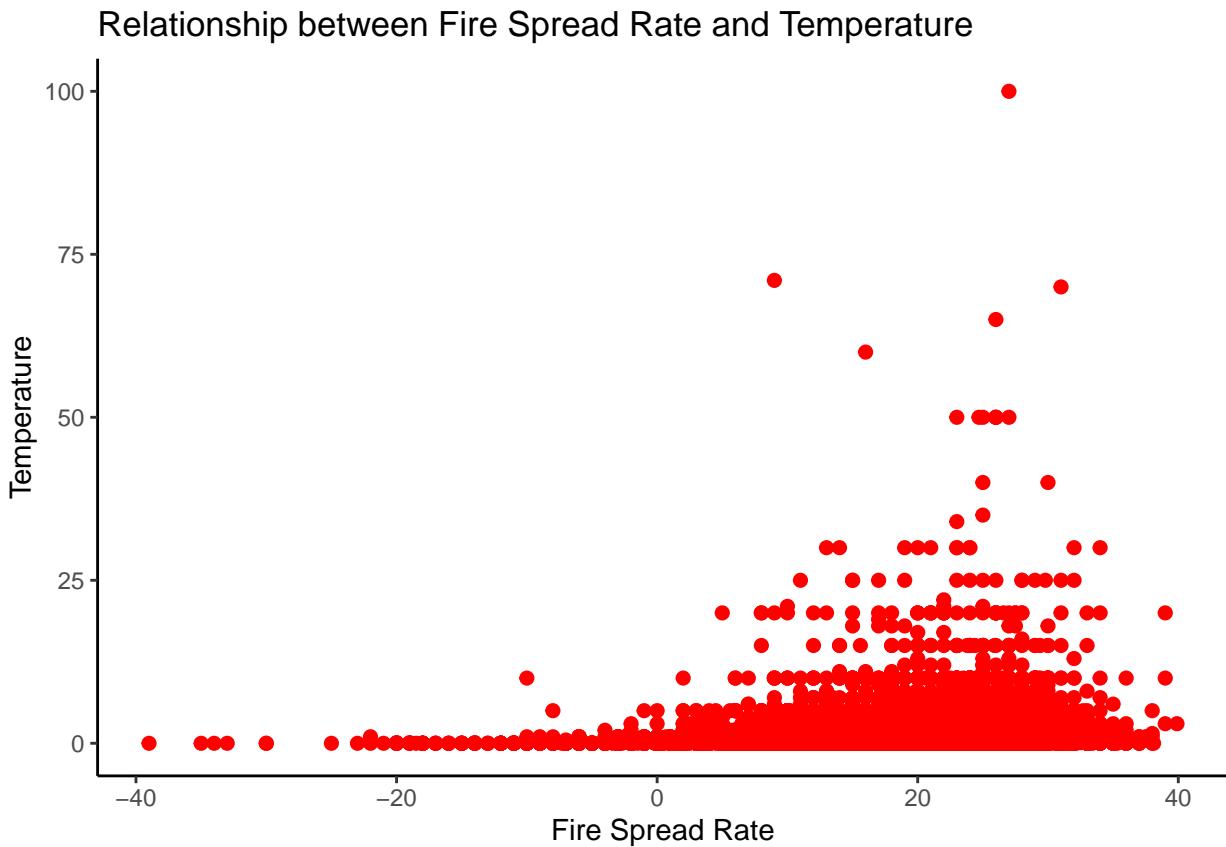
```

#geom_smooth(method = "lm", se = FALSE, color = "red")

labs(title = "Relationship between Fire Spread Rate and Temperature",
     x = "Fire Spread Rate",
     y = "Temperature") +
theme_classic()

## Warning: Removed 80 rows containing missing values or values outside the scale range
## ('geom_point()').

```



EDA

```

# Check the structure of the data
str(data)

## 'data.frame': 22562 obs. of 50 variables:
## $ fire_year          : int  2006 2006 2006 2006 2006 2006 2006 2006 2006 2006 ...
## $ fire_number        : chr  "PWF001" "EWF002" "EWF001" "EWF003" ...
## $ fire_name          : chr  NA NA NA NA ...
## $ current_size       : num  0.1 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ size_class          : chr  "A" "B" "B" "A" ...
## $ fire_location_latitude: num  56.2 53.6 53.6 53.6 56.2 ...

```

```

## $ fire_location_longitude : num -117 -116 -116 -116 -117 ...
## $ fire_origin : chr "Private Land" "Provincial Land" "Provincial Land" "Provincial ...
## $ general_cause_desc : chr "Resident" "Incendiary" "Incendiary" "Incendiary" ...
## $ industry_identifier_desc : chr NA NA NA NA ...
## $ responsible_group_desc : chr "Resident" "Others (explain in remarks)" "Others (explain in r ...
## $ activity_class : chr "Grass" "Lighting Fires" "Lighting Fires" "Lighting Fires" ...
## $ true_cause : chr "Permit Related" "Arson Suspected" "Arson Suspected" "Arson Su ...
## $ fire_start_date : chr "2006-04-02 12:00" "2006-04-03 12:10" "2006-04-03 12:15" "2006- ...
## $ det_agent_type : chr "UNP" "UNP" "UNP" "UNP" ...
## $ det_agent : chr "310" "310" "310" "PUB" ...
## $ discovered_date : chr NA NA NA NA ...
## $ discovered_size : num NA NA NA NA NA NA NA NA NA ...
## $ reported_date : chr "2006-04-02 20:46" "2006-04-03 12:27" "2006-04-03 12:36" "2006- ...
## $ dispatched_resource : chr "FPD Staff" "FPD Staff" "FPD Staff" "FPD Staff" ...
## $ dispatch_date : chr "2006-04-02 21:10" "2006-04-03 12:33" "2006-04-03 12:36" "2006- ...
## $ start_for_fire_date : chr "2006-04-02 21:20" "2006-04-03 12:35" "2006-04-03 12:42" "2006- ...
## $ assessment_resource : chr "IA Forces" "IA Forces" "IA Forces" "IA Forces" ...
## $ assessment_datetime : chr "2006-04-02 22:00" "2006-04-03 13:20" "2006-04-03 13:23" "2006- ...
## $ assessment_hectares : num 0.01 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ fire_spread_rate : num 0 0 0 0.1 0 0 0 0 0 ...
## $ fire_type : chr "Surface" "Surface" "Surface" "Surface" ...
## $ fire_position_on_slope : chr "Flat" "Lower 1/3" "Bottom" "Flat" ...
## $ weather_conditions_over_fire: chr "Clear" "Clear" "Clear" "Clear" ...
## $ temperature : num 18 12 12 12 6 11 11 16 11 11 ...
## $ relative_humidity : int 10 22 22 22 37 32 25 17 35 44 ...
## $ wind_direction : chr "SW" "SW" "SW" "SW" ...
## $ wind_speed : int 2 10 10 10 2 20 10 2 7 4 ...
## $ fuel_type : chr "01a" "01a" "01a" "01b" ...
## $ initial_action_by : chr "Land Owner" "Fire Department" "Fire Department" "Industry" ...
## $ ia_arrival_at_fire_date : chr NA NA NA NA ...
## $ ia_access : chr NA NA NA NA ...
## $ fire_fighting_start_date : chr NA NA NA NA ...
## $ fire_fighting_start_size : num NA NA NA NA NA NA NA NA 0.01 NA 0.6 ...
## $ bucketing_on_fire : chr NA NA NA NA ...
## $ distance_from_water_source : num NA NA NA NA NA NA NA NA NA ...
## $ first_bucket_drop_date : chr NA NA NA NA ...
## $ bh_fs_date : chr "2006-04-02 22:00" "2006-04-03 13:20" "2006-04-03 13:23" "2006- ...
## $ bh_hectares : num 0.01 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ uc_fs_date : chr "2006-04-02 22:00" "2006-04-03 13:20" "2006-04-03 13:23" "2006- ...
## $ uc_hectares : num 0.01 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ to_fs_date : chr NA NA NA NA ...
## $ to_hectares : num NA NA NA NA 0.1 NA NA 0.01 0.2 NA ...
## $ ex_fs_date : chr "2006-04-03 10:20" "2006-04-03 14:00" "2006-04-03 15:00" "2006- ...
## $ ex_hectares : num 0.1 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...

```

```

# View unique values in Temperature and FireSpreadRate
unique(data$temperature)

```

```

## [1] 18.0 12.0 6.0 11.0 16.0 28.0 26.0 25.0 35.0 15.0 10.0 13.2
## [13] 27.0 20.0 17.0 22.0 24.0 23.0 21.0 29.0 19.0 21.4 14.5 14.0
## [25] 33.0 31.0 32.0 4.0 24.5 30.0 27.5 22.5 21.5 6.5 9.0 13.0
## [37] 25.5 2.0 30.6 17.5 5.0 12.5 22.4 36.0 26.4 18.7 16.1 21.6
## [49] 16.5 19.5 7.0 24.6 -0.6 20.5 23.5 24.3 22.1 18.4 23.7 26.5
## [61] 8.0 15.6 22.8 12.8 13.5 29.5 14.2 8.3 11.5 8.5 13.3 17.6

```

```

## [73] 26.2 19.6 9.4 7.5 16.7 21.2 5.4 3.0 1.0 18.5 23.2 16.3
## [85] 24.8 26.7 19.8 31.5 25.6 28.2 28.7 20.2 22.7 17.3 30.8 24.7
## [97] 23.6 17.4 22.3 28.6 -18.0 0.1 -1.0 30.9 -8.0 -7.0 11.2 0.0
## [109] 19.4 -10.0 -5.0 6.2 11.7 -2.0 14.4 21.1 25.7 26.6 10.8 15.3
## [121] 0.2 -7.3 -4.0 7.6 8.9 -11.0 28.5 13.6 30.7 -6.0 -15.0 -3.0
## [133] 5.5 26.1 9.5 14.9 15.5 13.4 16.9 24.4 17.8 19.9 15.4 26.8
## [145] 25.4 21.8 18.2 27.2 -3.5 -12.0 -14.0 -13.0 20.9 2.5 14.6 12.1
## [157] 16.2 12.2 19.2 19.1 10.6 -9.0 8.2 6.6 20.6 17.7 3.2 20.7
## [169] 18.6 8.6 20.1 15.2 11.8 22.6 28.3 3.5 -2.7 6.1 4.8 13.8
## [181] 10.5 2.4 14.3 28.9 34.0 -21.0 26.3 -18.5 7.4 22.2 11.9 24.1
## [193] 14.8 20.3 18.3 29.1 8.7 7.2 5.1 27.8 15.7 -20.0 31.6 2.6
## [205] 5.7 13.7 12.4 27.4 -33.0 8.1 9.1 NA -25.0 8.8 30.5 14.1
## [217] 19.7 6.4 4.5 10.4 24.9 24.2 -17.0 10.2 13.9 18.1 23.9 -22.0
## [229] 10.3 -19.0 8.4 -16.0 -30.0 14.7 23.4 20.4 19.3 32.1 16.6 12.3
## [241] 18.8 25.8 37.0 1.5 16.4 -1.5 25.2 5.6 29.4 21.3 27.6 28.1
## [253] 27.1 9.3 18.9 12.7 6.3 27.7 32.5 23.8 33.2 25.3 20.8 17.9
## [265] 23.3 27.9 29.3 15.9 21.9 17.1 21.7 28.4 17.2 15.1 16.8 0.5
## [277] 32.4 29.8 30.4 39.0 -35.0 -34.0 2.3 13.1 28.8 3.3 6.7 23.1
## [289] 2.2 29.2 11.6 5.3 -2.5 29.6 33.4 27.3 11.1 1.7 31.8 22.9
## [301] 15.8 9.7 29.7 9.2 31.7 -3.4 35.4 31.2 7.3 30.1 1.3 12.6
## [313] 5.8 25.9 1.9 31.3 6.8 0.7 3.6 3.7 26.9 -23.0 25.1 38.0
## [325] 0.3 34.5 35.3 29.9 39.9 9.9 38.1 1.6 11.3 32.6 5.9 32.7
## [337] 37.5 4.7 12.9 -39.0 -0.2 10.1 7.9 9.8

```

```
unique(data$fire_spread_rate)
```

```

## [1] 0.0 0.1 1.0 12.0 0.5 1.5 3.0 2.0 5.0 10.0 0.2 35.0
## [13] 4.0 30.0 0.4 50.0 0.9 8.0 20.0 3.5 7.0 0.3 6.0 11.0
## [25] 17.0 9.0 25.0 18.0 40.0 2.5 100.0 0.7 15.0 0.8 1.2 1.1
## [37] 1.8 1.9 13.0 4.5 1.7 8.5 70.0 0.6 2.2 4.9 16.0 65.0
## [49] 21.0 5.7 34.0 71.0 19.0 60.0 22.0 5.5

```

We will also look at a summary of the data

```
summary(data$temperature)
```

```

##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.    NA's
## -39.00  14.00  19.00  17.85  23.00  39.90     80

```

```
summary(data$fire_spread_rate)
```

```

##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
## 0.0000 0.0000 0.0000 0.8962 1.0000 100.0000

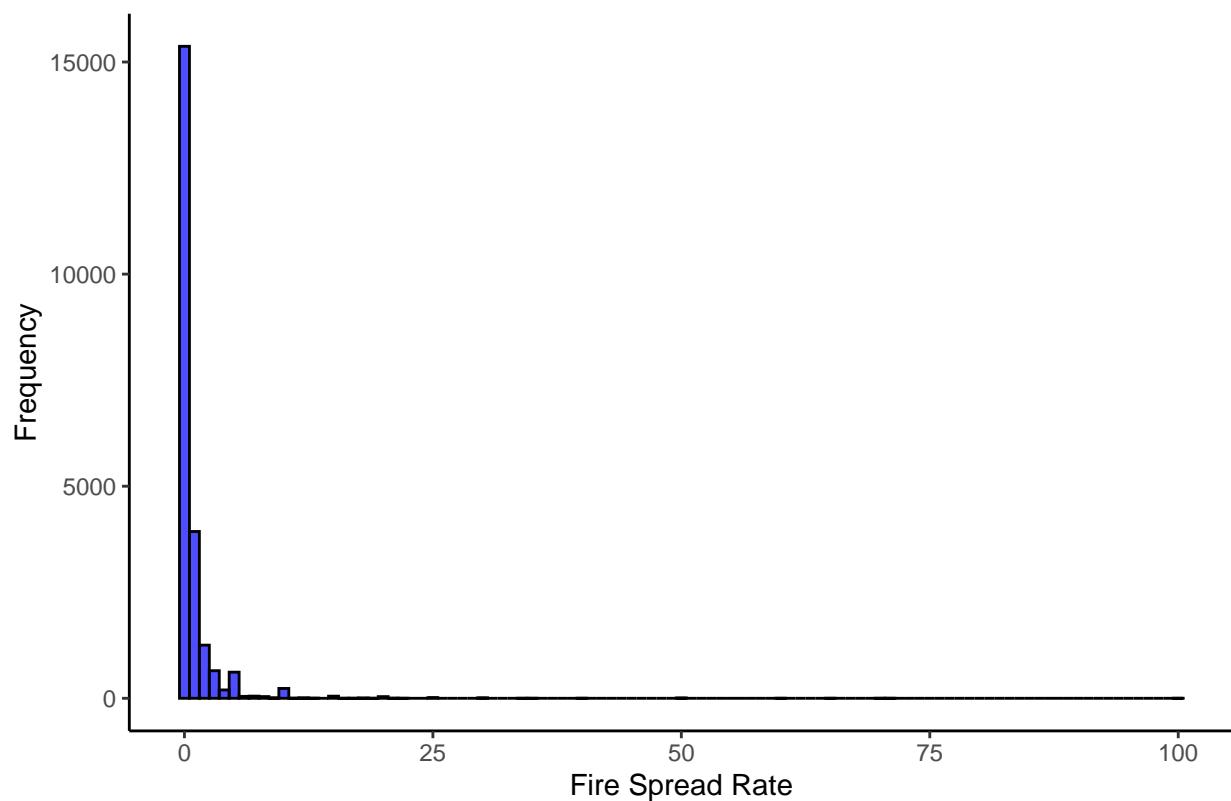
```

```

# Plot a histogram
ggplot(data, aes(x = fire_spread_rate)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Distribution of Fire Spread Rate",
       x = "Fire Spread Rate",
       y = "Frequency") +
  theme_classic()

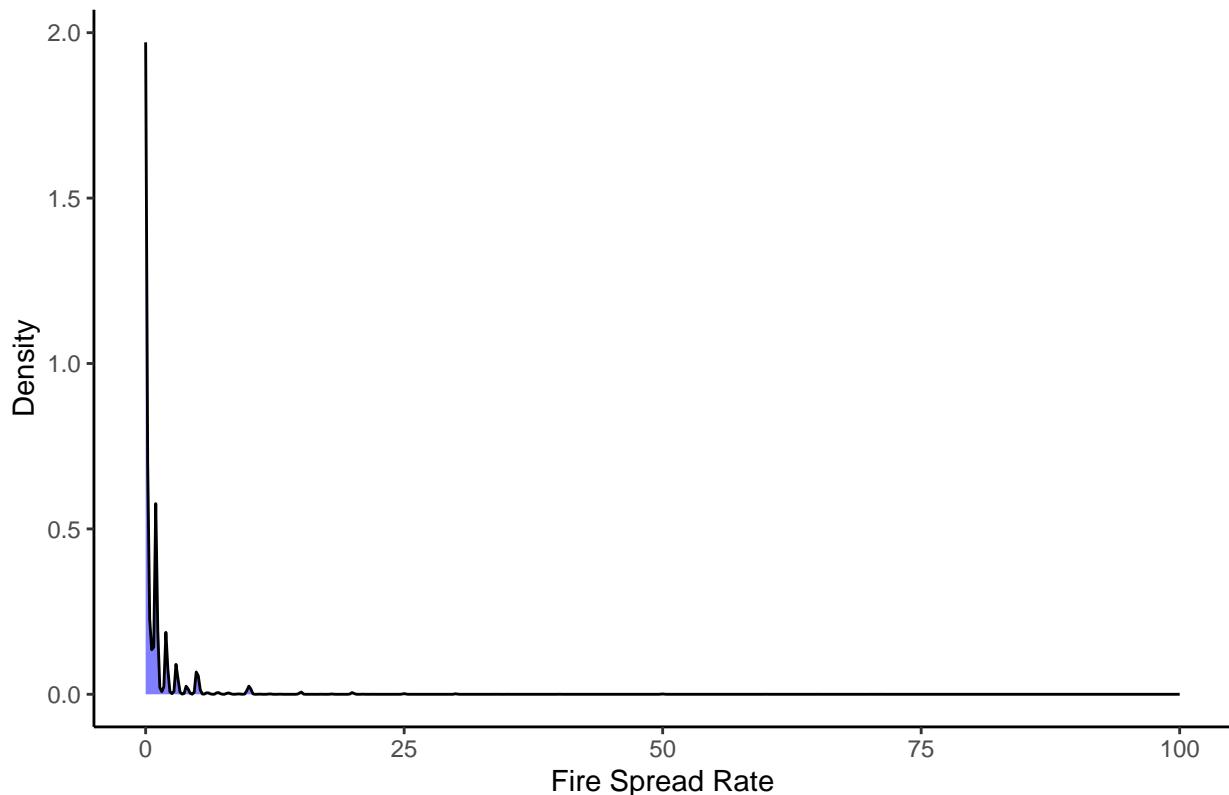
```

Distribution of Fire Spread Rate



```
# Alternatively, plot a density plot
ggplot(data, aes(x = fire_spread_rate)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Density Plot of Fire Spread Rate",
       x = "Fire Spread Rate",
       y = "Density") +
  theme_classic()
```

Density Plot of Fire Spread Rate



```
# Calculate skewness
spread_rate_skewness <- skewness(data$fire_spread_rate)
print(paste("Skewness of Fire Spread Rate:", spread_rate_skewness))
```

```
## [1] "Skewness of Fire Spread Rate: 11.2632058482397"
```

The value 11.22 suggests that the distribution of fire spread rates is heavily skewed to the right, meaning that most of the fire spread rates are relatively low, but there are a few extremely high values (outliers) that pull the tail of the distribution to the right.

Before performing a regression analysis, it is important to investigate the correlation between both variables. To normalize the distribution of the fire spread rate we can attempt a log transformation

```
# Log Transformation
data$log_fire_spread_rate <- log(data$fire_spread_rate + 1)

log_spread_rate_skewness <- skewness(data$log_fire_spread_rate)
print(paste("Skewness of Log Transformed Fire Spread Rate:", log_spread_rate_skewness))
```

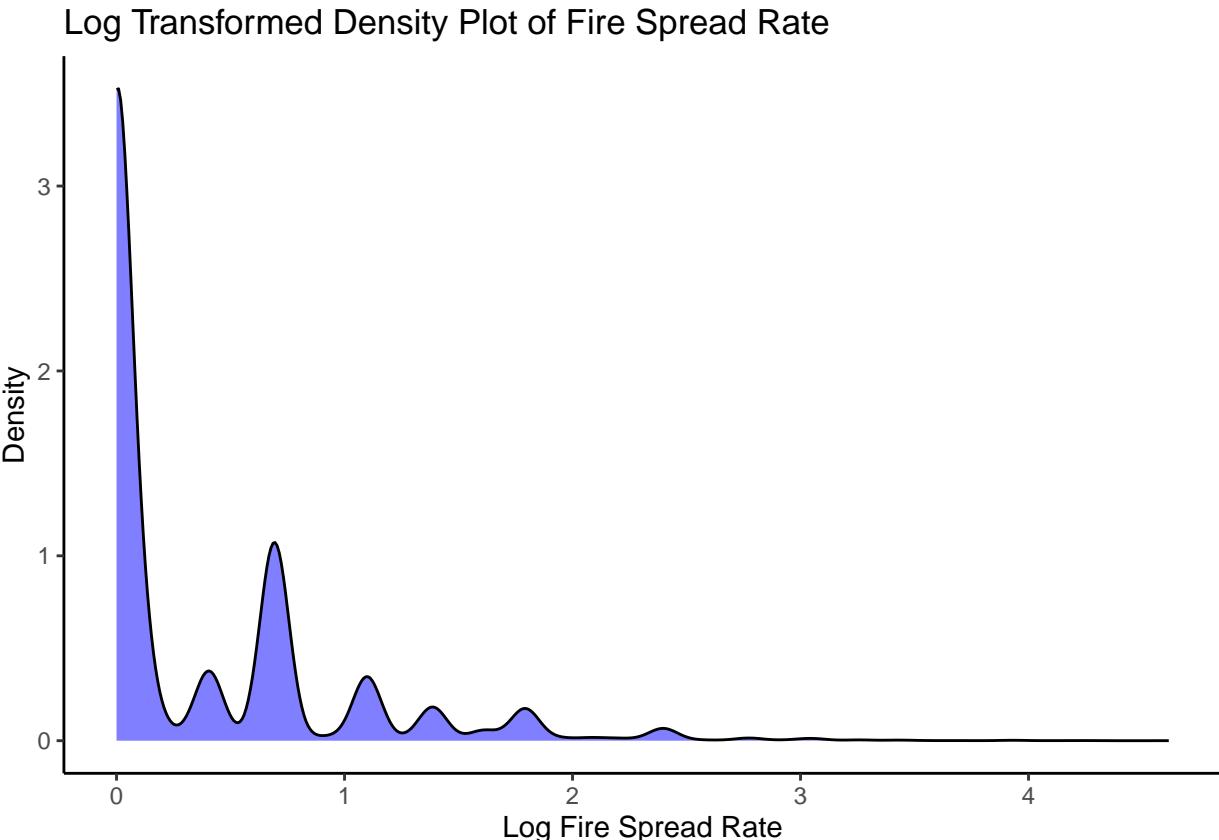
```
## [1] "Skewness of Log Transformed Fire Spread Rate: 1.92868885555048"
```

```
ggplot(data, aes(x = log_fire_spread_rate)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Log Transformed Density Plot of Fire Spread Rate",
```

```

x = "Log Fire Spread Rate",
y = "Density") +
theme_classic()

```



Following this, we will calculate Pearson correlation between log-transformed fire spread rate and temperature

```

correlation_log <- cor(data$log_fire_spread_rate, data$temperature, use = "complete.obs")
print(paste("Pearson correlation coefficient (Log Fire Spread Rate and Temperature):", correlation_log))

```

```
## [1] "Pearson correlation coefficient (Log Fire Spread Rate and Temperature): 0.250089700065433"
```

We will now rechecking correlation fire spread rate and temperature to see if the relationship has improved now.

The regression model can be coded as:

```

# Fit a linear regression model
log_model <- lm(log_fire_spread_rate ~ temperature, data = data)
summary(log_model)

```

```

##
## Call:
## lm(formula = log_fire_spread_rate ~ temperature, data = data)
##
## Residuals:
##
```

```

##      Min     1Q   Median     3Q     Max
## -0.7690 -0.3839 -0.2119  0.2326  4.0652
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0389989  0.0096034   4.061  4.9e-05 ***
## temperature 0.0191612  0.0004948  38.727 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5651 on 22480 degrees of freedom
##   (80 observations deleted due to missingness)
## Multiple R-squared:  0.06254,    Adjusted R-squared:  0.0625
## F-statistic:  1500 on 1 and 22480 DF,  p-value: < 2.2e-16

ggplot(data, aes(x = temperature, y = log_fire_spread_rate)) +
  geom_point(color = "red") +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Log Fire Spread Rate vs Temperature",
       x = "Temperature",
       y = "Log Fire Spread Rate")

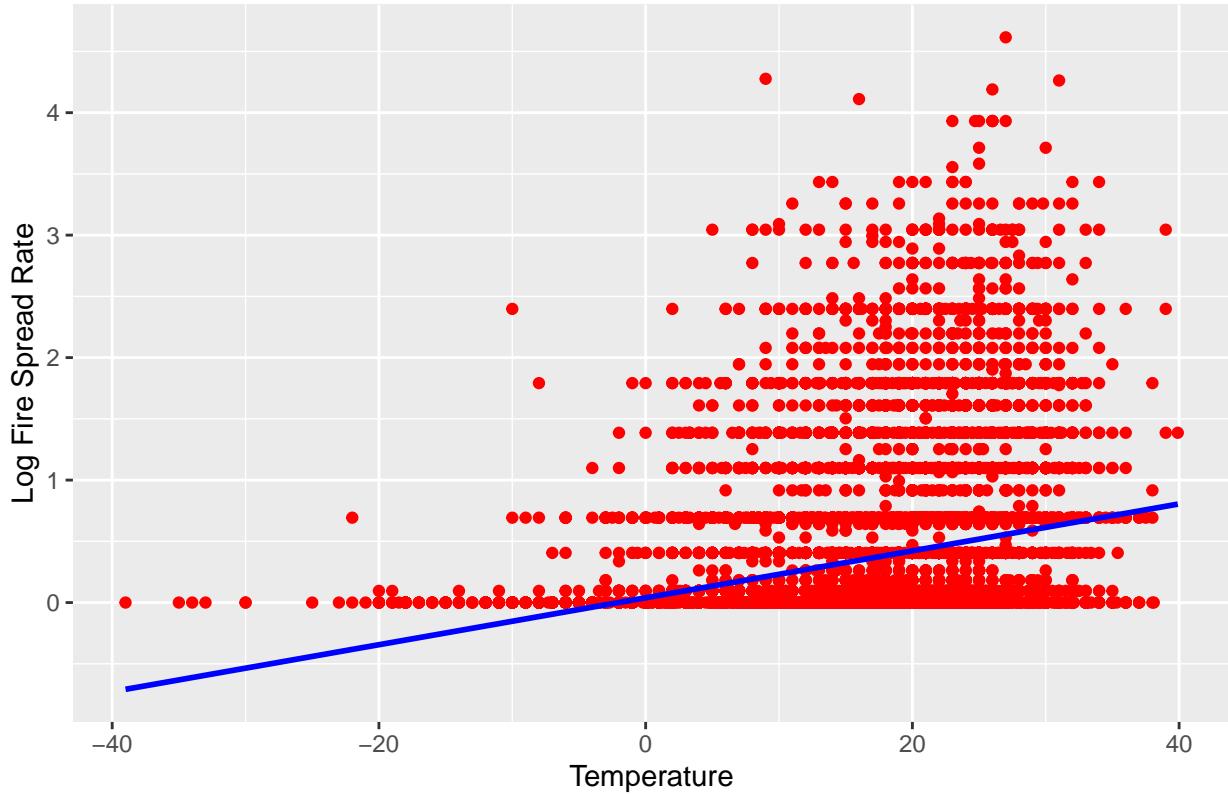
## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 80 rows containing non-finite outside the scale range
## ('stat_smooth()').

## Warning: Removed 80 rows containing missing values or values outside the scale range
## ('geom_point()').

```

Log Fire Spread Rate vs Temperature

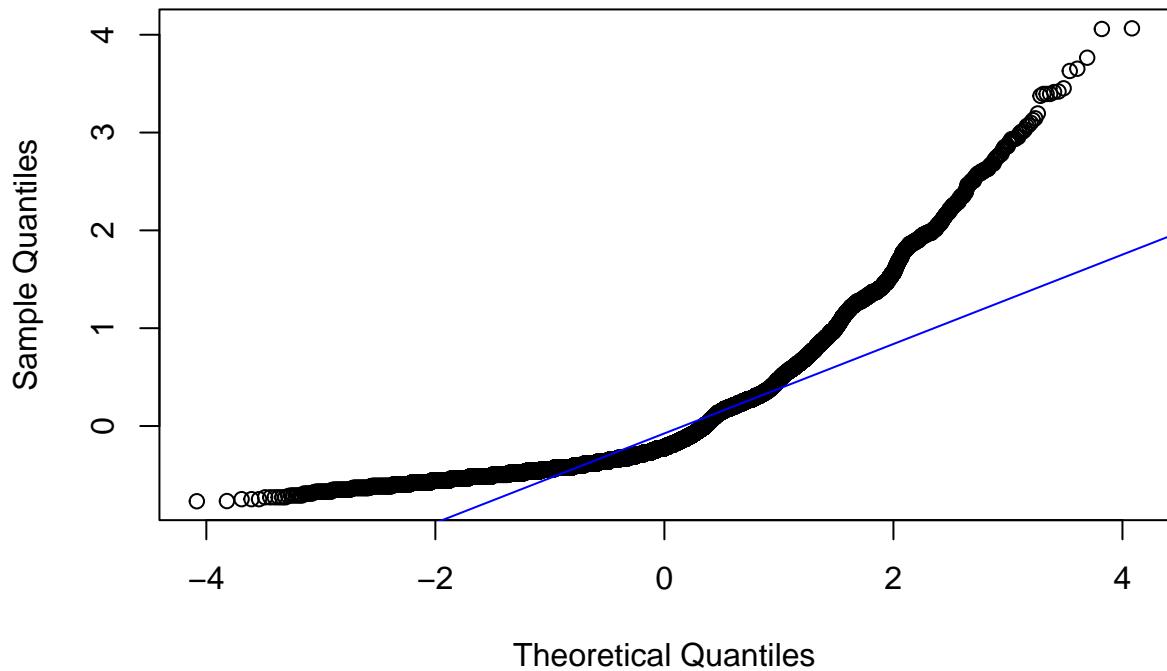


The graph shows the relationship between temperature and the rate of fire spread. The red dots represent data points where the x-axis represents the temperature and the y-axis represents the log of the fire spread rate. The blue line represents a line of best fit, indicating that there is a positive correlation between the two variables. This means that as temperature increases, the fire spread rate also increases, and this is illustrated by the upward trend of the blue line.

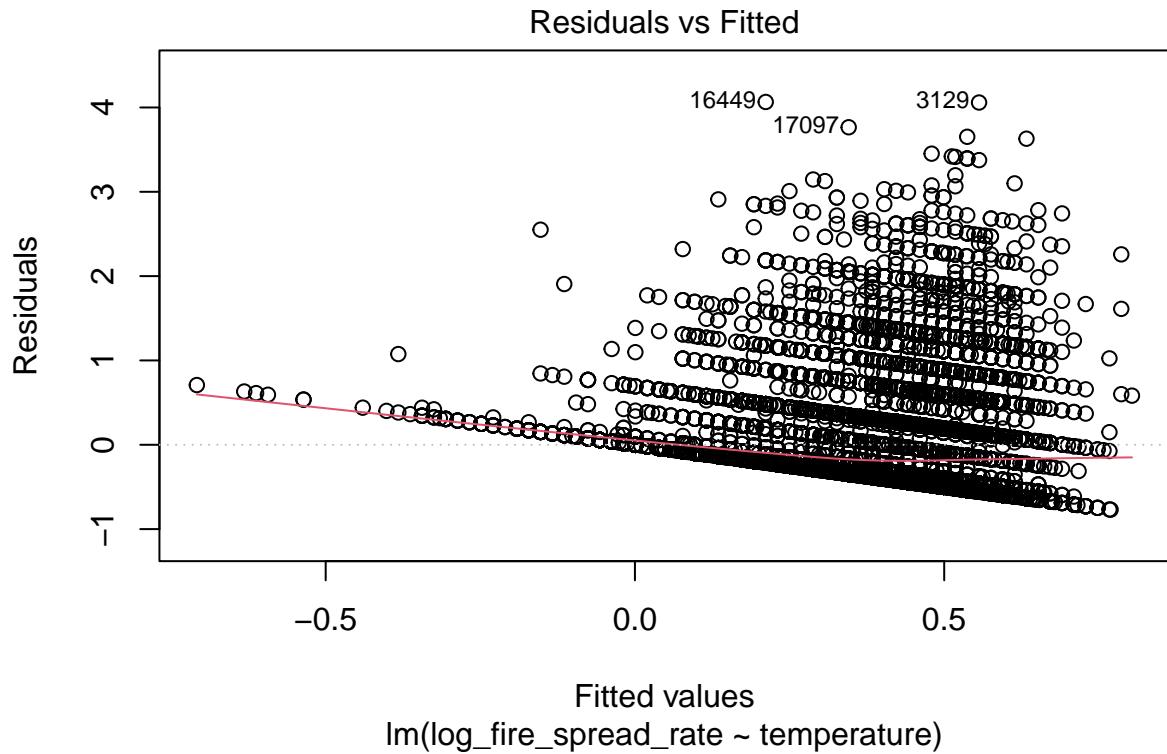
Q-Q Plot and Residuals Plot to check the normality

```
# Q-Q plot to check normality of residuals
qqnorm(residuals(log_model))
qqline(residuals(log_model), col = "blue")
```

Normal Q-Q Plot

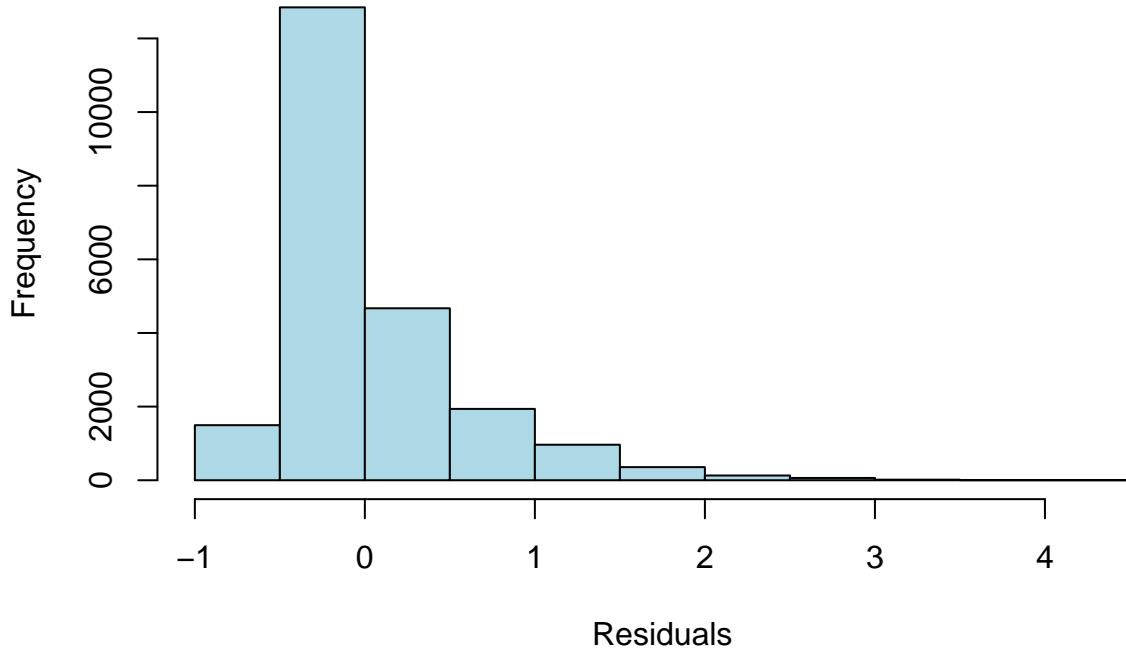


```
# Residuals vs Fitted plot  
plot(log_model, which = 1)
```



```
# Histogram of residuals
hist(residuals(log_model), main = "Histogram of Residuals", xlab = "Residuals", col = "lightblue")
```

Histogram of Residuals



Residual plot: This plot suggests that the linear model is not a good fit for the data, and alternative models might be considered.

Q-Q Residual: This plot indicates that the data is not normally distributed because it deviates from a straight line. The points are curved and have a few outliers. The data is skewed to the right as it deviates from the straight line on the right-hand side.

```
# Get model summary and extract statistics
log_model_summary <- summary(log_model)
r_squared_log <- log_model_summary$r.squared
adj_r_squared_log <- log_model_summary$adj.r.squared
p_value_log <- log_model_summary$coefficients[2, 4]

print(paste("R-squared (log model):", r_squared_log))

## [1] "R-squared (log model): 0.0625448580788178"

print(paste("Adjusted R-squared (log model):", adj_r_squared_log))

## [1] "Adjusted R-squared (log model): 0.062503156337629"

print(paste("p-value for temperature (log model):", p_value_log))

## [1] "p-value for temperature (log model): 1.1251791730511e-317"
```

To avoid the complexity by temperature variable Adjusted R-squared -the relationship between the temperature and fire spread rate is weak, meaning temperature does not explain much of the variation in fire spread rate.P Value suggests that the temperature is likely statistically significant, meaning it has a real effect on the dependent variable.

To check this since the p-value only reflects the significance of the relationship, it's also worth considering if other variables (like humidity , wind speed , vegetation_type

```
# Extra work--Consider including other relevant features
data$humidity <- data$relative_humidity
data$wind_speed <- data$wind_speed
data$vegetation_type <- data$fuel_type

# Update the model to include additional features
model <- lm(fire_spread_rate ~ temperature + humidity + wind_speed + vegetation_type, data = data)

# Summarize the updated model
summary(model)

## 
## Call:
## lm(formula = fire_spread_rate ~ temperature + humidity + wind_speed +
##     vegetation_type, data = data)
## 
## Residuals:
##      Min    1Q Median    3Q   Max 
## -4.849 -1.028 -0.475  0.221 98.075 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.241233  0.161542  7.684 1.63e-14 ***
## temperature  0.031667  0.003164 10.009 < 2e-16 ***
## humidity    -0.017181  0.001239 -13.866 < 2e-16 ***
## wind_speed   0.044757  0.002431 18.413 < 2e-16 ***
## vegetation_typeC2 0.223939  0.121777  1.839 0.065942 .  
## vegetation_typeC3 -0.246459  0.156810 -1.572 0.116036  
## vegetation_typeC4 -0.443133  0.259062 -1.711 0.087185 .  
## vegetation_typeC7 -0.845929  0.621074 -1.362 0.173202  
## vegetation_typeD1 -0.759180  0.176130 -4.310 1.64e-05 ***
## vegetation_typeM1 -0.559852  0.160891 -3.480 0.000503 *** 
## vegetation_typeM2 -0.699145  0.130445 -5.360 8.44e-08 *** 
## vegetation_typeM3 -1.718422  1.577507 -1.089 0.276024  
## vegetation_typeM4 -0.739332  2.726585 -0.271 0.786274  
## vegetation_type01a -0.760666  0.127503 -5.966 2.48e-09 *** 
## vegetation_type01b -0.719319  0.134277 -5.357 8.57e-08 *** 
## vegetation_typeS1 -0.892669  0.173143 -5.156 2.55e-07 *** 
## vegetation_typeS2 -1.039302  0.174980 -5.940 2.91e-09 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.724 on 17918 degrees of freedom
##     (4627 observations deleted due to missingness)
```

```

## Multiple R-squared:  0.07169,   Adjusted R-squared:  0.07086
## F-statistic: 86.48 on 16 and 17918 DF,  p-value: < 2.2e-16

```

This thing concludes that not only one factor is responsible for the fire spread these other factors combinely also affect that.

Creating a new column according to the size_class to represnt that in the visualization as HIgh or low spread in the particular area.

```
str(data)
```

```

## 'data.frame':  22562 obs. of  53 variables:
## $ fire_year          : int  2006 2006 2006 2006 2006 2006 2006 2006 2006 ...
## $ fire_number        : chr  "PWF001" "EWF002" "EWF001" "EWF003" ...
## $ fire_name          : chr  NA NA NA NA ...
## $ current_size       : num  0.1 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ size_class         : chr  "A" "B" "B" "A" ...
## $ fire_location_latitude: num  56.2 53.6 53.6 53.6 56.2 ...
## $ fire_location_longitude: num  -117 -116 -116 -116 -117 ...
## $ fire_origin         : chr  "Private Land" "Provincial Land" "Provincial Land" "Provincial ...
## $ general_cause_desc: chr  "Resident" "Incendiary" "Incendiary" "Incendiary" ...
## $ industry_identifier_desc: chr  NA NA NA NA ...
## $ responsible_group_desc: chr  "Resident" "Others (explain in remarks)" "Others (explain in re ...
## $ activity_class      : chr  "Grass" "Lighting Fires" "Lighting Fires" "Lighting Fires" ...
## $ true_cause          : chr  "Permit Related" "Arson Suspected" "Arson Suspected" "Arson Su ...
## $ fire_start_date     : chr  "2006-04-02 12:00" "2006-04-03 12:10" "2006-04-03 12:15" "2006-04-03 12:15" ...
## $ det_agent_type      : chr  "UNP" "UNP" "UNP" "UNP" ...
## $ det_agent           : chr  "310" "310" "310" "PUB" ...
## $ discovered_date     : chr  NA NA NA NA ...
## $ discovered_size      : num  NA NA NA NA NA NA NA NA NA ...
## $ reported_date        : chr  "2006-04-02 20:46" "2006-04-03 12:27" "2006-04-03 12:36" "2006-04-03 12:36" ...
## $ dispatched_resource: chr  "FPD Staff" "FPD Staff" "FPD Staff" "FPD Staff" ...
## $ dispatch_date        : chr  "2006-04-02 21:10" "2006-04-03 12:33" "2006-04-03 12:36" "2006-04-03 12:36" ...
## $ start_for_fire_date: chr  "2006-04-02 21:20" "2006-04-03 12:35" "2006-04-03 12:42" "2006-04-03 12:42" ...
## $ assessment_resource: chr  "IA Forces" "IA Forces" "IA Forces" "IA Forces" ...
## $ assessment_datetime: chr  "2006-04-02 22:00" "2006-04-03 13:20" "2006-04-03 13:23" "2006-04-03 13:23" ...
## $ assessment_hectares: num  0.01 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ fire_spread_rate     : num  0 0 0 0 0.1 0 0 0 0 0 ...
## $ fire_type            : chr  "Surface" "Surface" "Surface" "Surface" ...
## $ fire_position_on_slope: chr  "Flat" "Lower 1/3" "Bottom" "Flat" ...
## $ weather_conditions_over_fire: chr  "Clear" "Clear" "Clear" "Clear" ...
## $ temperature          : num  18 12 12 12 6 11 11 16 11 11 ...
## $ relative_humidity    : int  10 22 22 22 37 32 25 17 35 44 ...
## $ wind_direction       : chr  "SW" "SW" "SW" "SW" ...
## $ wind_speed           : int  2 10 10 10 2 20 10 2 7 4 ...
## $ fuel_type             : chr  "01a" "01a" "01a" "01b" ...
## $ initial_action_by    : chr  "Land Owner" "Fire Department" "Fire Department" "Industry" ...
## $ ia_arrival_at_fire_date: chr  NA NA NA NA ...
## $ ia_access             : chr  NA NA NA NA ...
## $ fire_fighting_start_date: chr  NA NA NA NA ...
## $ fire_fighting_start_size: num  NA NA NA NA NA NA 0.01 NA 0.6 ...
## $ bucketing_on_fire     : chr  NA NA NA NA ...
## $ distance_from_water_source: num  NA NA NA NA NA NA NA NA NA ...
## $ first_bucket_drop_date: chr  NA NA NA NA ...

```

```

## $ bh_fs_date : chr "2006-04-02 22:00" "2006-04-03 13:20" "2006-04-03 13:23" "2006-
## $ bh_hectares : num 0.01 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ uc_fs_date : chr "2006-04-02 22:00" "2006-04-03 13:20" "2006-04-03 13:23" "2006-
## $ uc_hectares : num 0.01 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ to_fs_date : chr NA NA NA NA ...
## $ to_hectares : num NA NA NA NA 0.1 NA NA 0.01 0.2 NA ...
## $ ex_fs_date : chr "2006-04-03 10:20" "2006-04-03 14:00" "2006-04-03 15:00" "2006-
## $ ex_hectares : num 0.1 0.2 0.5 0.01 0.1 0.2 0.01 0.01 0.2 0.6 ...
## $ log_fire_spread_rate : num 0 0 0 0.0953 ...
## $ humidity : int 10 22 22 22 37 32 25 17 35 44 ...
## $ vegetation_type : chr "01a" "01a" "01a" "01b" ...

```

Create SpreadCategory based on size_class

```

data$SpreadCategory <- ifelse(data$size_class %in% c("D", "E"), "High", "Low")

head(data)

##   fire_year fire_number fire_name current_size size_class
## 1    2006      PWF001     <NA>       0.10      A
## 2    2006      EWF002     <NA>       0.20      B
## 3    2006      EWF001     <NA>       0.50      B
## 4    2006      EWF003     <NA>       0.01      A
## 5    2006      PWF002     <NA>       0.10      A
## 6    2006      CWF001     <NA>       0.20      B
##   fire_location_latitude fire_location_longitude   fire_origin
## 1           56.24996            -117.1820 Private Land
## 2           53.60637            -115.9157 Provincial Land
## 3           53.61093            -115.5943 Provincial Land
## 4           53.60887            -115.6095 Provincial Land
## 5           56.24996            -117.0502 Provincial Land
## 6           51.15293            -115.0346 Indian Reservation
##   general_cause_desc industry_identifier_desc   responsible_group_desc
## 1        Resident             <NA>                   Resident
## 2      Incendiary             <NA>  Others (explain in remarks)
## 3      Incendiary             <NA>  Others (explain in remarks)
## 4      Incendiary             <NA>  Others (explain in remarks)
## 5 Other Industry           Waste Disposal          Employees
## 6        Resident             <NA>                   Resident
##   activity_class true_cause fire_start_date det_agent_type det_agent
## 1      Grass Permit Related 2006-04-02 12:00        UNP     310
## 2 Lighting Fires Arson Suspected 2006-04-03 12:10        UNP     310
## 3 Lighting Fires Arson Suspected 2006-04-03 12:15        UNP     310
## 4 Lighting Fires Arson Suspected 2006-04-03 12:10        UNP     PUB
## 5      Refuse Permit Related 2006-04-03 17:00        UNP     LFS
## 6 Unclassified      Unsafe Fire 2006-04-02 14:25        UNP     310
##   discovered_date discovered_size reported_date dispatched_resource
## 1             <NA>           NA 2006-04-02 20:46      FPD Staff
## 2             <NA>           NA 2006-04-03 12:27      FPD Staff
## 3             <NA>           NA 2006-04-03 12:36      FPD Staff
## 4             <NA>           NA 2006-04-03 13:23      FPD Staff
## 5 2006-04-03 19:11             NA 2006-04-03 19:12      FPD Staff
## 6 2006-04-02 14:27             NA 2006-04-02 14:30      FPD Staff
##   dispatch_date start_for_fire_date assessment_resource assessment_datetime
## 1 2006-04-02 21:10      2006-04-02 21:20      IA Forces 2006-04-02 22:00

```

```

## 2 2006-04-03 12:33    2006-04-03 12:35      IA Forces    2006-04-03 13:20
## 3 2006-04-03 12:36    2006-04-03 12:42      IA Forces    2006-04-03 13:23
## 4 2006-04-03 13:50    2006-04-03 13:50      IA Forces    2006-04-03 14:08
## 5 2006-04-03 19:19    2006-04-03 19:22      Other       2006-04-03 19:57
## 6 2006-04-02 14:40    2006-04-02 14:45      IA Forces    2006-04-02 16:00
##   assessment_hectares fire_spread_rate fire_type fire_position_on_slope
## 1             0.01          0.0  Surface           Flat
## 2             0.20          0.0  Surface          Lower 1/3
## 3             0.50          0.0  Surface           Bottom
## 4             0.01          0.0  Surface           Flat
## 5             0.10          0.1  Surface           Flat
## 6             0.20          0.0  Surface           Flat
##   weather_conditions_over_fire temperature relative_humidity wind_direction
## 1                   Clear        18            10        SW
## 2                   Clear        12            22        SW
## 3                   Clear        12            22        SW
## 4                   Clear        12            22        SW
## 5                   Clear         6            37        SW
## 6                   Clear        11            32         S
##   wind_speed fuel_type initial_action_by ia_arrival_at_fire_date ia_access
## 1         2      01a     Land Owner           <NA>      <NA>
## 2        10      01a   Fire Department        <NA>      <NA>
## 3        10      01a   Fire Department        <NA>      <NA>
## 4        10      01b     Industry           <NA>      <NA>
## 5         2     <NA>   Fire Department        <NA>      <NA>
## 6        20      01b   Fire Department        <NA>      <NA>
##   fire_fighting_start_date fire_fighting_start_size bucketing_on_fire
## 1                  <NA>            NA           <NA>
## 2                  <NA>            NA           <NA>
## 3                  <NA>            NA           <NA>
## 4                  <NA>            NA           <NA>
## 5                  <NA>            NA           <NA>
## 6                  <NA>            NA           <NA>
##   distance_from_water_source first_bucket_drop_date      bh_fs_date
## 1                      NA            <NA> 2006-04-02 22:00
## 2                      NA            <NA> 2006-04-03 13:20
## 3                      NA            <NA> 2006-04-03 13:23
## 4                      NA            <NA> 2006-04-03 14:08
## 5                      NA            <NA> 2006-04-03 19:57
## 6                      NA            <NA> 2006-04-02 16:00
##   bh_hectares      uc_fs_date uc_hectares      to_fs_date to_hectares
## 1       0.01 2006-04-02 22:00      0.01           <NA>      NA
## 2       0.20 2006-04-03 13:20      0.20           <NA>      NA
## 3       0.50 2006-04-03 13:23      0.50           <NA>      NA
## 4       0.01 2006-04-03 14:08      0.01           <NA>      NA
## 5       0.10 2006-04-03 20:19      0.10 2006-04-03 20:20      0.1
## 6       0.20 2006-04-02 16:00      0.20           <NA>      NA
##   ex_fs_date ex_hectares log_fire_spread_rate humidity vegetation_type
## 1 2006-04-03 10:20      0.10 0.0000000000      10      01a
## 2 2006-04-03 14:00      0.20 0.0000000000      22      01a
## 3 2006-04-03 15:00      0.50 0.0000000000      22      01a
## 4 2006-04-03 15:05      0.01 0.0000000000      22      01b
## 5 2006-04-05 10:18      0.10 0.0953101837      37      <NA>
## 6 2006-04-03 18:00      0.20 0.0000000000      32      01b

```

```

##   SpreadCategory
## 1      Low
## 2      Low
## 3      Low
## 4      Low
## 5      Low
## 6      Low

```

Geospatial Visualization: Map the fire occurrences geographically with spread rate and temperature as visual layers

```

#For the following code, the output is an HTML output, therefore exporting it in a PDF format is not possible

#data <- data[!is.na(data$fire_location_longitude) & !is.na(data$fire_location_latitude), ]
# Ensure data is not NULL and has the required columns
#if (!is.null(data)) && all(c("fire_location_longitude", "fire_location_latitude", "fire_spread_rate", "temperature") %in% names(data))

# Print debugging information

# Filter out rows with missing lat/long
#data <- data[!is.na(data$fire_location_longitude) & !is.na(data$fire_location_latitude), ]

#leaflet(data) %>%
#  addTiles() %>%
#  addCircleMarkers(~fire_location_longitude, ~fire_location_latitude,
#    radius = ~fire_spread_rate * 0.1,
#    color = ~ifelse(SpreadCategory == "High", "red", "green"),
#    fillOpacity = 0.5,
#    popup = ~paste("Spread Rate:", fire_spread_rate, "<br>",
#                  #Temperature:", temperature)) %>%
#  setView(lng = mean(data$fire_location_longitude, na.rm = TRUE),
#         #lat = mean(data$fire_location_latitude, na.rm = TRUE),
#         #zoom = 6) %>%
#  addLegend("bottomright",
#            colors = c("red", "green"),
#            labels = c("High Spread Rate", "Low Spread Rate"),
#            title = "Spread Rate Category")
#} else {
#  print("Data is NULL or required columns are missing.")
#}

```

Added clustering Points for Better Performance on this geospatial visualization. This will interactively shows the area where the fire spread rate and temperature is high or low.

Hypothesis testing:

```

# Extract the p-value for temperature in the transformed model
p_value_log <- summary(log_model)$coefficients[2, 4]

# Hypothesis testing
if (p_value_log < 0.05) {
  print("Reject the null hypothesis. There is a statistically significant relationship between temperature and fire spread rate")
} else {
  print("Fail to reject the null hypothesis. There is no statistically significant relationship between temperature and fire spread rate")
}

```

```

## [1] "Reject the null hypothesis. There is a statistically significant relationship between temperature and fire spread rate (log-transformed)."

# Print p-value for confirmation
print(paste("P-value:", p_value_log))

## [1] "P-value: 1.1251791730511e-317"

```

There is a statistically significant relationship between temperature and fire spread rate (log-transformed).

The analysis of the residual plot and Q-Q residual plot suggests that the linear regression model is not an appropriate fit for the data. The residual plot indicates a poor linear relationship, and the Q-Q residual plot reveals that the data is not normally distributed, with a right-skew and the presence of outliers. Given these observations, alternative modeling approaches, such as non-linear regression or robust regression techniques, should be considered to better capture the underlying patterns in the data and avoid the reliance on normality assumptions.

Question 3 - What is the Relationship Between Fire Size to Response and Extinguished Time

Introduction

Analyze the relationship between the Fire Size against the Response /Extinguished Time

Response Time

Duration between the detection of a forest fire and the arrival of firefighting resources at the scene.

Extinguished Time

Duration between the detection of a forest fire the moment when the fire is completely put out and no longer poses a threat.

Data Cleaning and Transformation

Reading the file and converting the blank into NA

```
data1=fire_data
```

Step 1: Identifying the targetted Variable:

Below are the list of variables which we are focusing to continue our statistical analysis for this problem statement

- Fire_Size (in Hectares)
- fire_start_date
- discovered date
- ex_fs_date -(Extinguished time)
- assessment_datetime-(Actual Response Starts)

Step 2: Mapping the NA assessment_datetime,ex_fs_date

Identified the blank dates with the next earliest date for fire_start_date with Discovered_date, assessment date with the arrival date in the fire control date,bh_fs_date. Main Objective of this transformation is not to eliminate the data which will be really supporting our regression

```
data2 <- data1 %>%
  mutate(discovered_date = ifelse(discovered_date == 'NA', reported_date, discovered_date),
         ex_fs_date = ifelse(ex_fs_date == 'NA', bh_fs_date, ex_fs_date),
         fire_start_date = ifelse(fire_start_date == 'NA', discovered_date, fire_start_date),
         assessment_datetime = ifelse(assessment_datetime == 'NA', ia_arrival_at_fire_date, assessment_datetime),
         discovered_date = ifelse(is.na(discovered_date), reported_date, discovered_date),
         ex_fs_date = ifelse(is.na(ex_fs_date), bh_fs_date, ex_fs_date),
         fire_start_date = ifelse(is.na(fire_start_date), discovered_date, fire_start_date),
         assessment_datetime = ifelse(is.na(assessment_datetime), ia_arrival_at_fire_date, assessment_datetime))
```

Step 3: Converting the date variables into date format

```
data2$ex_fs_date <- as.POSIXct(data2$ex_fs_date, format = "%Y-%m-%d %H:%M", tz = "America/Edmonton")
data2$discovered_date <- as.POSIXct(data2$discovered_date, format = "%Y-%m-%d %H:%M", tz = "America/Edmonton")
data2$fire_start_date <- as.POSIXct(data2$fire_start_date, format = "%Y-%m-%d %H:%M", tz = "America/Edmonton")
data2$assessment_datetime <- as.POSIXct(data2$assessment_datetime, format = "%Y-%m-%d %H:%M", tz = "America/Edmonton")
# Check the classes after conversion
class(data2$ex_fs_date)

## [1] "POSIXct" "POSIXt"

class(data2$fire_start_date)

## [1] "POSIXct" "POSIXt"
```

Step 4: Calculating the response and extinguished time in minutes

- ResponseTime = assessment_datetime - fire_start_date
- ExtTime = ex_fs_date - fire_start_date

```
# Calculate response time and ext time in minutes
data2$ResponseTime <- as.numeric(difftime(data2$assessment_datetime, data2$fire_start_date, units = "mins"))
data2$ExtTime <- as.numeric(difftime(data2$ex_fs_date, data2$fire_start_date, units = "mins"))
```

Step 5: Logarithmic Transformation

- Performed a log linear transformation on the dependent variables(current_size,ResponseTime,ExtTime)

```
data2$current_size=log10(data2$current_size)
data2$ResponseTime=log10(data2$ResponseTime)
data2$ExtTime=log10(data2$ExtTime)
```

Step 6: Filtering Variables

-Filtering Out the required variables to continue with the further regression analysis of the size and the times which we are targetting to produce the correlation.

```
data_filtered<- data2 %>%
  select(fire_number, fire_year, current_size, Responsetime, Exttime)
head(data_filtered)

##   fire_number fire_year current_size Responsetime   Exttime
## 1      PWF001     2006      -1.00000  2.778151 3.127105
## 2      EWF002     2006      -0.69897  1.845098 2.041393
## 3      EWF001     2006      -0.30103  1.832509 2.217484
## 4      EWF003     2006      -2.00000  2.071882 2.243038
## 5      PWF002     2006      -1.00000  2.247973 3.394101
## 6      CWF001     2006      -0.69897  1.977724 3.218798
```

Step 6: Calculating the mean response time and extinguished time

- To continue analysis of the newly derived variables and conducting analysis, we performed grouping the time with respect to the affected size and storing it into different data frames
- result1 - (Response Time)
- result2 - (Extinguished Time)

Below are the cleaned data set which are going to be in our further regression Analysis

```
result1 <- data2 %>%
  group_by(current_size) %>%
  summarise(
    meanresponsetime = mean(Responsetime),
  )

result1 <- result1 %>%
  filter(!is.na(meanresponsetime), !is.infinite(meanresponsetime),
        !is.na(current_size), !is.infinite(current_size))

result2 <- data2 %>%
  group_by(current_size) %>%
  summarise(
    meanExttime = mean(Exttime),
  )%>%
  arrange(meanExttime)

result2 <- result2 %>%
  filter(!is.na(meanExttime), !is.infinite(meanExttime),
        !is.na(current_size), !is.infinite(current_size))
cat("Data Frame for Response Time")

## Data Frame for Response Time
```

```

head(result1)

## # A tibble: 6 x 2
##   current_size meanresponsetime
##       <dbl>           <dbl>
## 1      -1.52        2.34
## 2      -1.40        2.41
## 3      -1.22        2.57
## 4      -1.10        2.38
## 5      -1.05        2.45
## 6     -0.959       2.25

```

```

summary(result1)

##    current_size      meanresponsetime
##  Min.   :-1.5229   Min.   :0.4771
##  1st Qu.: 0.7846   1st Qu.:1.7482
##  Median  : 1.5428   Median  :2.1533
##  Mean    : 1.7219   Mean    :2.3599
##  3rd Qu.: 2.5152   3rd Qu.:2.9807
##  Max.    : 5.7617   Max.    :6.0263

```

```

cat(".\n")

```

```

## .

```

```

cat("Data Frame for Extinguished Time")

```

```

## Data Frame for Extinguished Time

```

```

head(result2)

```

```

## # A tibble: 6 x 2
##   current_size meanExttime
##       <dbl>        <dbl>
## 1      4.08        1.99
## 2      0.420       2.05
## 3      0.897       2.06
## 4      0.185       2.11
## 5      0.246       2.13
## 6      0.117       2.14

```

```

summary(result2)

```

```

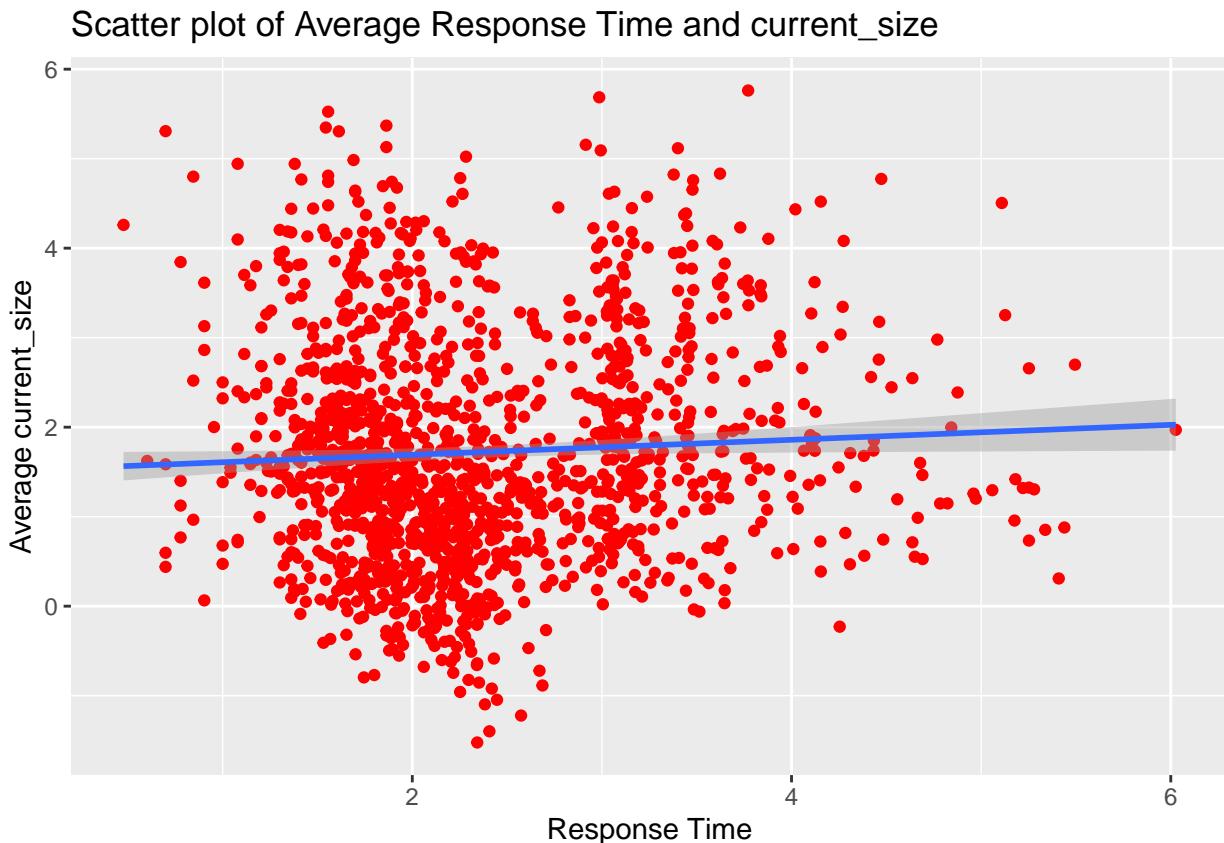
##    current_size      meanExttime
##  Min.   :-1.6990   Min.   :1.987
##  1st Qu.: 0.7645   1st Qu.:3.650
##  Median  : 1.5358   Median  :4.191
##  Mean    : 1.7056   Mean    :4.200
##  3rd Qu.: 2.5110   3rd Qu.:4.759
##  Max.    : 5.7617   Max.    :6.031

```

Regression Analysis

Plotted the scattered plots to validate the density of the variables being analyzed

```
ggplot1= (ggplot(result1, aes(x=meanresponsetime, y = current_size))+
  geom_point (color = "red") +
  stat_smooth(method = "lm", formula = y ~ x, geom = "smooth")+
  labs (title = "Scatter plot of Average Response Time and current_size",
        x = "Response Time", y ="Average current_size"))
ggplot1
```

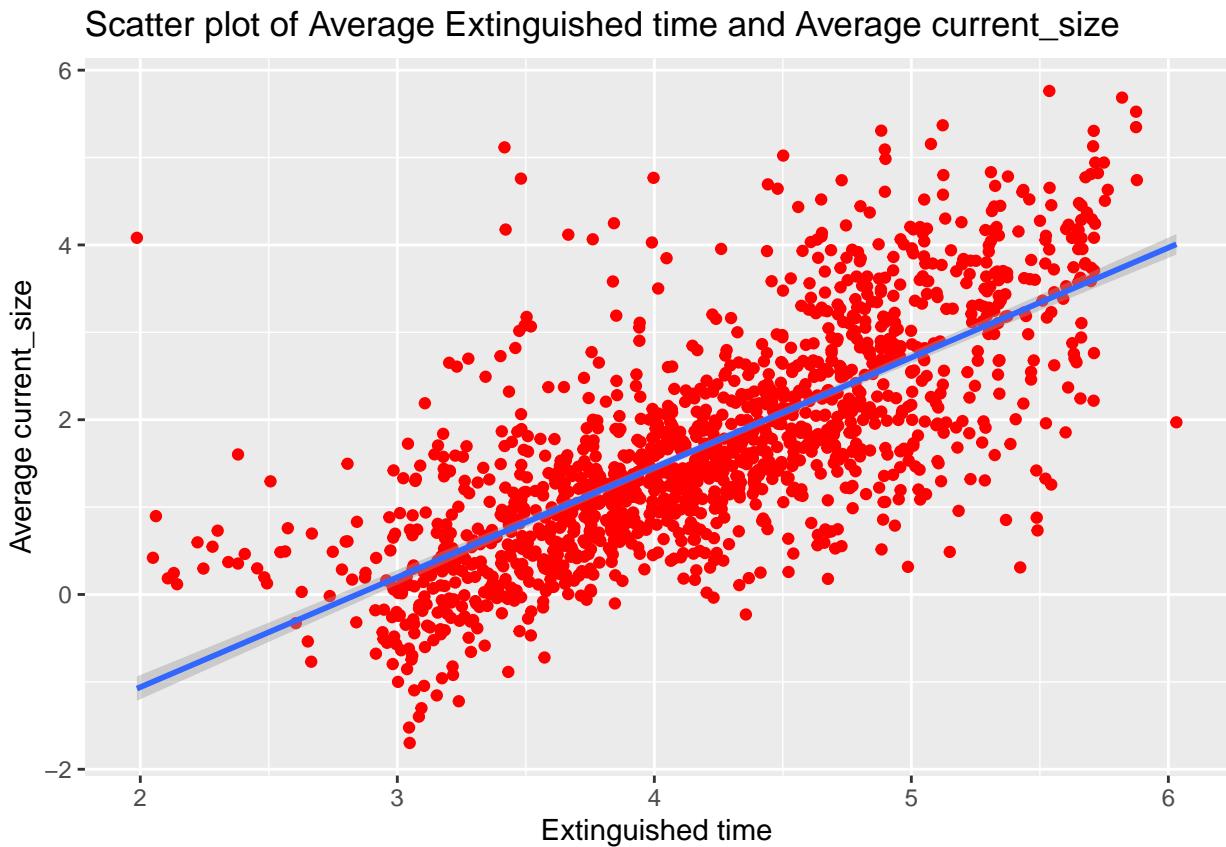


Inference:

- Positive linear trend. Response time increases, the current size also slightly increases, though the relationship seems weak (the slope of the line is small).
- Data points (in red) are densely scattered, with many points clustering around lower response times and a current size. However, there are a few points that deviate significantly from this dense cluster, especially for higher response times and current sizes.

```
ggplot2 =(ggplot(result2, aes(x=meanExttime, y = current_size))+
  geom_point (color = "red") +
  stat_smooth(method = "lm", formula = y ~ x, geom = "smooth")+
  labs (title = "Scatter plot of Average Extinguished time and Average current_size",
```

```
x = "Extinguished_time", y ="Average_current_size"))
ggplot2
```



Inference:

-Strong Positive Trend -Slope of the line is noticeably steeper compared to the response time plot, indicating that changes in extinguished time have a more pronounced effect on the average current size.

Generating Models and Intrepreting the Model Summary with Hypothesis Test

Mean Response Time

-Null Hypothesis (H_0): There is no relationship between the Response time and Average current_size.
-Alternative Hypothesis (H_a): There is a relationship between the Response time and Average current_size.

```
model_clean1 = lm(current_size ~ meanresponsetime, data = result1)
summary(model_clean1)
```

```
##
## Call:
## lm(formula = current_size ~ meanresponsetime, data = result1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.2433 -0.9376 -0.1591  0.7704  3.9218
```

```

## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.52463   0.09867 15.451 <2e-16 ***
## meanresponsetime 0.08359   0.03937  2.123  0.0339 *  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.274 on 1465 degrees of freedom
## Multiple R-squared:  0.003068, Adjusted R-squared:  0.002388 
## F-statistic: 4.509 on 1 and 1465 DF, p-value: 0.03388

```

Interpretation:

p-value of 0.03388 relationship between meanresponsetime and current_size is statistically significant at the 5% significance level ($p < 0.05$)

Model Fit:

R-squared value of 0.002388 indicates a Low Fit.

Extinguished Time

-Null Hypothesis (H0): There is no relationship between the Extinguished time and Average current_size.
-Alternative Hypothesis (Ha): There is a relationship between the Extinguished time and Average current_size.

```

model_clean2 = lm(current_size ~ meanExttime, data = result2)
summary(model_clean2)

```

```

## 
## Call:
## lm(formula = current_size ~ meanExttime, data = result2)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -2.9340 -0.5563 -0.0782  0.4719  5.1584 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -3.57474   0.12930 -27.65 <2e-16 ***
## meanExttime  1.25715   0.03031  41.48 <2e-16 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.8741 on 1480 degrees of freedom
## Multiple R-squared:  0.5376, Adjusted R-squared:  0.5373 
## F-statistic: 1721 on 1 and 1480 DF, p-value: < 2.2e-16

```

Interpretation:

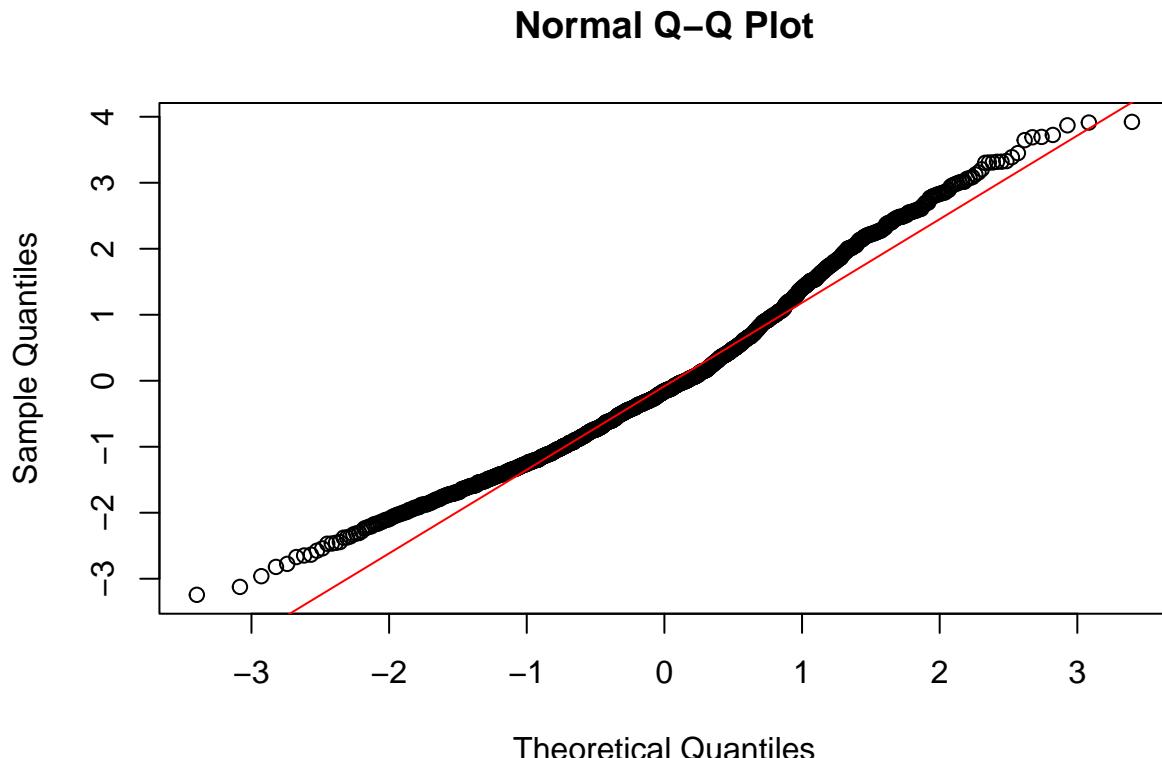
p-value of 2.2e-16 is extremely low, indicating strong evidence against the null hypothesis means that Response time has significant effect on the Size of the forest impact

Model Fit:

R-squared: 0.5373 of the variance in current_size which indicates a moderate fit

Analyzing the Normal Distribution - QQ Plot for Response Time

```
qqnorm(model_clean1$residuals)
qqline(model_clean1$residuals, col = "red")
```

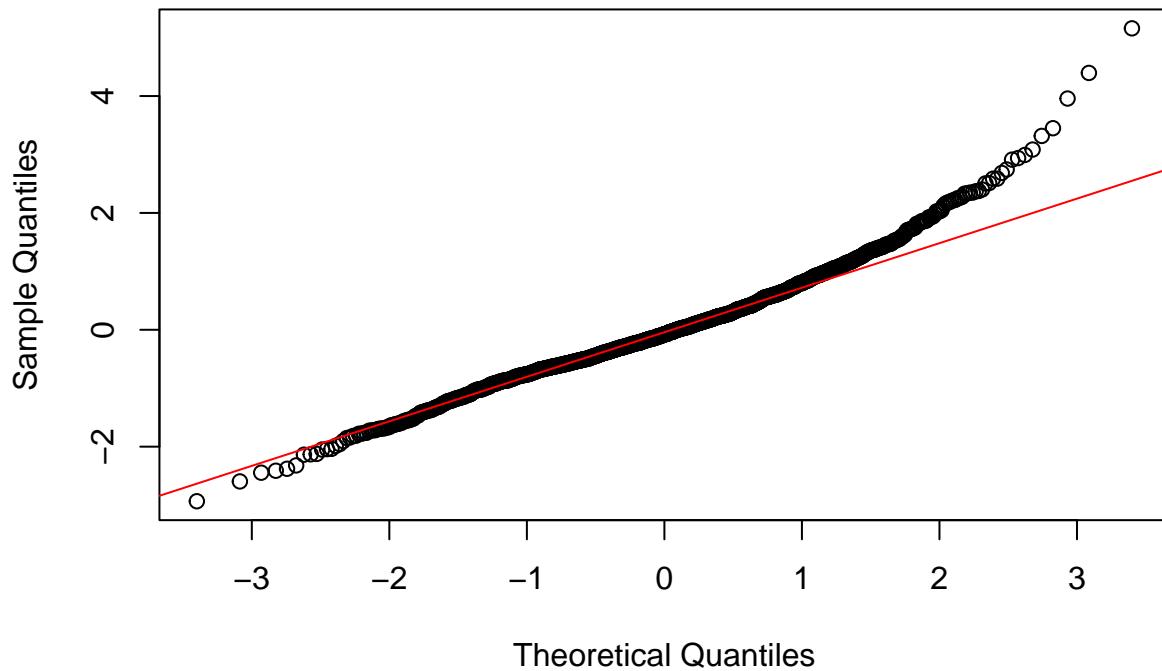


```
### Interpretation: - Model appears to follow a normal distribution reasonably well. - Some deviations at the extreme ends (in the tails) may indicate slight departures from normality, possibly showing mild outliers or heavier tails than a normal distribution
```

Analyzing the Normal Distribution - QQ Plot for Extinguished Time

```
qqnorm(model_clean2$residuals)
qqline(model_clean2$residuals, col = "red")
```

Normal Q-Q Plot

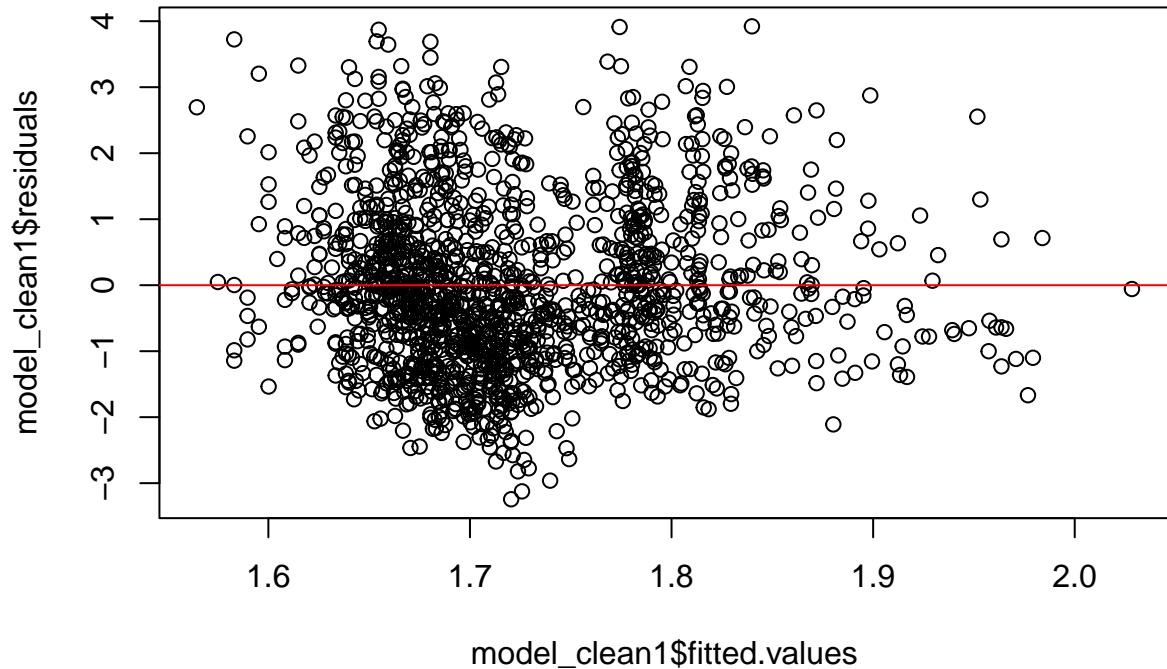


Interpretation:

- Model is approximately normal
- Model may have heavier tails or more extreme values than would be expected in a normal distribution- When the extinguished time increases there will be huge impact on the affected area

Analyzing the Normal Distribution - Residual Plot for Response Time

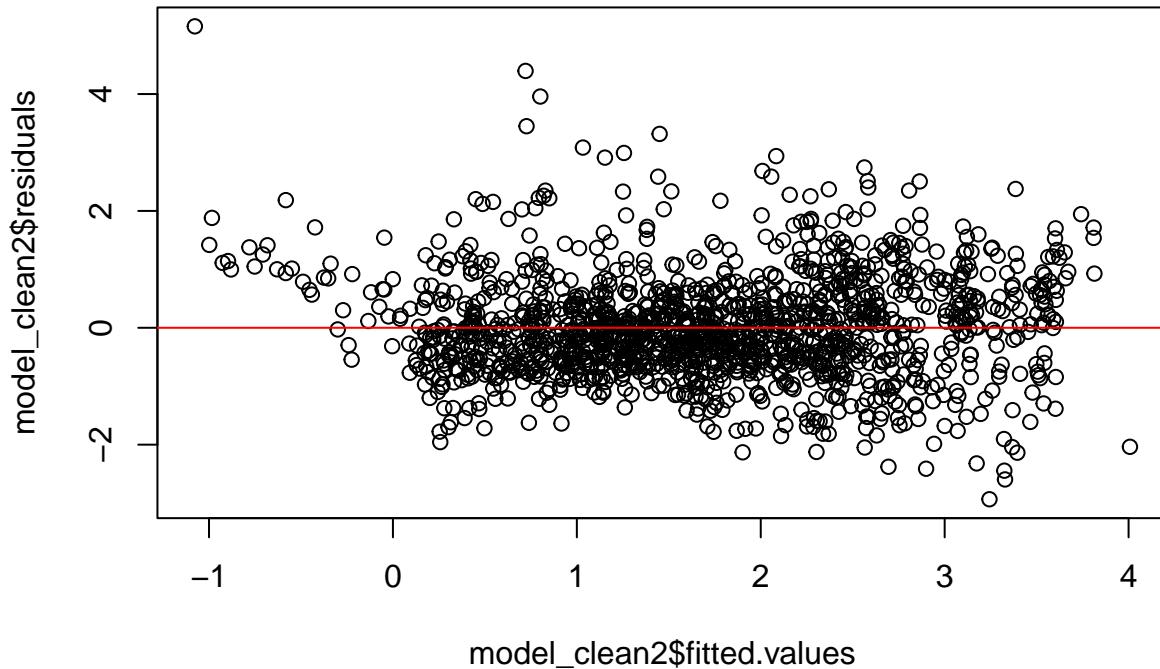
```
plot(model_clean1$fitted.values, model_clean1$residuals)
abline(h = 0, col = "red")
```



```
### Inference: -Residuals seem to be randomly scattered around the horizontal line at zero though there
are many outliers in the data set
```

Analyzing the Normal Distribution - Residual Plot for Extinguished Time

```
plot(model_clean2$fitted.values, model_clean2$residuals)
abline(h = 0, col = "red")
```



Inference:
 -Residuals are centered around zero, which is a good sign for a well-fitted model and shows the strong relationship between time and size

Conclusion

- Overall, Models of Response and Extinguished time has a significant relationship between the affected area - size.
- However, the Extinguished Time Model which shows the strong relationship with the relatively low p-value
- Hence improving the Response and extinguished time of a forest fire involving with following strategies ideally impact the size of the forest being affected
- Early detection,Rapid response,Effective firefighting techniques,Training and Coordination & Public awareness

Question 4 - Does Fire Size Relate To The Weather Conditions When The Fire Starts

The relationship I will be investigating is if there a potential relationship between the recorded fires size once under control and the weather that was present during the fires start. I will have uc_hectares as my dependent variable and I will use a multivariable regression with the temperature, relative humidity, and wind speed as the independent variables.

```
#assign initial dataset to a new dataset, to prevent errors with preexisting code for the other question
data1=fire_data
```

EDA

```
head(data1)
```

```
##   fire_year fire_number fire_name current_size size_class
## 1      2006     PWF001      <NA>       0.10       A
## 2      2006     EWF002      <NA>       0.20       B
## 3      2006     EWF001      <NA>       0.50       B
## 4      2006     EWF003      <NA>       0.01       A
## 5      2006     PWF002      <NA>       0.10       A
## 6      2006     CWF001      <NA>       0.20       B
##   fire_location_latitude fire_location_longitude      fire_origin
## 1             56.24996          -117.1820    Private Land
## 2             53.60637          -115.9157  Provincial Land
## 3             53.61093          -115.5943  Provincial Land
## 4             53.60887          -115.6095  Provincial Land
## 5             56.24996          -117.0502  Provincial Land
## 6             51.15293          -115.0346 Indian Reservation
##   general_cause_desc industry_identifier_desc responsible_group_desc
## 1        Resident           <NA>             Resident
## 2     Incendiary           <NA>  Others (explain in remarks)
## 3     Incendiary           <NA>  Others (explain in remarks)
## 4     Incendiary           <NA>  Others (explain in remarks)
## 5 Other Industry          Waste Disposal            Employees
## 6        Resident           <NA>             Resident
##   activity_class true_cause fire_start_date det_agent_type det_agent
## 1      Grass   Permit Related 2006-04-02 12:00        UNP     310
## 2 Lighting Fires Arson Suspected 2006-04-03 12:10        UNP     310
## 3 Lighting Fires Arson Suspected 2006-04-03 12:15        UNP     310
## 4 Lighting Fires Arson Suspected 2006-04-03 12:10        UNP     PUB
## 5      Refuse   Permit Related 2006-04-03 17:00        UNP     LFS
## 6  Unclassified   Unsafe Fire 2006-04-02 14:25        UNP     310
##   discovered_date discovered_size reported_date dispatched_resource
## 1           <NA>              NA 2006-04-02 20:46      FPD Staff
## 2           <NA>              NA 2006-04-03 12:27      FPD Staff
## 3           <NA>              NA 2006-04-03 12:36      FPD Staff
## 4           <NA>              NA 2006-04-03 13:23      FPD Staff
## 5 2006-04-03 19:11              NA 2006-04-03 19:12      FPD Staff
## 6 2006-04-02 14:27              NA 2006-04-02 14:30      FPD Staff
##   dispatch_date start_for_fire_date assessment_resource assessment_datetime
## 1 2006-04-02 21:10      2006-04-02 21:20      IA Forces 2006-04-02 22:00
## 2 2006-04-03 12:33      2006-04-03 12:35      IA Forces 2006-04-03 13:20
## 3 2006-04-03 12:36      2006-04-03 12:42      IA Forces 2006-04-03 13:23
## 4 2006-04-03 13:50      2006-04-03 13:50      IA Forces 2006-04-03 14:08
## 5 2006-04-03 19:19      2006-04-03 19:22          Other 2006-04-03 19:57
## 6 2006-04-02 14:40      2006-04-02 14:45      IA Forces 2006-04-02 16:00
##   assessment_hectares fire_spread_rate fire_type fire_position_on_slope
## 1             0.01          0.0 Surface           Flat
## 2             0.20          0.0 Surface        Lower 1/3
## 3             0.50          0.0 Surface           Bottom
## 4             0.01          0.0 Surface           Flat
## 5             0.10          0.1 Surface           Flat
## 6             0.20          0.0 Surface           Flat
```

```

##   weather_conditions_over_fire temperature relative_humidity wind_direction
## 1                         Clear      18            10          SW
## 2                         Clear      12            22          SW
## 3                         Clear      12            22          SW
## 4                         Clear      12            22          SW
## 5                         Clear       6            37          SW
## 6                         Clear     11            32           S
##   wind_speed fuel_type initial_action_by ia_arrival_at_fire_date ia_access
## 1        2       01a      Land Owner             <NA>      <NA>
## 2       10       01a    Fire Department           <NA>      <NA>
## 3       10       01a    Fire Department           <NA>      <NA>
## 4       10       01b      Industry             <NA>      <NA>
## 5        2      <NA>    Fire Department           <NA>      <NA>
## 6       20       01b    Fire Department           <NA>      <NA>
##   fire_fighting_start_date fire_fighting_start_size bucketing_on_fire
## 1                  <NA>                 NA      <NA>
## 2                  <NA>                 NA      <NA>
## 3                  <NA>                 NA      <NA>
## 4                  <NA>                 NA      <NA>
## 5                  <NA>                 NA      <NA>
## 6                  <NA>                 NA      <NA>
##   distance_from_water_source first_bucket_drop_date      bh_fs_date
## 1                      NA             <NA> 2006-04-02 22:00
## 2                      NA             <NA> 2006-04-03 13:20
## 3                      NA             <NA> 2006-04-03 13:23
## 4                      NA             <NA> 2006-04-03 14:08
## 5                      NA             <NA> 2006-04-03 19:57
## 6                      NA             <NA> 2006-04-02 16:00
##   bh_hectares      uc_fs_date uc_hectares      to_fs_date to_hectares
## 1     0.01 2006-04-02 22:00      0.01      <NA>      NA
## 2     0.20 2006-04-03 13:20      0.20      <NA>      NA
## 3     0.50 2006-04-03 13:23      0.50      <NA>      NA
## 4     0.01 2006-04-03 14:08      0.01      <NA>      NA
## 5     0.10 2006-04-03 20:19      0.10 2006-04-03 20:20      0.1
## 6     0.20 2006-04-02 16:00      0.20      <NA>      NA
##   ex_fs_date ex_hectares
## 1 2006-04-03 10:20      0.10
## 2 2006-04-03 14:00      0.20
## 3 2006-04-03 15:00      0.50
## 4 2006-04-03 15:05      0.01
## 5 2006-04-05 10:18      0.10
## 6 2006-04-03 18:00      0.20

```

Based off of the potential relationship I wish to investigate I will create a new data frame with the columns of interest.

```

reg_df=data.frame(uc_hectares=data1$uc_hectares,temperature=data1$temperature,relative_humidity=data1$relative_humidity)
summary(reg_df)

```

```

##   uc_hectares      temperature      relative_humidity      wind_speed
## Min.   : 0.0   Min.   :-39.00   Min.   : 0.00   Min.   : 0.000
## 1st Qu.: 0.0   1st Qu.: 14.00   1st Qu.: 31.00   1st Qu.: 3.000
## Median : 0.0   Median : 19.00   Median : 40.00   Median : 6.000

```

```

##  Mean    : 245.6   Mean    : 17.85   Mean    : 45.38   Mean    : 8.813
##  3rd Qu.: 0.3     3rd Qu.: 23.00   3rd Qu.: 56.00   3rd Qu.: 12.000
##  Max.    : 707648.0 Max.    : 39.90   Max.    : 100.00  Max.    : 90.000
##                NA's    : 2820     NA's    : 2822     NA's    : 2823

```

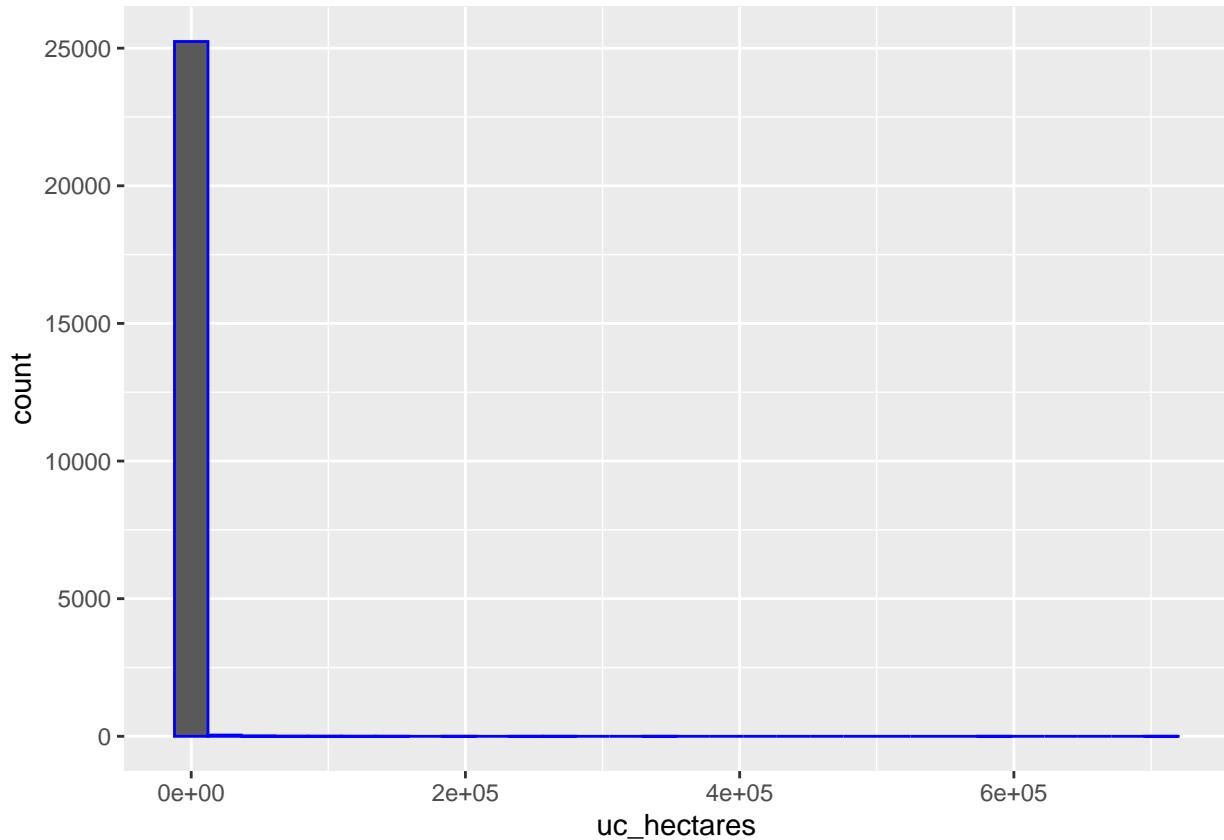
Based off of the summary stats the dependent variable uc_hectares seems to be strongly skewed. To explore the data I will create histograms to display the data's distribution.

```

ggplot(data = reg_df, aes(x = uc_hectares)) +
  geom_histogram(color='blue')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



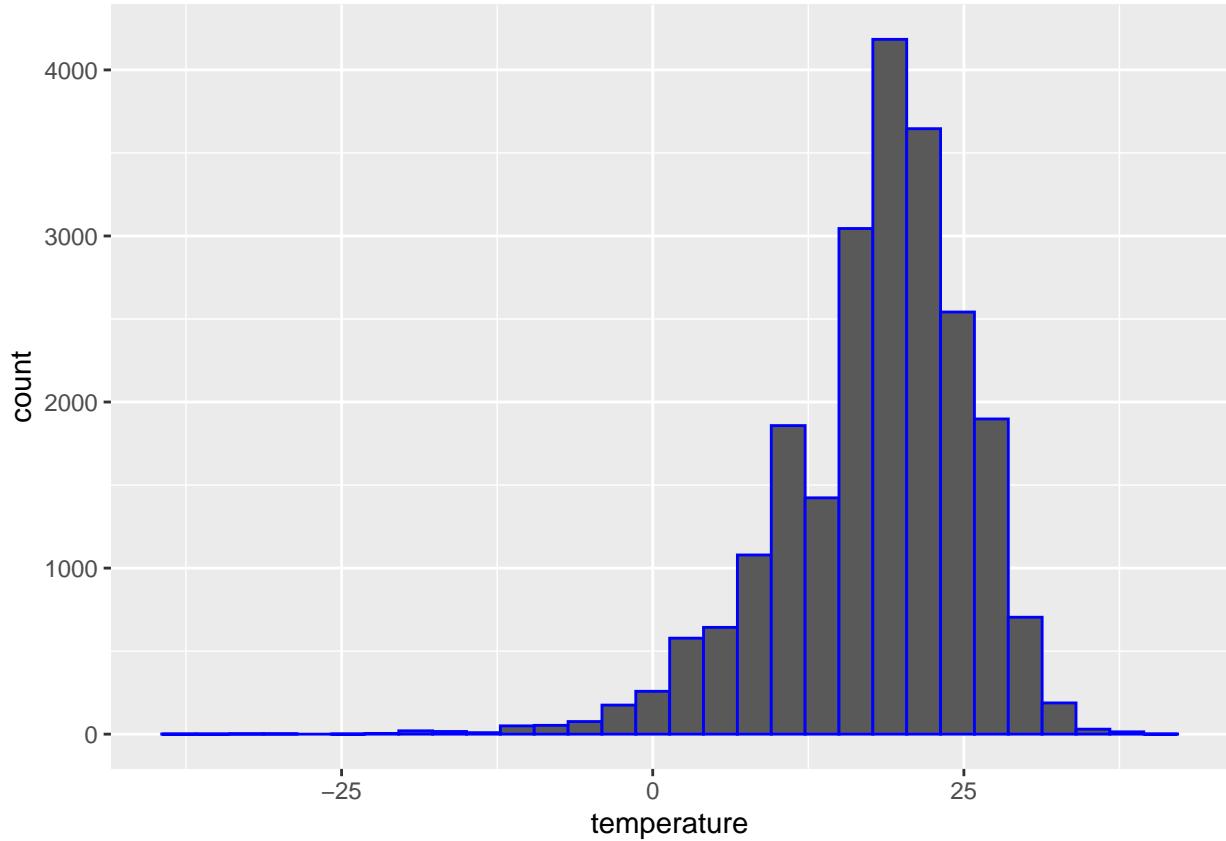
```

ggplot(data = reg_df, aes(x = temperature)) +
  geom_histogram(color='blue')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2820 rows containing non-finite outside the scale range
## (`stat_bin()`).

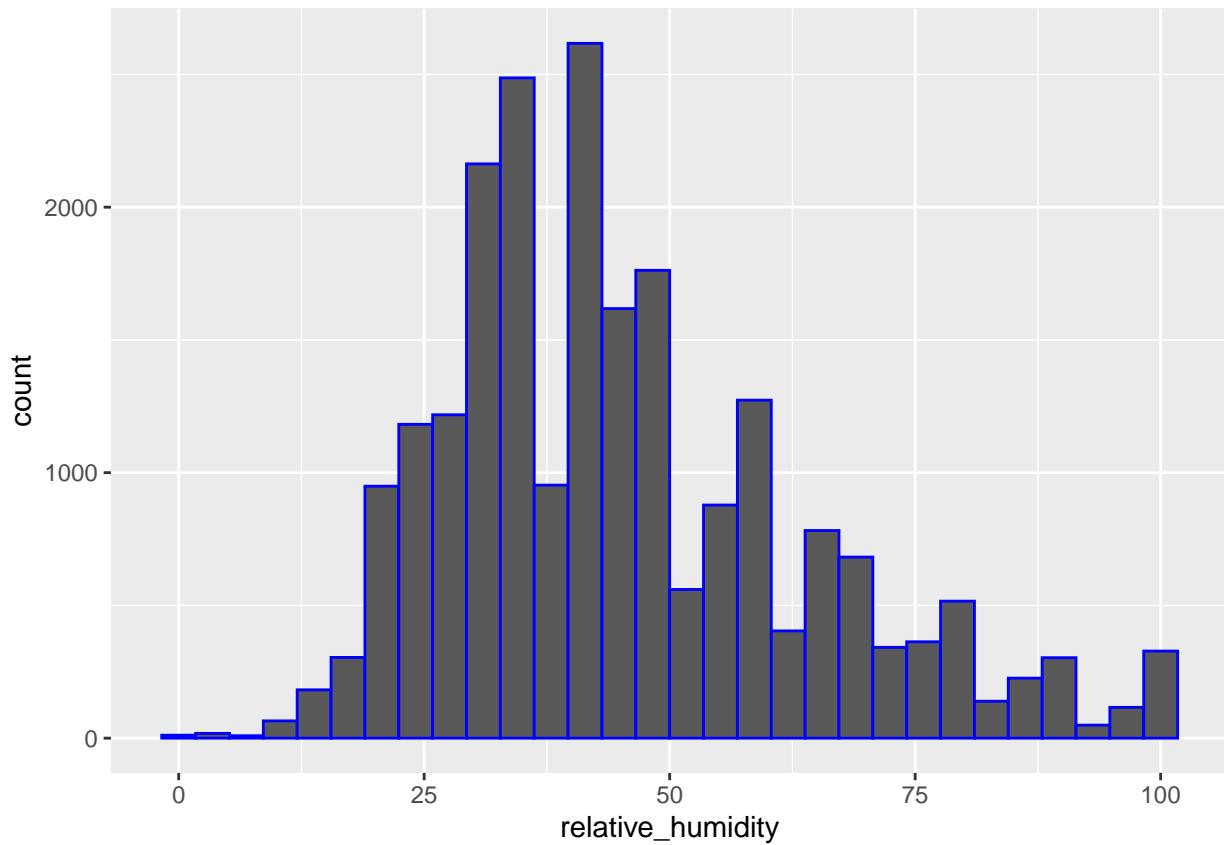
```



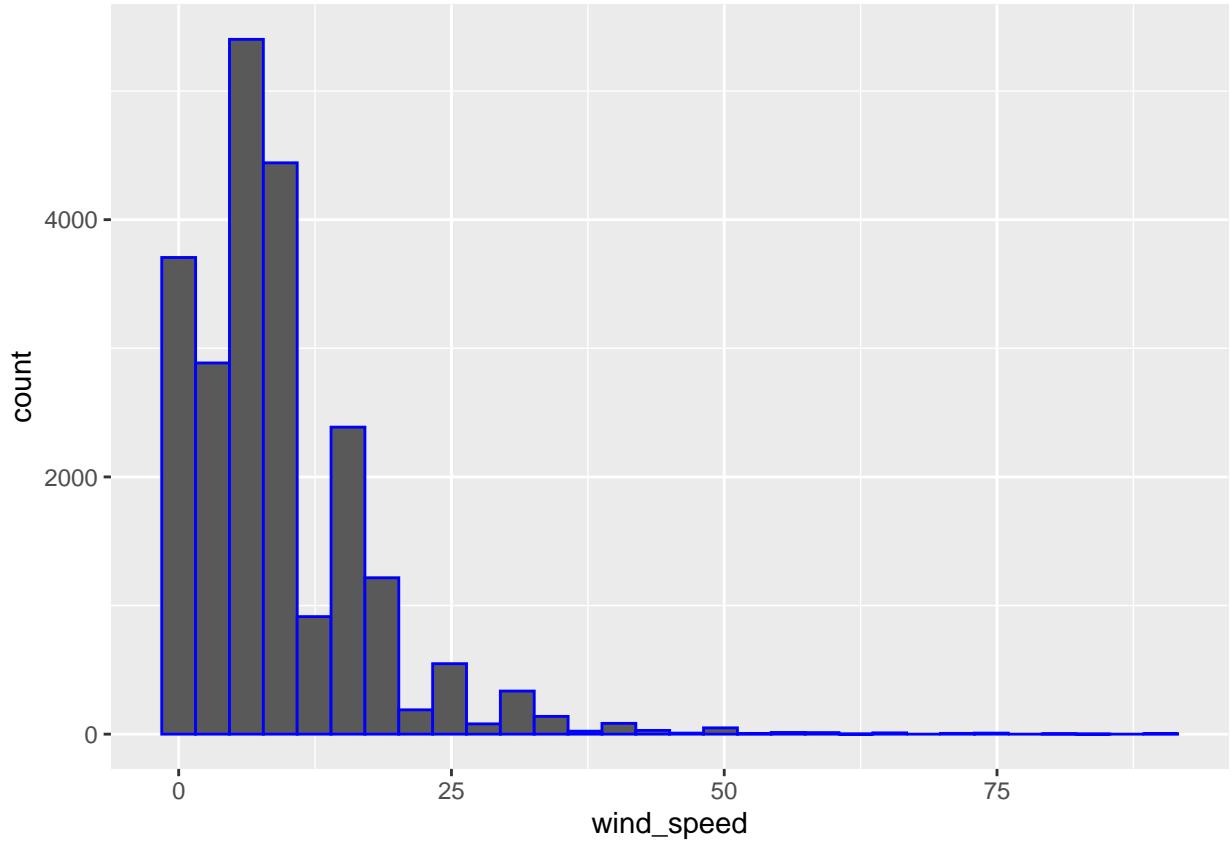
```
ggplot(data = reg_df, aes(x = relative_humidity)) +
  geom_histogram(color='blue')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2822 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



```
ggplot(data = reg_df, aes(x = wind_speed)) +  
  geom_histogram(color='blue')  
  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.  
  
## Warning: Removed 2823 rows containing non-finite outside the scale range  
## (`stat_bin()`).
```



Unsurprisingly the distribution of the uc_hectares variable is greatly skewed confirming what could be seen in the summary statistics.

Regression Analysis

```
reglarge=lm(formula=reg_df$uc_hectares~reg_df$temperature+reg_df$relative_humidity+reg_df$wind_speed)
summary(reglarge)

##
## Call:
## lm(formula = reg_df$uc_hectares ~ reg_df$temperature + reg_df$relative_humidity +
##     reg_df$wind_speed)
##
## Residuals:
##      Min      1Q Median      3Q      Max 
## -3902   -479   -220      29  705357 
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             -279.842    229.400  -1.220 0.222521    
## reg_df$temperature       23.514     7.070   3.326 0.000883 ***  
## reg_df$relative_humidity -5.442     2.927  -1.860 0.062956 .    
## reg_df$wind_speed        42.551     6.231   6.829 8.76e-12 ***  
## ---                        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

## 
## Residual standard error: 7734 on 22492 degrees of freedom
##   (2825 observations deleted due to missingness)
## Multiple R-squared:  0.003188, Adjusted R-squared:  0.003055
## F-statistic: 23.98 on 3 and 22492 DF, p-value: 1.733e-15

```

To start I'll state the hypothesis test for all coefficients, H0: B=0 The coefficient in question is not statistically significant from 0 H1: B!=0 The coefficient is statistically significant from 0

The intercept is -279.8 hectares if all other variables are 0 which does not make sense and it is unsurprisingly not significant with a P value of 0.22 so we cannot reject the null hypothesis. The coefficient for temperature, 23.5 hectares per degree increase in temperature has a P value of 0.00088 allowing us to state that is significant and we can reject the null hypothesis. The coefficient for relative humidity is -5.44 hectares per %increase in humidity, with a P value of 0.062 the coefficient is not significant at the 5% level but is close. Under the stated hypothesis test we cannot reject the null hypothesis. Lastly The coefficient for wind speed is 42.5 hectares burned per 1 kilometer per hour increase. With a P value of near 0 the coefficient is significant and we can reject the null hypothesis.

With an adj R-squared of 0.003055 indicating the regression only explains 0.32% of variation in uc_hectares is explained by this model, this indicates that the estimated model is likely a very poor fit for the data.

```

reg_cor= round(cor(reg_df,use = "complete.obs"), 2)
reg_cor

```

	uc_hectares	temperature	relative_humidity	wind_speed
## uc_hectares	1.00	0.03	-0.03	0.05
## temperature	0.03	1.00	-0.28	-0.02
## relative_humidity	-0.03	-0.28	1.00	-0.17
## wind_speed	0.05	-0.02	-0.17	1.00

From the correlation matrix we can see that there is very little correlation between uc_hectares and any of the independent variables. To better visualize this I will create scatter plots with estimated models for each independent variable and uc_hectares as visualizing a multivariate regression model is difficult.

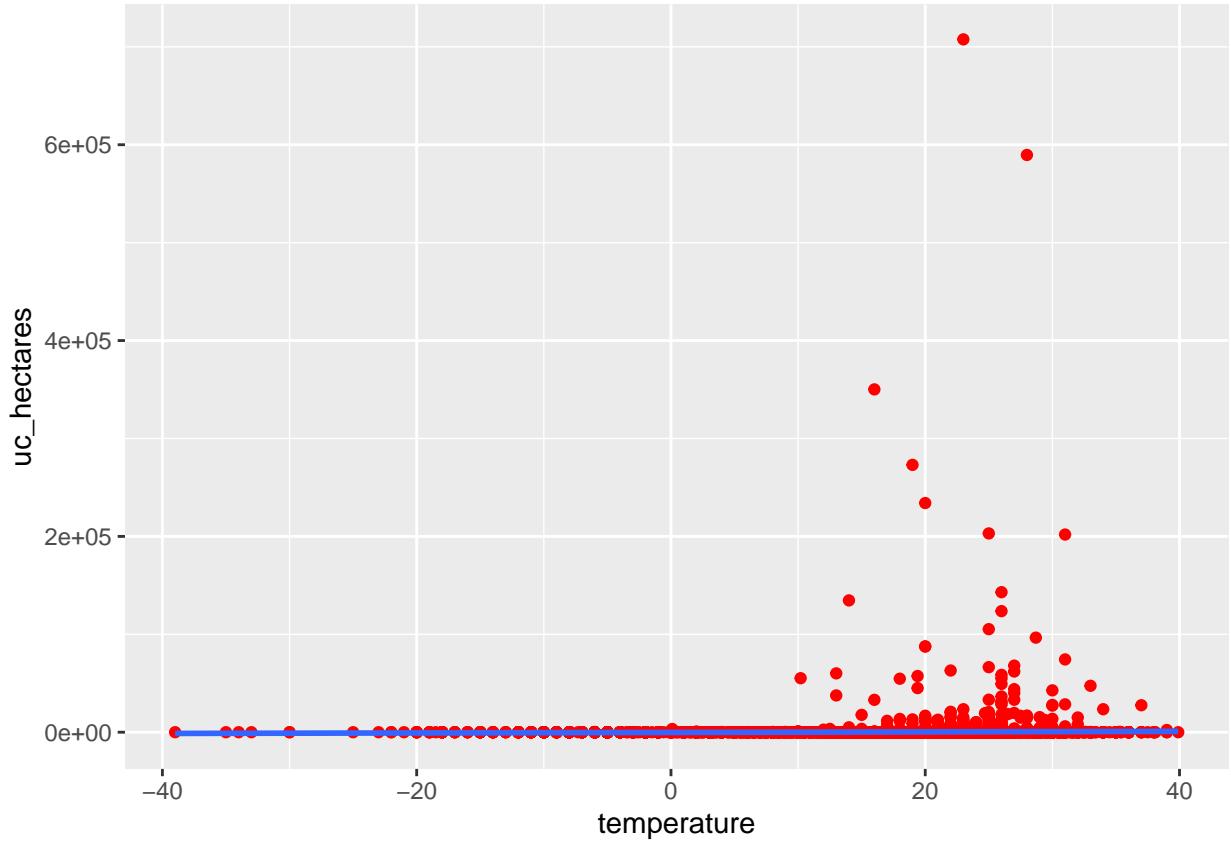
```

ggplot(reg_df, aes(x=temperature, y=uc_hectares )) +
  geom_point(color = "red") + stat_smooth(method = "lm",
                                             formula = y ~ x, geom = "smooth")

## Warning: Removed 2820 rows containing non-finite outside the scale range
## ('stat_smooth()').

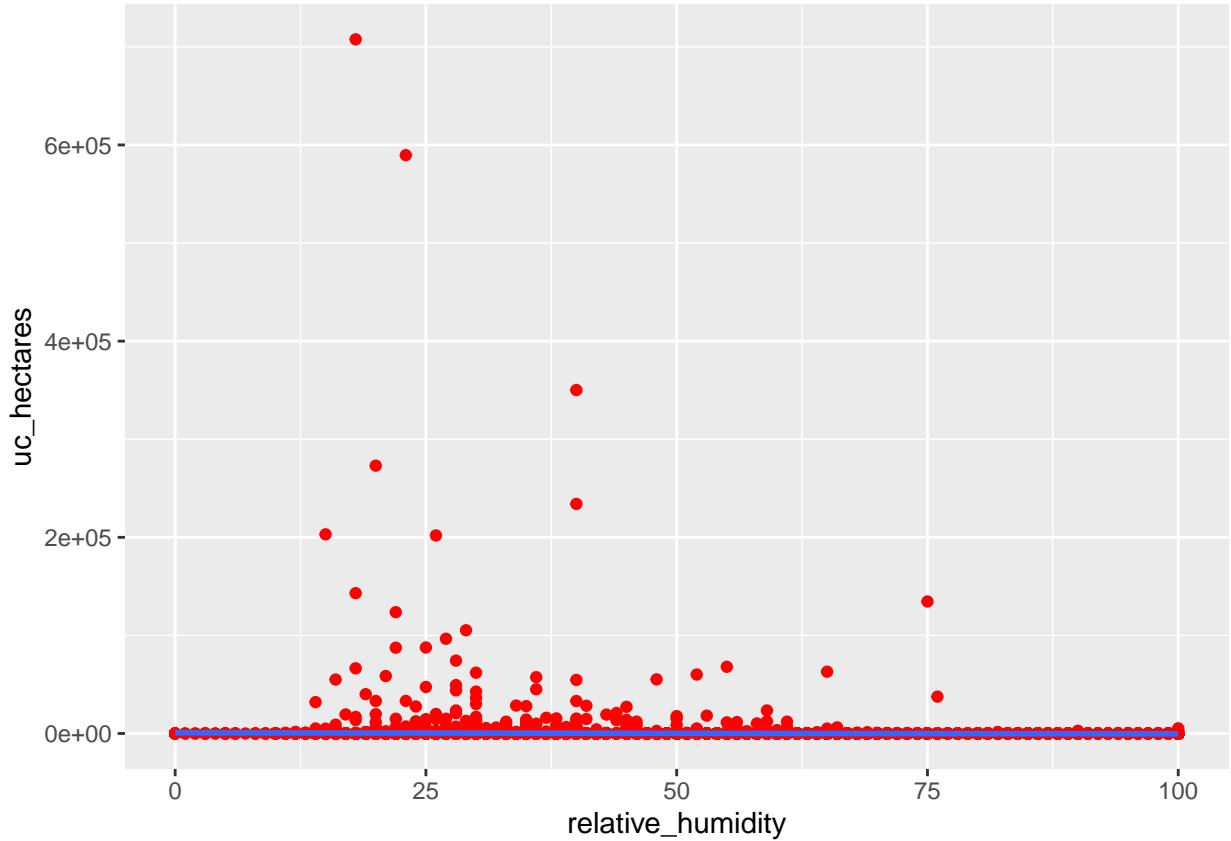
## Warning: Removed 2820 rows containing missing values or values outside the scale range
## ('geom_point()').

```



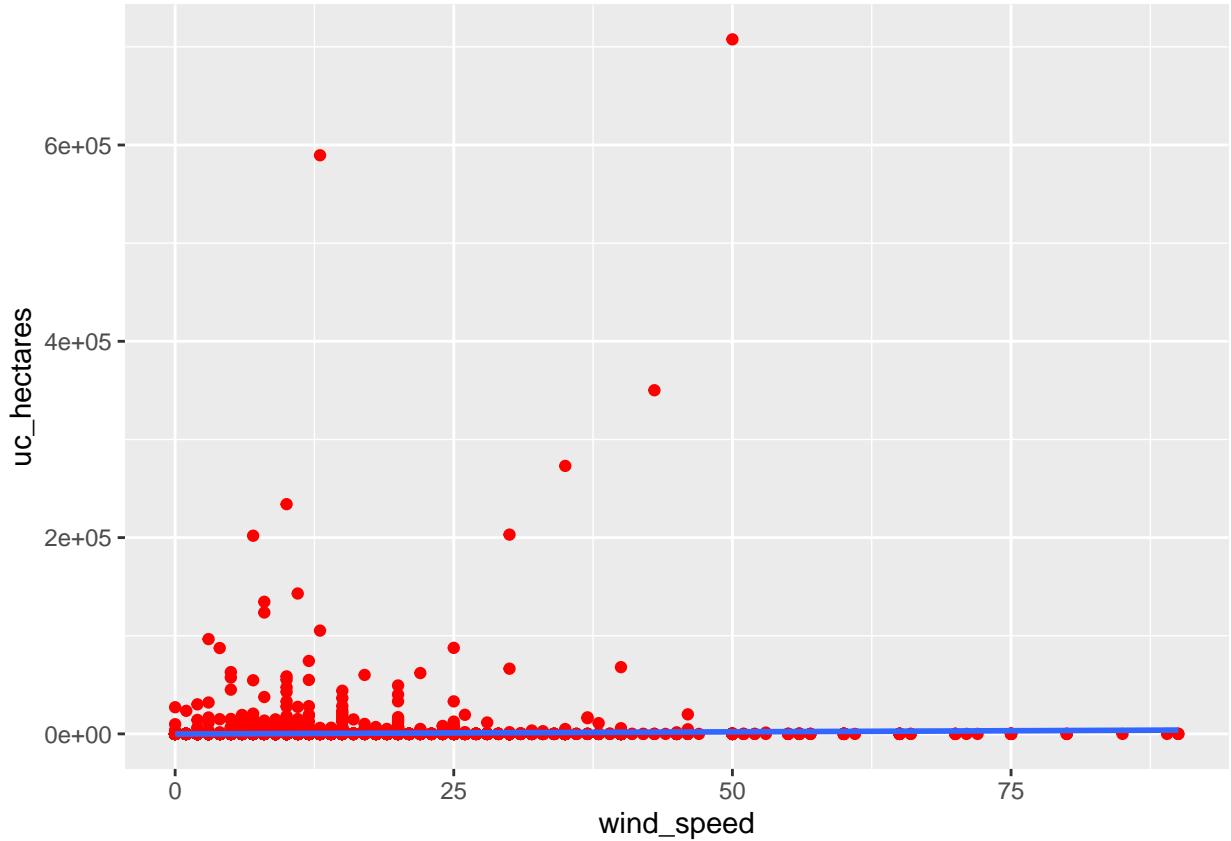
```
ggplot(reg_df, aes(x=relative_humidity, y=uc_hectares )) +  
  geom_point(color = "red") + stat_smooth(method = "lm",  
    formula = y ~ x, geom = "smooth")
```

```
## Warning: Removed 2822 rows containing non-finite outside the scale range  
## ('stat_smooth()').  
  
## Warning: Removed 2822 rows containing missing values or values outside the scale range  
## ('geom_point()').
```

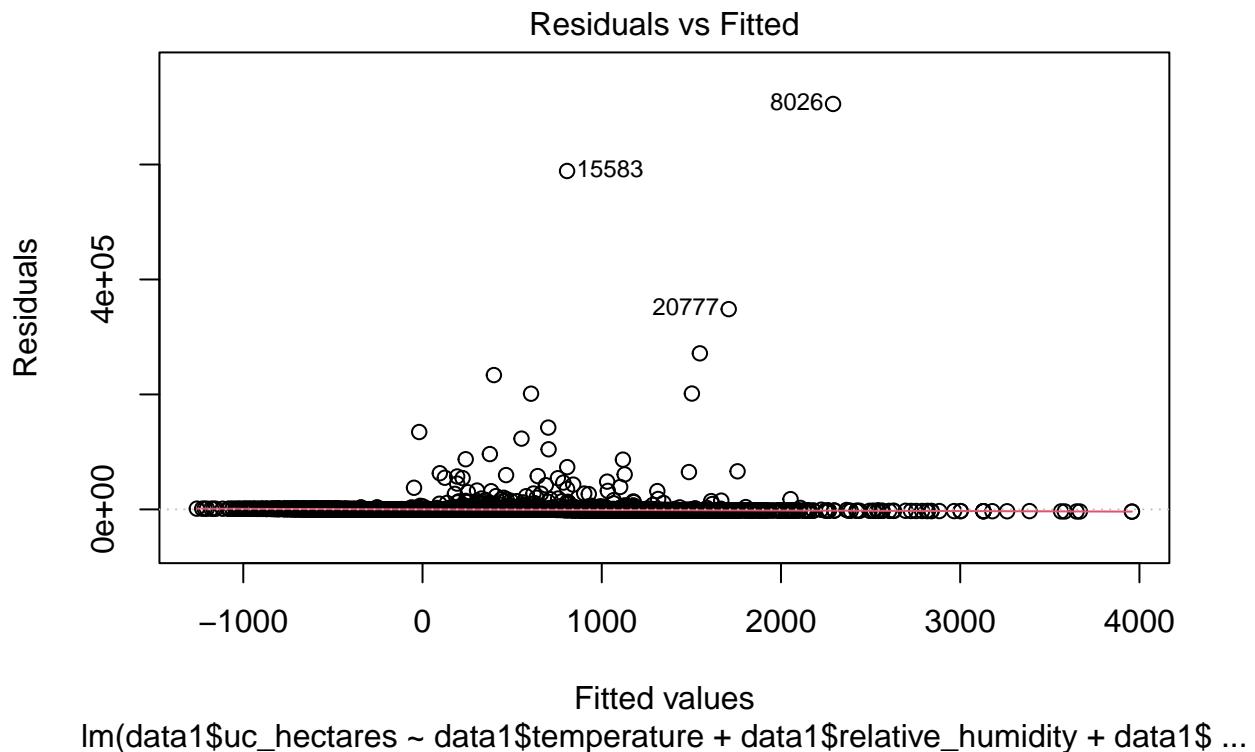


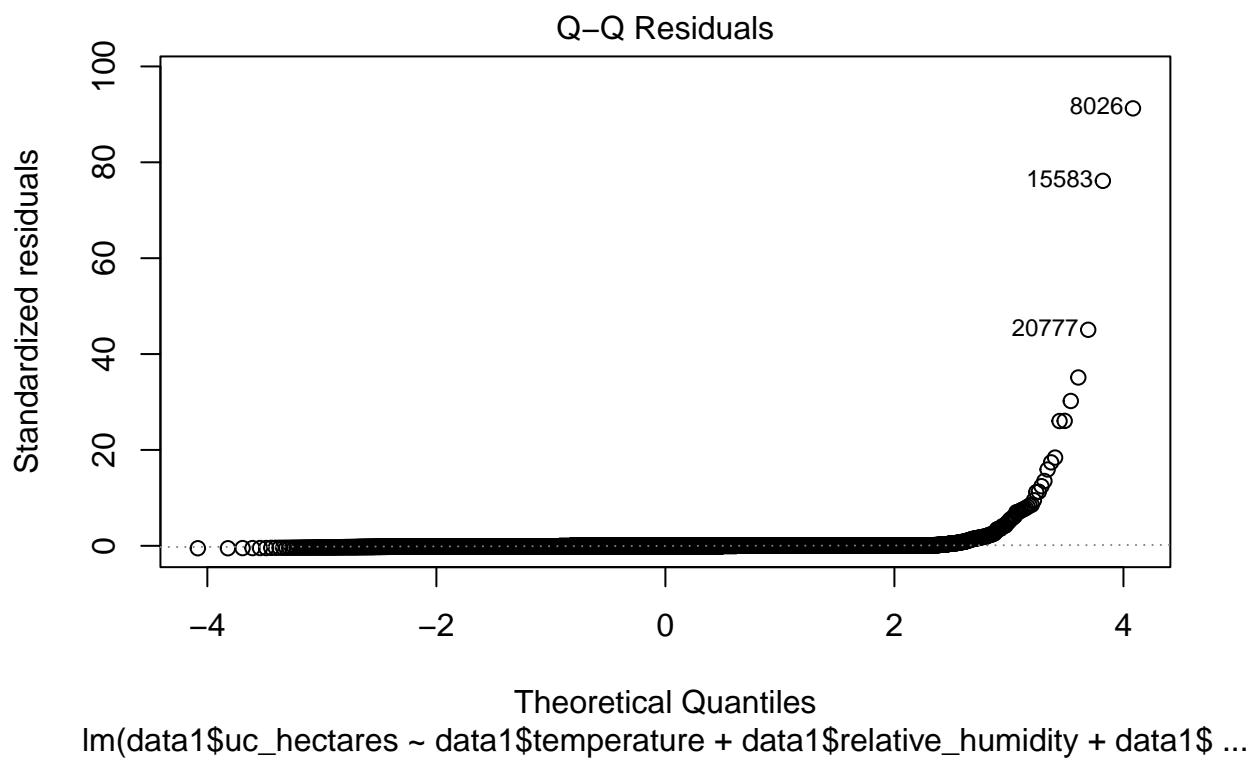
```
ggplot(reg_df, aes(x=wind_speed, y=uc_hectares)) +
  geom_point(color = "red") + stat_smooth(method = "lm",
  formula = y ~ x, geom = "smooth")
```

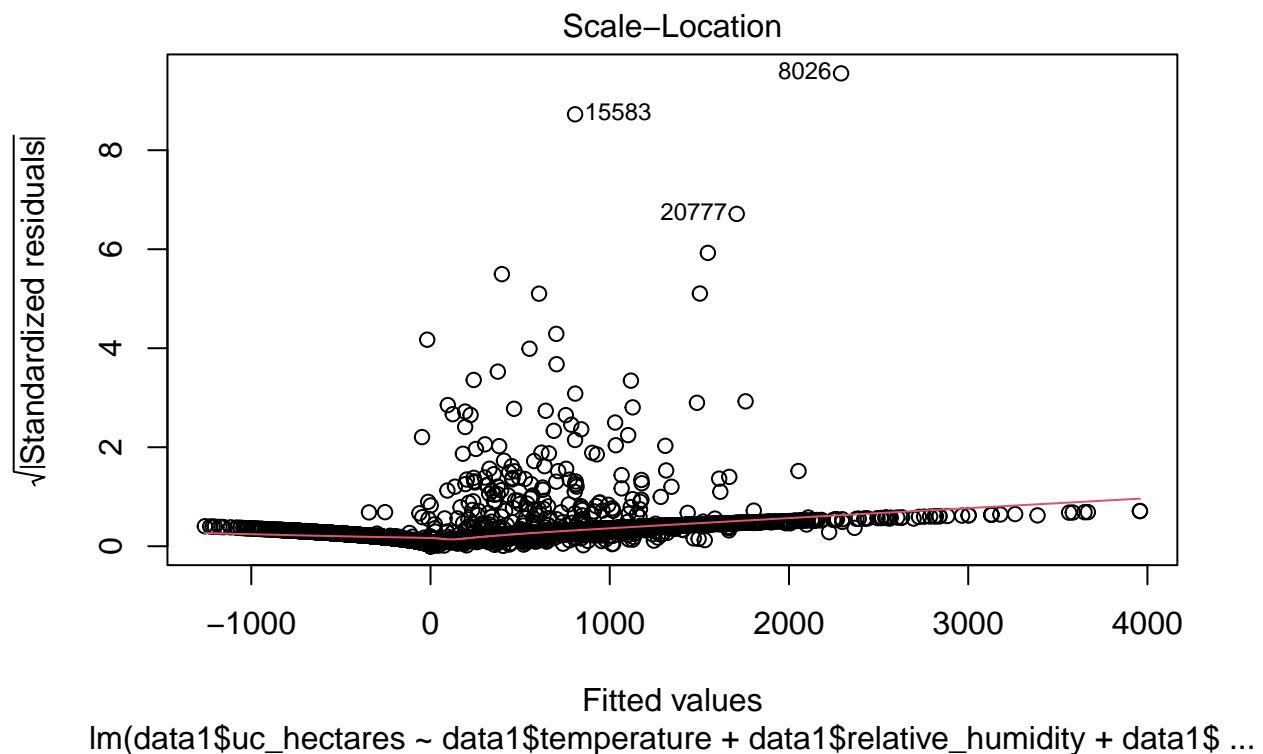
```
## Warning: Removed 2823 rows containing non-finite outside the scale range
## ('stat_smooth()').
## Warning: Removed 2823 rows containing missing values or values outside the scale range
## ('geom_point()').
```

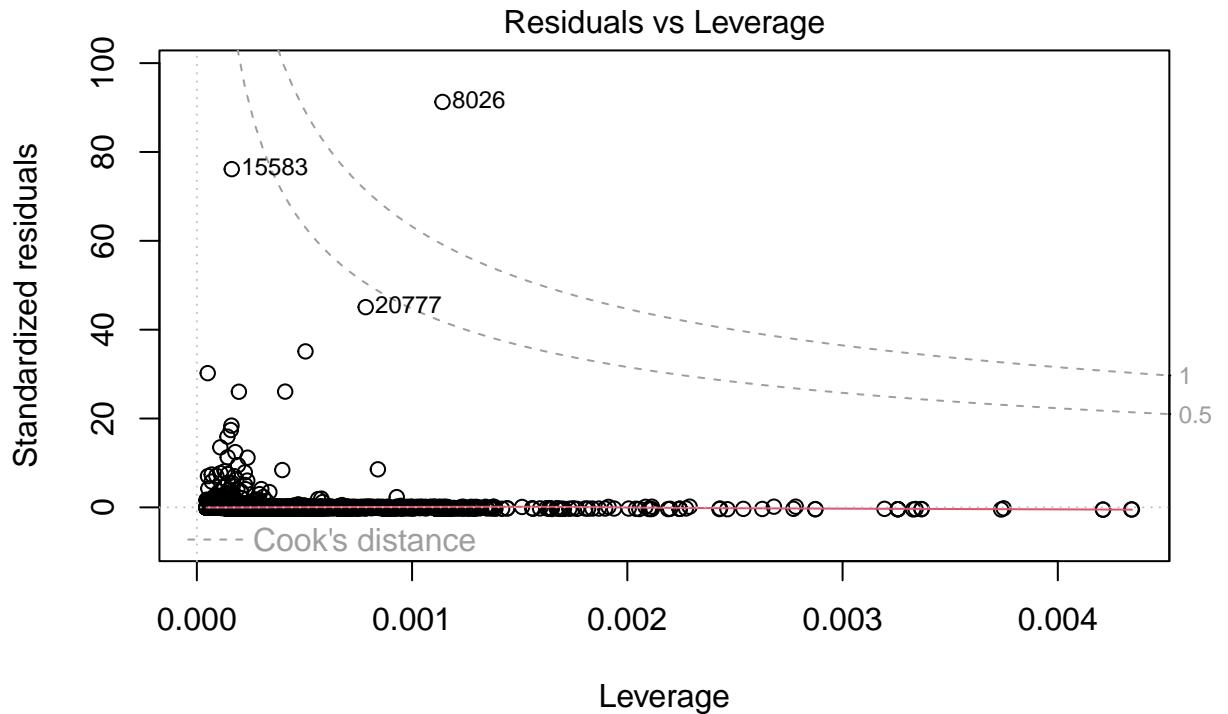


```
plot(lm(formula=uc_hectares~temperature+relative_humidity+wind_speed))
```









The plots above show that the residuals are not independent or normally distributed. The first plot shows that there are some large outliers in the data and the normality plot also shows that the the residuals are not normally distributed. This indicates that the model is likely unreliable. A log log transformation could perhaps improve this somewhat.

```

reg_dflog=data.frame(uc_hectares=log10(data1$uc_hectares),temperature=data1$temperature,relative_humidi

reglog=lm(formula=reg_dflog$uc_hectares~reg_dflog$temperature+reg_dflog$relative_humidity+reg_dflog$wind
summary(reglog)

## 
## Call:
## lm(formula = reg_dflog$uc_hectares ~ reg_dflog$temperature +
##     reg_dflog$relative_humidity + reg_dflog$wind_speed)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -3.1129 -0.8514 -0.2214  0.6036  6.4842 
## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             -1.2367098  0.0328594 -37.64   <2e-16 ***
## reg_dflog$temperature    0.0193262  0.0010127  19.08   <2e-16 ***
## reg_dflog$relative_humidity -0.0079249  0.0004192 -18.91   <2e-16 ***
## reg_dflog$wind_speed      0.0256644  0.0008925  28.76   <2e-16 ***
## ---
```

```

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.108 on 22492 degrees of freedom
##   (2825 observations deleted due to missingness)
## Multiple R-squared:  0.08394,   Adjusted R-squared:  0.08382
## F-statistic:    687 on 3 and 22492 DF,  p-value: < 2.2e-16

```

Transforming the data in log format did improve the regression and the P values of all coefficients are below 0.05 allowing us to state that all 3 coefficients significant along with the intercept but still does not make sense in the context of the data. With an adj R-squared of 0.083 the fit of the model does improve in this case drastically from the first model.

```

ggplot(reg_dflog, aes(x=temperature, y=uc_hectares )) +
  geom_point(color = "red") + stat_smooth(method = "lm",
                                           formula = y ~ x, geom = "smooth")

```

```

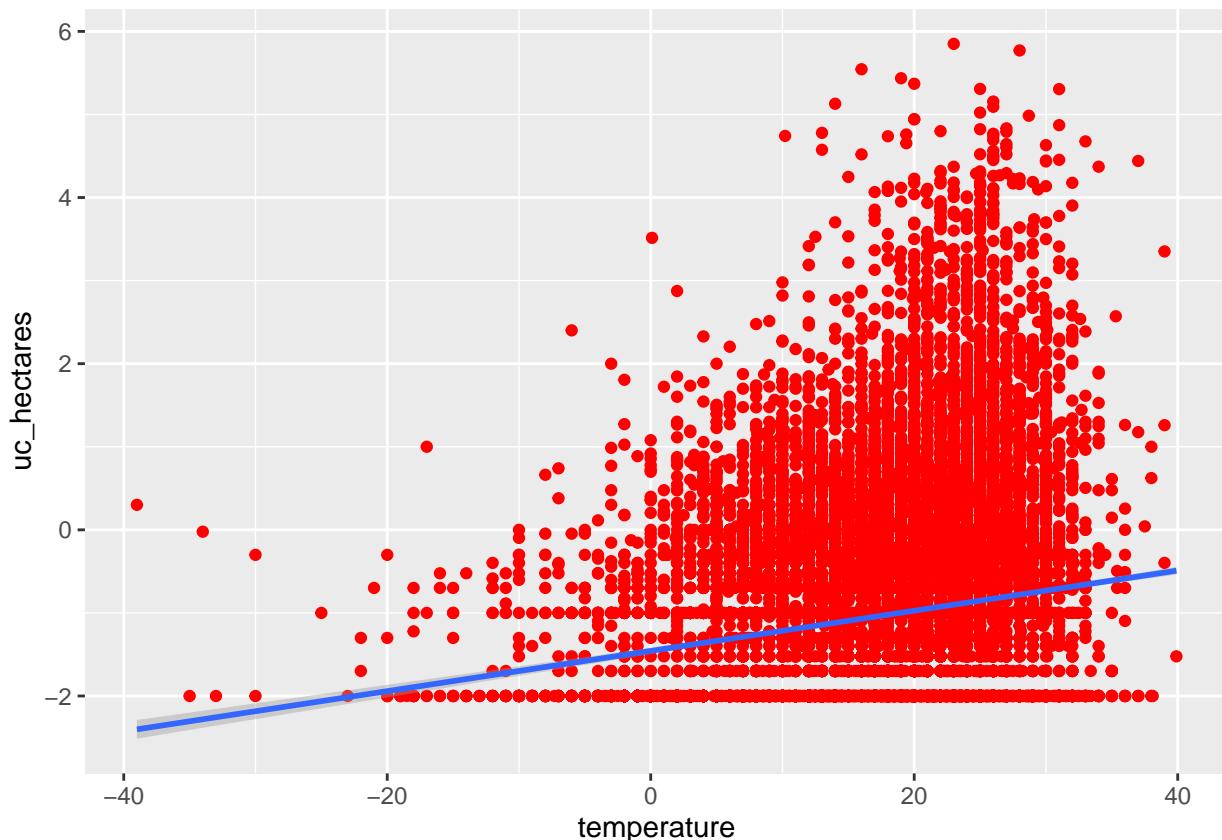
## Warning: Removed 2820 rows containing non-finite outside the scale range
## ('stat_smooth()').

```

```

## Warning: Removed 2820 rows containing missing values or values outside the scale range
## ('geom_point()').

```



```

ggplot(reg_dflog, aes(x=relative_humidity, y=uc_hectares )) +
  geom_point(color = "red") + stat_smooth(method = "lm",
                                           formula = y ~ x, geom = "smooth")

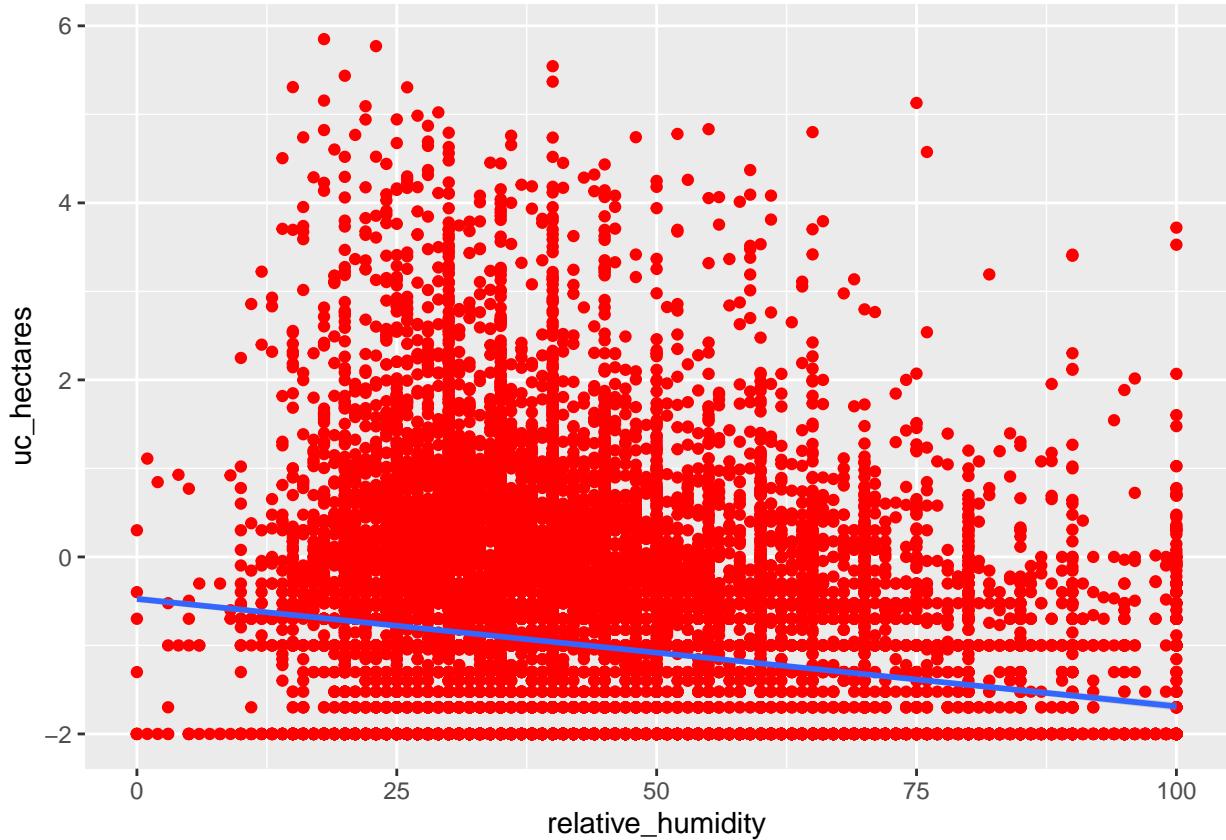
```

```

## Warning: Removed 2822 rows containing non-finite outside the scale range
## ('stat_smooth()').

## Warning: Removed 2822 rows containing missing values or values outside the scale range
## ('geom_point()').

```



```

ggplot(reg_dflog, aes(x=wind_speed, y=uc_hectares)) +
  geom_point(color = "red") + stat_smooth(method = "lm",
                                           formula = y ~ x, geom = "smooth")

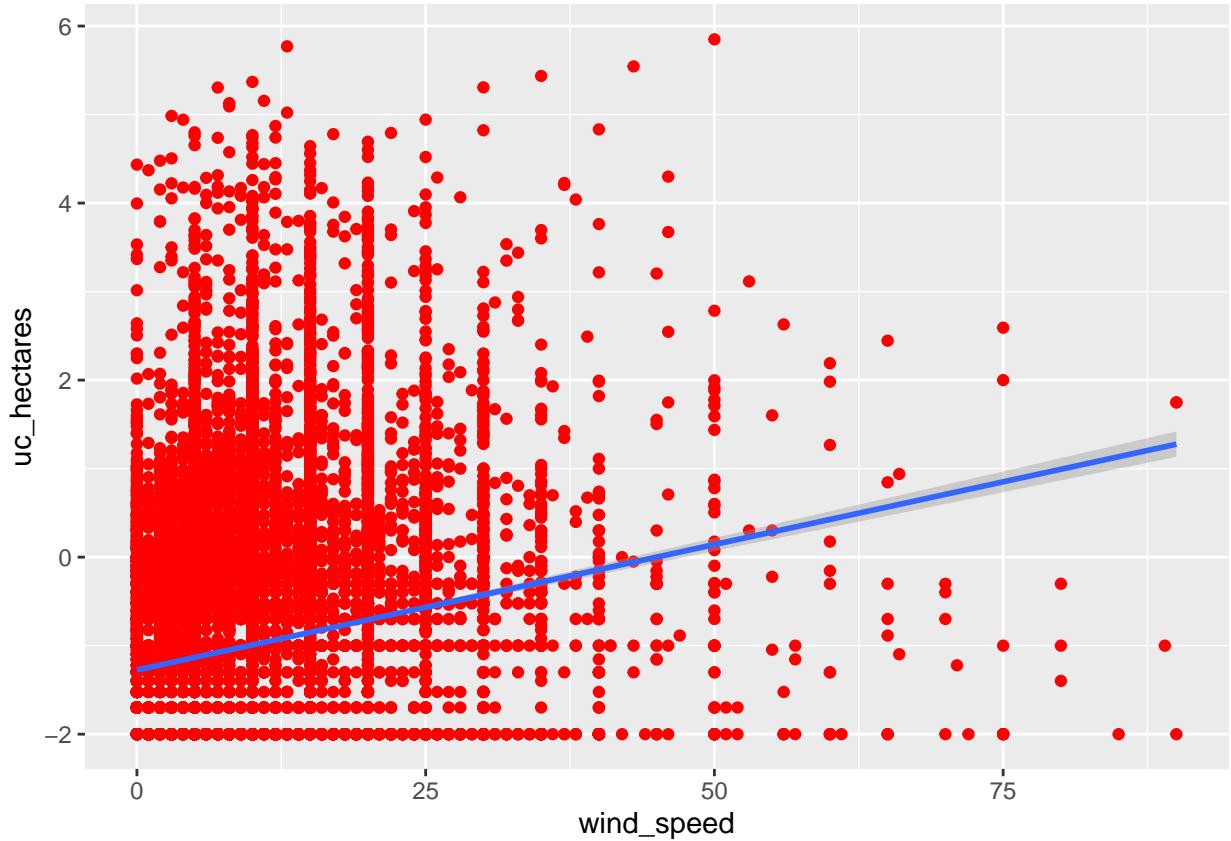
```

```

## Warning: Removed 2823 rows containing non-finite outside the scale range
## ('stat_smooth()').

## Warning: Removed 2823 rows containing missing values or values outside the scale range
## ('geom_point()').

```

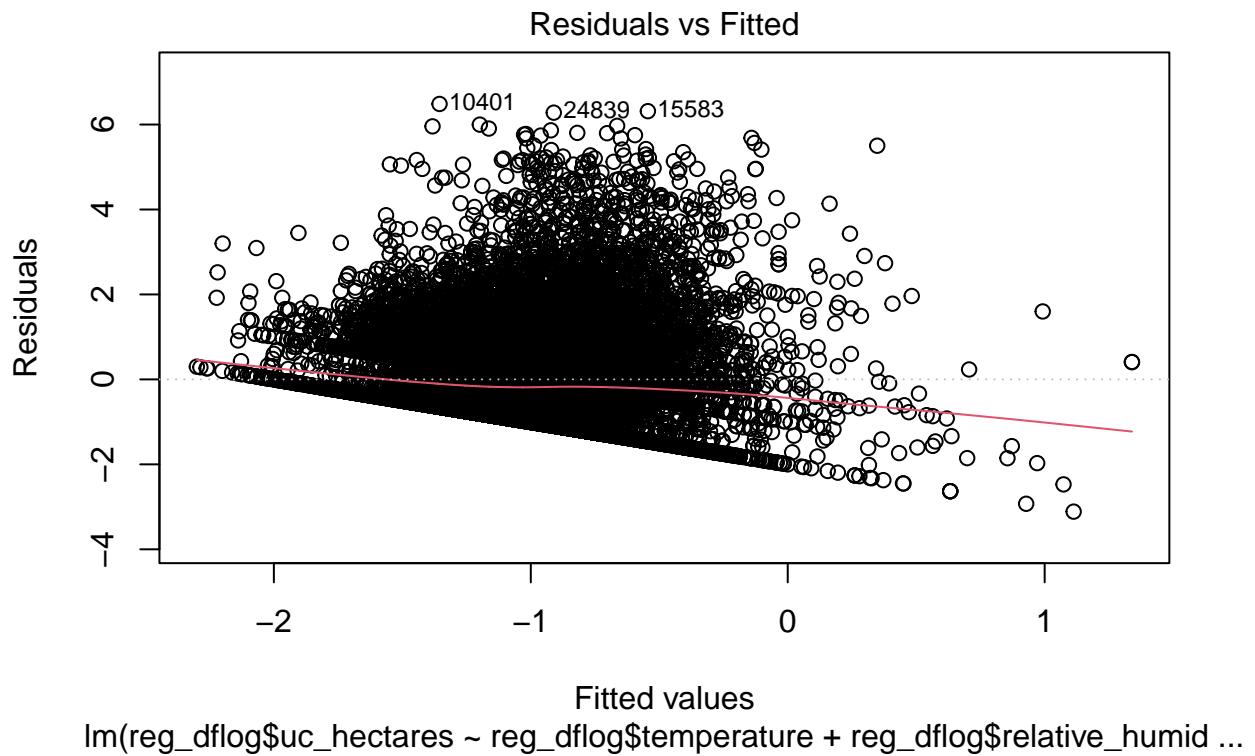


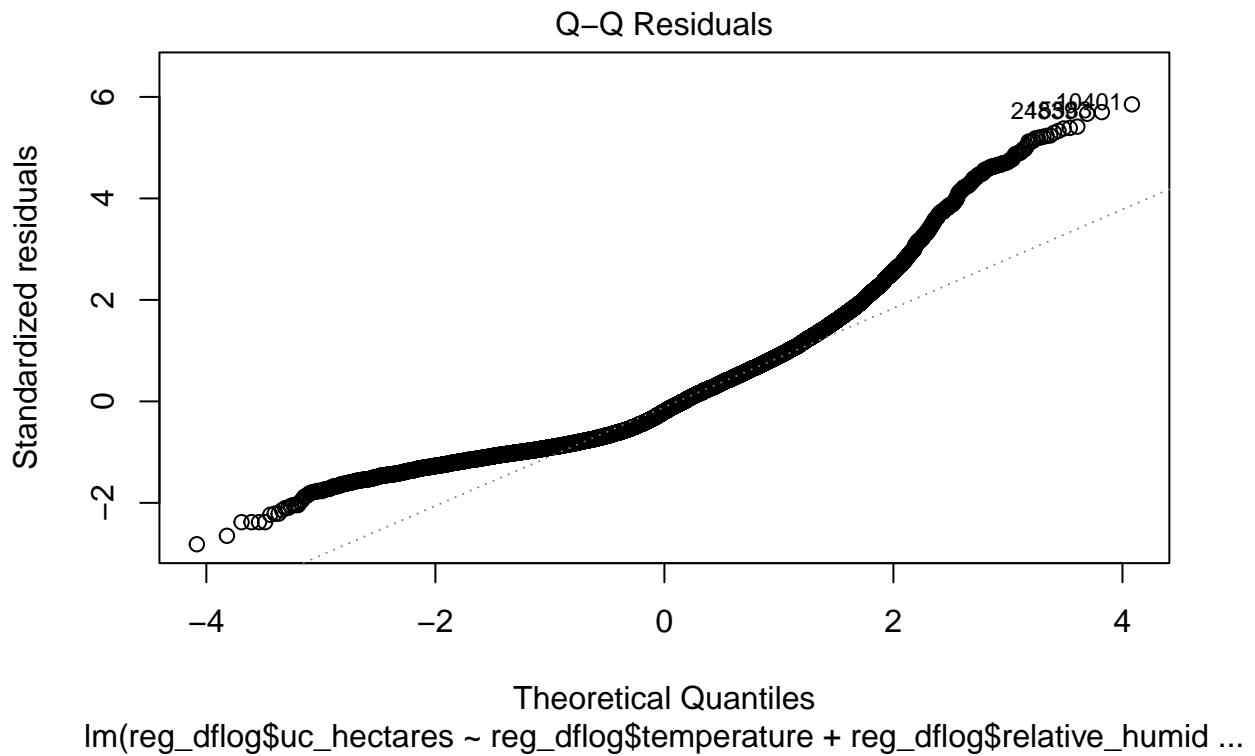
```
reg_corlog= round(cor(reg_dflog,use = "complete.obs"), 2)
reg_corlog
```

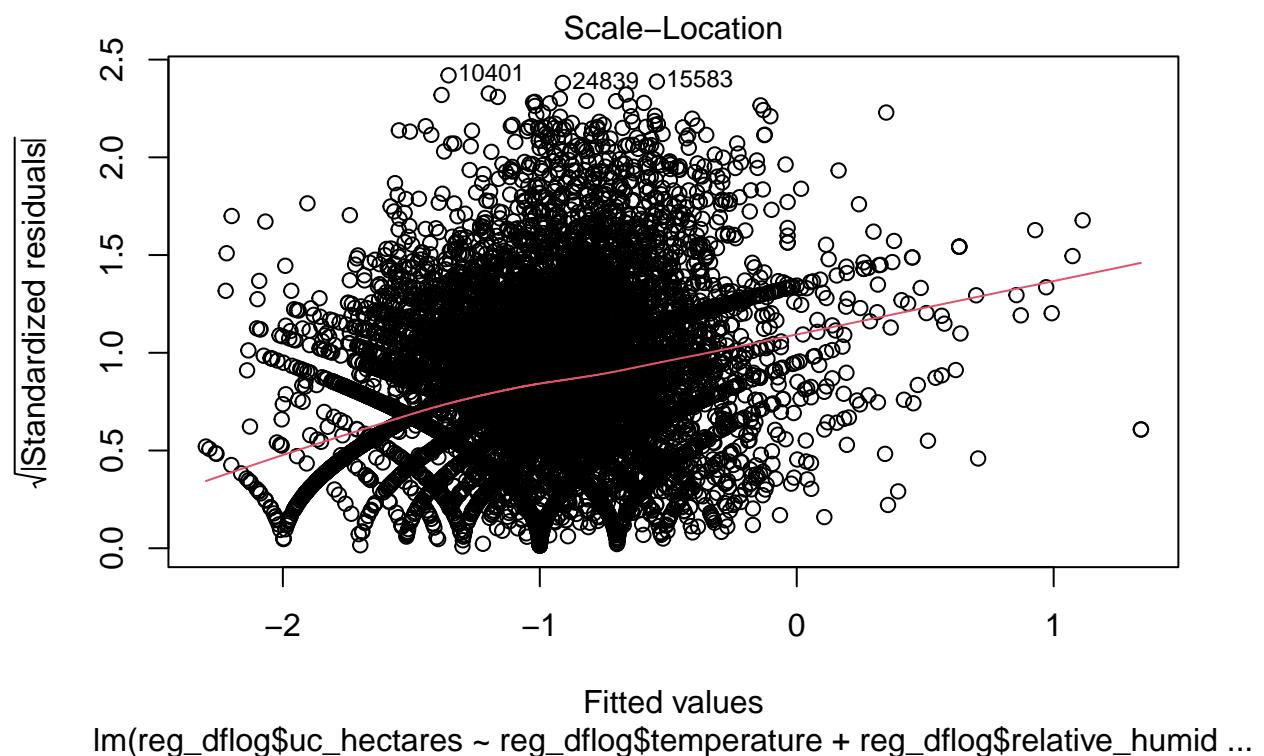
	uc_hectares	temperature	relative_humidity	wind_speed
## uc_hectares	1.00	0.16	-0.20	0.21
## temperature	0.16	1.00	-0.28	-0.02
## relative_humidity	-0.20	-0.28	1.00	-0.17
## wind_speed	0.21	-0.02	-0.17	1.00

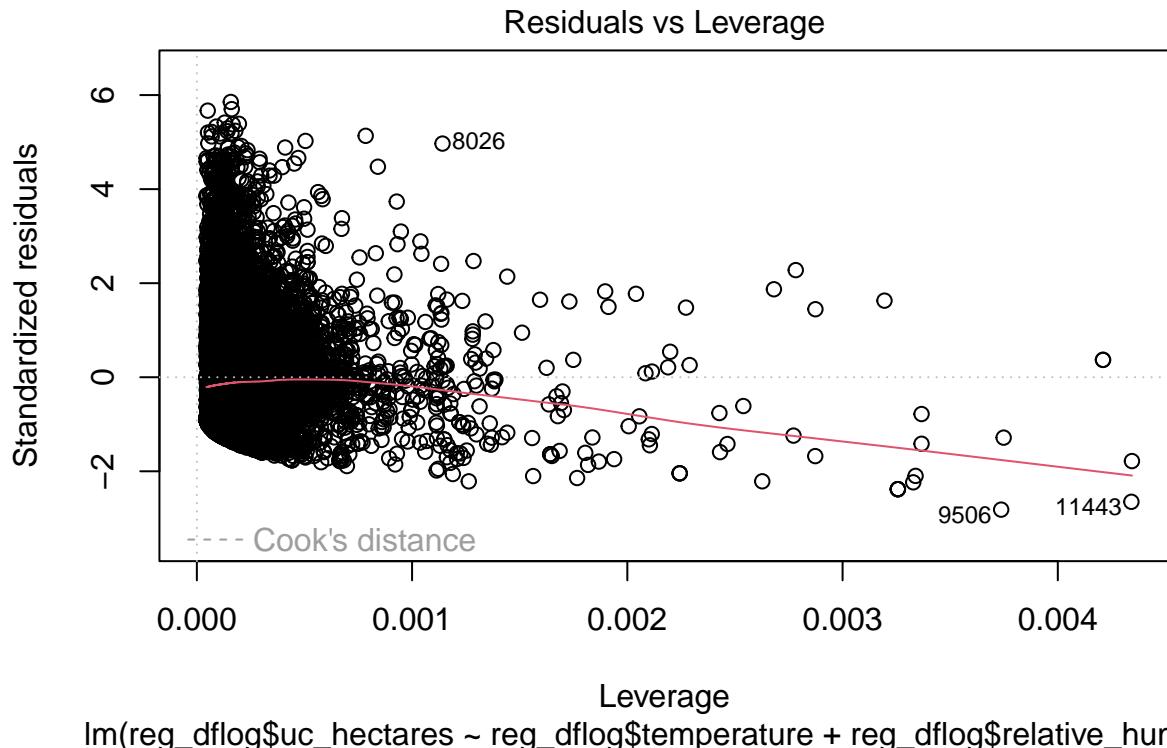
With the fire size transformed into log form the data shows slightly more of a linear relationship. Additionally, the correlation matrix shows an improved correlation between the independent and dependent variables.

```
plot(lm(formula=reg_dflog$uc_hectares~reg_dflog$temperature+reg_dflog$relative_humidity+reg_dflog$wind_
```









While still not independent or normally distributed it is still a large improvement on the original model with the untransformed data.

Overall the model with the total dataset does not fit the data well and the visualizations showed that there was almost no linear relationship between the variables and the fire size. However, once the fire size data is transformed so the model is in log linear format the model and the visualizations showed a much better relationship between the weather and fire size. With the size of the dataset and other variables available perhaps a larger or non linear model could be built to better fit the fire size data but given the regressions above temperature, wind speed, and relative humidity alone are not enough to sufficiently explain the data while there is at least something of a relationship between them.

Conclusion

Both questions investigating potential determinants of fire spread rates did have significant coefficients their R-squared values were low. The model looking at the relationship between fire size, response, and extinguished time did have significant coefficients and did show a good linear relationship between fire size and time to extinguishment time. The multivariable model looking at if weather conditions can predict fire size did show a linear relationship in the variables that were all significant, but the R-squared value was also low.

Overall, many of our results came down to the fact that the skewed nature of much of the data affected our linear model's ability to fit the data. Even with transformations and slightly larger multivariable models they were not sufficient. Perhaps a more comprehensive model containing more of the dataset's variables or perhaps a non-linear model may fit the data better.

References

1. Friedman, E. S., Kauffman, M. J., Burch, J. W., & Cottam, R. (2019). Trends in conifer regeneration following disturbance in a high-elevation forest: Implications for forest management and restoration. *Canadian Journal of Forest Research*, 49(5), 502-511. <https://doi.org/10.1139/cjfr-2018-0293>
2. Government of Alberta. (n.d.). Wildfire data [Data set]. Open Alberta. <https://open.alberta.ca/opendata/wildfire-data#summary>

torontocrimedrivers

July 13, 2025

1 Dynamics of Crime in Toronto

1.1 DATA 604 Group L01-05

Aaron Gelfand

David Griffin

Jackson Meier

Steen Rasmussen

Venkateshwaran Balu Soundararajan

```
[1]: import pandas as pd
import sqlalchemy as sq
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import statsmodels.api as sm
from scipy.stats import linregress
import geopandas as gpd
from shapely.geometry import shape
from shapely import wkt
import json
import re
import folium
import warnings
import mysql.connector
from mysql.connector import Error
import plotly.express as px
warnings.filterwarnings('ignore')
sq.__version__
```

```
[1]: '2.0.36'
```

```
[2]: USERNAME='project'
PASSWORD='ErCjJdGXlcFTr'
```

```
DATABASE='project'
PORT=3306
engine = sq.create_engine(f'''mysql+mysqlconnector://{{USERNAME}}:
    {{PASSWORD}}@localhost:{{PORT}}/{{DATABASE}}'''
```

```
[3]: # Function to run SQL queries
def SQL(query_string):
    try:
        query = pd.read_sql_query(query_string, con=engine)
        return query
    except Exception as e:
        print("An error occurred:", e)
        return None
```

```
[4]: # Function to create mysql table using DDL
def create_table(db_name, table_name, create_table_query):
    try:
        connection = mysql.connector.connect(
            host='localhost',
            user=USERNAME,
            password=PASSWORD,
            port=PORT,
            database=db_name
        )

        if connection.is_connected():
            cursor = connection.cursor()

            drop_table_query= f"""DROP TABLE IF EXISTS {table_name};"""
            cursor.execute(drop_table_query)
            print("Table "+table_name+" dropped successfully.")
            cursor.execute(create_table_query)
            connection.commit()
            print("Table "+table_name+" created successfully.")

    except Error as e:
        print(f"Error while connecting to MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
```

```
[5]: SQL("SHOW TABLES")
```

```
[5]:          Tables_in_project
0           data604_Bike_crime
```

```
1      data604_address_points  
2      data604_converted_budget  
3      data604_converted_crime  
4      data604_daily_shelter_occupancy  
5  data604_daily_shelter_occupancy_points  
6      data604_daily_shelter_occupancy_raw  
7      data604_income
```

2 Introduction

The goal of this project is to investigate the dynamics of crime-related trends and rates in Toronto, focusing on the factors that drive these changes. By looking at how the severity of crimes varies across Toronto neighbourhoods, we aim to help uncover contributors to crime and reveal high-risk areas with the hope of unveiling mitigation strategies. Specifically, our project will analyze yearly police budgets, household income, household education levels, homeless shelter occupancy rates, and current bicycle parking racks to determine what factors most influence crime statistics.

In particular, we will compare trends in annual police funds to crime rates over the same time period. We will also link any correlations between household income and average education level to crime rates in the corresponding neighbourhood. Lastly, we will be looking at whether homeless shelter occupancy has any impact on surrounding petty crime and whether bike parking racks and bike concentrations have any effect on bike theft rates. By linking the trends of these variables, we hope to answer the questions regarding crime contributors and how to address the tendencies. Ultimately, we hope to reveal insights on Toronto crime that may be useful to governing parties to help make data-informed decisions. These discoveries and implementations may not only be useful for Toronto neighbourhoods but also for other major Canadian cities with similar challenges.

Guiding Questions Overall Objective of this project is to evaluate the crime statistics in Toronto neighbourhoods by analyzing the guiding Questions.

Q1: Police Budget and Crime Data Trends: How does the Operating Budget impact the crime statistics? Research indicates that police budgets have generally increased over the years, even as crime rates have fluctuated (Toronto Police Service 2024). We will be analyzing the Police Budget with the Crime Rates, so we can compare police budget data with crime trends to see if an increase or decrease in funds plays a role in neighbourhood crime rates for different years.

Q2: Household Income and Crime Rates: How does mean household income change crime statistics? Higher Household income is generally associated with lower crime rates. Analyzing household income data in relation to crime rates can reveal important trends. We will analyze census data from 2020 to see if lower household income results in higher crime rates.

Q3: Shelter Occupancy and Neighbourhood Crime: How does shelter occupancy rate affect crime statistics per neighbourhood? Investigating the relationship between shelter occupancy rates and neighbourhood crime can provide insights into how homelessness and crime are interconnected. We will be investigating the shelter occupancy rates and link them to crime patterns by neighbourhood and year.

Q4: Bike racks and Bikes Theft: How does the presence of bike racks affect bike thefts per neighbourhood? Comparing crime data for areas with and without bike racks can help determine

if the presence of bike racks reduces bike theft. Studies often show that well-placed bike racks, combined with other security measures, can deter theft and improve overall safety in the area (Bike Finder, 2024). We want to examine whether a higher concentration of bike racks helps prevent bike thefts.

Q5: Highest Education Level and Crime Rates: How does the highest certificate, diploma, or degree per household, affect crime rate per neighbourhood? Generally, higher education levels are associated with lower crime rates (Lochner & Moretti, 2004), as education can provide better economic opportunities and reduce the likelihood of engaging in criminal activities. Analyzing crime rates in relation to education levels in each neighbourhood can highlight the impact of education on crime.

3 Individual Datasets

We are using 6 datasets for this project. All of these datasets are from toronto open data and all subsequently hold the same open data license requiring use of the data to be acknowledged. You will find links to the datasets in the references below. Our first dataset contains data on crime rates in various neighbourhoods in Toronto from 2014 to 2023. The dataset comes in the form of a Geojson, csv, geopackage, and shapefile and contains 158 neighbourhoods and 18 separate crime statistics for each neighbourhood. Our second dataset contains the toronto police operating budget for 2014-2023 and is downloadable in csv, json, and xml format. The dataset contains 6 columns and 129 rows. The third dataset contains data on the neighbourhoods in Toronto. It contains information on each neighbourhood's demographic data such as income distribution and education levels and many more. The dataset is released every 5 years alongside the census and we will be using 2020 dataset. They can be downloaded in csv, json, and xml and xlsx. The 2020 dataset has 158 columns and 2383 rows. The fourth dataset contains geographic data on bike rack locations in Toronto, the dataset has 27 columns and 241 rows. It can be downloaded as a Geojson, csv, geopackage, and shapefile. The fifth dataset contains data on daily shelter occupancy in Toronto from 2017-2020. The dataset contains information such as shelter name, location and occupancy on each day. Each year's dataset can be downloaded as a csv, json, and xml and the most recent 2020 dataset contains 13 columns and 41061 rows. The final dataset contains data on obrt 500,000 addresses in Toronto. The dataset contains information on street addresses, as well as geometric coordinates for each address.

Below we have listed the team member that worked on the cleaning of each dataset and the variables of interest. The shelter occupancy and address points datasets were particularly difficult to utilize as we needed to attach the geometric coordinates from the address points dataset to the shelter occupancy dataset. As both datasets were very large, any process run on either dataset took considerable time. Therefore any mistakes in our code required that additional time when running, which considerably slowed the process here.

Toronto Crime Data - Jackson Meier - AREA_NAME: Name of the neighbourhood - POPULATION_2023: Population of neighbourhood - Crime Columns: CRIME_XXXX: the crime followed by the year (XXXX). Replace CRIME with the following: Assault (ASSAULT), Auto Theft (AUTOTHEFT), Bike Theft (BIKETHEFT), Break & Enter (BREAKENTER), Homicide (HOMICIDE), Robbery (ROBBERY), Shooting (SHOOTING), Theft From Motor Vehicle (THEFTFROMMV), Theft Over \$5,000 (THEFTOVER). Years range from 2014-2023 - Each crime has a rate column, which is the same crime, and same year, but as a rate per 100,000 people. It is the crime, followed by RATE then the year (XXXX). Example: ASSAULT_RATE_XXXX

Gross Operating Budget - Aaron Gelfand - YEAR: year of the value collected - COUNT: Value of the subtype - SUBTYPE: Factor that is being measured. Main interest in **Gross operating budget** (\$) and **Absolute change** (\$)

Neighbourhood Profiles - David Griffin - Neighbourhood Name: name of the neighbourhood - Rest of the columns are specific neighbourhoods - Average total income among recipients (\$): The average income among recipients aged 15 and over, in a private household - Total - Highest certificate, diploma or degree for the population aged 15 years and over in private households - 25% sample data: Data on the highest degree obtained in a private household

Bike Rack Data - Steen Rasmussen - ADDRESS_FULL: displays full address of the location of the bike rack - POSTAL_CODE: postal code to go along with the address - WARD_NAME: ward the bike rack is located in - CAPACITY: number of bikes that can fit on the rack - SHELTERED: whether or not the bike rack is sheltered/covered - STATUS: whether the bike rack is installed, approved or planned. Only interested in installed.

Shelter Occupancy - Venkateshwaran Balu Soundararajan/Aaron Gelfand - SHELTER_ADDRESS: Location of the shelter - OCCUPANCY: occupancy of the shelter - CAPACITY: amount of people the shelter can host

Address Points - Venkateshwaran Balu Soundararajan/Aaron Gelfand - ADDRESS_FULL: The full address of the location - geometry: The geometric coordinate of the respective address

4 Data Exploration

4.1 Data Cleaning and Uploading Datasets to Database Tables

```
[6]: #TABLE DEFINITIONS
CRIMETABLE="data604_converted_crime"
BUDGETTABLE="data604_converted_budget"
BIKECRIMETABLE="data604_Bike_crime"
INCOMETABLE="data604_income"
SHELTEROCCUPANCYTABLE="data604_daily_shelter_occupancy"

#ReferenceTable
ADDRESSTABLE="data604_address_points"
SHELTEROCCUPANCY_RAW="data604_daily_shelter_occupancy_raw"
SHELTEROCCUPANCY_POINTS="data604_daily_shelter_occupancy_points"
```

Budget cleaning and uploading to SQL table

```
[7]: ### Creation of budget table
BUDGETTABLE_CREATE = f"""CREATE TABLE {BUDGETTABLE} (
    ID bigint NOT NULL AUTO_INCREMENT,
    year bigint(20) DEFAULT NULL,
    percent_change double DEFAULT NULL,
    absolute_budget_change double DEFAULT NULL,
    Chief double DEFAULT NULL,
```

```

    Communities_and_Neighbourhoods double,
    ↵DEFAULT NULL,
    Community_Safety double DEFAULT NULL,
    Community_Safety_Command double DEFAULT,
    ↵NULL,
    Corporate_Services double DEFAULT NULL,
    Corporate_Support double DEFAULT NULL,
    Equipment double DEFAULT NULL,
    gross_operating_budget double DEFAULT,
    ↵NULL,
    Human_Resources double DEFAULT NULL,
    Information_Technology double DEFAULT,
    ↵NULL,
    Materials double DEFAULT NULL,
    Operational_Support double DEFAULT NULL,
    Priority_Response double DEFAULT NULL,
    Salaries_and_Benefits double DEFAULT NULL,
    Services_and_Rents double DEFAULT NULL,
    Specialized_Operations double DEFAULT,
    ↵NULL,
    PRIMARY KEY (ID)
);"""
create_table(DATABASE, BUDGETTABLE,BUDGETTABLE_CREATE)

```

Table data604_converted_budget dropped successfully.
Table data604_converted_budget created successfully.

[8]: *### Budget cleaning to SQL table*

```

#First we want to upload our CSV
budgetdf=pd.read_csv("Gross Operating Budget.csv")

#Our csv is not in an ideal format, so we pivot the table so that columns are
#now the subtypes, like gross operating budget ($), Absolute Change ($) etc.
budget_df_pivot=budgetdf.pivot_table(
    index='YEAR',
    columns='SUBTYPE',
    values='COUNT_'
).reset_index()

# some generic cleaning replace NA values w/o. Shouldn't be an issue as all
#the NA values are for % values
budget_df_pivot.fillna(0, inplace=True)

#strip and replace blanks with _, and & with and for readability
budget_df_pivot.columns= budget_df_pivot.columns.str.strip().str.replace('
', '_').str.replace('&', 'and')

```

```

#convert to csv so we have the changed csv for later
budget_df_pivot.to_csv('converted_budget.csv',index=False)

#Change column names to prevent issues with SQL queries

budget_df=pd.read_csv("converted_budget.csv")
budget_df.rename(columns={"YEAR":"year","%_Change":
    "percent_change","Absolute_Change_(\$)": "absolute_budget_change",
    "Gross_Operating_Budget_(\$)": "gross_operating_budget"},inplace=True)
budget_df['_id']= budget_df.index + 1
budget_df.rename(columns = {'_id':'ID'}, inplace = True)

#Upload csv as SQL table
budget_df.to_sql(BUDGETTABLE,engine,index=False,if_exists='append')
budget_table_df=pd.read_sql_table(BUDGETTABLE,engine)
budget_table_df.head()

```

```

[8]:   ID  year  percent_change  absolute_budget_change  Chief  \
0    1  2014           0.062          63610200.0  0.006
1    2  2015           0.017          17216200.0  0.006
2    3  2016           0.026          28666300.0  0.008
3    4  2017          -0.003          -3267300.0  0.010
4    5  2018           0.007          8209800.0  0.012

               Communities_and_Neighbourhoods  Community_Safety  Community_Safety_Command  \
0                  0.000                 0.540                   0.0
1                  0.000                 0.536                   0.0
2                  0.311                 0.000                   0.0
3                  0.317                 0.000                   0.0
4                  0.295                 0.000                   0.0

      Corporate_Services  Corporate_Support  Equipment  gross_operating_budget  \
0            0.068          0.000       0.002      1.086002e+09
1            0.071          0.000       0.003      1.103218e+09
2            0.000          0.101       0.002      1.131884e+09
3            0.000          0.103       0.002      1.128617e+09
4            0.000          0.102       0.002      1.136827e+09

      Human_Resources  Information_Technology  Materials  Operational_Support  \
0            0.000                  0.0       0.019        0.191
1            0.000                  0.0       0.018        0.189
2            0.047                  0.0       0.016        0.000
3            0.044                  0.0       0.016        0.000
4            0.045                  0.0       0.016        0.000

```

	Priority_Response	Salaries_and_Benefits	Services_and_Rents	\
0	0.000	0.890	0.089	
1	0.000	0.890	0.089	
2	0.386	0.895	0.087	
3	0.374	0.891	0.091	
4	0.374	0.884	0.098	

	Specialized_Operations
0	0.196
1	0.198
2	0.147
3	0.152
4	0.172

Crime data cleaning and uploading to SQL table

```
[9]: ### Creation of crime table
CRIMETABLE_CREATE = f"""
CREATE TABLE {CRIMETABLE} (
    ID BIGINT(20) NOT NULL AUTO_INCREMENT,
    AREA_NAME LONGTEXT DEFAULT NULL,
    HOOD_ID BIGINT(20) DEFAULT NULL,
    POPULATION_2023 BIGINT(20) DEFAULT NULL,
    GEOMETRY LONGTEXT DEFAULT NULL,
    YEAR BIGINT(20) DEFAULT NULL,
    ASSAULT BIGINT(20) DEFAULT NULL,
    ASSAULT_RATE DOUBLE DEFAULT NULL,
    AUTOTHEFT DOUBLE DEFAULT NULL,
    AUTOTHEFT_RATE DOUBLE DEFAULT NULL,
    BIKETHEFT DOUBLE DEFAULT NULL,
    BIKE THEFT RATE DOUBLE DEFAULT NULL,
    BREAKENTER BIGINT(20) DEFAULT NULL,
    BREAKENTER_RATE DOUBLE DEFAULT NULL,
    HOMICIDE DOUBLE DEFAULT NULL,
    HOMICIDE_RATE DOUBLE DEFAULT NULL,
    ROBBERY DOUBLE DEFAULT NULL,
    ROBBERY_RATE DOUBLE DEFAULT NULL,
    SHOOTING DOUBLE DEFAULT NULL,
    SHOOTING_RATE DOUBLE DEFAULT NULL,
    THEFTFROMMV BIGINT(20) DEFAULT NULL,
    THEFTFROMMV_RATE DOUBLE DEFAULT NULL,
    THEFTOVER DOUBLE DEFAULT NULL,
    THEFTOVER_RATE DOUBLE DEFAULT NULL,
    PRIMARY KEY (ID)
);"""
create_table(DATABASE, CRIMETABLE, CRIMETABLE_CREATE)
```

Table data604_converted_crime dropped successfully.

Table data604_converted_crime created successfully.

```
[10]: #Loading Neighbour Crime Rates
polygons_gdf=gpd.read_file('./neighbourhood-crime-rates.geojson')
polygons_gdf.dtypes
polygons_gdf = polygons_gdf.to_crs(epsg=4326)

polygons_gdf.head(5)
polygons_gdf.columns
polygons_gdf.to_csv("neighbourhood-crime-rates.csv")

[11]: ### Crime cleaning to SQL table

#Clean and add crimeset to database
crime_df=pd.read_csv("neighbourhood-crime-rates.csv")
crime_df.head

crime_df = crime_df.loc[:, ~crime_df.columns.str.contains('^\d+')]
#wide_to_long to squish the ASSAULT_2014, ASSAULT_2015, etc. into just ASSAULT
#and add a Year column
id_cols = ['_id', 'AREA_NAME', 'HOOD_ID', 'POPULATION_2023','geometry']

crime_long = pd.wide_to_long(
    crime_df,
    stubnames=[
        'ASSAULT', 'ASSAULT_RATE', 'AUTOTHEFT', 'AUTOTHEFT_RATE',
        'BIKETHEFT', 'BIKETHEFT_RATE', 'BREAKENTER', 'BREAKENTER_RATE',
        'HOMICIDE', 'HOMICIDE_RATE', 'ROBBERY', 'ROBBERY_RATE',
        'SHOOTING', 'SHOOTING_RATE', 'THEFTFROMMV', 'THEFTFROMMV_RATE',
        'THEFTOVER', 'THEFTOVER_RATE'
    ],
    i=id_cols,
    j='Year',
    sep='_'
).reset_index()

#store changes as new csv
crime_long.to_csv('converted_crime.csv',index=False)

#Some general cleaning so the csv has no issues with SQL syntax
crime_df=pd.read_csv('converted_crime.csv')
crime_df.rename(columns={"Year":"year"},inplace=True)
crime_df['_id']= crime_df.index + 1
crime_df.rename(columns = {'_id':'ID'}, inplace = True)

# Comfortable replacing the NAs with 0, because if you inspect the data you can
# see that the NAs only appear in columns
# where the totals are low values already. This makes me believe that the
# missing data may simply just be 0's
```

```

crime_df[['ASSAULT', 'ASSAULT_RATE', 'AUTOTHEFT', 'AUTOTHEFT_RATE',
          'BIKETHEFT', 'BIKETHEFT_RATE', 'BREAKENTER', 'BREAKENTER_RATE',
          'HOMICIDE', 'HOMICIDE_RATE', 'ROBBERY', 'ROBBERY_RATE',
          'SHOOTING', 'SHOOTING_RATE', 'THEFTFROMMV', 'THEFTFROMMV_RATE',
          'THEFTOVER', 'THEFTOVER_RATE']] = crime_df[['ASSAULT', 'ASSAULT_RATE', '←
          'AUTOTHEFT', 'AUTOTHEFT_RATE',
          'BIKETHEFT', 'BIKETHEFT_RATE', 'BREAKENTER', 'BREAKENTER_RATE',
          'HOMICIDE', 'HOMICIDE_RATE', 'ROBBERY', 'ROBBERY_RATE',
          'SHOOTING', 'SHOOTING_RATE', 'THEFTFROMMV', 'THEFTFROMMV_RATE',
          'THEFTOVER', 'THEFTOVER_RATE']].fillna(0)

#Upload csv as SQL table
crime_df.to_sql(CRIMETABLE, engine, index=False, if_exists='append', chunksize=1000)
crime_table_df = pd.read_sql_table(CRIMETABLE, engine)
crime_table_df.head(10)

```

[11]:

	ID	AREA_NAME	HOOD_ID	POPULATION_2023	\
0	1	South Eglinton-Davisville	174	21987	
1	2	South Eglinton-Davisville	174	21987	
2	3	South Eglinton-Davisville	174	21987	
3	4	South Eglinton-Davisville	174	21987	
4	5	South Eglinton-Davisville	174	21987	
5	6	South Eglinton-Davisville	174	21987	
6	7	South Eglinton-Davisville	174	21987	
7	8	South Eglinton-Davisville	174	21987	
8	9	South Eglinton-Davisville	174	21987	
9	10	South Eglinton-Davisville	174	21987	

	GEOMETRY	YEAR	ASSAULT	\
0	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2014	63	
1	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2015	61	
2	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2016	70	
3	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2017	82	
4	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2018	85	
5	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2019	70	
6	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2020	82	
7	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2021	121	
8	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2022	128	
9	MULTIPOLYGON (((-79.3863542900264 43.697839855...,	2023	101	

	ASSAULT_RATE	AUTOTHEFT	AUTOTHEFT_RATE	...	HOMICIDE	HOMICIDE_RATE	\
0	344.978638	5.0	27.379257	...	0.0	0.000000	
1	332.135468	4.0	21.779375	...	0.0	0.000000	
2	377.826965	3.0	16.192583	...	1.0	5.397528	
3	429.454285	8.0	41.897980	...	0.0	0.000000	
4	431.581635	15.0	76.161461	...	0.0	0.000000	
5	345.866882	8.0	39.527645	...	1.0	4.940956	

6	396.844605	15.0	72.593521	...	1.0	4.839568
7	577.455383	15.0	71.585381	...	1.0	4.772358
8	597.628174	10.0	46.689701	...	0.0	0.000000
9	459.362366	21.0	95.510986	...	0.0	0.000000
	ROBBERY	ROBBERY_RATE	SHOOTING	SHOOTING_RATE	THEFTFROMMV	\
0	12.0	65.710220	1.0	5.475852	18	
1	10.0	54.448437	0.0	0.000000	19	
2	9.0	48.577751	1.0	5.397528	13	
3	7.0	36.660732	0.0	0.000000	17	
4	17.0	86.316322	1.0	5.077431	19	
5	5.0	24.704779	1.0	4.940956	24	
6	16.0	77.433090	1.0	4.839568	43	
7	11.0	52.495945	0.0	0.000000	22	
8	16.0	74.703522	1.0	4.668970	43	
9	3.0	13.644426	0.0	0.000000	40	
	THEFTFROMMV RATE	THEFTOVER	THEFTOVER RATE			
0	98.565331	4.0	21.903406			
1	103.452034	3.0	16.334532			
2	70.167862	4.0	21.590111			
3	89.033203	1.0	5.237247			
4	96.471184	3.0	15.232292			
5	118.582932	3.0	14.822866			
6	208.101440	5.0	24.197842			
7	104.991890	6.0	28.634151			
8	195.570110	3.0	14.006910			
9	181.925690	8.0	36.385136			

[10 rows x 24 columns]

Shelter cleaning, addition of neighborhood and uploading to SQL table

```
[12]: ### Cleaning Shelter Occupancy data and adding a neighborhood column
#First we will upload our daily shelter occupancy data for all years
#daily_shelter_occupancy='daily_shelter_occupancy'
df = pd.read_csv("Data_Shelter_Occupancy_Merged.csv")
df=df.drop(columns=['FileUniqueID'],errors='ignore')
df['ID']= df.index + 1

#To upload our df as an SQL we make sure to do it in chunks because the dataset ↴
#is too large
df.
    ↴to_sql(SHELTEROCCUPANCY_RAW,engine,index=False,if_exists='replace',chunksize=10000)
```

[12]: 156977

```
[13]: ### We then upload our address points dataset, so that we can attach geometric points to the addresses in our shelter data
ADDRESS_POINTS=pd.read_csv('Address Points_Neighbourhoods.csv')
#Only intered in the id, full address, and geometric points
ADDRESS_POINTS2=ADDRESS_POINTS[['_id','ADDRESS_FULL','geometry']]
ADDRESS_POINTS2.rename(columns = {'_id':'ID',"geometry":"GEOMETRY"}, inplace =True)

#This needs to be done in chunks due to the size of the dataset
ADDRESS_POINTS2.
    ↵to_sql(ADDRESSTABLE,engine,index=False,if_exists='replace',chunksize=10000)
```

[13]: 525460

```
[14]: ### Attempt to load our crime dataset geojson file to our SQL tables
#geo_test=gpd.read_file('neighbourhood-crime-rates.geojson')
#geo_test.to_postgis(name="geo_test", con=engine, if_exists="replace")

# When running this code, we see that we are denied permissions to do this
# Therefore we will simply do the merging outside of it, then convert it to a csv and upload the merged
# dataset to a SQL table
```

```
[15]: #Normalizes the address line so that we can merge occupancy and neighbourhood geo data
def normalize_address(address):

    if isinstance(address, str):
        abbreviations = {
            r'\bSt\b': 'Street',
            r'\bst\b': 'Street',
            r'\bAve\b': 'Avenue',
            r'\bAve.\b': 'Avenue',
            r'\bBlvd\b': 'Boulevard',
            r'\bBlvd.\b': 'Boulevard',
            r'\bRd\b': 'Road',
            r'\bRd.\b': 'Road',
            r'\bLn\b': 'Lane',
            r'\bLn.\b': 'Lane',
            r'\bDr\b': 'Drive',
            r'\bDr.\b': 'Drive',
            r'\bPkwy\b': 'Parkway',
            r'\bPkwy.\b': 'Parkway',
            r'\bPl\b': 'Place',
            r'\bPl.\b': 'Place',
            r'\bCt\b': 'Court',
            r'\bCt.\b': 'Court',
```

```

        r'\bCrt\b': 'Court',
        r'\bW\b': 'West',
        r'\bE\b': 'East',
        r'\bN\b': 'North',
        r'\bS\b': 'South',
        r'\bBathrust\b': 'Bathurst',
        r'\bToronto\b': '',
        r'\b2nd\b': '',
        r'\bfloor\b': ''
    }

    for abbr, full in abbreviations.items():
        address = re.sub(abbr, full, address)

    address = re.sub(r'[.,]', '', address)
    address = address.strip()

    return address

```

```

[16]: #Normalize our shelter occupancy data
df=SQL(f"""SELECT * FROM {SHELTEROCCUPANCY_RAW};""")
df['Normalized_ADDRESS']=df['SHELTER_ADDRESS'].apply(lambda x:normalize_address(str(x)))

#Normalize our address points data
df2=SQL(f"""SELECT ADDRESS_FULL, GEOMETRY FROM {ADDRESSTABLE};""")
df2['Normalized_ADDRESS']=df2['ADDRESS_FULL'].apply(lambda x:normalize_address(str(x)))

#Merge the 2 datasets on the Normalized_ADDRESS column, this ensures that our
#shelter occupancy data now has
#geometric points for all the shelter addresses
merged_df = df.merge(df2[['Normalized_ADDRESS', 'GEOMETRY']], on='Normalized_ADDRESS', how='left')

#Upload this dataset as an SQL table
merged_df = merged_df.loc[:, ~merged_df.columns.str.contains('^\u2022Unnamed')]

#Write DataFrame to SQL in chunks
merged_df.
    ↪to_sql(SHELTEROCCUPANCY_POINTS, engine, index=False, if_exists='replace', chunksize=10000)

```

[16]: 159751

[17]: #Now we will call upon this table, converting the geometric points to a
 ↪geometric datatype to be used for merging later
geopoints = SQL(f"""SELECT * FROM {SHELTEROCCUPANCY_POINTS};""")

```

geopoints = geopoints[(geopoints['OCCUPANCY'] != 0) | (geopoints['CAPACITY'] != 0)]
geopoints = geopoints.dropna(subset=['GEOMETRY'])
geopoints['GEOMETRY'] = geopoints['GEOMETRY'].apply(lambda x: shape(json.loads(x)))
geopoints['GEOMETRY'] = geopoints['GEOMETRY'].apply(shape)
points_gdf = gpd.GeoDataFrame(geopoints, geometry='GEOMETRY')
points_gdf.set_crs(epsg=4326, inplace=True)

```

	ID	OCCUPANCY_DATE	ORGANIZATION_NAME	\
0	1	2017-01-01	COSTI Immigrant Services	
1	2	2017-01-01	Christie Ossington Neighbourhood Centre	
2	3	2017-01-01	Christie Ossington Neighbourhood Centre	
3	4	2017-01-01	Christie Refugee Welcome Centre, Inc.	
4	5	2017-01-01	City of Toronto	
...	
159744	156971	2020-12-31	YWCA Toronto	
159747	156974	2020-12-31	Youth Without Shelter	
159748	156975	2020-12-31	Youth Without Shelter	
159749	156976	2020-12-31	YouthLink	
159750	156977	2020-12-31	YouthLink	
		SHELTER_NAME	SHELTER_ADDRESS	SHELTER_CITY \
0		COSTI Reception Centre	100 Lippincott Street	Toronto
1	Christie Ossington Men's Hostel		973 Lansdowne Avenue	Toronto
2	Christie Ossington Men's Hostel		973 Lansdowne Avenue	Toronto
3	Christie Refugee Welcome Centre		43 Christie Street	Toronto
4	Birchmount Residence		1673 Kingston Road	Toronto
...
159744	YWCA - First Stop Woodlawn	80 Woodlawn Ave. East		Toronto
159747	Youth Without Shelter	6 Warrendale Court		Etobicoke
159748	Youth Without Shelter	6 Warrendale Court		Etobicoke
159749	YouthLink Shelter	747 Warden Ave		Toronto
159750	YouthLink Shelter	747 Warden Ave		Toronto
	SHELTER_PROVINCE	SHELTER_POSTAL_CODE	FACILITY_NAME	\
0	ON	M5S 2P1	COSTI Reception Centre	
1	ON	M6H 3Z5	Christie Ossington Men's Hostel	
2	ON	M6H 3Z5	Christie Ossington Men's Hostel	
3	ON	M6G 3B1	Christie Refugee Welcome Centre	
4	ON	None	Birchmount Res 1673 Kingston Rd	
...	
159744	ON	M4T 1C1	YWCA-80 Woodlawn Ave. E.-Youth	
159747	ON	M9V 1P9	Youth w/o Shelter Emerg Shelter	
159748	ON	M9V 1P9	Youth w/o Shltr Transitional Res	
159749	ON	M1L 4A1	YouthLink - 747 Warden Ave	
159750	ON	M1L 4A1	YouthLink - 747 Warden Ave	

		PROGRAM_NAME	SECTOR	\
0	COSTI Reception Ctr	CITY Program	Co-ed	
1	Christie Ossington Extreme Weather Program		Men	
2	Christie Ossington Men's Hostel		Men	
3	Christie Refugee Welcome Ctr - Settlement and ...	Families		
4	Birchmount Residence		Men	
...		
159744	YWCA - Youth Shelter		Youth	
159747	Youth without Shelter Emergency Shelter Program		Youth	
159748	Youth without Shelter Stay In School Program		Youth	
159749	YouthLink Emergency Program		Co-ed	
159750	YouthLink Transitional Program		Co-ed	
OCCUPANCY	CAPACITY	Normalized_ADDRESS	\	
0	16	16	100 Lippincott Street	
1	13	17	973 Lansdowne Avenue	
2	63	63	973 Lansdowne Avenue	
3	66	70	43 Christie Street	
4	58	60	1673 Kingston Road	
...	
159744	21	24	80 Woodlawn Avenue East	
159747	11	17	6 Warrendale Court	
159748	11	16	6 Warrendale Court	
159749	10	10	747 Warden Avenue	
159750	29	29	747 Warden Avenue	

GEOMETRY

0	MULTIPOINT	(-79.40716 43.65766)	
1	MULTIPOINT	(-79.44591 43.66618)	
2	MULTIPOINT	(-79.44591 43.66618)	
3	MULTIPOINT	(-79.41885 43.66516)	
4	MULTIPOINT	(-79.26399 43.69151)	
...		...	
159744	MULTIPOINT	(-79.38971 43.68478)	
159747	MULTIPOINT	(-79.58027 43.73636)	
159748	MULTIPOINT	(-79.58027 43.73636)	
159749	MULTIPOINT	(-79.28283 43.71784)	
159750	MULTIPOINT	(-79.28283 43.71784)	

[148683 rows x 15 columns]

```
[18]: ### Creation of shelter occupancy table
SHELTERTABLE_CREATE = f"""CREATE TABLE {SHELTEROCCUPANCYTABLE} (
    ID bigint(20) NOT NULL AUTO_INCREMENT,
    OCCUPANCY_DATE text DEFAULT NULL,
    ORGANIZATION_NAME text DEFAULT NULL,
```

```

        SHELTER_NAME text DEFAULT NULL,
        SHELTER_ADDRESS text DEFAULT NULL,
        SHELTER_CITY text DEFAULT NULL,
        SHELTER_PROVINCE text DEFAULT NULL,
        SHELTER_POSTAL_CODE text DEFAULT NULL,
        FACILITY_NAME text DEFAULT NULL,
        PROGRAM_NAME text DEFAULT NULL,
        SECTOR text DEFAULT NULL,
        OCCUPANCY bigint(20) DEFAULT NULL,
        CAPACITY bigint(20) DEFAULT NULL,
        Normalized_ADDRESS text DEFAULT NULL,
        index_right bigint(20) DEFAULT NULL,
        AREA_NAME text DEFAULT NULL,
        PRIMARY KEY (ID)
    );"""
create_table(DATABASE, SHELTEROCCUPANCYTABLE,SHELTERTABLE_CREATE)

```

Table data604_daily_shelter_occupancy dropped successfully.
Table data604_daily_shelter_occupancy created successfully.

```
[19]: # We will be merging this dataset to our crime dataset on the geometric field, ↴
      ↴to see any shelter locations
      # that fall within the geometric polygon of our crime dataset, therefore ↴
      ↴telling us which neighborhood this shelter
      # is located

crime_geo=gpd.read_file('neighbourhood-crime-rates.geojson')
crime_geo=crime_geo[['AREA_NAME','geometry']]

#Use a spatial join to determine if a geometric point falls within a certain ↴
      ↴geometric polygon
merged_df = gpd.sjoin(points_gdf, crime_geo, how='inner', predicate='within')

#Drop our geometry data as it is no longer needed, and upload data to SQL table
merged_df=merged_df.drop(columns=['GEOMETRY'], errors='ignore')

merged_df['ID']= merged_df.index + 1

#This needs to be done in chunks due to the size of the dataset
merged_df.
      ↴to_sql(SHELTEROCCUPANCYTABLE,engine,index=False,if_exists='append',chunksize=10000)
```

[19]: 148683

Income and education cleaning and uploading to SQL table

```
[20]: #We begin by uploading our dataset
df20= pd.read_csv("neighbourhood-profiles-2021-158-model (1).csv",header=None)
```

```

#Select relevant columns
df20_clean=pd.DataFrame(columns=df20.columns, data=[df20.iloc[0,:],df20.
    ↪iloc[67,:],df20.iloc[1981,:],df20.iloc[36,:]])
#Make neighbourhoods rows
display(df20_clean)
pivot=df20_clean.T
head=pivot.iloc[0]
pivot=pivot[1:]
pivot.columns=head
#Rename long column names
incomedf=pivot.rename(columns={'Neighbourhood Name':'Neighbourhood',
    'Total - Highest certificate, diploma or degree for the population aged 15 years and over in private households - 25%':'Degree',
    'sample data':'Degree',
    'Total - Persons in private households - 25%':'Population',
    'sample data':'Population',
    'Average after-tax income in 2020 among recipients ($)' :'Income'})
#Create year variable
incomedf['YEAR']=2020
#Convert number columns to int
incomedf=incomedf.astype({'Degree':'int'})
incomedf=incomedf.astype({'Population':'int'})
incomedf=incomedf.astype({'Income':'int'})
#Create degree rate variable
incomedf['Degree Rate']=incomedf['Degree']/incomedf['Population']
incomedf.insert(0, 'ID',incomedf.index)
incomedf.rename(columns = {'Degree Rate':'Degree_Rate'}, inplace = True)
incomedf

```

	0	\
0	Neighbourhood Name	
67	Average after-tax income in 2020 among rec...	
1981	Total - Highest certificate, diploma or degree...	
36	Total - Persons in private households - 25% sa...	

	1	2	\
0	West Humber-Clairville	Mount Olive-Silverstone-Jamestown	
67	35800	31760	
1981	29000	25655	
36	33300	31345	

	3	4	5	\
0	Thistletown-Beaumont Heights	Rexdale-Kipling	Elms-Old Rexdale	
67	36360	36880	36440	
1981	8350	8805	7745	
36	9850	10375	9355	

			6		7	\
0	Kingsview Village-The Westway	Willowridge-Martingrove-Richview				
67		38640		45160		
1981		18090		18945		
36		22005		22445		
			8	9	...	\
0	Humber Heights-Westmount	Edenbridge-Humber Valley		...		
67		50160		68700	...	
1981		8635		13120	...	
36		10005		15190	...	
			149		150	\
0	Harbourfront-CityPlace	St Lawrence-East Bayfront-The Islands				
67		57700		61650		
1981		26060		29010		
36		28135		31285		
			151	152	153	\
0	Church-Wellesley	Downtown Yonge East	Bay-Cloverhill			
67		46360	52900	50960		
1981		21420	16640	15930		
36		22320	17700	16670		
			154	155	156	\
0	Yonge-Bay Corridor	Junction-Wallace Emerson	Dovercourt Village			
67		53050	45480	46200		
1981		11675	20105	11015		
36		12645	23180	12380		
			157	158		
0	North Toronto	South Eglinton-Davisville				
67		48200	54700			
1981		14570	20545			
36		15885	22735			

[4 rows x 159 columns]

[20]:	0	ID	Neighbourhood	Income	Degree	Population	YEAR	\
	1	1	West Humber-Clairville	35800	29000	33300	2020	
	2	2	Mount Olive-Silverstone-Jamestown	31760	25655	31345	2020	
	3	3	Thistletown-Beaumont Heights	36360	8350	9850	2020	
	4	4	Rexdale-Kipling	36880	8805	10375	2020	
	5	5	Elms-Old Rexdale	36440	7745	9355	2020	
	
	154	154	Yonge-Bay Corridor	53050	11675	12645	2020	

155	155	Junction-Wallace Emerson	45480	20105	23180	2020
156	156	Dovercourt Village	46200	11015	12380	2020
157	157	North Toronto	48200	14570	15885	2020
158	158	South Eglinton-Davisville	54700	20545	22735	2020
0	Degree_Rate					
1		0.870871				
2		0.818472				
3		0.847716				
4		0.848675				
5		0.827900				
..		..				
154		0.923290				
155		0.867343				
156		0.889742				
157		0.917218				
158		0.903673				

[158 rows x 7 columns]

```
[21]: INCOMETABLE_CREATE = f"""CREATE TABLE {INCOMETABLE} (
    ID bigint(20) NOT NULL AUTO_INCREMENT,
    Neighbourhood text DEFAULT NULL,
    Income bigint(20) DEFAULT NULL,
    Degree bigint(20) DEFAULT NULL,
    Population bigint(20) DEFAULT NULL,
    YEAR bigint(20) DEFAULT NULL,
    Degree_Rate double DEFAULT NULL,
    PRIMARY KEY (ID)
);"""

create_table(DATABASE, INCOMETABLE, INCOMETABLE_CREATE)
```

Table data604_income dropped successfully.

Table data604_income created successfully.

```
[22]: #Make table
incomedf.to_sql(INCOMETABLE,engine,index=False,if_exists='append')
income_table=pd.read_sql_table(INCOMETABLE,engine)
crime_table_df=pd.read_sql_table(CRIMETABLE,engine)
incomedf
```

	ID	Neighbourhood	Income	Degree	Population	YEAR	\
0	1	West Humber-Clairville	35800	29000	33300	2020	
1	2	Mount Olive-Silverstone-Jamestown	31760	25655	31345	2020	
2	3	Thistletown-Beaumont Heights	36360	8350	9850	2020	
3	4	Rexdale-Kipling	36880	8805	10375	2020	
4	5	Elms-Old Rexdale	36440	7745	9355	2020	
..	

154	154	Yonge-Bay Corridor	53050	11675	12645	2020
155	155	Junction-Wallace Emerson	45480	20105	23180	2020
156	156	Dovercourt Village	46200	11015	12380	2020
157	157	North Toronto	48200	14570	15885	2020
158	158	South Eglinton-Davisville	54700	20545	22735	2020

		Degree_Rate
0		0.870871
1		0.818472
2		0.847716
3		0.848675
4		0.827900
..		...
154		0.923290
155		0.867343
156		0.889742
157		0.917218
158		0.903673

[158 rows x 7 columns]

Bike rack cleaning and uploading to SQL table

```
[23]: ### Bike rack cleaning

# cleann bike rack csv
# read csv
bikerack_df = pd.read_csv('Bicycle Parking Racks Data - 4326.csv')

# drop columns that aren't needed
bikerack_df = bikerack_df.drop(columns=['ADDRESS_POINT_ID',
                                         'ADDRESS_NUMBER', 'LINEAR_NAME_FULL', 'CENTRELINE_ID', 'LO_NUM', 'LO_NUM_SUF', 'HI_NUM', 'HI_NUM_SUF'],
                                         errors='ignore')
bikerack_df_cleaned = bikerack_df[bikerack_df['STATUS'].str.lower() == 'installed'] # make sure the bike rack is installed

# drop rows with missing values
bikerack_df_cleaned = bikerack_df_cleaned.dropna(subset=['ADDRESS_FULL',
                                         'POSTAL_CODE', 'WARD_NAME'])

#make sure capacity is a number
bikerack_df_cleaned['CAPACITY'] = pd.to_numeric(bikerack_df_cleaned['CAPACITY'], errors='coerce')
bikerack_df_cleaned = bikerack_df_cleaned.dropna(subset=['CAPACITY']) # get rid of rows with no capacity

bikerack_df_cleaned = bikerack_df_cleaned.reset_index(drop=True)
```

```

# save new csv
bikerack_df_cleaned.to_csv('Cleaned_Bicycle_Parking_Racks_Data.csv', index =False)

### Crime data pivot and clean

# read geojson
crime_df = gpd.read_file('neighbourhood-crime-rates.geojson')

# set column names
id_cols = ['_id', 'AREA_NAME', 'HOOD_ID', 'POPULATION_2023','geometry']

#wide_to_long to squish the ASSAULT_2014, ASSAULT_2015, etc. into just ASSAULT
#and add a Year column
crime_long = pd.wide_to_long(
    crime_df,
    stubnames=[
        'ASSAULT', 'ASSAULT_RATE', 'AUTOTHEFT', 'AUTOTHEFT_RATE',
        'BIKETHEFT', 'BIKETHEFT_RATE', 'BREAKENTER', 'BREAKENTER_RATE',
        'HOMICIDE', 'HOMICIDE_RATE', 'ROBBERY', 'ROBBERY_RATE',
        'SHOOTING', 'SHOOTING_RATE', 'THEFTFROMMV', 'THEFTFROMMV_RATE',
        'THEFTOVER', 'THEFTOVER_RATE'],
    i=id_cols,
    j='Year', #makes it so Year is a column
    sep='_'
).reset_index()

# save new csv
crime_long.to_csv('crime_long2.csv', index=False)

### clean crime_long2

#read file
crime_df = pd.read_csv('crime_long2.csv')

# drop rows with missing values
crime_df = crime_df.dropna(subset=['BIKETHEFT', 'BIKETHEFT_RATE', 'HOOD_ID'])

# drop unnecessary columns
columns_to_drop = ['ASSAULT', 'ASSAULT_RATE', 'AUTOTHEFT', 'AUTOTHEFT_RATE',
                   'BREAKENTER', 'BREAKENTER_RATE',
                   'HOMICIDE', 'HOMICIDE_RATE', 'ROBBERY', 'ROBBERY_RATE',
                   'SHOOTING', 'SHOOTING_RATE', 'THEFTFROMMV', 'THEFTFROMMV_RATE',
                   'THEFTOVER', 'THEFTOVER_RATE']
crime_df_cleaned = crime_df.drop(columns=columns_to_drop)

```

```

# convert POPULATION_2023 column to numeric
crime_df_cleaned['POPULATION_2023'] = pd.
    ↪to_numeric(crime_df_cleaned['POPULATION_2023'], errors='coerce')
crime_df_cleaned = crime_df_cleaned.dropna(subset=['POPULATION_2023'])

# reset the index
crime_df_cleaned = crime_df_cleaned.reset_index(drop=True)

# save new csv
crime_df_cleaned.to_csv('Cleaned_crime_long_Data.csv', index=False)

### Merging Bike and crime

# read cleaned files
bikerack_df = pd.read_csv('Cleaned_Bicycle_Parking_Racks_Data.csv')
crime_stats_df = pd.read_csv('Cleaned_crime_long_Data.csv')

# turn geometry to a string for bike racks
if isinstance(bikerack_df['geometry'].iloc[0], str):
    try:
        bikerack_df['geometry'] = bikerack_df['geometry'].apply(lambda x: ↪
            ↪shape(json.loads(x)))
    except (json.JSONDecodeError, TypeError):
        bikerack_df['geometry'] = gpd.GeoSeries.
            ↪from_wkt(bikerack_df['geometry']))

# turn geometry to a string for crime
if isinstance(crime_stats_df['geometry'].iloc[0], str):
    try:
        crime_stats_df['geometry'] = crime_stats_df['geometry'].apply(lambda x: ↪
            ↪shape(json.loads(x)))
    except (json.JSONDecodeError, TypeError):
        crime_stats_df['geometry'] = gpd.GeoSeries.
            ↪from_wkt(crime_stats_df['geometry']))

# turn dataframes into geodataframes
bikerack_gdf = gpd.GeoDataFrame(bikerack_df, geometry='geometry')
crime_stats_gdf = gpd.GeoDataFrame(crime_stats_df, geometry='geometry')

# set coordinate reference system
bikerack_gdf = bikerack_gdf.set_crs(epsg=4326)
crime_stats_gdf = crime_stats_gdf.set_crs(epsg=4326)

# function to check if points are inside polygons
def check_points_in_polygons(points_gdf, polygons_gdf, output_file=None):

```

```

if not isinstance(points_gdf, gpd.GeoDataFrame) or not
    ↪isinstance(polygons_gdf, gpd.GeoDataFrame):
    raise ValueError("Both inputs must be GeoDataFrames.")

# join on points and polygons
result = gpd.sjoin(points_gdf, polygons_gdf, how='inner', ↪
    ↪predicate='within')

# save file to right type
if output_file:
    if output_file.endswith('.csv'):
        result.to_csv(output_file, index=False)
    elif output_file.endswith('.geojson'):
        result.to_file(output_file, driver='GeoJSON')
    elif output_file.endswith('.shp'):
        result.to_file(output_file)
    else:
        raise ValueError("Unsupported file format. Use .csv, .geojson, or .
    ↪shp.")
return result

# save new joined csv
output = check_points_in_polygons(bikerack_gdf, crime_stats_gdf, ↪
    ↪output_file='Bike_crime.csv')

```

[24]: BIKETABLE_CREATE = f"""CREATE TABLE {BIKECRIMETABLE} (
 ID bigint(20) NOT NULL AUTO_INCREMENT,
 id_left bigint(20) DEFAULT NULL,
 ADDRESS_FULL text DEFAULT NULL,
 POSTAL_CODE text DEFAULT NULL,
 MUNICIPALITY text DEFAULT NULL,
 CITY text DEFAULT NULL,
 WARD_NAME text DEFAULT NULL,
 CAPACITY bigint(20) DEFAULT NULL,
 SHELTERED text DEFAULT NULL,
 STATUS text DEFAULT NULL,
 MAP_CLASS text DEFAULT NULL,
 geometry text DEFAULT NULL,
 index_right bigint(20) DEFAULT NULL,
 id_right bigint(20) DEFAULT NULL,
 AREA_NAME text DEFAULT NULL,
 HOOD_ID double DEFAULT NULL,
 POPULATION_2023 bigint(20) DEFAULT NULL,
 Year bigint(20) DEFAULT NULL,
 BIKETHEFT bigint(20) DEFAULT NULL,
 BIKETHEFT_RATE double DEFAULT NULL,
 PRIMARY KEY (ID)

```
        );"""
create_table(DATABASE, BIKECRIMETABLE,BIKETABLE_CREATE)
```

Table data604_Bike_crime dropped successfully.
Table data604_Bike_crime created successfully.

[25]: *### Create Bike_crime table*

```
# read file
df = pd.read_csv('Bike_crime.csv')
df.insert(0, 'ID', df.index+1)
df.rename(columns = {'_id_left':'id_left','_id_right':'id_right'}, inplace =True)

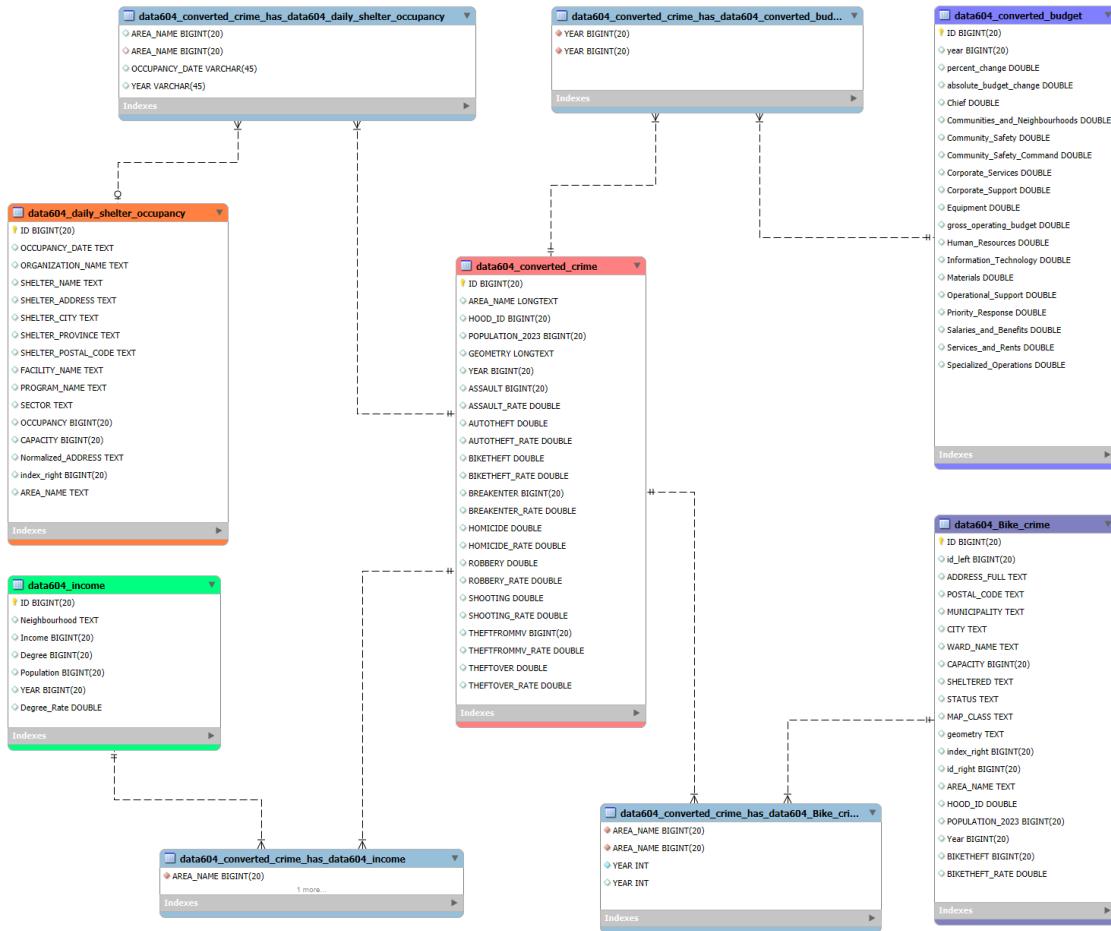
# save file to a table
df.to_sql(BIKECRIMETABLE, engine, if_exists='append', index=False)
```

[25]: 1390

[26]: SQL("SHOW TABLES")

```
[26]:          Tables_in_project
 0            data604_Bike_crime
 1            data604_address_points
 2            data604_converted_budget
 3            data604_converted_crime
 4            data604_daily_shelter_occupancy
 5  data604_daily_shelter_occupancy_points
 6            data604_daily_shelter_occupancy_raw
 7            data604_income
```

4.1.1 Entity relationship Diagram



4.2 Data Exploration for Each Guiding Question

Police Budget vs Crime Statistics - Aaron Gelfand

[27]: # query for inspecting relationship between police budget and crime stats

```
# create a dataframe from our budget table and crime table which looks at all
# crime total counts and avg crime rates
#Also take our total budget in billions and budget change in billions
# merges the two into one table using the year column
crime_query= """
WITH CrimeTotals AS (
    SELECT
        year,
        SUM(ASSAULT) as total_assault,
        SUM(AUTOTHEFT) as total_autotheft,
        SUM(BIKETHEFT) as total_biketheft,
        SUM(BREAKENTER) as total_breakenter,
        SUM(HOMICIDE) as total_homicide,
```

```

        SUM(ROBBERY) as total_robbery,
        SUM(SHOOTING) as total_shooting,
        SUM(THEFTFROMMV) as total_theftfrommv,
        SUM(THEFTOVER) as total_theftover,
        AVG(ASSAULT_RATE) as avg_assault_rate,
        AVG(AUTOTHEFT_RATE) as avg_autotheft_rate,
        AVG(BIKETHEFT_RATE) as avg_biketheft_rate,
        AVG(BREAKENTER_RATE) as avg_breakenter_rate,
        AVG(HOMICIDE_RATE) as avg_homicide_rate,
        AVG(ROBBERY_RATE) as avg_robbery_rate,
        AVG(SHOOTING_RATE) as avg_shooting_rate,
        AVG(THEFTFROMMV_RATE) as avg_theftfrommv_rate,
        AVG(THEFTOVER_RATE) as avg_theftover_rate

    FROM {CRIMETABLE}
    GROUP BY year
),
BudgetData AS (
    SELECT
        year,
        CAST(SUM(gross_operating_budget) / 1000000000 AS FLOAT) AS ↵
        ↵total_budget_in_billions,
        CAST(SUM(absOLUTE_budget_change) / 1000000000 AS FLOAT) AS ↵
        ↵budget_change_in_billions,
        percent_change
    FROM {BUDGETTABLE}
    GROUP BY year
)
SELECT
    B.year,
    B.total_budget_in_billions,
    B.budget_change_in_billions,
    B.percent_change,
    C.total_assault,
    C.total_autotheft,
    C.total_biketheft,
    C.total_breakenter,
    C.total_homicide,
    C.total_robbery,
    C.total_shooting,
    C.total_theftfrommv,
    C.total_theftover,
    C.avg_assault_rate,
    C.avg_autotheft_rate,
    C.avg_biketheft_rate,
    C.avg_breakenter_rate,

```

```

C.avg_homicide_rate,
C.avg_robbery_rate,
C.avg_shooting_rate,
C.avg_theftfrommv_rate,
C.avg_theftover_rate
FROM BudgetData B
Join CrimeTotals C
On B.year=C.year
ORDER BY B.year;
"""

result_df=SQL(crime_query)
SQL(crime_query)

```

	year	total_budget_in_billions	budget_change_in_billions	percent_change	\
0	2014	1.08600	0.063610	0.062000	
1	2015	1.10322	0.017216	0.017000	
2	2016	1.13188	0.028666	0.026000	
3	2017	1.12862	-0.003267	-0.003000	
4	2018	1.13683	0.008210	0.007000	
5	2019	1.20194	0.065113	0.057000	
6	2020	1.22122	0.019276	0.016000	
7	2021	1.22000	-0.001215	-0.000995	
8	2022	1.26243	0.042427	0.033610	
9	2023	1.33063	0.068197	0.054020	

	total_assault	total_autotheft	total_biketheft	total_breakenter	\
0	16530.0	3576.0	3023.0	7185.0	
1	17875.0	3262.0	3294.0	6911.0	
2	18626.0	3311.0	3798.0	6402.0	
3	18928.0	3568.0	3856.0	6883.0	
4	19586.0	4774.0	3967.0	7565.0	
5	20602.0	5285.0	3704.0	8438.0	
6	17974.0	5731.0	3915.0	6914.0	
7	19012.0	6579.0	3154.0	5671.0	
8	21024.0	9662.0	2940.0	6038.0	
9	24376.0	12013.0	3008.0	7632.0	

	total_homicide	total_robbery	...	total_theftover	avg_assault_rate	\
0	58.0	3668.0	...	993.0	610.401795	
1	59.0	3472.0	...	1035.0	656.997006	
2	75.0	3654.0	...	1030.0	670.169852	
3	65.0	4008.0	...	1167.0	673.530257	
4	98.0	3623.0	...	1230.0	680.128628	
5	79.0	3508.0	...	1368.0	706.708804	
6	71.0	2773.0	...	1209.0	605.126466	
7	85.0	2243.0	...	1053.0	637.650870	

```

8          71.0      2807.0 ...      1444.0      703.030473
9          72.0      3149.0 ...      1724.0      806.310513

    avg_autotheft_rate  avg_biketheft_rate  avg_breakenter_rate \
0          122.894735      127.664053      268.183005
1          113.846110      133.504008      255.802629
2          113.862644      149.693942      234.708454
3          120.558686      148.402819      248.018052
4          159.456077      150.450802      264.205776
5          175.019654      135.200693      293.590049
6          186.981110      142.251404      241.965059
7          217.026578      112.199870      193.507392
8          316.850521      105.244566      205.555160
9          382.173675      102.847534      258.216067

    avg_homicide_rate  avg_robbery_rate  avg_shooting_rate \
0          2.112469      136.057003      6.274449
1          2.212523      124.980440      10.707640
2          2.696912      131.174051      14.310603
3          2.438607      141.363362      13.704951
4          3.630148      127.874208      14.227553
5          2.882434      120.688364      16.807147
6          2.220783      92.661019      14.691605
7          2.974706      75.648864      13.390221
8          2.344280      92.461004      12.401718
9          2.251066      103.874248      11.212980

    avg_theftfrommv_rate  avg_theftover_rate
0          353.516855      36.385880
1          320.224055      37.096303
2          276.356090      36.700544
3          294.381803      40.144456
4          312.275849      41.853118
5          339.260389      46.163937
6          345.983044      40.760462
7          270.300717      34.193171
8          300.305751      46.940361
9          277.921576      54.832638

[10 rows x 22 columns]

```

```
[28]: #Plot out operating budget vs avg crime rates, with budget as bars and crime
      ↪rates as lines, so you can see the change in both over time
fig, axs = plt.subplots(2,2,figsize=(15,12))

axs[0, 0].bar(result_df['year'], result_df['total_budget_in_billions'], color='tab:blue', alpha=0.4, label='Total Budget (Billions)')
```

```

axs[0, 0].set_xlabel('Year')
axs[0, 0].set_ylabel('Total Budget (in billions)', color='tab:blue')
axs[0, 0].tick_params(axis='y', labelcolor='tab:blue')
ax2 = axs[0, 0].twinx()
ax2.plot(result_df['year'], result_df['avg_assault_rate'], label='Average  
Assault Rates', color='magenta', linewidth=2)
ax2.set_ylabel('Crime Rates', color='green')
ax2.tick_params(axis='y', labelcolor='green')
axs[0, 0].set_title('Operating Budget vs. Average Assault Rates per 100,000  
Population')

axs[0, 1].bar(result_df['year'], result_df['total_budget_in_billions'],  
    color='tab:blue', alpha=0.4)
axs[0, 1].set_xlabel('Year')
axs[0, 1].set_ylabel('Total Budget (in billions)', color='tab:blue')
axs[0, 1].tick_params(axis='y', labelcolor='tab:blue')
ax2 = axs[0, 1].twinx()
ax2.plot(result_df['year'], result_df['avg_autotheft_rate'], label='Average  
Auto Theft Rates', color='tab:orange', linewidth=2)
ax2.plot(result_df['year'], result_df['avg_breakenter_rate'], label='Average  
Break and Enter Rates', color='tab:green', linewidth=2)
ax2.plot(result_df['year'], result_df['avg_biketheft_rate'], label='Average  
Bike Theft Rates', color='tab:red', linewidth=2)
ax2.plot(result_df['year'], result_df['avg_robbery_rate'], label='Average  
Robbery Rates', color='tab:blue', linewidth=2)
ax2.plot(result_df['year'], result_df['avg_theftfrommv_rate'], label='Average  
Theft from Moving Vehicle Rates', color='black', linewidth=2)
ax2.plot(result_df['year'], result_df['avg_theftover_rate'], label='Average  
Thefts Over $5000 Rates', color='tab:brown', linewidth=2)
ax2.set_ylabel('Crime Rates', color='green')
ax2.tick_params(axis='y', labelcolor='green')
axs[0, 1].set_title('Operating Budget vs. Average Rates for Multiple Crimes per  
100,000 Population')

axs[1, 0].bar(result_df['year'], result_df['total_budget_in_billions'],  
    color='tab:blue', alpha=0.4)
axs[1, 0].set_xlabel('Year')
axs[1, 0].set_ylabel('Total Budget (in billions)', color='tab:blue')
axs[1, 0].tick_params(axis='y', labelcolor='tab:blue')
ax2 = axs[1, 0].twinx()
ax2.plot(result_df['year'], result_df['avg_homicide_rate'], label='Average  
Homicide Rates', color='purple', linewidth=2)
ax2.plot(result_df['year'], result_df['avg_shooting_rate'], label='Average  
Shooting Rates', color='teal', linewidth=2)
ax2.set_ylabel('Crime Rates', color='green')
ax2.tick_params(axis='y', labelcolor='green')

```

```

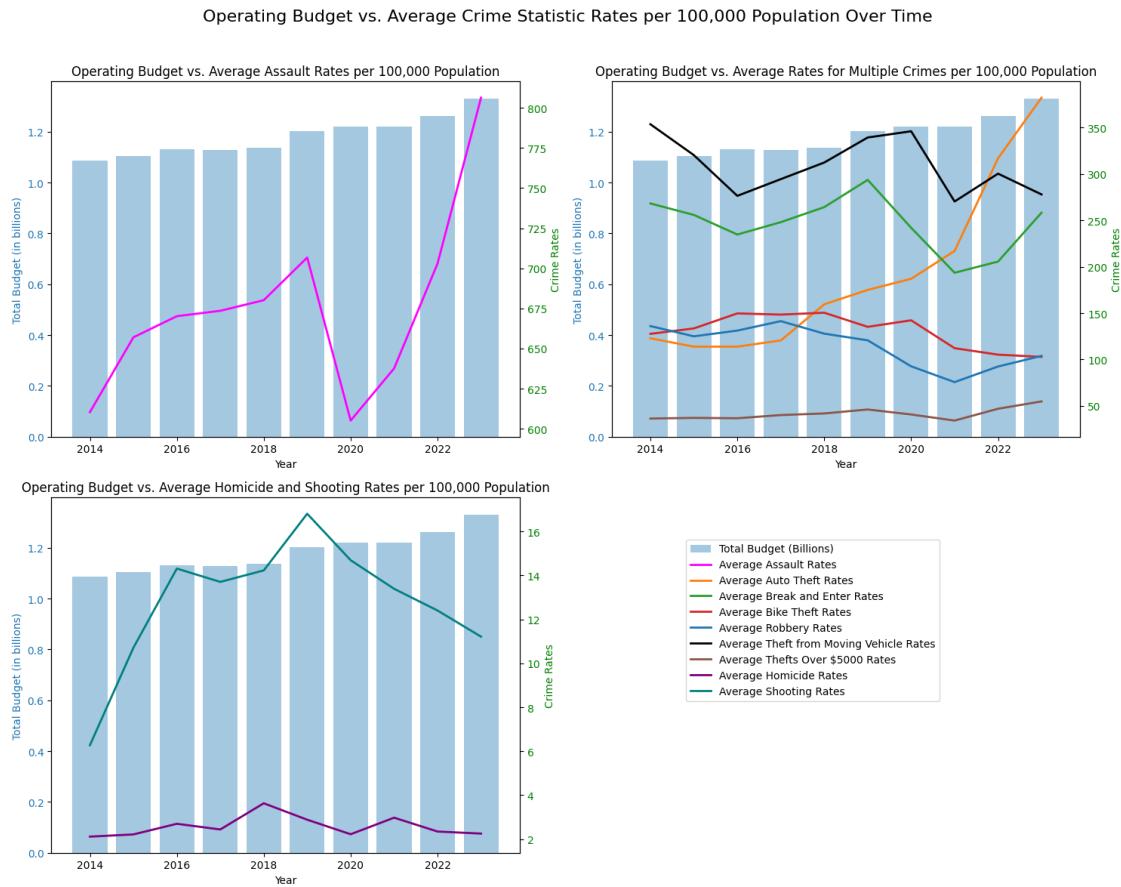
axs[1, 0].set_title('Operating Budget vs. Average Homicide and Shooting Rates per 100,000 Population')

axs[1, 1].axis('off')

fig.suptitle('Operating Budget vs. Average Crime Statistic Rates per 100,000 Population Over Time', fontsize=16)
fig.tight_layout(rect=[0, 0, 1, 0.96]) # Adjust layout to avoid overlap
fig.legend(loc='upper left', bbox_to_anchor=(0.6, 0.4))

plt.show()

```



```

[29]: # Our sample size for budgetary data is too small (n=10) therefore we decided to run bootstrapping
target_columns = [
    'avg_assault_rate', 'avg_autotheft_rate', 'avg_biketheft_rate',
    'avg_breakenter_rate',
    'avg_homicide_rate', 'avg_robbery_rate', 'avg_shooting_rate',
    'avg_theftfrommv_rate',
]

```

```

    'avg_theftover_rate'
]

n_iter=1000
boot_results={}
for target in target_columns:
    y = result_df[target]
    X = result_df[['total_budget_in_billions']]
    X = sm.add_constant(X)

    boot_coefs=[]
    for _ in range(n_iter):
        resample_indices=np.random.choice(len(result_df), len(result_df), replace = True)
        X_resample=X.iloc[resample_indices]
        y_resample=y.iloc[resample_indices]
        model = sm.OLS(y_resample, X_resample).fit()
        boot_coefs.append(model.params.values)
    boot_coefs_df=pd.DataFrame(boot_coefs,columns=['Intercept', 'Slope'])
    boot_results[target]=boot_coefs_df

    intercept_mean = boot_coefs_df['Intercept'].mean()
    intercept_std = boot_coefs_df['Intercept'].std()
    slope_mean = boot_coefs_df['Slope'].mean()
    slope_std = boot_coefs_df['Slope'].std()

    print(f"Bootstrap Results for {target}:")
    print(f"Intercept: Mean = {intercept_mean:.3f}, Std Dev = {intercept_std:.3f}")
    print(f"Slope: Mean = {slope_mean:.3f}, Std Dev = {slope_std:.3f}")
    print(f"95% CI for Slope: {np.percentile(boot_coefs_df['Slope'], [2.5, 97.5])}")
    print()

```

Bootstrap Results for avg_assault_rate:
 Intercept: Mean = 200.268, Std Dev = 295.023
 Slope: Mean = 399.191, Std Dev = 256.663
 95% CI for Slope: [-219.29111869 723.56022077]

Bootstrap Results for avg_autotheft_rate:
 Intercept: Mean = -1089.792, Std Dev = 210.396
 Slope: Mean = 1080.958, Std Dev = 181.100
 95% CI for Slope: [587.81773088 1344.30914803]

Bootstrap Results for avg_biketheft_rate:
 Intercept: Mean = 323.732, Std Dev = 87.799
 Slope: Mean = -162.449, Std Dev = 74.914

```

95% CI for Slope: [-290.01173223 -5.53325095]

Bootstrap Results for avg_breakenter_rate:
Intercept: Mean = 400.265, Std Dev = 151.882
Slope: Mean = -130.903, Std Dev = 133.942
95% CI for Slope: [-414.43424544 80.83283158]

Bootstrap Results for avg_homicide_rate:
Intercept: Mean = 3.224, Std Dev = 2.802
Slope: Mean = -0.515, Std Dev = 2.346
95% CI for Slope: [-5.15405213 4.147113 ]

Bootstrap Results for avg_robbery_rate:
Intercept: Mean = 384.456, Std Dev = 97.799
Slope: Mean = -228.910, Std Dev = 84.969
95% CI for Slope: [-433.54062493 -114.94065184]

Bootstrap Results for avg_shooting_rate:
Intercept: Mean = 0.052, Std Dev = 19.554
Slope: Mean = 10.984, Std Dev = 16.467
95% CI for Slope: [-15.91244638 48.76534223]

Bootstrap Results for avg_theftfrommv_rate:
Intercept: Mean = 456.387, Std Dev = 157.016
Slope: Mean = -125.196, Std Dev = 135.189
95% CI for Slope: [-338.05800975 187.33348451]

Bootstrap Results for avg_theftover_rate:
Intercept: Mean = -23.802, Std Dev = 24.808
Slope: Mean = 55.020, Std Dev = 21.780
95% CI for Slope: [-2.67892237 79.21359648]

```

[30]: #Plotted out our significant relationships from the output of our bootstrapped analysis (instances where 95% CI did not contain 0)
#These were auto theft, bike theft, theft over \$5000, and robbery
#Plotted out 100 of the bootstrapped regression lines, as well as the mean regression line

```

def add_bootstrap_lines(ax, x_data, boot_results, n_lines=100, color='gray', alpha=0.2):
    """Add bootstrapped regression lines to the plot"""
    for i in range(min(n_lines, len(boot_results))):
        slope = boot_results.iloc[i]['Slope']
        intercept = boot_results.iloc[i]['Intercept']
        x_vals = np.array([x_data.min(), x_data.max()])
        y_vals = slope * x_vals + intercept
        ax.plot(x_vals, y_vals, color=color, alpha=alpha)

```

```

def add_mean_regression_line(ax, x_data, boot_results, data_color, label='Mean Regression Line'):
    """Add the mean regression line to the plot"""
    mean_slope = boot_results['Slope'].mean()
    mean_intercept = boot_results['Intercept'].mean()
    x_vals = np.array([x_data.min(), x_data.max()])
    y_vals = mean_slope * x_vals + mean_intercept
    ax.plot(x_vals, y_vals, color=data_color, label=label, linewidth=2)

fig, axs = plt.subplots(2, 1, figsize=(10, 12))

axs[0].scatter(result_df['total_budget_in_billions'],
               result_df['avg_autotheft_rate'],
               color='tab:orange', label='Auto Theft Rates')
add_bootstrap_lines(axs[0], result_df['total_budget_in_billions'],
                    boot_results['avg_autotheft_rate'])
add_mean_regression_line(axs[0], result_df['total_budget_in_billions'],
                        boot_results['avg_autotheft_rate'], data_color='tab:orange')
axs[0].scatter(result_df['total_budget_in_billions'],
               result_df['avg_biketheft_rate'],
               color='tab:green', label='Bike Theft Rates')
add_bootstrap_lines(axs[0], result_df['total_budget_in_billions'],
                    boot_results['avg_biketheft_rate'])
add_mean_regression_line(axs[0], result_df['total_budget_in_billions'],
                        boot_results['avg_biketheft_rate'], data_color='tab:green')
axs[0].set_xlabel('Total Budget (in billions)', fontsize=14)
axs[0].set_ylabel('Average Crime Rates', fontsize=14)
axs[0].set_title('Operating Budget vs. Average Rates of Auto and Bike Theft', fontsize=14)

axs[1].scatter(result_df['total_budget_in_billions'],
               result_df['avg_theftover_rate'],
               color='purple', label='Theft Over $5000 Rates')
add_bootstrap_lines(axs[1], result_df['total_budget_in_billions'],
                    boot_results['avg_theftover_rate'])
add_mean_regression_line(axs[1], result_df['total_budget_in_billions'],
                        boot_results['avg_theftover_rate'], data_color='purple')
axs[1].scatter(result_df['total_budget_in_billions'],
               result_df['avg_robbery_rate'],
               color='teal', label='Robbery Rates')
add_bootstrap_lines(axs[1], result_df['total_budget_in_billions'],
                    boot_results['avg_robbery_rate'])

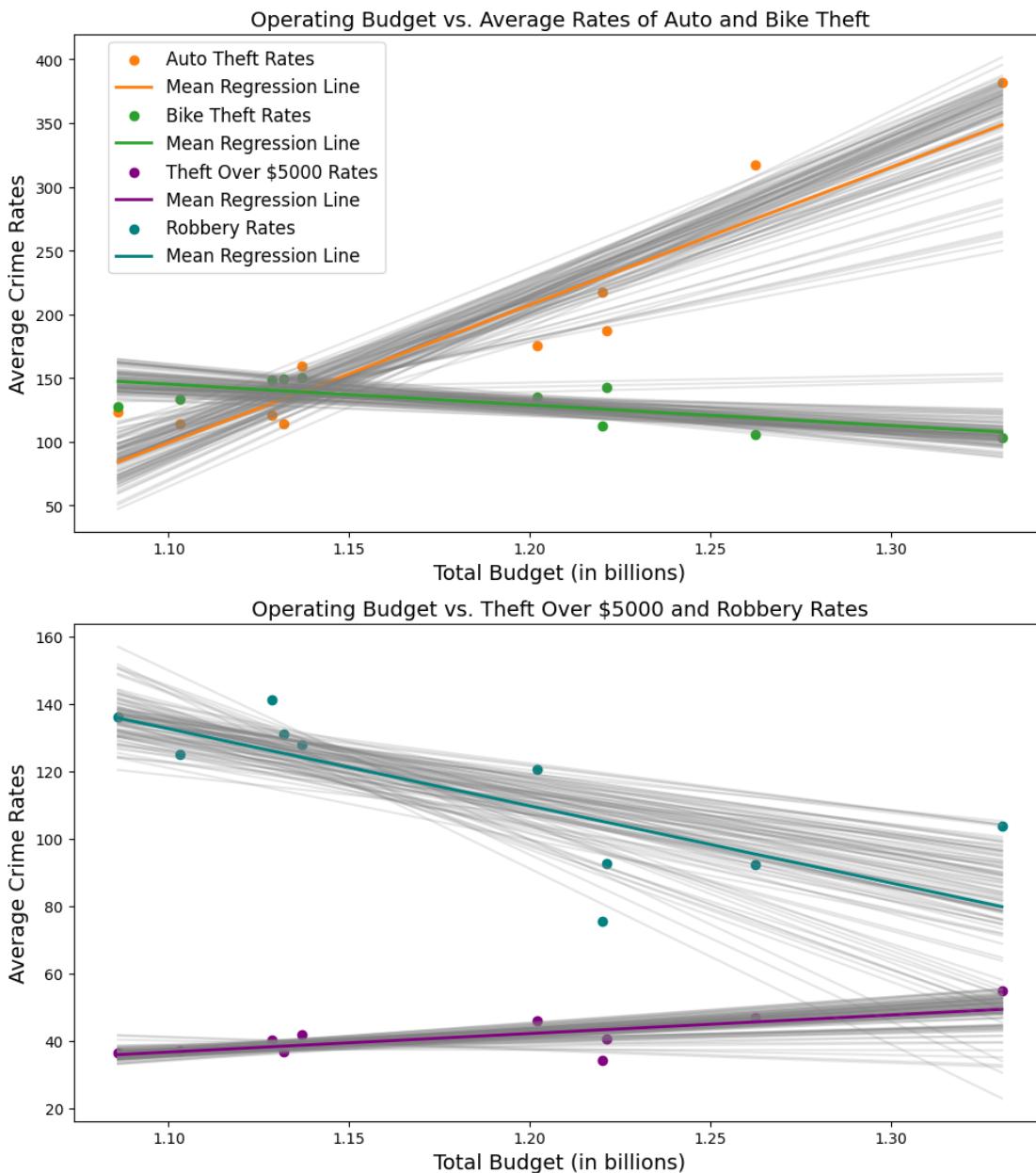
```

```
add_mean_regression_line(axes[1], result_df['total_budget_in_billions'],
                        boot_results['avg_robbery_rate'], data_color='teal')
axes[1].set_xlabel('Total Budget (in billions)', fontsize=14)
axes[1].set_ylabel('Average Crime Rates', fontsize=14)
axes[1].set_title('Operating Budget vs. Theft Over $5000 and Robbery Rates', fontsize=14)

fig.suptitle('Bootstrapped Regression Analysis of Operating Budget Vs. Crime Rates', fontsize=16)
fig.tight_layout(rect=[0, 0, 1, 0.96])
fig.legend(loc='upper left', bbox_to_anchor=(0.1, 0.9), fontsize=12)

plt.show()
```

Bootstrapped Regression Analysis of Operating Budget Vs. Crime Statistic Rates



Income vs Crime Statistics - David Griffin

```
[31]: #join crime and income datasets and create dataframe
query= f"""
WITH CrimeTotals AS (
    SELECT
        AREA_NAME,
```

```

year,
ASSAULT,
AUTOTHEFT,
BIKETHEFT,
BREAKENTER,
HOMICIDE,
ROBBERY,
SHOOTING,
THEFTFROMMV,
THEFTOVER,
ASSAULT_RATE,
HOMICIDE_RATE,
ROBBERY_RATE,
SHOOTING_RATE,
AUTOTHEFT_RATE,
BREAKENTER_RATE
FROM {CRIMETABLE}
WHERE year='2020'
),
IncomeData AS (
SELECT
    Neighbourhood,
    Income,
    YEAR,
    Population
FROM {INCOMETABLE}
)

SELECT
    I.YEAR,
    I.Neighbourhood,
    I.Income,
    I.Population,
    C.ASSAULT,
    C.HOMICIDE,
    C.ROBBERY,
    C.AUTOTHEFT,
    C.ASSAULT_RATE,
    C.HOMICIDE_RATE,
    C.ROBBERY_RATE,
    C.AUTOTHEFT_RATE,
    C.BREAKENTER,
    C.BREAKENTER_RATE
FROM IncomeData I
Join CrimeTotals C
On I.Neighbourhood=C.AREA_NAME
ORDER BY I.Neighbourhood;

```

```
#####
crime_df=pd.read_sql_query(query, engine)
crime_df.describe()
```

[31]:

	YEAR	Income	Population	ASSAULT	HOMICIDE	\
count	151.0	151.000000	151.000000	151.000000	151.000000	
mean	2020.0	50083.642384	17526.456954	113.576159	0.456954	
std	0.0	18275.444527	6225.065451	98.205800	0.745968	
min	2020.0	30800.000000	6260.000000	11.000000	0.000000	
25%	2020.0	37100.000000	12395.000000	55.500000	0.000000	
50%	2020.0	45760.000000	16670.000000	85.000000	0.000000	
75%	2020.0	55375.000000	22325.000000	133.000000	1.000000	
max	2020.0	141200.000000	33300.000000	756.000000	3.000000	

	ROBBERY	AUTOTHEFT	ASSAULT_RATE	HOMICIDE_RATE	ROBBERY_RATE	\
count	151.000000	151.000000	151.000000	151.000000	151.000000	
mean	17.443709	37.099338	601.820149	2.230829	91.718967	
std	16.628744	40.088694	471.090176	3.530964	80.170084	
min	1.000000	4.000000	110.430679	0.000000	11.983224	
25%	7.000000	17.000000	339.540222	0.000000	52.815395	
50%	15.000000	26.000000	508.151611	0.000000	77.433090	
75%	21.500000	44.500000	689.026428	4.691375	108.235043	
max	136.000000	407.000000	3582.768555	15.490267	644.519226	

	AUTOTHEFT_RATE	BREAKENTER	BREAKENTER_RATE
count	151.000000	151.000000	151.000000
mean	190.831371	43.748344	240.835126
std	141.915053	36.576170	182.616767
min	37.606354	3.000000	29.228371
25%	103.571892	21.000000	116.272907
50%	160.057831	34.000000	196.055359
75%	232.629288	50.500000	304.359421
max	1139.545288	204.000000	1105.997681

[32]:

```
#create variable that sorts each neighbourhoods income into quantiles
crime_df['Income Group']=0
for i in range(len(crime_df)):
    if crime_df['Income'][i]<=37100:
        crime_df.iloc[i,14]=1
    if crime_df['Income'][i]>37100 and crime_df['Income'][i]<=45760:
        crime_df.iloc[i,14]=2
    if crime_df['Income'][i]>45760 and crime_df['Income'][i]<=55375:
        crime_df.iloc[i,14]=3
    if crime_df['Income'][i]>55375:
        crime_df.iloc[i,14]=4
```

```
[33]: #make scatter plots for assault, robbery, autotheft and break and enter rates
      ↪with regressions for each plot
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes[0,0].scatter(crime_df['Income'],crime_df['ASSAULT_RATE'], color='tab:blue')
axes[0,0].set_xlabel('Income')
axes[0,0].set_ylabel('Assault Rate')
slope, intercept,r1,p1,_ = linregress(crime_df['Income'], ↪
      ↪crime_df['ASSAULT_RATE'])
x_vals=np.array([min(crime_df['Income']), max(crime_df['Income'])])
y_vals= slope*x_vals + intercept
axes[0,0].plot(x_vals,y_vals, color='tab:blue', linestyle='-', ↪
      ↪label='Regression Line')

axes[0,1].scatter(crime_df['Income'],crime_df['AUTOTHEFT_RATE'], color='tab:blue')
axes[0,1].set_xlabel('Income')
axes[0,1].set_ylabel('Auto Rate')
slope, intercept,r2,p2,_ = linregress(crime_df['Income'], ↪
      ↪crime_df['AUTOTHEFT_RATE'])
x_vals=np.array([min(crime_df['Income']), max(crime_df['Income'])])
y_vals= slope*x_vals + intercept
axes[0,1].plot(x_vals,y_vals, color='tab:blue', linestyle='-', ↪
      ↪label='Regression Line')

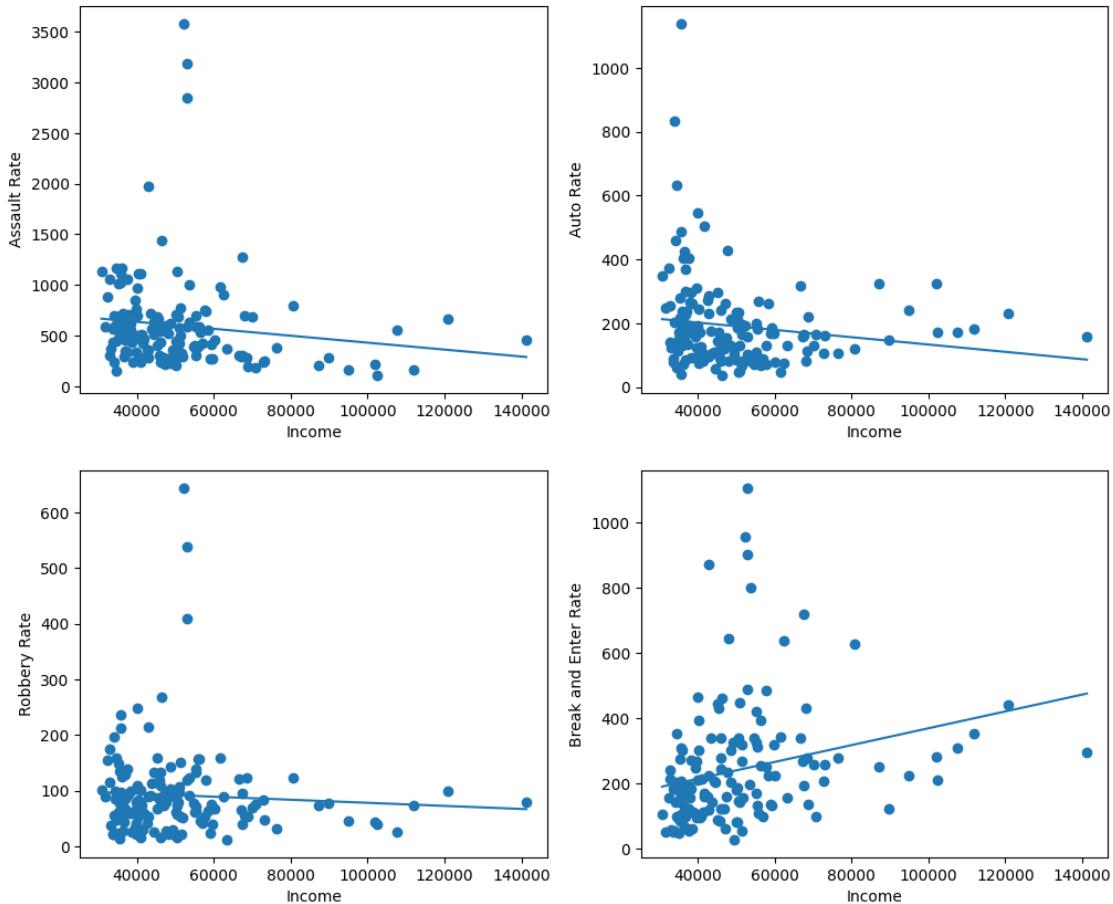
axes[1,0].scatter(crime_df['Income'],crime_df['ROBBERY_RATE'], color='tab:blue')
axes[1,0].set_xlabel('Income')
axes[1,0].set_ylabel('Robbery Rate')
slope, intercept,r3,p3,_ = linregress(crime_df['Income'], ↪
      ↪crime_df['ROBBERY_RATE'])
x_vals=np.array([min(crime_df['Income']), max(crime_df['Income'])])
y_vals= slope*x_vals + intercept
axes[1,0].plot(x_vals,y_vals, color='tab:blue', linestyle='-', ↪
      ↪label='Regression Line')

axes[1,1].scatter(crime_df['Income'],crime_df['BREAKENTER_RATE'], color='tab:blue')
axes[1,1].set_xlabel('Income')
axes[1,1].set_ylabel('Break and Enter Rate')
slope, intercept,r4,p4,_ = linregress(crime_df['Income'], ↪
      ↪crime_df['BREAKENTER_RATE'])
x_vals=np.array([min(crime_df['Income']), max(crime_df['Income'])])
y_vals= slope*x_vals + intercept
axes[1,1].plot(x_vals,y_vals, color='tab:blue', linestyle='-', ↪
      ↪label='Regression Line')\
```

```

plt.show()
print(f"Assault R-squared:{r1**2}, coefficient P-value:{p1}")
print(f"Autotheft R-squared:{r2**2}, coefficient P-value:{p2}")
print(f"Robbery R-squared:{r3**2}, coefficient P-value:{p3}")
print(f"Break and Enter R-squared:{r4**2}, coefficient P-value:{p4}")

```



Assault R-squared:0.017645701711067212, coefficient P-value:0.1039539860617245
 Autotheft R-squared:0.021867904397415174, coefficient P-value:0.06998142980679437
 Robbery R-squared:0.0039613181600412035, coefficient P-value:0.4426404066165349
 Break and Enter R-squared:0.06665997361769427, coefficient P-value:0.001370828658722553

```

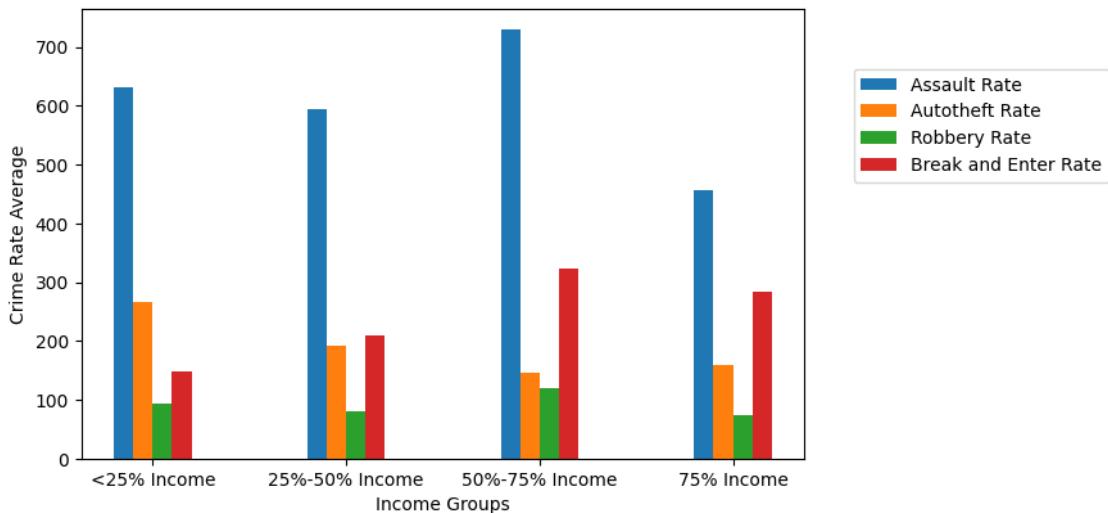
[34]: #plot crime rates against income quartiles
crime_dfg=crime_df.drop(columns=['YEAR','Neighbourhood'])
group1=crime_dfg.groupby('Income Group',as_index=False).mean()
fig, axes = plt.subplots(2, 2, figsize=(16, 10))
axes[0,0].bar(group1['Income Group'].unique()-0.
               ↪15,group1['ASSAULT RATE'],width=0.1,label='Assault Rate')

```

```

axes[0,0].bar(group1['Income Group'].unique()-0.
    ↪5,group1['AUTOTHEFT_RATE'],width=0.1,label='Autotheft Rate')
axes[0,0].bar(group1['Income Group'].unique()+0.
    ↪5,group1['ROBBERY_RATE'],width=0.1,label='Robbery Rate')
axes[0,0].bar(group1['Income Group'].unique()+0.
    ↪15,group1['BREAKENTER_RATE'],width=0.1,label='Break and Enter Rate')
axes[0,0].set_xticks(group1['Income Group'].unique(),('<25% Income','25%-50% Income',
    ↪'50%-75% Income','75% Income'))
axes[0,0].set_xlabel('Income Groups')
axes[0,0].set_ylabel('Crime Rate Average')
fig.delaxes(axes[0, 1])
fig.delaxes(axes[1, 0])
fig.delaxes(axes[1, 1])
fig.legend(bbox_to_anchor=(0.63,0.84))
plt.show()

```



```

[35]: #plot monicide rates against income for each neighbourhood
income_sort = crime_df.sort_values(by='Income', ascending=False)

fig, ax1 = plt.subplots(figsize=(14, 8))

x = np.arange(len(income_sort['Neighbourhood']))
width = 0.4
remove=[]

ax1.bar(x - width/2, income_sort['Income'], width, color='tab:blue',
    ↪label='Income')
ax1.set_xlabel('Neighbourhood', fontsize=14)
ax1.set_ylabel('Income', color='tab:blue', fontsize=14)

```

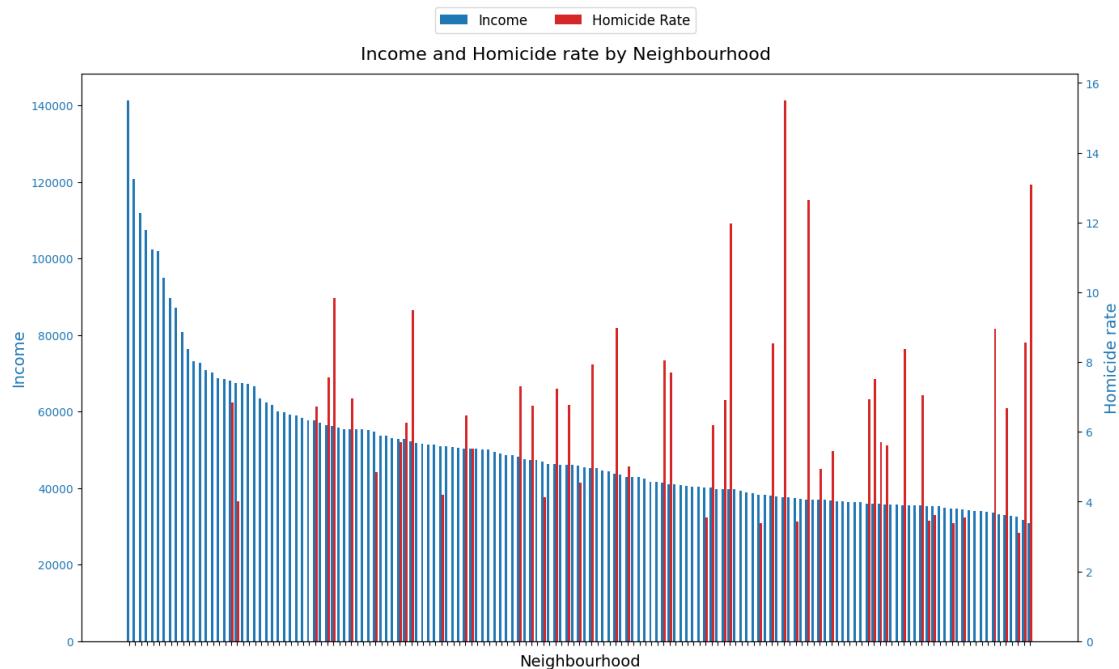
```

ax1.tick_params(axis='y', labelcolor='tab:blue')
ax1.set_xticks(x)
ax1.set_xticklabels(remove)

ax2 = ax1.twinx()
ax2.bar(x + width/2, income_sort['HOMICIDE_RATE'], width, color='tab:red', □
    ↪label='Homicide Rate')
ax2.set_ylabel('Homicide rate', color='tab:blue', fontsize=14)
ax2.tick_params(axis='y', labelcolor='tab:blue')

fig.suptitle('Income and Homicide rate by Neighbourhood', fontsize=16)
fig.legend(loc="upper center", bbox_to_anchor=(0.5, 1.05), ncol=2, fontsize=12)
plt.tight_layout()

```



Education Level vs Crime Statistics - Venkateshwaran Balu Soundararajan This Analysis examines the relationship between the highest certificate, diploma, or degree per household and the crime rate per neighborhood in Toronto. It aims to highlight the impact of education on crime rates across different neighborhoods in Toronto.

To begin our analysis we extracted the crime statistics from the database. In this step, we execute an SQL query to join two tables: one containing crime statistics (CRIMETABLE) and the other containing income and education data (INCOMETABLE). The objective is to create a comprehensive dataset that includes information on crime rates, education levels, and other relevant variables for each neighborhood. This resulting dataset will be used for further analysis, including calculating correlation matrices, performing regression analysis, and creating visualizations to understand

the relationship between education and crime in Toronto neighborhoods.

```
[36]: degreeandcrimestats=pd.read_sql_query(f"""SELECT
    ccr.area_name,
    ccr.geometry,
    ine.Degree_Rate,
    Population,
    ASSAULT_RATE,
    AUTOTHEFT_RATE,
    BIKETHEFT_RATE,
    BREAKENTER_RATE,
    HOMICIDE_RATE,
    ROBBERY_RATE,
    SHOOTING_RATE,
    THEFTFROMMV_RATE,
    THEFTOVER_RATE
FROM
    {CRIMETABLE} ccr
    INNER JOIN
    {INCOMETABLE} ine ON ccr.AREA_NAME = ine.Neighbourhood
    AND ccr.year = ine.year;""",engine)
degreeandcrimestats.head(5)
```

```
[36]:          area_name \
0  South Eglinton-Davisville
1           North Toronto
2      Dovercourt Village
3  Junction-Wallace Emerson
4     Yonge-Bay Corridor

                                     geometry  Degree_Rate  Population \
0  MULTIPOLYGON (((-79.3863542900264 43.697839855...  0.903673   22735
1  MULTIPOLYGON (((-79.3974398976879 43.706938508...  0.917218   15885
2  MULTIPOLYGON (((-79.4341164165158 43.660153857...  0.889742   12380
3  MULTIPOLYGON (((-79.4387032547807 43.667669388...  0.867343   23180
4  MULTIPOLYGON (((-79.3840431592607 43.644973675...  0.923290   12645

    ASSAULT_RATE  AUTOTHEFT_RATE  BIKETHEFT_RATE  BREAKENTER_RATE \
0    396.844605       72.593521      454.919434      338.769775
1    525.969727       80.356491      365.256775      642.851929
2    542.731018      130.255447      202.619583      340.111450
3    661.992248      172.353012      211.524155      430.882538
4   2842.933594     103.919243     1284.144897     1105.997681

    HOMICIDE_RATE  ROBBERY_RATE  SHOOTING_RATE  THEFTFROMMV_RATE \
0      4.839568     77.433090      4.839568      208.101440
1      7.305136     21.915407      0.000000      175.323257
```

2	7.236413	108.546204	0.000000	484.839722
3	0.000000	86.176506	19.585569	544.478821
4	0.000000	408.254150	14.845606	1002.078369

	THEFTOVER_RATE
0	24.197842
1	43.830814
2	21.709240
3	35.254025
4	259.798096

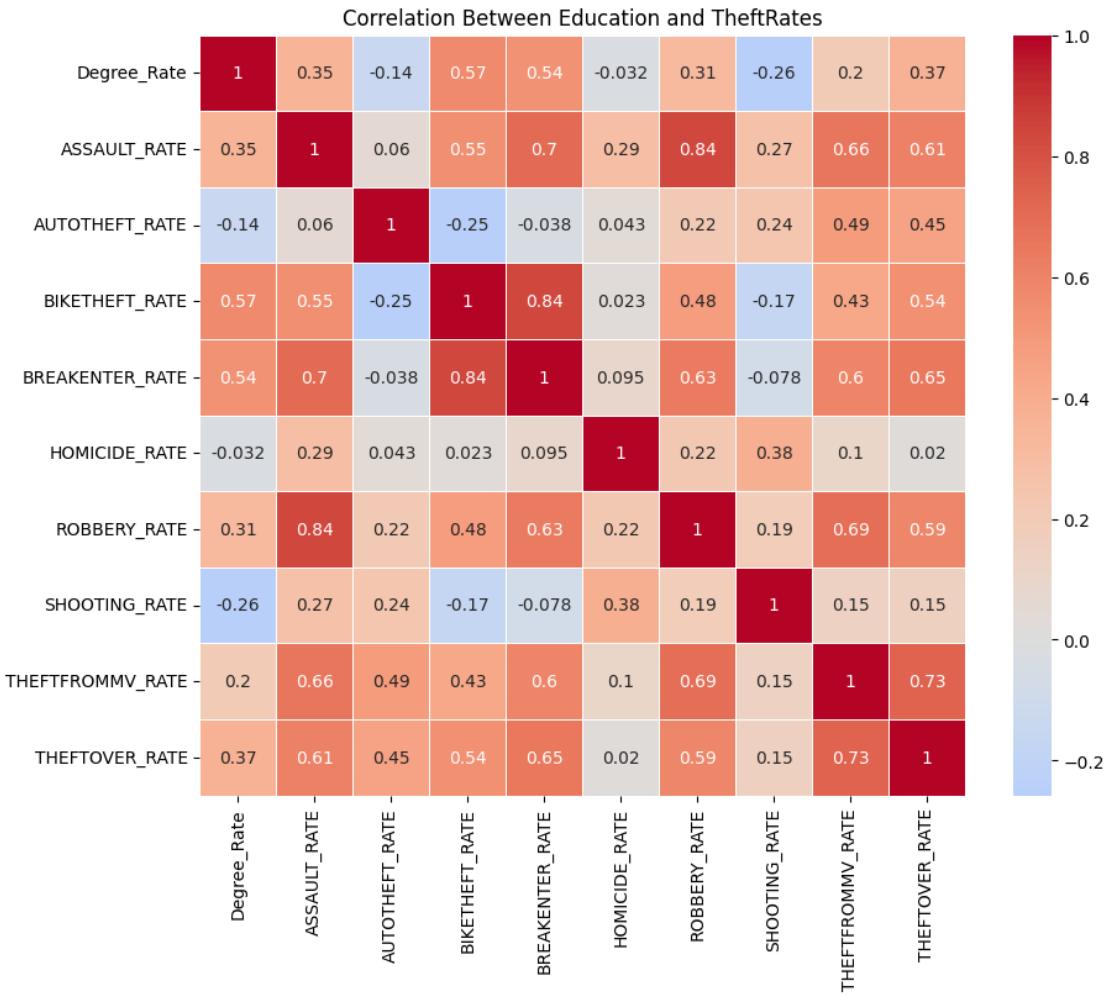
Correlation Analysis

We then created a correlation matrix to examine the strength and direction of the relationships between the degree rate and various crime rates. The correlation matrix provides valuable insights into how education impacts different types of crime.

```
[37]: import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = degreeandcrimestats[['Degree_Rate', 'ASSAULT_RATE',
                                         'AUTOTHEFT_RATE', 'BIKETHEFT_RATE', 'BREAKENTER_RATE',
                                         'HOMICIDE_RATE', 'ROBBERY_RATE',
                                         'SHOOTING_RATE', 'THEFTFROMMV_RATE',
                                         'THEFTOVER_RATE']].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0,
            linewidths=0.5)
plt.title('Correlation Between Education and TheftRates')
plt.show()
```



From the correlation matrix we infer that SHOOTING RATE has a weaker negative correlation (-0.26) with education, indicating that higher education levels may be associated with lower shooting incidents. Conversely, metrics like BIKETHEFT RATE (0.55) and BREAKENTER RATE (0.54) demonstrate moderate positive correlations, suggesting these crimes might be more prevalent in areas with higher education levels, possibly due to economic factors or population density.

Regression Analysis

We then performed regression analyses to verify the significance of the relationships observed. By fitting regression models for each crime rate variable against the degree rate, we determined the statistical significance and the extent to which education levels explain the variability in crime rates.

```
[38]: import statsmodels.api as sm

variables = ['ASSAULT_RATE', 'AUTOTHEFT_RATE', 'BIKETHEFT_RATE',
             'BREAKENTER_RATE', 'HOMICIDE_RATE',
```

```

    'ROBBERY_RATE', 'SHOOTING_RATE', 'THEFTFROMMV_RATE', ↴
    'THEFTOVER_RATE']

X = degreeandcrimestats['Degree_Rate']
X = sm.add_constant(X)
for var in variables:
    y = degreeandcrimestats[var]
    model_assault = sm.OLS(y, X).fit()
    p_value=model_assault.pvalues[1]
    adj_r_squared = model_assault.rsquared_adj
    print(f'Regression Results for {var}:')
    print(f' p-value for Degree_Rate: {p_value:.4f}')
    print(f' Adjusted R-squared: {adj_r_squared:.4f}')
    print('-' * 40)

```

Regression Results for ASSAULT_RATE:

 p-value for Degree_Rate: 0.0000
 Adjusted R-squared: 0.1190

Regression Results for AUTOTHEFT_RATE:

 p-value for Degree_Rate: 0.0839
 Adjusted R-squared: 0.0133

Regression Results for BIKETHEFT_RATE:

 p-value for Degree_Rate: 0.0000
 Adjusted R-squared: 0.3260

Regression Results for BREAKENTER_RATE:

 p-value for Degree_Rate: 0.0000
 Adjusted R-squared: 0.2822

Regression Results for HOMICIDE_RATE:

 p-value for Degree_Rate: 0.7003
 Adjusted R-squared: -0.0057

Regression Results for ROBBERY_RATE:

 p-value for Degree_Rate: 0.0001
 Adjusted R-squared: 0.0917

Regression Results for SHOOTING_RATE:

 p-value for Degree_Rate: 0.0014
 Adjusted R-squared: 0.0601

Regression Results for THEFTFROMMV_RATE:

 p-value for Degree_Rate: 0.0132
 Adjusted R-squared: 0.0341

Regression Results for THEFTOVER_RATE:

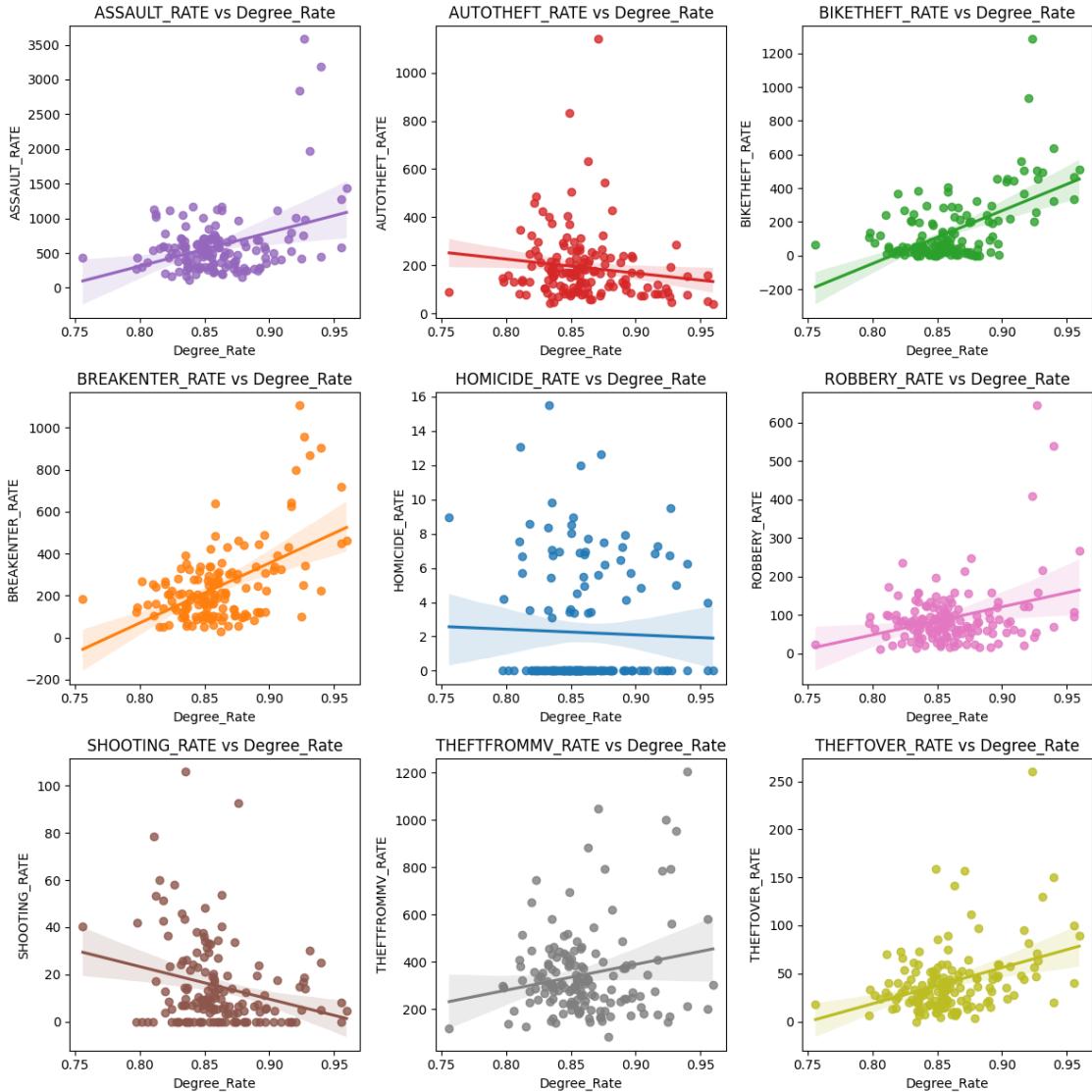
```
p-value for Degree_Rate: 0.0000
Adjusted R-squared: 0.1319
```

Following the correlation and regression analyses, we visualized the relationships between degree rate and various crime rates. This step involved creating a series of regression plots to provide a clear and intuitive representation of how education levels correlate with different types of crime across neighborhoods in Toronto.

We use the seaborn and matplotlib libraries to create a 3x3 grid of subplots, each displaying a regression plot for a specific crime rate variable against the degree rate. The regression plots help us visually assess the impact of education levels on different types of crime.

```
[39]: fig, axes = plt.subplots(3, 3, figsize=(12, 12))
colors = ['tab:purple', 'tab:red', 'tab:green', 'tab:orange', 'tab:blue', 'tab:pink',
          'tab:brown', 'tab:gray', 'tab:olive']
for i, var in enumerate(variables):
    row, col = divmod(i, 3)
    sns.regplot(y=var, x='Degree_Rate', data=degreeandcrimestats, ax=axes[row, col], color=colors[i])
    axes[row, col].set_title(f'{var} vs Degree_Rate')

plt.tight_layout()
plt.show()
```



To further enhance our understanding, we create spatial visualizations that map the distribution of crime rates and degree rates across different neighborhoods. These visualizations provide a geographical perspective, highlighting areas with high concentrations of crime and education levels

```
[40]: geopolygon = pd.read_sql_query(f'''SELECT
    ccr.AREA_NAME,
    ccr.geometry,
    ine.Degree,
    ine.Degree_Rate,
    (SUM(ASSAULT_RATE) + SUM(AUTOTHEFT_RATE) +
    SUM(BIKETHEFT_RATE) + SUM(BREAKENTER_RATE) + SUM(HOMICIDE_RATE) +
    SUM(ROBBERY_RATE) + SUM(SHOOTING_RATE) +
    SUM(THEFTFROMMV_RATE) + SUM(THEFTOVER_RATE)) / 9 AS AVG_CRIME_RATE
    FROM ccr
    JOIN ine ON ccr.GEO_ID = ine.GEO_ID
    GROUP BY ccr.AREA_NAME, ccr.geometry, ine.Degree, ine.Degree_Rate''')
```

```

        FROM
          {CRIMETABLE} ccr
        INNER JOIN
          {INCOMETABLE} ine ON ccr.AREA_NAME = ine.
        ↵Neighbourhood
          AND ccr.year = ine.year
        GROUP BY ccr.AREA_NAME, ccr.geometry, ine.
        ↵Degree, ine.Degree_Rate
          ORDER BY degree_rate DESC; ''', engine)
geopolygons.dtypes
geopolygons['geometry'] = geopolygons['geometry'].apply(wkt.loads)
geopolygons_gdf = gpd.GeoDataFrame(geopolygons, geometry='geometry')
geopolygons_gdf = geopolygons_gdf.set_crs(epsg=4326)
geopolygons_gdf.head()

```

[40]:

	AREA_NAME \	geometry	Degree	Degree_Rate \
0	Church-Wellesley		21420	0.959677
1	Wellington Place		24440	0.955808
2	Bay-Cloverhill		15930	0.955609
3	Downtown Yonge East		16640	0.940113
4	Fort York-Liberty Village		18710	0.939493

	AVG_CRIME_RATE
0	345.769547
1	379.555556
2	195.020355
3	756.388558
4	153.465399

[41]:

```

# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(20, 10))

# Plot the AVG_CRIME RATE
geopolygons_gdf.plot(column="AVG_CRIME RATE", cmap="Blues", legend=True, ↵
    ↵ax=axes[0])
axes[0].set_title("Crime Rates by Area", fontsize=16)
axes[0].set_xlabel("Longitude")
axes[0].set_ylabel("Latitude")

# Plot the Degree Rate

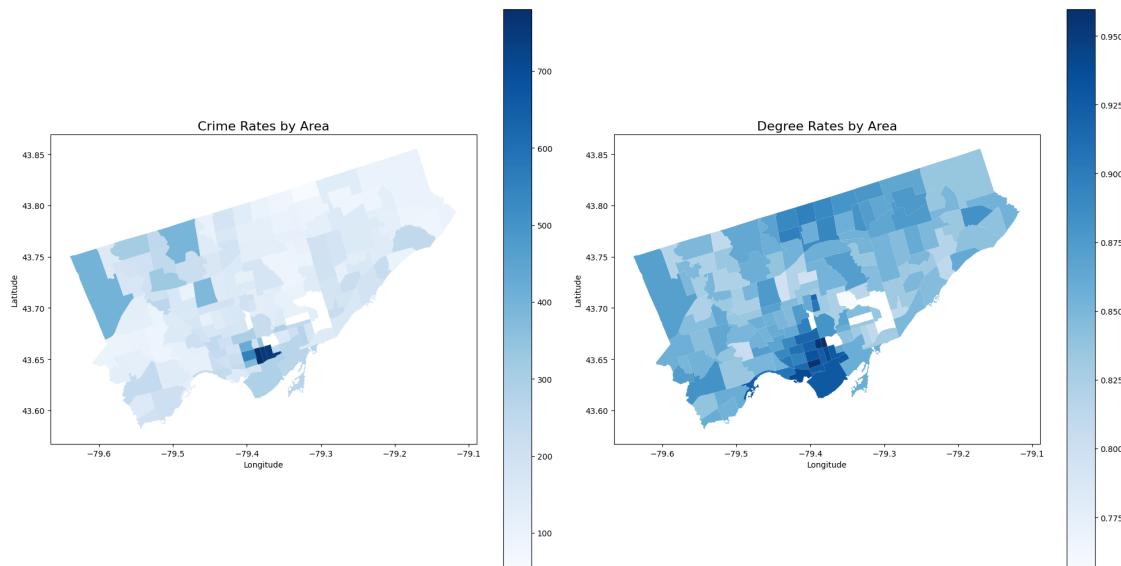
```

```

geopolygon_gdf.plot(column="Degree_Rate", cmap="Blues", legend=True, ax=axes[1])
axes[1].set_title("Degree Rates by Area", fontsize=16)
axes[1].set_xlabel("Longitude")
axes[1].set_ylabel("Latitude")

# Adjust layout and show the plot
plt.tight_layout()
plt.show()

```



Shelter Occupancy vs Crime Statistics - Jackson Meier

[42]: #creating combined table with shelter and crime data joined on neighbourhood

```

sheltercrime = ("""WITH CrimeTotals AS (
    SELECT
        AREA_NAME,
        year,
        ASSAULT_RATE,
        HOMICIDE_RATE,
        ROBBERY_RATE,
        SHOOTING_RATE,
        AUTOTHEFT_RATE,
        BREAKENTER_RATE
    FROM data604_converted_crime
    WHERE year='2020'
),
ShelterData AS (
    SELECT
        Occupancy_Date,
        Shelter_Name,

```

```

        Sector,
        Occupancy,
        Capacity,
        AREA_NAME
    FROM data604_daily_shelter_occupancy
)

SELECT
    S.Occupancy_Date,
    S.Shelter_Name,
    S.Sector,
    S.Occupancy,
    S.Capacity,
    S.AREA_NAME,
    C.AREA_NAME,
    C.ASSAULT_RATE,
    C.HOMICIDE_RATE,
    C.ROBBERY_RATE,
    C.AUTOTHEFT_RATE,
    C.BREAKENTER_RATE
FROM ShelterData S
Join CrimeTotals C
On S.AREA_NAME=C.AREA_NAME
ORDER BY S.AREA_NAME; """")

```

[43]: shelter_crime_df = SQL(sheltercrime)

[44]: #drop duplicate AREA_NAME column
shelter_crime_df = shelter_crime_df.loc[:,~shelter_crime_df.columns.duplicated()]

[45]: shelter_crime_df.head(3)

	Occupancy_Date	Shelter_Name	Sector	Occupancy	\
0	2017-01-01	Christie Refugee Welcome Centre	Families	66	
1	2017-01-01	Toronto Community Hostel	Families	26	
2	2017-01-01	YMCA Sprott House	Youth	24	

	Capacity	AREA_NAME	ASSAULT_RATE	HOMICIDE_RATE	ROBBERY_RATE	\
0	70	Annex	795.62561	0.0	122.876541	
1	24	Annex	795.62561	0.0	122.876541	
2	25	Annex	795.62561	0.0	122.876541	

	AUTOTHEFT_RATE	BREAKENTER_RATE
0	119.804626	626.670349
1	119.804626	626.670349
2	119.804626	626.670349

```
[46]: #creating occupancy rate
shelter_crime_df['Occupancy_Rate'] = shelter_crime_df['Occupancy']/
    ↪shelter_crime_df['Capacity']
#clean data - drop 'inf' rows
shelter_crime_df['Occupancy_Rate'] = shelter_crime_df['Occupancy_Rate'].replace([np.inf,-np.inf],np.nan)
shelter_crime_df = shelter_crime_df.dropna(subset=['Occupancy_Rate'])
```

```
[47]: occupancy_sorted = shelter_crime_df.groupby('AREA_NAME',as_index = False)[['Occupancy_Rate','ASSAULT_RATE','HOMICIDE_RATE','ROBBERY_RATE','AUTOTHEFT_RATE','BREAKENTER_RATE']].mean().sort_values(by='Occupancy_Rate',ascending = False)
occupancy_sorted[(occupancy_sorted != 0).all(axis=1)].shape
```

[47]: (16, 7)

```
[48]: import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import linregress

# List of crime rates to plot
crime_columns = ['ASSAULT_RATE', 'HOMICIDE_RATE', 'ROBBERY_RATE',
                  'AUTOTHEFT_RATE', 'BREAKENTER_RATE']

# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10)) # Adjust rows/
    ↪columns as needed
axes = axes.flatten()

# Plot each crime rate against Occupancy_Rate with correlation line
for i, crime in enumerate(crime_columns):
    # Scatter plot
    axes[i].scatter(occupancy_sorted['Occupancy_Rate'], ↪
                    occupancy_sorted[crime], alpha=0.7)

    # Calculate the linear regression line
    slope, intercept, r_value, p_value, std_err = ↪
        linregress(occupancy_sorted['Occupancy_Rate'], occupancy_sorted[crime])

    # Generate the line using the slope and intercept
    line = slope * occupancy_sorted['Occupancy_Rate'] + intercept
    axes[i].plot(occupancy_sorted['Occupancy_Rate'], line, label=f'{crime} fit', ↪
                line, linestyle='--')

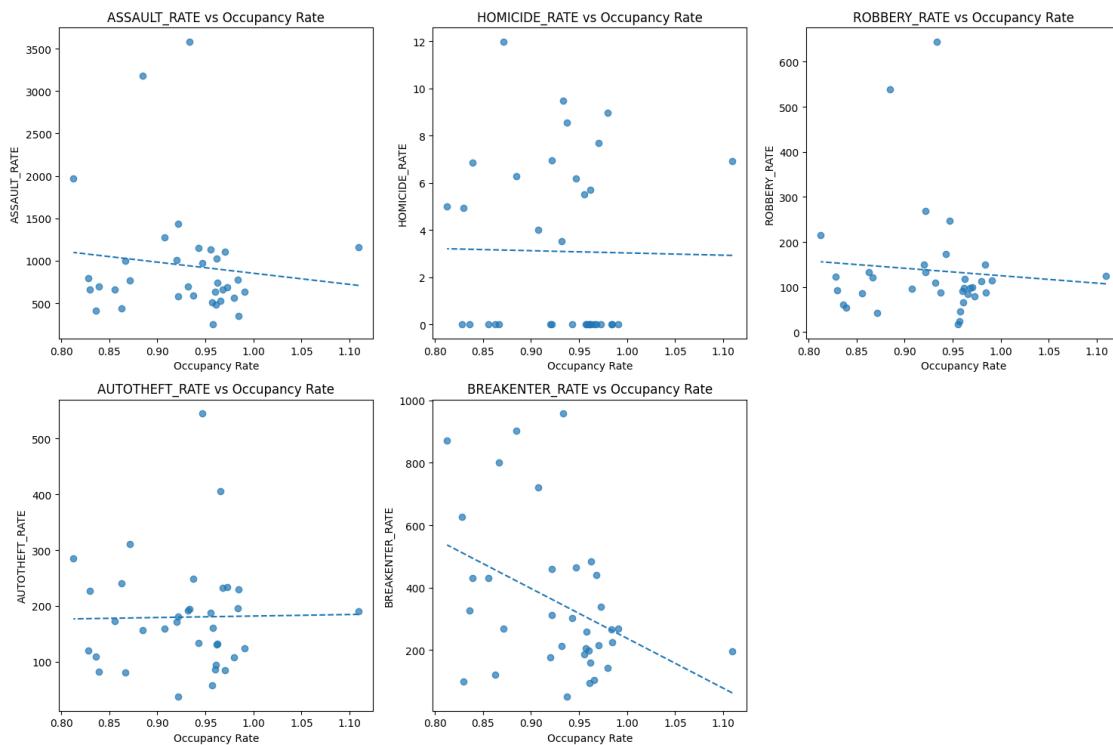
    # Titles and labels
    axes[i].set_title(f'{crime} vs Occupancy Rate')
    axes[i].set_xlabel('Occupancy Rate')
    axes[i].set_ylabel(crime)
```

```

# Remove empty subplot (if any)
for j in range(len(crime_columns), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()

```



```

[49]: #calculate correlation and p-vals
from scipy.stats import pearsonr
correlations = {}
p_values = {}

for crime in crime_columns:
    corr, p_value = pearsonr(occupancy_sorted['Occupancy_Rate'], ↴
                             occupancy_sorted[crime])
    correlations[crime] = corr
    p_values[crime] = p_value

# Print the correlations and p-values
for crime in crime_columns:
    print(f"Correlation between Occupancy Rate and {crime}: ↴
          {correlations[crime]:.2f}")

```

```

print(f"P-value for {crime}: {p_values[crime]:.4f}")
if p_values[crime] < 0.05:
    print("The correlation is statistically significant.")
else:
    print("The correlation is not statistically significant.")
print()

```

Correlation between Occupancy Rate and ASSAULT_RATE: -0.11

P-value for ASSAULT_RATE: 0.5110

The correlation is not statistically significant.

Correlation between Occupancy Rate and HOMICIDE_RATE: -0.02

P-value for HOMICIDE_RATE: 0.9287

The correlation is not statistically significant.

Correlation between Occupancy Rate and ROBBERY_RATE: -0.08

P-value for ROBBERY_RATE: 0.6475

The correlation is not statistically significant.

Correlation between Occupancy Rate and AUTOTHEFT_RATE: 0.02

P-value for AUTOTHEFT_RATE: 0.9241

The correlation is not statistically significant.

Correlation between Occupancy Rate and BREAKENTER_RATE: -0.40

P-value for BREAKENTER_RATE: 0.0173

The correlation is statistically significant.

Bike Theft vs Bike Crimes - Steen Rasmussen

```

[50]: # query for neighbourhoods with the most bike rack capacity
capacity_query = pd.read_sql_query(f"""
    SELECT
        AREA_NAME,
        SUM(CAPACITY) AS total_bike_rack_capacity,
        SUM(BIKETHEFT) AS total_bike_thefts,
        SUM(POPULATION_2023) AS total_population
    FROM
        {BIKECRIMETABLE}
    WHERE
        Year = 2023
    GROUP BY
        AREA_NAME
    "", engine)

display(capacity_query)

capacity_query.to_csv('capacity_query.csv', index=False)

```

	AREA_NAME	total_bike_rack_capacity	\
0	Annex	109.0	
1	Avondale	8.0	
2	Bendale-Glen Andrew	8.0	
3	Birchcliffe-Cliffside	8.0	
4	Cabbagetown-South St.James Town	16.0	
5	Church-Wellesley	8.0	
6	Danforth	50.0	
7	Dovercourt Village	42.0	
8	Downtown Yonge East	61.0	
9	Dufferin Grove	8.0	
10	Harbourfront-CityPlace	48.0	
11	High Park North	20.0	
12	Humber Bay Shores	8.0	
13	Junction-Wallace Emerson	44.0	
14	Kensington-Chinatown	116.0	
15	Lawrence Park North	8.0	
16	Little Portugal	64.0	
17	Mimico-Queensway	16.0	
18	Moss Park	70.0	
19	Mount Olive-Silverstone-Jamestown	32.0	
20	New Toronto	8.0	
21	O'Connor-Parkview	8.0	
22	Palmerston-Little Italy	118.0	
23	Roncesvalles	8.0	
24	Rosedale-Moore Park	8.0	
25	South Riverdale	40.0	
26	St Lawrence-East Bayfront-The Islands	124.0	
27	Steeles	16.0	
28	Tam O'Shanter-Sullivan	8.0	
29	The Beaches	8.0	
30	Thorncliffe Park	16.0	
31	Trinity-Bellwoods	46.0	
32	University	30.0	
33	Wellington Place	114.0	
34	West Queen West	56.0	
35	Willowdale West	64.0	
36	Yonge-Bay Corridor	224.0	
37	Yonge-Eglinton	56.0	
38	Yonge-St.Clair	18.0	

	total_bike_thefts	total_population
0	891.0	379522.0
1	9.0	16149.0
2	10.0	20773.0
3	6.0	23417.0
4	62.0	23936.0
5	102.0	22938.0

6	39.0	30519.0
7	228.0	83022.0
8	1435.0	126161.0
9	58.0	12709.0
10	528.0	188862.0
11	28.0	24474.0
12	16.0	20822.0
13	220.0	104960.0
14	1096.0	169832.0
15	6.0	15581.0
16	85.0	86995.0
17	10.0	36626.0
18	495.0	111035.0
19	11.0	35386.0
20	9.0	12205.0
21	8.0	19901.0
22	297.0	132291.0
23	21.0	15939.0
24	45.0	23053.0
25	340.0	120540.0
26	424.0	124012.0
27	6.0	52790.0
28	16.0	29770.0
29	25.0	23097.0
30	20.0	45016.0
31	230.0	89070.0
32	159.0	22692.0
33	880.0	317570.0
34	168.0	60544.0
35	6.0	19546.0
36	2964.0	191503.0
37	80.0	68715.0
38	28.0	27906.0

```
[51]: # most capacity by neighbourhood

capacity_df = pd.read_csv('capacity_query.csv')

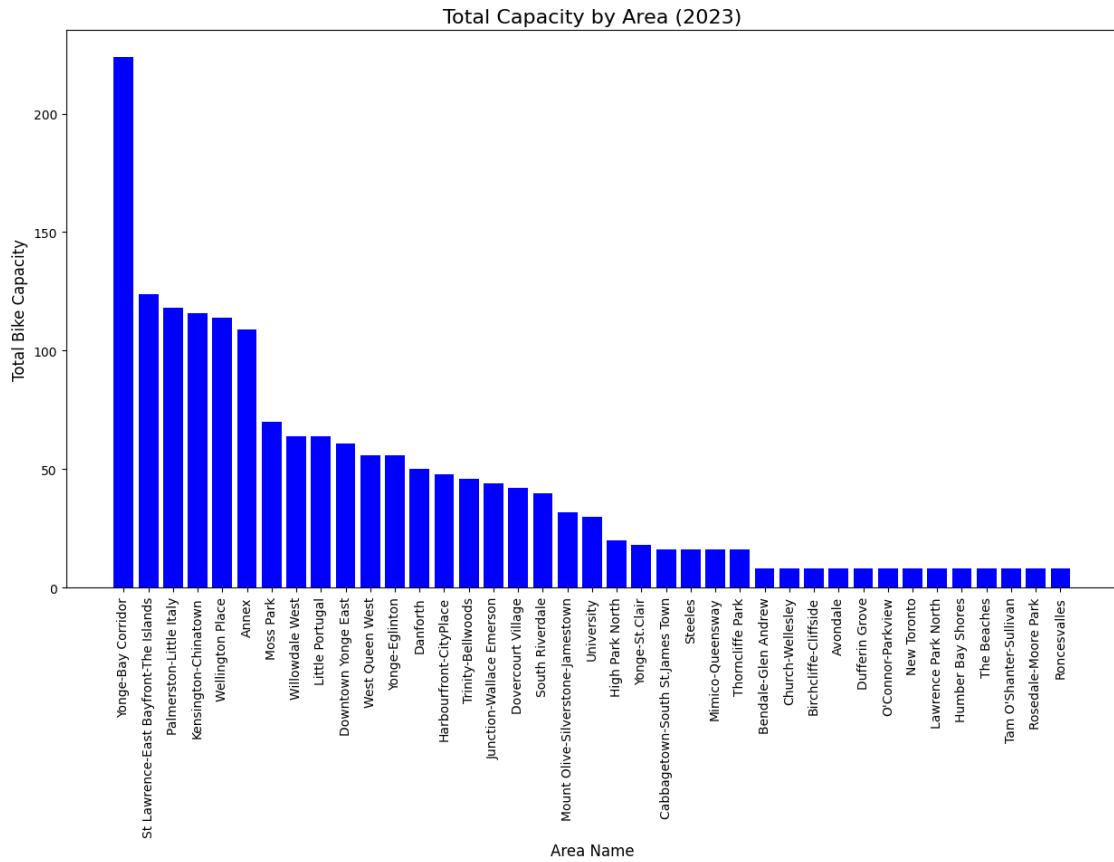
# sort by most to least
data_sorted = capacity_df.sort_values(by='total_bike_rack_capacity', ↴
                                         ascending=False)

# bar plot
plt.figure(figsize=(15, 8))
plt.bar(data_sorted['AREA_NAME'], data_sorted['total_bike_rack_capacity'], ↴
         color='blue')
plt.title('Total Capacity by Area (2023)', fontsize=16)
```

```

plt.xlabel('Area Name', fontsize=12)
plt.ylabel('Total Bike Capacity', fontsize=12)
plt.xticks(rotation=90)
plt.show()

```



[52]: # graph of bike capacity by bike thefts and correlation term

```

Bike_capacity_plot_query = pd.read_csv('capacity_query.csv')

x = Bike_capacity_plot_query['total_bike_rack_capacity']
y = Bike_capacity_plot_query['total_bike_thefts']

slope = np.cov(x, y, ddof=0)[0, 1] / np.var(x, ddof=0)
intercept = np.mean(y) - slope * np.mean(x)

correlation_matrix = np.corrcoef(x, y)
r_value = correlation_matrix[0, 1]

regression_line = slope * x + intercept

```

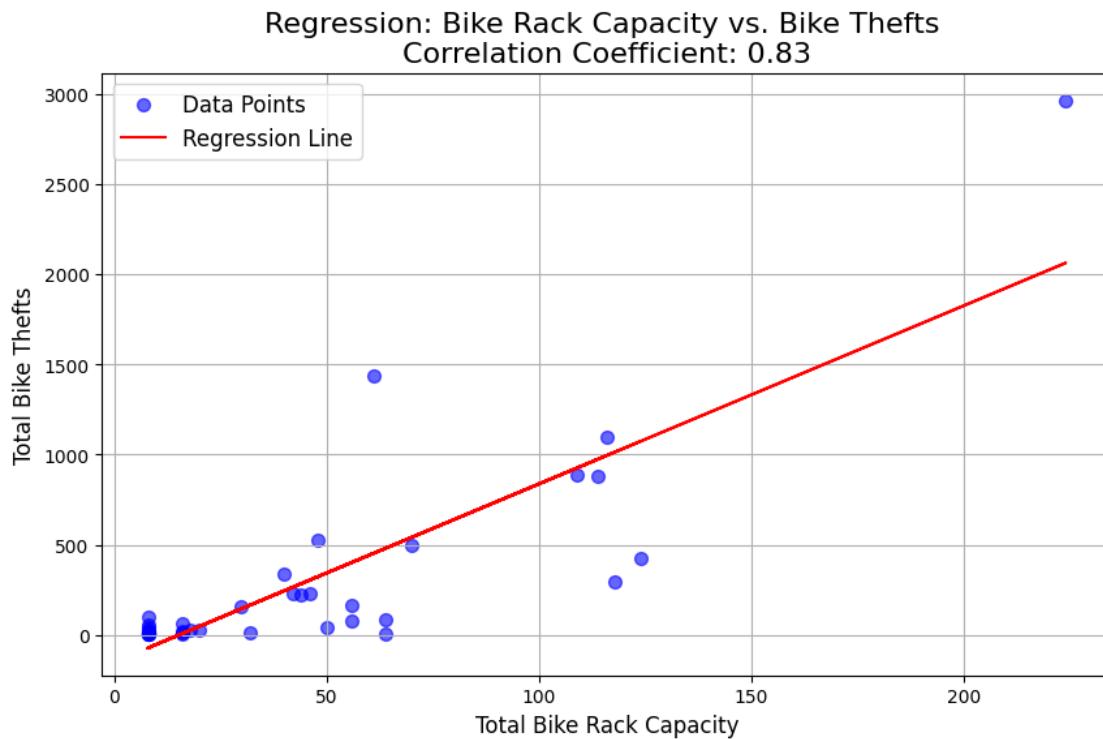
```

plt.figure(figsize=(10, 6))
plt.scatter(x, y, color='blue', s=50, alpha=0.6, label='Data Points') # ↵scatter points
plt.plot(x, regression_line, color='red', label='Regression Line') # ↵regression line

# add titles, labels, and the correlation coefficient
plt.title(f'Regression: Bike Rack Capacity vs. Bike Thefts \nCorrelation ↵Coefficient: {r_value:.2f}', fontsize=16)
plt.xlabel('Total Bike Rack Capacity', fontsize=12)
plt.ylabel('Total Bike Thefts', fontsize=12)
plt.legend(fontsize=12)
plt.grid(True)

# show the plot
plt.show()

```



Bike Theft vs Total Bike Racks

```
[53]: Bike_Theft_query = pd.read_sql_query(f"""
SELECT
    AREA_NAME,
    SUM(BIKETHEFT) AS total_bike_thefts,
```

```

        COUNT(*) AS total_racks,
        AVG(BIKETHEFT_RATE) AS bike_theft_rate,
        SUM(POPULATION_2023) AS total_population
    FROM
        {BIKECRIMETABLE}
    WHERE
        Year = 2023
    GROUP BY
        AREA_NAME
    """", engine)

display(Bike_Theft_query)

Bike_Theft_query.to_csv('bike_theft_area.csv', index=False)

```

	AREA_NAME	total_bike_thefts	total_racks	\
0	Annex	891.0	11	
1	Avondale	9.0	1	
2	Bendale-Glen Andrew	10.0	1	
3	Birchcliffe-Cliffside	6.0	1	
4	Cabbagetown-South St. James Town	62.0	2	
5	Church-Wellesley	102.0	1	
6	Danforth	39.0	3	
7	Dovercourt Village	228.0	6	
8	Downtown Yonge East	1435.0	7	
9	Dufferin Grove	58.0	1	
10	Harbourfront-CityPlace	528.0	6	
11	High Park North	28.0	1	
12	Humber Bay Shores	16.0	1	
13	Junction-Wallace Emerson	220.0	4	
14	Kensington-Chinatown	1096.0	8	
15	Lawrence Park North	6.0	1	
16	Little Portugal	85.0	5	
17	Mimico-Queensway	10.0	2	
18	Moss Park	495.0	5	
19	Mount Olive-Silverstone-Jamestown	11.0	1	
20	New Toronto	9.0	1	
21	O'Connor-Parkview	8.0	1	
22	Palmerston-Little Italy	297.0	9	
23	Roncesvalles	21.0	1	
24	Rosedale-Moore Park	45.0	1	
25	South Riverdale	340.0	4	
26	St Lawrence-East Bayfront-The Islands	424.0	4	
27	Steeles	6.0	2	
28	Tam O'Shanter-Sullivan	16.0	1	
29	The Beaches	25.0	1	
30	Thorncliffe Park	20.0	2	
31	Trinity-Bellwoods	230.0	5	

32	University	159.0	3
33	Wellington Place	880.0	11
34	West Queen West	168.0	4
35	Willowdale West	6.0	1
36	Yonge-Bay Corridor	2964.0	13
37	Yonge-Eglinton	80.0	5
38	Yonge-St.Clair	28.0	2

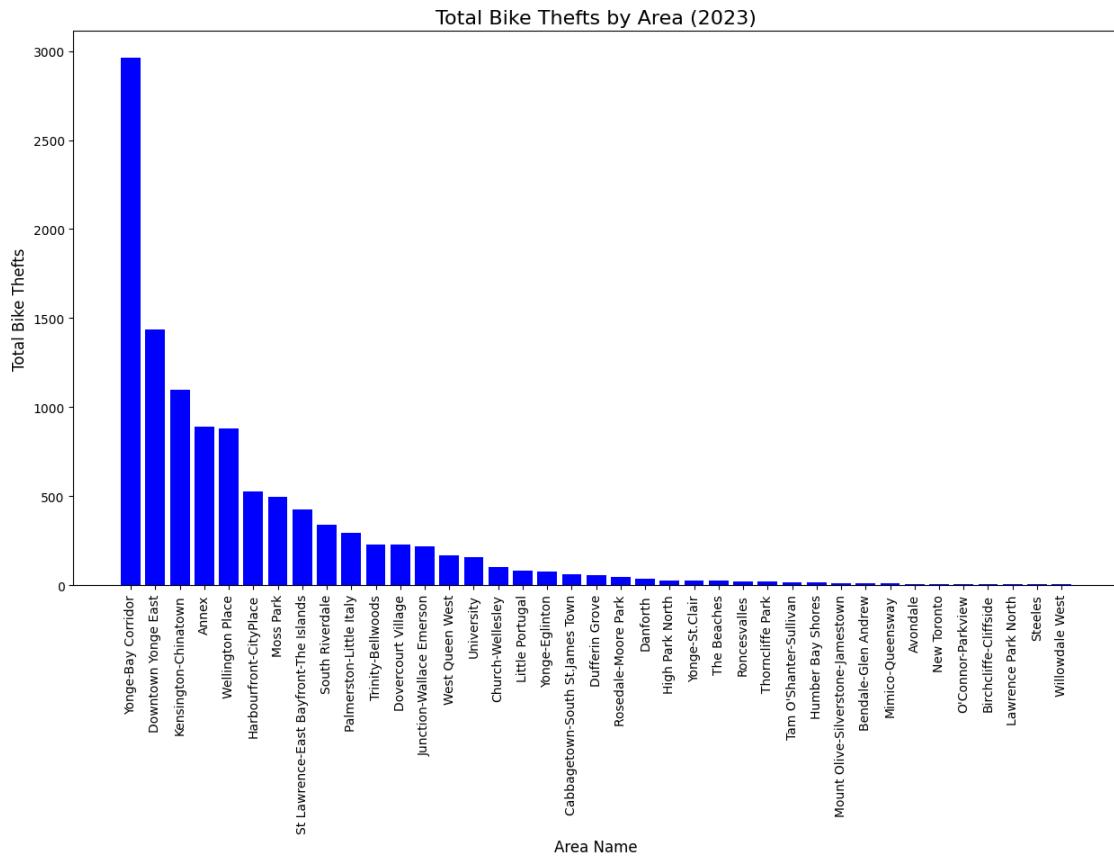
	bike_theft_rate	total_population
0	234.768997	379522.0
1	55.731007	16149.0
2	48.139412	20773.0
3	25.622412	23417.0
4	259.024078	23936.0
5	444.676941	22938.0
6	127.789246	30519.0
7	274.626007	83022.0
8	1137.435547	126161.0
9	456.369507	12709.0
10	279.569214	188862.0
11	114.407127	24474.0
12	76.841804	20822.0
13	209.603653	104960.0
14	645.343628	169832.0
15	38.508438	15581.0
16	97.706764	86995.0
17	27.303009	36626.0
18	445.805389	111035.0
19	31.085741	35386.0
20	73.740273	12205.0
21	40.198986	19901.0
22	224.505066	132291.0
23	131.752304	15939.0
24	195.202362	23053.0
25	282.064056	120540.0
26	341.902405	124012.0
27	11.365789	52790.0
28	53.745380	29770.0
29	108.239166	23097.0
30	44.428646	45016.0
31	258.223877	89070.0
32	700.687439	22692.0
33	277.104248	317570.0
34	277.484131	60544.0
35	30.696817	19546.0
36	1547.756470	191503.0
37	116.422905	68715.0
38	100.336845	27906.0

```
[54]: # bar plot for neighbourhoods with the most bike thefts

# read file
Bike_crime_df = pd.read_csv('bike_theft_area.csv')

# sort by most to least
data_sorted = Bike_crime_df.sort_values(by='total_bike_thefts', ascending=False)

# bar plot
plt.figure(figsize=(15, 8))
plt.bar(data_sorted['AREA_NAME'], data_sorted['total_bike_thefts'], color='blue')
plt.title('Total Bike Thefts by Area (2023)', fontsize=16)
plt.xlabel('Area Name', fontsize=12)
plt.ylabel('Total Bike Thefts', fontsize=12)
plt.xticks(rotation=90)
plt.show()
```



```
[55]: # comparing total bike racks to total bike thefts

df = pd.read_csv('bike_theft_area.csv')

df = df.sort_values('total_racks', ascending=False)

fig, ax1 = plt.subplots(figsize=(14, 8))

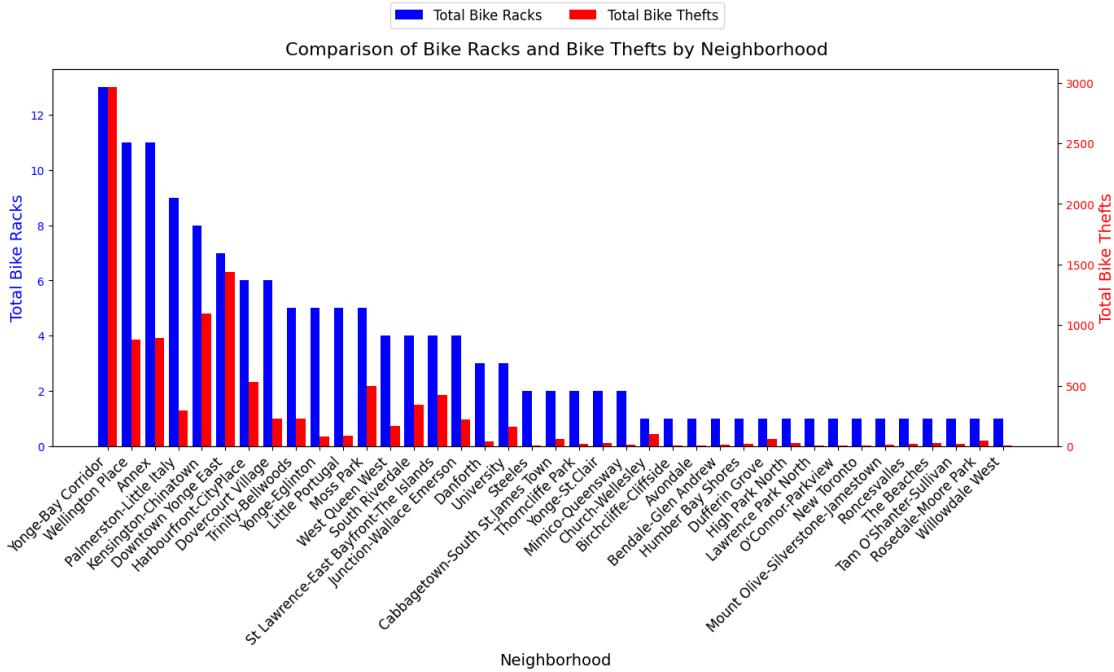
x = np.arange(len(df['AREA_NAME']))
width = 0.4

ax1.bar(x - width/2, df['total_racks'], width, color='b', label='Total Bike Racks')
ax1.set_xlabel('Neighborhood', fontsize=14)
ax1.set_ylabel('Total Bike Racks', color='b', fontsize=14)
ax1.tick_params(axis='y', labelcolor='b')
ax1.set_xticks(x)
ax1.set_xticklabels(df['AREA_NAME'], rotation=45, ha='right', fontsize=12)

ax2 = ax1.twinx()
ax2.bar(x + width/2, df['total_bike_thefts'], width, color='r', label='Total Bike Thefts')
ax2.set_ylabel('Total Bike Thefts', color='r', fontsize=14)
ax2.tick_params(axis='y', labelcolor='r')

fig.suptitle('Comparison of Bike Racks and Bike Thefts by Neighborhood', font-size=16)
fig.legend(loc="upper center", bbox_to_anchor=(0.5, 1.05), ncol=2, font-size=12)
plt.tight_layout()

plt.show()
```



```
[56]: # graph of number of bike racks by bike thefts and correlation term

Bike_Theft_query = pd.read_csv('bike_theft_area.csv')

x = Bike_Theft_query['total_racks']
y = Bike_Theft_query['total_bike_thefts']

slope = np.cov(x, y, ddof=0)[0, 1] / np.var(x, ddof=0)
intercept = np.mean(y) - slope * np.mean(x)

correlation_matrix = np.corrcoef(x, y)
r_value = correlation_matrix[0, 1]

regression_line = slope * x + intercept

plt.figure(figsize=(10, 6))
plt.scatter(x, y, color='blue', s=50, label='Data Points') # scatter points
plt.plot(x, regression_line, color='red', label='Regression Line') # regression line

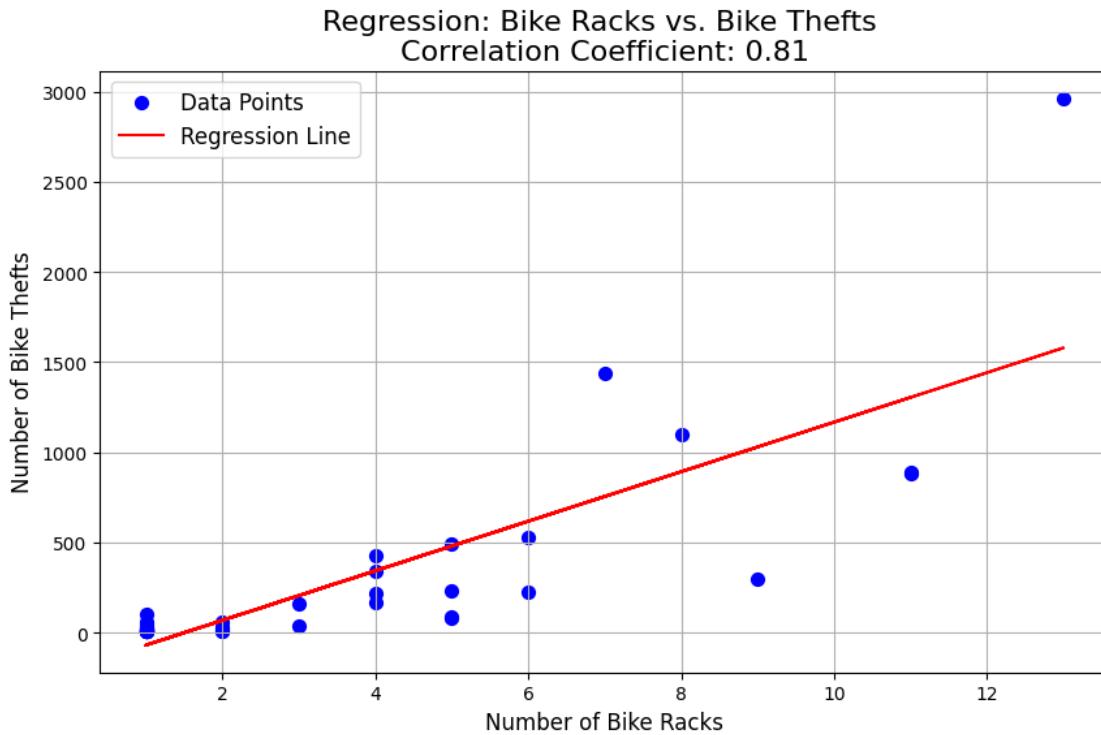
plt.title(f'Regression: Bike Racks vs. Bike Thefts \nCorrelation Coefficient: {r_value:.2f}', fontsize=16)
plt.xlabel('Number of Bike Racks', fontsize=12)
plt.ylabel('Number of Bike Thefts', fontsize=12)
plt.legend(fontsize=12)
```

```

plt.grid(True)

plt.show()

```



Bike rack and neighbourhood map

```

[57]: # load GeoJSON Files
bikerack_gdf = gpd.read_file("Bicycle Parking Racks Data.geojson")
crime_gdf = gpd.read_file("neighbourhood-crime-rates.geojson")

print("Geometry types in bike rack data:")
print(bikerack_gdf.geometry.geom_type.value_counts())

bikerack_gdf["geometry"] = bikerack_gdf["geometry"].apply(lambda geom: geom.
    if geom.geom_type == "MultiPoint" else None)

bikerack_gdf = bikerack_gdf[bikerack_gdf.geometry.notna()]

if bikerack_gdf.empty:
    print("No valid Point geometries found. Using fallback.")
    map_center = [43.7, -79.4]
else:
    map_center = [

```

```

        bikerack_gdf.geometry.y.mean(),
        bikerack_gdf.geometry.x.mean()
    ]
    print("Valid Point geometries found:", len(bikerack_gdf))

# make sure CRS Matches
if bikerack_gdf.crs != crime_gdf.crs:
    crime_gdf = crime_gdf.to_crs(bikerack_gdf.crs)

# create Folium Map
m = folium.Map(location=map_center, zoom_start=12)

for idx, row in bikerack_gdf.iterrows():
    folium.Marker(
        location=[row.geometry.y, row.geometry.x],
        popup=f"<b>Address:</b> {row.get('ADDRESS_FULL', 'N/A')}<br><b>Capacity:</b> {row.get('CAPACITY', 'N/A')}"
    ).add_to(m)

folium.GeoJson(
    crime_gdf,
    name="Neighborhoods",
    style_function=lambda x: {
        "fillColor": "yellow",
        "color": "black",
        "weight": 1,
        "fillOpacity": 0.3,
    },
    tooltip=folium.GeoJsonTooltip(
        fields=["AREA_NAME"],
        aliases=["Neighborhood:"],
    ),
).add_to(m)

m

```

Geometry types in bike rack data:
MultiPoint 241
Name: count, dtype: int64
Valid Point geometries found: 241

[57]: <folium.folium.Map at 0x715bc88e0370>

Top 10 Bike Racks with the Highest Capacity

[58]: # top 10 list of bike racks with the highest capacity

```

highest_capacity_query = f"""

```

```

SELECT DISTINCT ADDRESS_FULL, CAPACITY
FROM {BIKECRIMETABLE}
ORDER BY CAPACITY DESC
LIMIT 10;
"""

capacity_result = pd.read_sql(highest_capacity_query, engine)

display(capacity_result)

```

	ADDRESS_FULL	CAPACITY
0	61 Front St W	72
1	5100 Yonge St	64
2	675 Bloor St W	48
3	100 Queen St W	40
4	175 Mount Olive Dr	32
5	65 Dundas St W	32
6	743 Pape Ave	32
7	35 York St	30
8	2 Bloor St W	24
9	236 Augusta Ave	24

Totals of Each Bike Rack Class

```

[59]: map_class_query = f"""
        SELECT MAP_CLASS, COUNT(DISTINCT ADDRESS_FULL) AS total_bike_racks
        FROM {BIKECRIMETABLE}
        GROUP BY MAP_CLASS
        ORDER BY total_bike_racks DESC;
"""

map_class_result = pd.read_sql(map_class_query, engine)
display(map_class_result)

```

	MAP_CLASS	total_bike_racks
0	Multi-Bike Rack	110
1	Bike Corral	15
2	Multi-Bike Rack (Angled)	6
3	Bike Shelter	3

5 Discussion

When analyzing the impact of police budgetary data on crime statistics in Toronto, we observed a couple of unique trends. When simply looking at the raw data we noticed that as the total operating budget increased over time, so did the average rates for assault, auto theft, and shooting. Although this may not seem intuitive, it may have been due to a couple of factors. An increased budget could mean there was an improvement in reporting or detection of specific crimes, which could lead to an increase observed in those crime rates. It may also be an indication of improper allocation of resources for various years. Although there was some breakdown of allocation provided, it was not very stringent, so it is possible that despite an increase in budget, areas useful in auto theft were

underfunded, as an example.

When trying to run regressional analysis on our data, we realized that our sample size for budgetary data was too small, therefore we could not rely on standard regression methods. To overcome this problem we decided it would be best to bootstrap our data. Upon doing so we found which crimes had significant relationships with total operating budget by seeing which confidence intervals did not contain 0. Of all the crimes analyzed, we found significant relationships between gross operating budget and average rates of auto theft, theft over \$5000, bike theft, and robbery.

Once we determined which relationships were significant, we plotted the bootstrapped regression lines, as well as the mean regression line. Of all the plots, we observed the strongest positive relationship between total budget and auto theft rates, and the strongest negative relationship between total budget and robbery rates.

Analysing the impact of income in neighbourhoods on various crime rates showed some interesting results. We began by visualizing the relationships between assault, robbery, auto theft, and break-and-enter rates against income using scatter plots and regression lines. As can be seen from the results, income showed a slight negative relationship with assault and auto theft rates, with each regression having poor R squared values, but neither showing significant coefficients. Additionally, income showed a negative relationship with robbery rates, but did not have a significant coefficient. Lastly, income showed a strong positive relationship with break and enter rates, with a poor R squared, but a very significant coefficient.

For the same 4 crime rates we broke income into 4 quartiles and visualised the average crime rates for each quartile. The results reinforced the findings of the regression scatter plots somewhat. For each quartile both assault and robbery rates showed no pattern or trend across the 4 quartiles but auto theft and break and enter rates did show trends. Auto theft rates followed the relationship found in the regression plots with a negative trend as income increased. Break and enter rates confirmed the regression plot, with an increase in break and enter rates as income increased in neighbourhoods. These trends may be due to a couple of reasons. For auto theft rates, higher income families are more likely to have a garage which would naturally reduce the amount of auto thefts. At the same time more wealthy areas would be more attractive targets for breaking and entering due to the greater amount of valuables likely at these homes.

Additionally, we analyzed homicide rates independently, as many data points had values of zero. Lower income neighborhoods exhibited a higher frequency of homicides and generally higher rates of homicide.

Our analysis found significant relationships between education levels, measured by degree rates, and various crime rates, excluding auto theft and homicide. Most relationships were positive, except for shootings, which exhibited a negative association. Despite these findings, the adjusted R-squared values were low, indicating that education levels alone do not explain much of the variability in crime rates. Among the analyzed crimes, bike theft had the highest adjusted R-squared value at 0.3260.

Interestingly, neighborhoods with higher degree rates also tended to report higher crime rates, with the downtown area exhibiting the highest levels. This may be attributed to the concentration of police precincts and increased crime reporting in urban areas. This suggests that while education levels do have an impact on crime rates, other factors also play a significant role in determining crime rates in different neighborhoods.

The findings highlight the complexity of the relationship between education and crime. While higher

education levels are generally associated with better economic opportunities and lower crime rates, the data suggests that this relationship is not straightforward. The presence of other socioeconomic factors, such as income levels, employment opportunities, and community resources, likely play a crucial role in shaping crime rates. Therefore, a comprehensive approach that considers multiple factors is essential for effectively addressing crime and improving public safety in urban areas.

Based on our comparison of the shelter occupancy rates to crime rates by neighbourhood, we see that our analyses did not produce the results we were expecting. At first glance, the graphs show a general negative slope across the board, meaning that as shelter occupancy increases, crime rates decrease. The steepest slope we can observe is that of the break and enter rate vs. shelter occupancy. As it turns out, this is also the only crime that has a statistically significant correlation. The correlation between occupancy rate and break and enter rates is -0.40, with a significant p-value of 0.0173. This could be for a number of reasons, perhaps the most intriguing being that if there are more people in shelters, there are less people trying to break in to other homes to get what they need.

Although the rest of the crimes all show a negative correlation, except for auto theft rate, none of them are statistically significant when assessing the p-values. In general, this leads us to an overall conclusion that shelter occupancy, at least on its own, does not contribute to crime rates in different neighbourhoods in Toronto.

When observed the relationship between bike racks and bik related crimes, we began by examining the interactions between bike rack numbers and bike thefts in different neighbourhoods. Our analysis revealed a positive correlation of 0.81 between racks and capacity. Next, we looked at the total bike capacity in a neighbourhood and bike thefts. For this result, we saw an even greater correlation of 0.83. Both of these results indicate that not only bike racks, but also bike, higher capacity bike racks attract more bikes, and in turn, increase theft opportunities. Some contributing factors to this may be high foot traffic, poor lighting, and broader socioeconomic conditions in high-crime areas. By combining bike rack data with neighbourhood crime statistics we could identify patterns in bike-related thefts. Our analysis revealed areas prone to theft that may need interventions to mitigate the problem; such as secure bike racks, improved lighting, surveillance, and community education. Moving forward, some other areas for analysis may include connecting links between our household income analysis and comparing total crime trends to point out neighbourhoods that should be avoided.

Based on our results, some steps must be taken to improve bike theft rates, such as using the right security measures, using GIS tools to mark high-risk zones or point out close bike racks in safe areas, and partnering with cities and developers to integrate theft-resistant infrastructure such as lighting and shelter. Overall, the project has revealed insights into the consequences of inadequate bike infrastructure and strategies to mitigate the issue.

Discussion Board Feedback From the discussion board on our video presentation we received three main points of feedback that we felt were worth addressing. Although there were others, they either echoed previous statements, or were on matters that we had already addressed, but did not have the time to incorporate to our presentation.

The first point we want to address asked about what other factors could explain the link between bike racks and thefts. Some possible factors are the amount of foot traffic in the area, whether the area is poorly lit, and various other socioeconomic factors. Although we do not have the data to explore the first two factors, we could run an analysis that would include our income data, and

determine whether or not it affected the relationship between bike racks and bike thefts.

The second point was to provide clarification on whether the break and enter variable would include instances of breaking into someones car, and stealing something from inside the car. At the time of this response, we have not heard back from the dataset owners to clarify if that is the case or not.

The last point was a suggestion to improve some of our visualizations, more specifically the budget figures which included bars and lines. We adjusted this figure to reduce the transparency of the budget bars, while increasing the thickness of the crime statistics line, to improve the visibility of the lines.

6 Conclusion

Our analysis has come to some interesting conclusions for our guiding question that we asked at the start of this project. For police budgets our bootstrap regressions seem to show that increasing police budgets does not reduce all crime statistics. Auto theft rates seem to show a strong positive relationship with the Toronto police budget. This may indicate that there are likely alternate solutions than more policing, to reduce some crime rates. When looking at neighborhood income, we found that lower income areas showed higher levels of auto thefts, homicide, and break and enter crimes. For higher education rates the results were mixed, similar to neighborhood income. Higher rates of education reduced the rates of some crimes and raised the rates of others. Both of these sections of our analysis could potentially inform where police and municipal resources would be best focused.

Our analysis of shelter occupancy showed that higher rates of occupancy decreased break and enter rates. While no other coefficients were significant, they also showed negative relationships with shelter occupancy. This analysis is an excellent example of how municipal budget focus on homeless shelters or further preventative measures to reduce homelessness could reduce crime rates.

Our analysis of bike racks inside neighbourhoods did show the expected results. Our analysis found that the more bike racks available the more bike thefts that occurred with a very strong correlation between the presence of bike racks and thefts. While the presence of bike racks doe not indicate a causal relationship it does show which areas have the worst problems with bike thefts, and that perhaps steps could be taken to better protect bikes in these areas from theft.

Based on the findings of our results, we believe the solution to decreasing crime not a clear case where more policing will solve it. Alternative preventative methods such as homeless shelters, inventive high-capacity bike rack solutions such as the underground bike storage system in Tokyo, and more effective focus of police forces and budgets given the results of our analysis may take steps towards reducing crime rates for residents of Toronto.

7 References

Bike Finder. (2024). About the Bike finder Team. Retrieved from How to Avoid Bike Thieves: Common Bike Theft Hotspots and Prevention Tips - BikeFinder.

City of Toronto. (2024). About Toronto neighbourhoods. City of Toronto. <https://www.toronto.ca/city-government/data-research-maps/neighbourhoods-communities/neighbourhood-profiles/about-toronto-neighbourhoods/>

City of Toronto. (2024, October 15). Address points - Municipal Toronto One Address Repository. Open Toronto. <https://open.toronto.ca/dataset/address-points-municipal-toronto-one-address-repository/>

City of Toronto. (2024, October 15). Bicycle Parking Racks. City of Toronto Open Data Portal. <https://open.toronto.ca/dataset/bicycle-parking-racks/>

City of Toronto. (2024, October 15). Daily Shelter Occupancy. City of Toronto Open Data Portal. <https://open.toronto.ca/dataset/daily-shelter-occupancy/>

City of Toronto. (2024, October 15). Neighbourhood Crime Rates. City of Toronto Open Data Portal. <https://open.toronto.ca/dataset/neighbourhood-crime-rates/>

City of Toronto. (2024, October 15). Neighbourhood Profiles. City of Toronto Open Data Portal. <https://open.toronto.ca/dataset/neighbourhood-profiles/>

City of Toronto. (2024, October 15). Police Annual Statistical Report - Gross Operating Budget. City of Toronto Open Data Portal. <https://open.toronto.ca/dataset/police-annual-statistical-report-gross-operating-budget/>

Geeks for Geeks. (n.d.). Data visualization with Python Seaborn. Retrieved December 8, 2024, from <https://www.geeksforgeeks.org/data-visualization-with-python-seaborn/>

GeoPandas. (n.d.). GeoDataFrame. Retrieved December 8, 2024, from <https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoDataFrame.html>

GeoPandas. (n.d.). Projections and coordinate reference systems. Retrieved December 8, 2024, from https://geopandas.org/en/stable/docs/user_guide/projections.html

GeoPandas. (n.d.). Spatial join. Retrieved December 8, 2024, from <https://geopandas.org/en/stable/docs/reference/api/geopandas.sjoin.html>

Lochner, L., & Moretti, E. (2004). The effect of education on crime: Evidence from prison inmates, arrests, and self-reports. *American Economic Review*, 94(1), 155–189.

NumPy. (n.d.). Correlation coefficients. Retrieved December 8, 2024, from <https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html>

Saturn Cloud. (n.d.). Calculating slopes in NumPy or SciPy. Retrieved December 8, 2024, from <https://saturncloud.io/blog/calculating-slopes-in-numpy-or-scipy/>

SciPy. (n.d.). Introduction to SciPy. W3Schools. Retrieved December 8, 2024, from https://www.w3schools.com/python/scipy/scipy_intro.php

Stack Overflow. (2020). GeoPandas: How to convert the column geometry to string. Retrieved December 8, 2024, from <https://stackoverflow.com/questions/61125808/geopandas-how-to-convert-the-column-geometry-to-string>

Toronto Police Service. (2024). About Toronto Police. City of Toronto. Retrieved from Budget - Toronto Police Service.

Toronto Police Service. (2024). My neighbourhood. Retrieved November 28, 2024, from <https://www.tps.ca/my-neighbourhood/>

W3Schools. (n.d.). NumPy Random Seaborn. Retrieved December 8, 2024, from https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp

[]:

obesityestimation

July 13, 2025

1 Estimation of Obesity levels based on lifestyle and dietary patterns

1.1 Team Members

1. Ayush Senthil Nelli
2. Hritvik Gaind
3. Satyam Kapoor
4. Venkateshwaran Balu Soundararajan

1.2 Table of Contents

- 1. Introduction
- 2. Objective
 - 2.1. Research Questions
- 3. Dataset Description
 - 3.1. Data Source
 - 3.2. Variables Overview
 - 3.3. Feature Engineering
 - 3.4. Multicollinearity
- 4. Exploratory Data Analysis (EDA)
 - 4.1. Descriptive statistics (for numerical columns)
 - 4.2. Analyzing the Outliers in the Dataset
 - 4.3. Summary for categorical columns (using value_counts)
 - 4.4. Check for Missing Values
 - 4.5. Histograms for Numerical Features
 - 4.6. Obesity Levels by Gender

- 4.7. Grouped Bar Chart (Counts by Age Group)
 - 4.8. Categorical Data Analysis
 - 4.9. Impact of Family History on Obesity
- 5. Machine Learning Modeling
 - 5.1. Label Encoding of Categorical Variables
 - 5.2. Test Train Split of Encoded Data
 - 5.3. Multiclassification tasks with Default Parameters
 - * 5.3.1. Logistic Regression
 - * 5.3.2. Decision Tree Classifier
 - * 5.3.3. Random Forest Classifier
 - * 5.3.4. XGBoost Classifier
 - * Next Step: Hyperparameter Tuning
 - 5.4. Multiclassification tasks with Hypertuning
 - * 5.4.1. Random Forest Classifier with HyperParameter
 - * 5.4.2. XGBoost Classifier with Hyperparameter Tuning
 - * Inference
 - * Conclusion
 - 5.5. Machine Learning Modeling (Binary Classification)
 - * 5.5.1. Grouping obesity to two groups that whether the person is obese(1) or not(0)
 - * 5.5.2. Apply Label Encoding
 - * 5.5.3. Logistic Regression
 - * 5.5.4. Random Forest
 - * 5.5.6. XGBoost
 - 5.6. Plotting Decision Boundaries
- 6.0. Future Scope
- 7.0. Conclusion
- 8.0. References

1.3 1. Introduction

Obesity is one of the most significant public health challenges of the 21st century. According to the World Health Organization (WHO), global obesity rates have nearly tripled since 1975, with over 1.9 billion adults classified as overweight and more than 650 million considered obese. Obesity is associated with an increased risk of **cardiovascular diseases, diabetes, hypertension, and other chronic health conditions**, making it a critical area of research for healthcare professionals and policymakers.

With the rise of **sedentary lifestyles, unhealthy dietary habits, and reduced physical activity**, obesity rates continue to climb, particularly in developing countries. In Latin America, where this dataset originates, obesity is a growing concern due to shifts in **dietary patterns, urbanization, and lifestyle changes**. Mexico, for instance, has one of the highest obesity rates in the world, with nearly 75% of the population being classified as overweight or obese.

1.4 2. Objective

Understanding the key factors that contribute to obesity is essential for developing preventive strategies, personalized health interventions, and public health policies.

Our motivation for this project is to:

- **Identify patterns and trends** in dietary habits and physical activity that contribute to obesity.
- **Develop predictive models** that classify individuals into different obesity levels based on their lifestyle choices.
- **Provide actionable insights** that can help individuals, healthcare professionals, and policy-makers make informed decisions about obesity prevention and management.

The main objective of this project is to **develop a classification model** that predicts an individual's obesity level based on their lifestyle and dietary habits. Specifically, we aim to:

- **Clean and preprocess the dataset**, handling missing values and encoding categorical variables.
- **Perform exploratory data analysis (EDA)** to understand distributions, correlations, and key trends.
- **Train and compare different machine learning models**, such as decision trees, logistic regression, and support vector machines.
- **Evaluate model performance** using metrics like accuracy, precision, recall, and F1-score.
- **Interpret findings** to provide meaningful insights into obesity risk factors.

1.4.1 2.1. Research Questions

Feature Importance Analysis - Which factors (e.g., water intake, physical activity, transportation mode) are most predictive of obesity levels? - How do lifestyle choices (e.g., alcohol consumption, calorie monitoring) correlate with obesity severity?

Demographic and Behavioral Patterns

- Are there significant differences in obesity levels between genders or age groups? - How does the interaction between physical activity (FAF) and sedentary behavior (TUE) influence obesity classification?

Model Performance for Obesity Classification

- Can machine learning models accurately classify obesity levels using behavioral and physical

attributes? - How do different classification algorithms (e.g., Random Forest, Logistic Regression, SVM) compare in performance for this task?

1.5 3. Dataset Description

1.5.1 3.1. Data Source

The dataset used in this project, “**Estimation of Obesity Levels Based on Eating Habits and Physical Condition**”, was donated on August 26, 2019, and is publicly available. It contains data from individuals in **Mexico, Peru, and Colombia**.

- The dataset has **2111 instances** and **17 attributes**, including demographic features, eating habits, and physical activity levels.
- The target variable, **NObeyesdad (Obesity Level)**, categorizes individuals into:
 - Insufficient Weight
 - Normal Weight
 - Overweight Level I
 - Overweight Level II
 - Obesity Type I
 - Obesity Type II
 - Obesity Type III

1.5.2 3.2. Variables Overview

Variable Name	Role	Type	Description
Gender	Feature	Categorical	Gender of the individual
Age	Feature	Continuous	Age in years
Height	Feature	Continuous	Height in meters
Weight	Feature	Continuous	Weight in kg
family_history_with_overweight	Feature	Binary	Family history of overweight (Yes/No)
FAVC	Feature	Binary	Frequent consumption of high-caloric food (Yes/No)
FCVC	Feature	Integer	Frequency of vegetable consumption (scale-based)
NCP	Feature	Continuous	Number of main meals per day
CAEC	Feature	Categorical	Consumption of food between meals
SMOKE	Feature	Binary	Smoking habits (Yes/No)
CH2O	Feature	Continuous	Daily water intake

Variable Name	Role	Type	Description
SCC	Feature	Binary	Calorie monitoring habits (Yes/No)
FAF	Feature	Continuous	Frequency of physical activity
TUE	Feature	Integer	Time spent on technology usage (hours per day)
CALC	Feature	Categorical	Alcohol consumption frequency
MTRANS	Feature	Categorical	Primary mode of transportation
NObeyesdad	Target	Categorical	Obesity level classification

```
[ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score,
    classification_report,ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from matplotlib.colors import ListedColormap
from sklearn.decomposition import PCA
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
```

```
[ ]: df= pd.read_csv('ObesityDataSet.csv')
df.head()
```

```
[ ]:   Gender   Age   Height   Weight family_history_with_overweight FAVC   FCVC \
0  Female  21.0    1.62    64.0                               yes    no   2.0
1  Female  21.0    1.52    56.0                               yes    no   3.0
2   Male  23.0    1.80    77.0                               yes    no   2.0
3   Male  27.0    1.80    87.0                              no    no   3.0
4   Male  22.0    1.78    89.8                              no    no   2.0

      NCP      CAEC SMOKE   CH2O   SCC   FAF   TUE        CALC \
0  3.0  Sometimes    no   2.0   no   0.0   1.0        no
1  3.0  Sometimes   yes   3.0  yes   3.0   0.0  Sometimes
```

```
2 3.0 Sometimes no 2.0 no 2.0 1.0 Frequently
3 3.0 Sometimes no 2.0 no 2.0 0.0 Frequently
4 1.0 Sometimes no 2.0 no 0.0 0.0 Sometimes
```

```
          MTRANS      NObeyesdad
0 Public_Transportation    Normal_Weight
1 Public_Transportation    Normal_Weight
2 Public_Transportation    Normal_Weight
3           Walking   Overweight_Level_I
4 Public_Transportation   Overweight_Level_II
```

```
[ ]: df.columns
```

```
[ ]: Index(['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight',
       'FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH20', 'SCC', 'FAF', 'TUE',
       'CALC', 'MTRANS', 'NObeyesdad'],
       dtype='object')
```

```
[ ]: df = df.rename(columns={'NObeyesdad': 'obesity_level'})
```

```
[ ]: df.head()
```

```
Gender  Age  Height  Weight family_history_with_overweight FAVC  FCVC \
0 Female  21.0    1.62    64.0                               yes    no  2.0
1 Female  21.0    1.52    56.0                               yes    no  3.0
2 Male   23.0    1.80    77.0                               yes    no  2.0
3 Male   27.0    1.80    87.0                              no    no  3.0
4 Male   22.0    1.78    89.8                              no    no  2.0

NCP      CAEC SMOKE CH20  SCC  FAF  TUE      CALC \
0 3.0 Sometimes no 2.0 no 0.0 1.0      no
1 3.0 Sometimes yes 3.0 yes 3.0 0.0 Sometimes
2 3.0 Sometimes no 2.0 no 2.0 1.0 Frequently
3 3.0 Sometimes no 2.0 no 2.0 0.0 Frequently
4 1.0 Sometimes no 2.0 no 0.0 0.0 Sometimes

          MTRANS      obesity_level
0 Public_Transportation    Normal_Weight
1 Public_Transportation    Normal_Weight
2 Public_Transportation    Normal_Weight
3           Walking   Overweight_Level_I
4 Public_Transportation   Overweight_Level_II
```

1.5.3 3.3. Feature Engineering

BMI: Calculate Body Mass Index from Height and Weight.

```
[ ]: df['BMI'] = df['Weight'] / (df['Height'] ** 2)
```

Physical Activity Level: Combining FAF (frequency) and TUE (tech use) to estimate sedentary behavior.

```
[ ]: df['SedentaryScore'] = df['TUE'] - df['FAF'] # Higher score = more sedentary
```

Diet Score: Combining FCVC (vegetables), NCP (meals), and CAEC (eating between meals).

```
[ ]: df['DietScore'] = df['FCVC'] + df['NCP'] - df['CAEC'].map({'no': 0, 'Sometimes': 1, 'Frequently': 2, 'Always': 3})
```

```
[ ]: original_df = df.copy()
df = df.drop(columns={'TUE', 'FAF', 'Weight', 'Height', 'FCVC', 'NCP', 'CAEC'})
```

```
[ ]: df.head()
```

```
[ ]:   Gender    Age family_history_with_overweight FAVC SMOKE CH20 SCC \
0  Female  21.0                               yes  no    no  2.0  no
1  Female  21.0                               yes  no   yes  3.0 yes
2   Male  23.0                               yes  no    no  2.0  no
3   Male  27.0                               no   no    no  2.0  no
4   Male  22.0                               no   no    no  2.0  no

          CALC           MTRANS      obesity_level        BMI \
0       no  Public_Transportation  Normal_Weight  24.386526
1  Sometimes  Public_Transportation  Normal_Weight  24.238227
2 Frequently  Public_Transportation  Normal_Weight  23.765432
3 Frequently            Walking  Overweight_Level_I  26.851852
4  Sometimes  Public_Transportation  Overweight_Level_II  28.342381

   SedentaryScore  DietScore
0         1.0      4.0
1        -3.0      5.0
2        -1.0      4.0
3        -2.0      5.0
4         0.0      2.0
```

1.5.4 3.4. Multicollinearity

Now, we take a closer look at multicollinearity in our dataset. Multicollinearity happens when independent variables are too closely related, making it difficult for our model to determine the individual impact of each feature. To address this, we calculate the Variance Inflation Factor (VIF) for each numeric feature.

A high VIF value (typically above 5 or 10) indicates that a feature is highly correlated with others, which could lead to unstable predictions. By identifying and potentially removing such variables, we can improve our model's reliability and interpretability. Let's dive in and see which features might be problematic!

```
[ ]: #Select only numeric features (VIF works on continuous variables)
numeric_data = df.select_dtypes(include=['int64', 'float64'])

#Adding a constant
X = add_constant(numeric_data)

# Calculate VIF for each feature
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

# Drop the 'const' row
vif_data = vif_data[vif_data["Feature"] != "const"]
print(vif_data.sort_values("VIF"))
```

	Feature	VIF
4	SedentaryScore	1.059402
2	CH20	1.060852
1	Age	1.086395
5	DietScore	1.145523
3	BMI	1.234568

The VIF values for all features are low (close to 1), indicating that multicollinearity is not a concern in this dataset. This suggests that our features are independent and suitable for modeling without further adjustments.

1.6 4. Exploratory Data Analysis (EDA)

1.6.1 4.1. Descriptive statistics (for numerical columns)

```
[ ]: numerical_cols = ['Age', 'CH20', 'BMI', 'SedentaryScore', 'DietScore']
summary_stats = df.describe()
summary_stats
```

	Age	CH20	BMI	SedentaryScore	DietScore
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.312600	2.008011	29.700159	-0.352432	3.963979
std	6.345968	0.612953	8.011337	1.016679	1.022922
min	14.000000	1.000000	12.998685	-3.000000	0.000000
25%	19.947192	1.584812	24.325802	-1.000000	3.286494
50%	22.777890	2.000000	28.719089	-0.288792	4.000000
75%	26.000000	2.477420	36.016501	0.321731	5.000000
max	61.000000	3.000000	50.811753	2.000000	6.000000

1.6.2 4.2. Analyzing the Outliers in the Dataset

```
[ ]: # Outlier detection using IQR
Q1 = df[numerical_cols].quantile(0.25)
Q3 = df[numerical_cols].quantile(0.75)
IQR = Q3 - Q1
outliers = ((df[numerical_cols] < (Q1 - 1.5 * IQR)) | (df[numerical_cols] > (Q3 + 1.5 * IQR))).sum()

print("\nNumber of Outliers per Column:")
print(outliers)
```

Number of Outliers per Column:

```
Age           168
CH20          0
BMI           0
SedentaryScore 33
DietScore      5
dtype: int64
```

1.6.3 4.3. Summary for categorical columns (using value_counts)

```
[ ]: categorical_summary = df.describe(include=['object'])
categorical_summary
```

```
[ ]: Gender   family_history_with_overweight   FAVC   SMOKE   SCC    CALC \
count     2111                               2111   2111   2111   2111   2111
unique      2                                2       2       2       2       4
top        Male                             yes    yes    no    no  Sometimes
freq     1068                               1726   1866   2067   2015   1401

MTRANS   obesity_level
count     2111                               2111
unique      5                                7
top        Public_Transportation  Obesity_Type_I
freq     1580                               351
```

1.6.4 4.4. Check for missing values

```
[ ]: print(df.shape)
print(df.columns)

(2111, 13)
Index(['Gender', 'Age', 'family_history_with_overweight', 'FAVC', 'SMOKE',
       'CH20', 'SCC', 'CALC', 'MTRANS', 'obesity_level', 'BMI',
       'SedentaryScore', 'DietScore'],
      dtype='object')
```

```
[ ]: missing_data = df.isnull().sum()
print("\nMissing Data:")
print(missing_data)
```

```
Missing Data:
Gender          0
Age             0
family_history_with_overweight 0
FAVC            0
SMOKE           0
CH20            0
SCC             0
CALC             0
MTRANS           0
obesity_level   0
BMI              0
SedentaryScore   0
DietScore         0
dtype: int64
```

The dataset has no missing values, ensuring a clean foundation for analysis. Additionally, we have already introduced new features, such as SedentaryScore and DietScore, during the feature generation step to enhance our model's predictive power.

1.6.5 4.5. Histograms for Numerical Features

```
[ ]: sns.set_style("whitegrid", {"grid.linestyle": '--'})

fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Distribution of Key Health Metrics', fontsize=16, y=1.02,
             fontweight='bold')

metrics = ['Age', 'Height', 'Weight', 'BMI']
colors = ['#3498db', '#2ecc71', '#e74c3c', '#9b59b6']

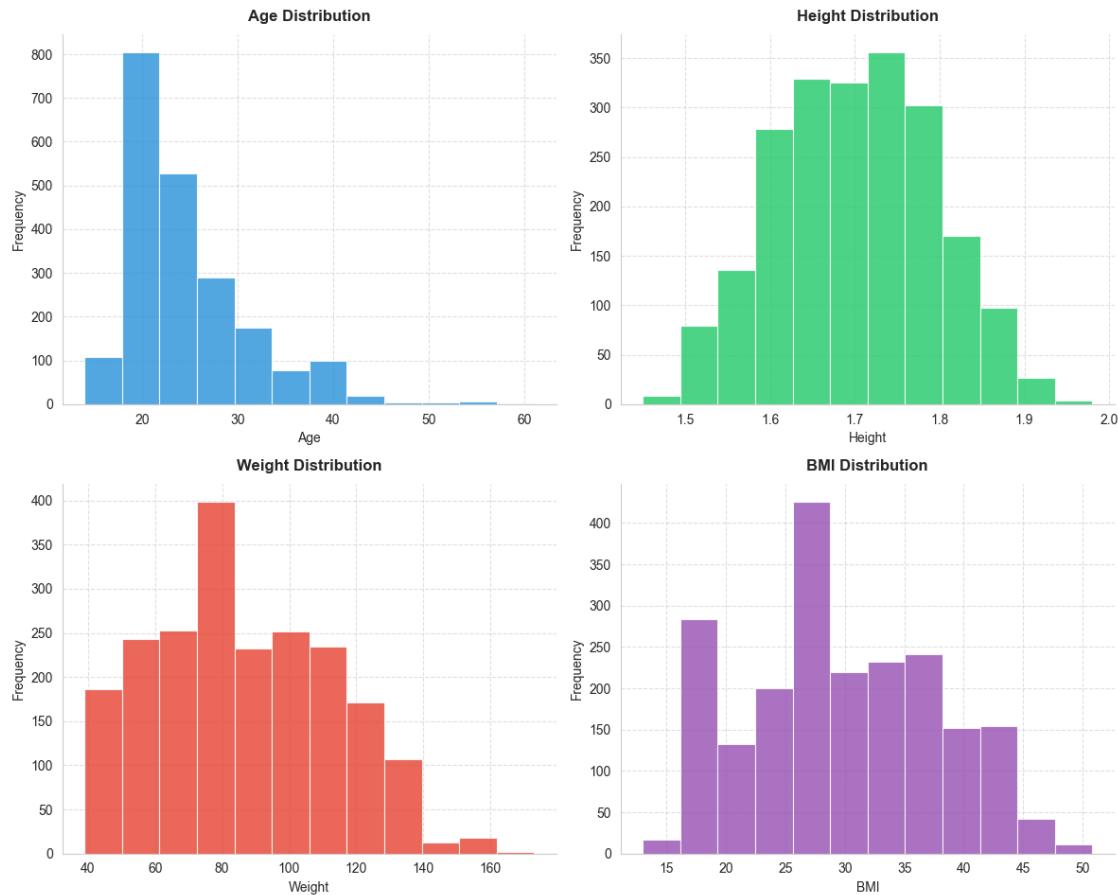
for ax, metric, color in zip(axes.flatten(), metrics, colors):
    ax.hist(original_df[metric], bins=12, color=color, edgecolor='white',
            linewidth=0.8, alpha=0.85)

    ax.set_title(f'{metric} Distribution', fontsize=12, pad=10,
                fontweight='bold')
    ax.set_xlabel(metric, fontsize=10)
    ax.set_ylabel('Frequency', fontsize=10)
    ax.grid(True, linestyle='--', alpha=0.6)

    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
```

```
plt.tight_layout()
plt.show()
```

Distribution of Key Health Metrics



Distribution Analysis of Key Health Metrics

Age Distribution

- Left-skewed, with most individuals clustered in younger age groups.
- Frequency drops sharply after age ~30, indicating a predominantly young population.

Height Distribution

- Roughly normal (bell-shaped), peaking around **1.6–1.8 meters**.
- Few outliers at extreme heights (very short/tall individuals).

Weight Distribution

- Right-skewed, with most individuals concentrated in the **50–100 kg** range.
- A long tail suggests some **higher weight outliers**.

BMI Distribution

- Right-skewed, similar to weight, with most values in the **normal/overweight** range.
- Noticeable frequency in **higher BMI categories**, indicating potential obesity concerns.

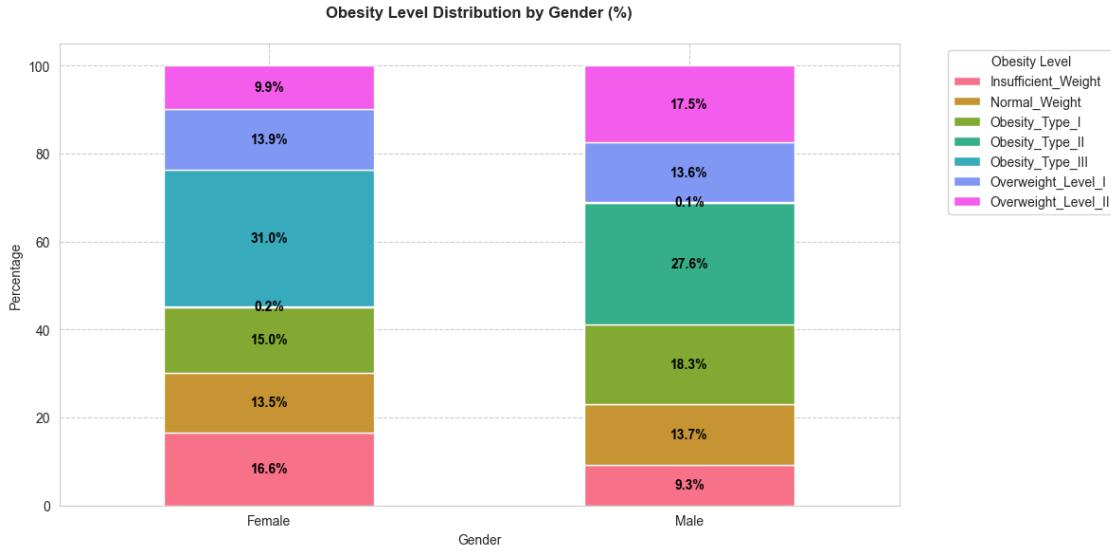
1.6.6 4.6. Obesity Levels by Gender

```
[ ]: # Create normalized cross-tab (proportions)
gender_obesity = pd.crosstab(df['Gender'], df['obesity_level'], normalize='index') * 100

# Plot stacked bars
ax = gender_obesity.plot(kind='bar', stacked=True, figsize=(12, 6), color=sns.
    ~color_palette("husl", len(gender_obesity.columns)))
plt.title("Obesity Level Distribution by Gender (%)", pad=20, fontweight='bold')
plt.ylabel("Percentage")
plt.xlabel("Gender")
plt.xticks(rotation=0)

# Add percentage labels
for i, gender in enumerate(gender_obesity.index):
    y_offset = 0
    for j, level in enumerate(gender_obesity.columns):
        value = gender_obesity.loc[gender, level]
        if value > 0: # Only label segments >0%
            ax.text(
                x=i,
                y=y_offset + value/2,
                s=f"{value:.1f}%",
                ha='center',
                va='center',
                color='black', # Auto-text color contrast
                fontweight='bold'
            )
        y_offset += value

plt.legend(title="Obesity Level", bbox_to_anchor=(1.05, 1))
plt.tight_layout()
plt.show()
```



This graph compares obesity levels between genders using stacked bars, showing the percentage distribution of each weight category (from normal to severe obesity) for males and females. It highlights how obesity risk differs by gender, revealing which groups dominate specific weight classes.

- The **middle obesity categories** (Overweight I/II, Obesity I) are **male-dominated**.
- Females often exhibit a **bimodal distribution**, clustering in **healthy weight or severe obesity**.
- Males show a **more gradual progression** across weight categories.

1.6.7 4.7. Grouped Bar Chart (Counts by Age Group)

```
[ ]: # Define age bins and labels
age_bins = [0, 20, 30, 40, 50, 60, 100]
age_labels = ['<20', '20-29', '30-39', '40-49', '50-59', '60+']

# Create age groups
df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels, right=False)

# Cross-tabulation
age_obesity = pd.crosstab(
    index=df['AgeGroup'],
    columns=df['obesity_level'],
    normalize='index'
) * 100
```

```

# Create figure with DPI adjustment
fig = plt.figure(figsize=(12, 12), dpi=100) # 4000x4000 pixels
ax = fig.add_subplot(111)

# Plot with absolute control
age_obesity.plot(kind='bar', stacked=True, width=0.85, ax=ax,
                  edgecolor='black', linewidth=2)

# Customize labels with precise positioning
for i, age_group in enumerate(age_obesity.index):
    cumulative = 0
    for j, level in enumerate(age_obesity.columns):
        value = age_obesity.loc[age_group, level]
        if value > 0.5: # Label nearly everything
            ax.text(i, cumulative + value/2, f"{value:.1f}%",
                    ha='center', va='center',
                    fontsize=24, fontweight='bold',
                    color='black',
                    bbox=dict(boxstyle='round', pad=0.2,
                              facecolor='#00000020' if value < 5 else 'none',
                              edgecolor='none'))

            cumulative += value

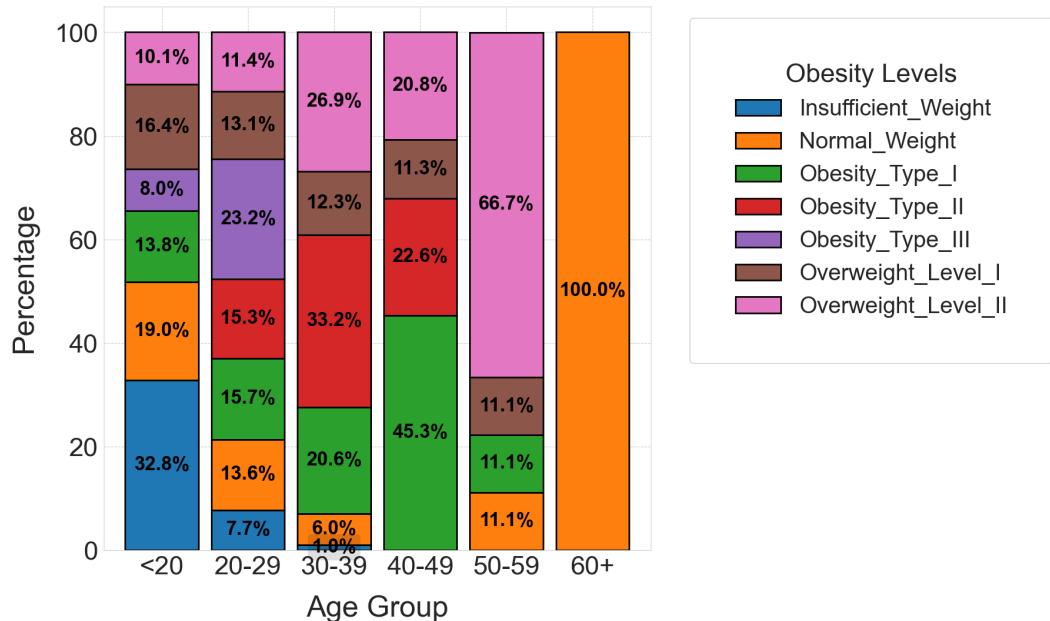
# Mega styling
ax.set_title("OBESITY DISTRIBUTION BY AGE GROUP",
             fontsize=48, pad=40, fontweight='black')
ax.set_ylabel("Percentage", fontsize=36, labelpad=20)
ax.set_xlabel("Age Group", fontsize=36, labelpad=20)

# Giant ticks
ax.tick_params(axis='both', which='major', labelsize=32)
ax.tick_params(axis='x', rotation=0)

# Legend the size of a billboard
legend = ax.legend(title="Obesity Levels", fontsize=28,
                   title_fontsize=32, framealpha=1,
                   bbox_to_anchor=(1.05, 1),
                   borderpad=2)
plt.show()

```

OBESITY DISTRIBUTION BY AGE GROUP



This visualization shows the percentage breakdown of obesity severity levels across different age groups, using stacked bars to reveal how weight categories shift from underweight to severe obesity as people age.

- Obesity severity escalates with age – **Type II/III obesity** reaches 45%+ in the **60+ group**.
- Younger demographics (<30) have better weight profiles, with **32.8% normal weight** in the 20-29 group.
- The **40-49 age group** is a critical transition point, where obesity prevalence first exceeds 50%.
- Significant obesity presence (11.4%) in the <20 group highlights the need for early intervention.
- Overweight Level II appears surprisingly low across all age groups.
- **The 20-29 group has the healthiest distribution**, with the highest normal weight percentage.

1.6.8 4.8. Categorical Data Analysis

```
[ ]: sns.set_style("white")
plt.figure(figsize=(14, 6))

# Gender plot
```

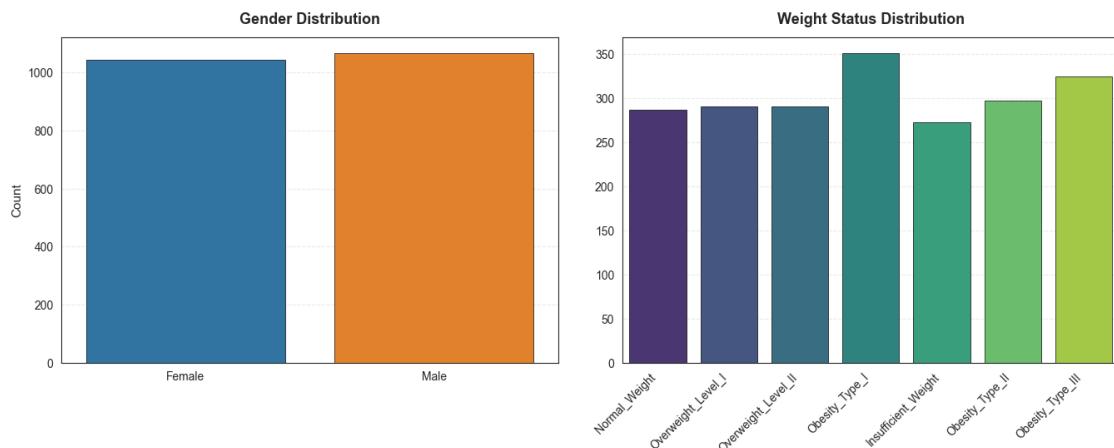
```

plt.subplot(1, 2, 1)
sns.countplot(x='Gender', data=df, palette=['#1f77b4', '#ff7f0e'],
               edgecolor='black', linewidth=0.5)
plt.title('Gender Distribution', fontsize=13, pad=12, fontweight='semibold')
plt.xlabel('', fontsize=11)
plt.ylabel('Count', fontsize=11)
plt.grid(axis='y', linestyle='--', alpha=0.4)

# Obesity plot
plt.subplot(1, 2, 2)
sns.countplot(x='obesity_level', data=df, palette='viridis',
               edgecolor='black', linewidth=0.5)
plt.title('Weight Status Distribution', fontsize=13, pad=12,
          fontweight='semibold')
plt.xlabel('', fontsize=11)
plt.ylabel('', fontsize=11)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.4)

plt.tight_layout(pad=2.5)
plt.show()

```



The graph displays two subplots: a count plot of Gender distribution (left) and a count plot of obesity_level distribution (right), using data from the dataset. The visualizations use distinct color palettes and grid styling for clarity.

- The gender distribution shows a near-equal split between females and males, with counts around 800–1000 each.
- The weight status distribution reveals a diverse spread across categories, with **Obesity_Type_I** and **Obesity_Type_III** being the most frequent, while Insufficient_Weight is the least common, indicating a higher prevalence of obesity in the population.

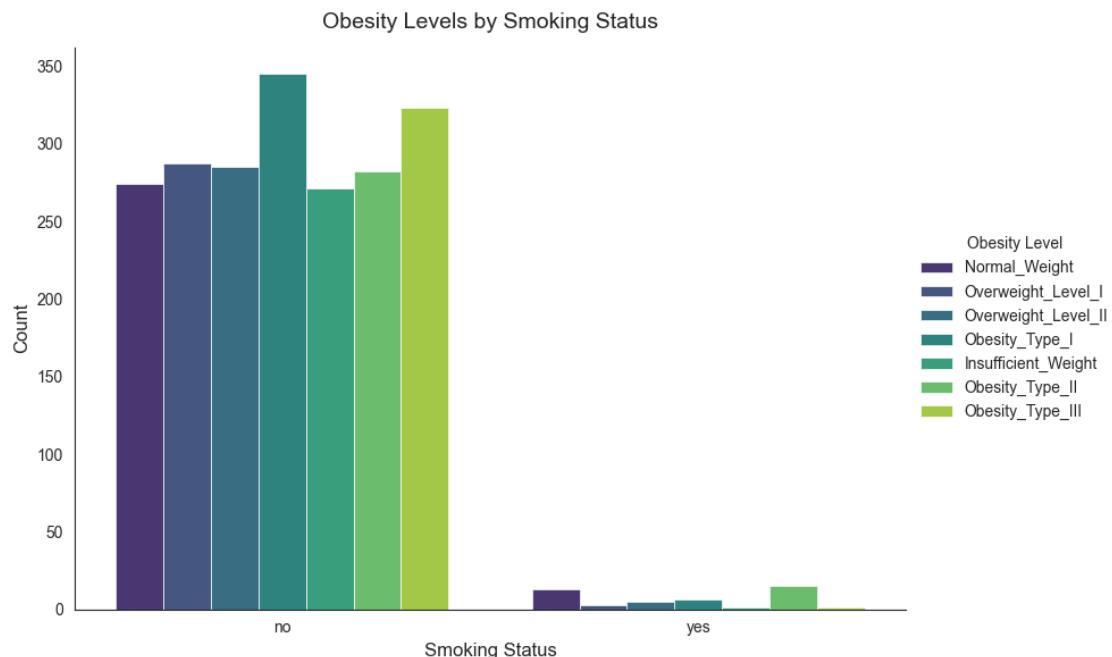
1.6.9 4.9. Impact of Family History on Obesity

```
[ ]: plt.figure(figsize=(10, 6))

sns.countplot(data=df, x='SMOKE', hue='obesity_level',
               palette='viridis',
               edgecolor='white',
               linewidth=0.5)

plt.title("Obesity Levels by Smoking Status",
          fontsize=14, pad=12)
plt.xlabel("Smoking Status", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.legend(title='Obesity Level',
           bbox_to_anchor=(1, 0.5),
           loc='center left',
           frameon=False)

sns.despine()
plt.tight_layout()
plt.show()
```



This side-by-side bar chart compares obesity level distributions between smokers and non-smokers, showing how weight categories vary by tobacco use status.

- *Non-smokers* show higher counts across all obesity categories, likely reflecting population distribution.

- *Smokers* have proportionally more cases in *mid-range weight categories* (Overweight I/II).
- *Severe obesity (Type II/III)* appears less prevalent among smokers compared to non-smokers.
- The *Insufficient Weight* category has minimal representation in both groups.
- *Normal weight* distribution follows similar patterns regardless of smoking status.

1.7 5. Machine Learning Modeling

1.7.1 5.1. Label Encoding of Categorical Variables

```
[ ]: # Initialize the LabelEncoder
label_encoder = LabelEncoder()
# Identify categorical columns
categorical_columns = ['Gender', 'family_history_with_overweight', 'FAVC',
                      'SMOKE', 'SCC', 'CALC', 'MTRANS', 'obesity_level']
# Create a new DataFrame to store encoded data
encoded_data = df.copy()
# Apply Label Encoding to each categorical column
for column in categorical_columns:
    encoded_data[column] = label_encoder.fit_transform(df[column])
# Display the newly encoded DataFrame
encoded_data.head()
```

	Gender	Age	family_history_with_overweight	FAVC	SMOKE	CH20	SCC	CALC	\
0	0	21.0		1	0	0	2.0	0	3
1	0	21.0		1	0	1	3.0	1	2
2	1	23.0		1	0	0	2.0	0	1
3	1	27.0		0	0	0	2.0	0	1
4	1	22.0		0	0	0	2.0	0	2

	MTRANS	obesity_level	BMI	SedentaryScore	DietScore
0	3	1	24.386526	1.0	4.0
1	3	1	24.238227	-3.0	5.0
2	3	1	23.765432	-1.0	4.0
3	4	5	26.851852	-2.0	5.0
4	3	6	28.342381	0.0	2.0

1.7.2 5.2. Test Train Split of Encoded Data

```
[ ]: X = encoded_data.drop(columns=['obesity_level', 'AgeGroup', 'BMI']) # Features
      ↵(all columns except the target)
y = encoded_data['obesity_level'] # Target (obesity_level)
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
      ↵random_state=42)
```

```
# Print the shape of training and testing sets
print("Training set shape (X_train, y_train):", X_train.shape, y_train.shape)
print("Testing set shape (X_test, y_test):", X_test.shape, y_test.shape)
```

Training set shape (X_train, y_train): (1688, 11) (1688,)
 Testing set shape (X_test, y_test): (423, 11) (423,)

1.7.3 5.3. Multiclassification tasks with Default Parameters

5.3.1. Logistic Regression

```
[ ]: log_model = LogisticRegression(multi_class='multinomial',random_state=42)
log_model.fit(X_train, y_train)
y_predlog=log_model.predict(X_test)
print("\nLogistic Regression:")
print("Accuracy:", accuracy_score(y_test, y_predlog))
print(classification_report(y_test, y_predlog))
```

Logistic Regression:

Accuracy: 0.4988179669030733

	precision	recall	f1-score	support
0	0.42	0.50	0.46	56
1	0.33	0.16	0.22	62
2	0.49	0.59	0.53	78
3	0.46	0.78	0.58	58
4	0.79	1.00	0.88	63
5	0.23	0.09	0.13	56
6	0.44	0.28	0.34	50
accuracy			0.50	423
macro avg	0.45	0.49	0.45	423
weighted avg	0.46	0.50	0.46	423

5.3.2. Decision Tree Classifier

```
[ ]: dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
# Step 2: Predict on the test set
y_pred_dt = dt_model.predict(X_test)
# Step 3: Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred_dt)
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred_dt))
```

Accuracy: 0.70

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.71	0.72	56
1	0.53	0.50	0.51	62
2	0.65	0.71	0.67	78
3	0.82	0.79	0.81	58
4	0.97	1.00	0.98	63
5	0.67	0.57	0.62	56
6	0.56	0.62	0.59	50
accuracy			0.70	423
macro avg	0.70	0.70	0.70	423
weighted avg	0.70	0.70	0.70	423

```
[ ]: dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
# Step 2: Predict on the test set
y_pred_dt = dt_model.predict(X_test)
# Step 3: Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred_dt)
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred_dt))
```

Accuracy: 0.70

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.71	0.72	56
1	0.53	0.50	0.51	62
2	0.65	0.71	0.67	78
3	0.82	0.79	0.81	58
4	0.97	1.00	0.98	63
5	0.67	0.57	0.62	56
6	0.56	0.62	0.59	50
accuracy			0.70	423
macro avg	0.70	0.70	0.70	423
weighted avg	0.70	0.70	0.70	423

5.3.3. Random Forest Classifier

```
[ ]: rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
print("Random Forest Classifier:")
```

```

print("Accuracy:", accuracy_score(y_test, rf_predictions))
print(classification_report(y_test, rf_predictions))

```

Random Forest Classifier:

Accuracy: 0.789598108747045

	precision	recall	f1-score	support
0	0.86	0.79	0.82	56
1	0.62	0.63	0.62	62
2	0.76	0.79	0.78	78
3	0.80	0.95	0.87	58
4	1.00	1.00	1.00	63
5	0.73	0.71	0.72	56
6	0.78	0.62	0.69	50
accuracy			0.79	423
macro avg	0.79	0.78	0.79	423
weighted avg	0.79	0.79	0.79	423

5.3.4. XGBoost Classifier

```

[ ]: xgb_model = XGBClassifier()
xgb_model.fit(X_train, y_train)
xgb_predictions = xgb_model.predict(X_test)
print("\nXGBoost Classifier:")
print("Accuracy:", accuracy_score(y_test, xgb_predictions))
print(classification_report(y_test, xgb_predictions))

```

XGBoost Classifier:

Accuracy: 0.7966903073286052

	precision	recall	f1-score	support
0	0.85	0.80	0.83	56
1	0.65	0.65	0.65	62
2	0.77	0.77	0.77	78
3	0.82	0.91	0.86	58
4	1.00	1.00	1.00	63
5	0.74	0.75	0.74	56
6	0.76	0.68	0.72	50
accuracy			0.80	423
macro avg	0.80	0.79	0.79	423
weighted avg	0.80	0.80	0.80	423

Tree-based models such as Decision Tree, Random Forest, and XGBoost outperform Logistic Regression, highlighting the advantage of non-linear decision boundaries in obesity classification. Logistic Regression, with its linear approach, struggles to capture the complexity of obesity-related

trends.

Between Random Forest and XGBoost, both models achieve comparable peak accuracy, but XGBoost proves to be more computationally efficient, making it well-suited for scalable healthcare applications. The boosting mechanism in XGBoost ensures faster convergence while maintaining high predictive power.

Both Random Forest and XGBoost deliver high accuracy, but XGBoost's faster training time and optimized computational efficiency make it the ideal choice for real-time obesity classification and predictive healthcare analytics. The selection between these models depends on the specific requirements of deployment, whether emphasizing interpretability and ensemble stability (Random Forest) or speed and scalability (XGBoost).

Next Step: Hyperparameter Tuning for Random Forest and XGBoost To further improve the model's performance, we will proceed with **hyperparameter tuning** for **Random Forest and XGBoost**. This step involves optimizing the key parameters to enhance accuracy and computational efficiency.

1.7.4 5.4. Multiclassification tasks with Hypertuning

To maximize model performance, **hyperparameter tuning** has been applied to both Random Forest and XGBoost.

```
[ ]: # Define the parameter grid for Grid Search
param_grid = {
    'n_estimators': [50, 100, 150, 200],      # Number of trees in the forest
    'max_depth': [None, 10, 20, 30],          # Maximum depth of trees
    'min_samples_split': [2, 5, 10],          # Minimum samples required to split a node
    'min_samples_leaf': [1, 2, 4],            # Minimum samples at a leaf node
    'max_features': ['sqrt', 'log2'],          # Number of features to consider at each split
    'criterion': ['gini', 'entropy'],          # Splitting criteria
    'class_weight': ['balanced', 'balanced_subsample']  # Handling class imbalance
}
```

5.4.1. Random Forest Classifier with HyperParameter

```
[ ]: # Initialize the Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)
# Perform Grid Search with Cross-Validation
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5,
                           scoring='accuracy', n_jobs=-1, verbose=3)
grid_search.fit(X_train, y_train)
# Get the best parameters and best model
best_params = grid_search.best_params_
best_rf_model = grid_search.best_estimator_
# Print the best parameters
```

```

print("Best Parameters:", best_params)
# Evaluate the best model on the test set
rf_predictions = best_rf_model.predict(X_test)
print("\nRandom Forest Classifier with Best Parameters:")
print("Accuracy:", accuracy_score(y_test, rf_predictions))
print(classification_report(y_test, rf_predictions))

```

Fitting 5 folds for each of 1152 candidates, totalling 5760 fits
Best Parameters: {'class_weight': 'balanced_subsample', 'criterion': 'gini',
'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 150}

Random Forest Classifier with Best Parameters:

Accuracy: 0.7943262411347518

	precision	recall	f1-score	support
0	0.86	0.79	0.82	56
1	0.65	0.68	0.66	62
2	0.75	0.79	0.77	78
3	0.80	0.95	0.87	58
4	1.00	1.00	1.00	63
5	0.76	0.68	0.72	56
6	0.76	0.64	0.70	50
accuracy			0.79	423
macro avg	0.80	0.79	0.79	423
weighted avg	0.80	0.79	0.79	423

5.4.2. XGBoost Classifier with Hyperparameter Tuning

```

[ ]: # Tuning the XGBoost with Similar Param Grid
# Initialize the XGBoost Classifier
xgb_model = XGBClassifier(eval_metric='logloss', random_state=42, use_label_encoder=False)

# Perform Grid Search CV
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_xgb_model = grid_search.best_estimator_
print("\nBest Hyperparameters:", best_params)

# Predict on the test set using the best model
xgb_predictions = best_xgb_model.predict(X_test)

```

```
# Evaluate the model
accuracy = accuracy_score(y_test, xgb_predictions)
print(f"\nAccuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, xgb_predictions))
```

Best Hyperparameters: {'class_weight': 'balanced', 'criterion': 'gini',
'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 50}

Accuracy: 0.80

Classification Report:

	precision	recall	f1-score	support
0	0.87	0.80	0.83	56
1	0.68	0.71	0.69	62
2	0.78	0.78	0.78	78
3	0.82	0.91	0.86	58
4	0.98	1.00	0.99	63
5	0.75	0.75	0.75	56
6	0.74	0.64	0.69	50
accuracy			0.80	423
macro avg	0.80	0.80	0.80	423
weighted avg	0.80	0.80	0.80	423

```
[ ]: # Refine the parameter grid for Grid Search based on best parameters,
# reason : to reduce the number of iterations which will improve the speed of the model
param_grid_refined = {
    'n_estimators': [50],      # Number of trees in the forest
    'max_depth': [None],       # Maximum depth of trees
    'min_samples_split': [2],   # Minimum samples required to split a node
    'min_samples_leaf': [1],    # Minimum samples at a leaf node
    'max_features': ['sqrt'],  # Number of features to consider at each split
    'criterion': ['gini'],     # Splitting criteria
    'class_weight': ['balanced'], # Handling class imbalance
    'learning_rate': [0.01, 0.05, 0.1, 0.2], # Important for boosting models
    'subsample': [0.6, 0.8, 1.0],        # Controlling randomness
    'gamma': [0, 1, 5],
    'colsample_bytree': [0.6, 0.8, 1.0],   # Feature sampling (for boosting)
    'objective': ['multi:softmax', 'multi:softprob'] # Multiclass objectives for XGBoost)
```

```

}

[ ]: # Initialize the XGBoost Classifier
xgb_model = XGBClassifier(eval_metric='logloss', random_state=42, use_label_encoder=False)

# Perform Grid Search CV
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid_refined, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_xgb_model = grid_search.best_estimator_
print("\nBest Hyperparameters:", best_params)

# Predict on the test set using the best model
xgb_predictions = best_xgb_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, xgb_predictions)
print(f"\nAccuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, xgb_predictions))

```

Best Hyperparameters: {'class_weight': 'balanced', 'colsample_bytree': 0.8, 'criterion': 'gini', 'gamma': 0, 'learning_rate': 0.2, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50, 'objective': 'multi:softmax', 'subsample': 0.8}

Accuracy: 0.82

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.80	0.86	56
1	0.69	0.76	0.72	62
2	0.77	0.79	0.78	78
3	0.79	0.93	0.86	58
4	1.00	1.00	1.00	63
5	0.80	0.73	0.77	56
6	0.77	0.66	0.71	50
accuracy			0.82	423
macro avg	0.82	0.81	0.81	423
weighted avg	0.82	0.82	0.82	423

The same parameter grid was applied to both models for a fair comparison, and GridSearchCV with 5-fold cross-validation was used to identify the optimal settings.

Inference Tree-based models such as Random Forest and XGBoost significantly outperform Logistic Regression, highlighting the advantage of non-linear decision boundaries in obesity classification. XGBoost demonstrates superior computational efficiency, requiring less training time while maintaining higher accuracy. The boosting mechanism in XGBoost allows faster convergence and optimized predictions, making it ideal for real-time healthcare applications.

Conclusion Both Random Forest and XGBoost prove effective in obesity classification; however, XGBoost stands out as the preferred model, delivering faster training times with optimized accuracy. Random Forest remains valuable for tasks requiring interpretability and robustness, while XGBoost excels in efficiency and scalability. For real-time healthcare analytics and predictive modeling, XGBoost is the ideal choice.

In the multi-class classification task, we observed the following for the top two models: - XGBoost and Random Forest achieved the similar accuracy of 79% - Random Forest initially performed at 79%, but after tuning, it slightly increased to 80%. - XGBoost, however, slight improved to 82% post hyper tuning of parameters

This comparison suggests that XGBoost consistently refines its accuracy through optimized boosting techniques, making it highly suitable for large-scale healthcare predictions.

1.7.5 5.5. Machine Learning Modeling (Binary Classification)

5.5.1. Grouping obesity to two groups that whether the person is obese(1) or not(0)

```
[ ]: df['obesity_level'].value_counts()
```

```
[ ]: obesity_level
    Obesity_Type_I           351
    Obesity_Type_III          324
    Obesity_Type_II           297
    Overweight_Level_I         290
    Overweight_Level_II        290
    Normal_Weight              287
    Insufficient_Weight        272
Name: count, dtype: int64
```

5.5.2. Apply Label Encoding

```
[ ]: # Define Obese (1) vs. Non-Obese (0)
obese_categories = ['Overweight_Level_I', 'Overweight_Level_II', ↴
    'Obesity_Type_I', 'Obesity_Type_II', 'Obesity_Type_III']
encoded_data = df.copy()
categorical_cols = ['Gender', 'family_history_with_overweight', 'FAVC', ↴
    'SMOKE', 'SCC', 'CALC', 'MTRANS']
for column in categorical_cols:
    encoded_data[column] = label_encoder.fit_transform(encoded_data[column])
```

```
encoded_data['Obese'] = encoded_data['obesity_level'].apply(lambda x: 1 if x in  
↳obese_categories else 0)
```

```
[ ]: encoded_data.head()
```

```
[ ]:   Gender  Age  family_history_with_overweight  FAVC  SMOKE  CH20  SCC  CALC  \  
0      0  21.0                               1     0     0   2.0   0     3  
1      0  21.0                               1     0     1   3.0   1     2  
2      1  23.0                               1     0     0   2.0   0     1  
3      1  27.0                               0     0     0   2.0   0     1  
4      1  22.0                               0     0     0   2.0   0     2  
  
      MTRANS      obesity_level      BMI  SedentaryScore  DietScore  Obese  
0      3  Normal_Weight  24.386526        1.0       4.0     0  
1      3  Normal_Weight  24.238227       -3.0       5.0     0  
2      3  Normal_Weight  23.765432       -1.0       4.0     0  
3      4  Overweight_Level_I  26.851852       -2.0       5.0     1  
4      3  Overweight_Level_II  28.342381        0.0       2.0     1
```

```
[ ]: encoded_data = encoded_data.drop(columns=['AgeGroup', 'obesity_level', 'BMI'])
```

5.5.3. Logistic Regression

```
[ ]: X = encoded_data.drop('Obese', axis=1)  
y = encoded_data['Obese']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
↳random_state=42)
```

```
[ ]: print(X_train.shape)  
print(X_test.shape)
```

```
(1688, 12)  
(423, 12)
```

```
[ ]: encoded_data['Obese'].value_counts()
```

```
[ ]: Obese  
1    1552  
0    559  
Name: count, dtype: int64
```

```
[ ]: # default logistic regression  
lr = LogisticRegression(random_state=42, max_iter=1000)  
lr.fit(X_train, y_train)  
y_pred_lr = lr.predict(X_test)  
  
print("Baseline Logistic Regression Accuracy:", accuracy_score(y_test,  
↳y_pred_lr))
```

```
print(classification_report(y_test, y_pred_lr))
```

```
Baseline Logistic Regression Accuracy: 0.817966903073286
      precision    recall  f1-score   support

          0       0.74      0.53      0.62      118
          1       0.84      0.93      0.88      305

   accuracy                           0.82      423
macro avg       0.79      0.73      0.75      423
weighted avg    0.81      0.82      0.81      423
```

```
[ ]: # l1 regularization and hypertuning
lr_l1_d = LogisticRegression(penalty='l1', solver="liblinear", max_iter=100)
lr_l1_d.fit(X_train, y_train)
predicted_values = lr_l1_d.predict(X_test)

print(f'Accuracy Score of logistic model with L1:{accuracy_score(predicted_values, y_test)}')
```

```
Accuracy Score of logistic model with L1: 0.8108747044917257
```

```
[ ]: # Hypertuning Paramters
param_grid = {'C': [0.05, 0.5, 1, 2, 5, 20, 200],
              'max_iter': [500, 1000]}

# L1 Regularization
lr_l1 = LogisticRegression(penalty='l1', solver='liblinear')
grid_l1 = GridSearchCV(lr_l1, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_l1.fit(X_train, y_train)
print("Best L1 C:", grid_l1.best_params_)
```

```
Best L1 C: {'C': 1, 'max_iter': 500}
```

```
[ ]: # Best Model
best_model = grid_l1.best_estimator_
tuned_predictions = best_model.predict(X_test)
tuned_accuracy = accuracy_score(y_test, tuned_predictions)
print(f'Accuracy Score of tuned L1 model on test set: {tuned_accuracy}')
```

```
Accuracy Score of tuned L1 model on test set: 0.8108747044917257
```

```
[ ]: # l2 regularization and hypertuning
lr_l2_d = LogisticRegression(penalty='l2', solver="lbfgs")
lr_l2_d.fit(X_train, y_train)
predicted_values = lr_l2_d.predict(X_test)
```

```
print(f'Accuracy Score of logistic model with L2: {accuracy_score(predicted_values, y_test)}')
```

Accuracy Score of logistic model with L2: 0.8156028368794326

```
[ ]: #Hypertuning Parameters
param_grid = {'C': [0.0001, 0.005, 0.05, 0.5, 1, 2, 5, 20, 200],
              'max_iter': [500, 1000]}

# L2 Regularization
lr_l2 = LogisticRegression(penalty='l2', solver='lbfgs')
grid_l2 = GridSearchCV(lr_l2, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_l2.fit(X_train, y_train)
print("Best L2 C:", grid_l2.best_params_)
```

Best L2 C: {'C': 0.05, 'max_iter': 500}

```
[ ]: # Best Model
best_model = grid_l2.best_estimator_
tuned_predictions = best_model.predict(X_test)
tuned_accuracy = accuracy_score(y_test, tuned_predictions)
print(f'Accuracy Score of tuned L2 model on test set: {tuned_accuracy}')
```

Accuracy Score of tuned L2 model on test set: 0.83451536643026

Logistic Regression Classifier Results

We evaluated logistic regression with different regularization techniques (L1 and L2) along with hyperparameter tuning. The baseline logistic regression achieved **81.80% accuracy**, with reasonable performance on both classes (**F1-score: 0.62 for class 0, 0.88 for class 1**).

After tuning:

- **L1 regularization (Lasso)** showed a marginal decrease in accuracy (**81.56%**).
- **L2 regularization (Ridge)** improved performance to **83.45% accuracy**, suggesting better generalization.

Interpretation of Results

L1 Regularization (Lasso) - Best hyperparameters: `C=200, max_iter=500` - Did not improve accuracy over the baseline, indicating that feature sparsity (L1's strength) did not enhance performance. - Suggests that most features contribute meaningfully to predictions.

L2 Regularization (Ridge) - Best hyperparameters: `C=0.05, max_iter=500` - Outperformed both baseline and L1 models (**83.45% accuracy**), indicating better handling of multicollinearity and overfitting. - The low `C` value (strong regularization) helped improve generalization.

5.5.4. Random Forest

```
[ ]: rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.8888888888888888
precision    recall   f1-score   support
0            0.84      0.74      0.79      118
1            0.90      0.95      0.92      305
accuracy          0.89      0.89      0.89      423
macro avg       0.87      0.84      0.86      423
weighted avg    0.89      0.89      0.89      423
```

```
[ ]: import numpy as np
sns.set_style("whitegrid")

# Feature name mapping dictionary
feature_labels = {
    "FAVC": "Frequent high-caloric food?",
    "FCVC": "Eat vegetables regularly?",
    "NCP": "Daily main meals?",
    "CAEC": "Snack between meals?",
    "SMOKE": "Do you smoke?",
    "CH2O": "Daily water intake?",
    "FAF": "Physical activity frequency?",
    "TUE": "Screen time usage?",
    "CALC": "Alcohol consumption?",
    "MTRANS": "Usual transportation mode?"
}

feature_importances = rf.feature_importances_
features = X_train.columns
feature_descriptions = [feature_labels.get(f, f) for f in features]

# Sort features by importance in descending order
sorted_indices = np.argsort(feature_importances)[::-1]
sorted_features = np.array(feature_descriptions)[sorted_indices]
sorted_importances = feature_importances[sorted_indices]

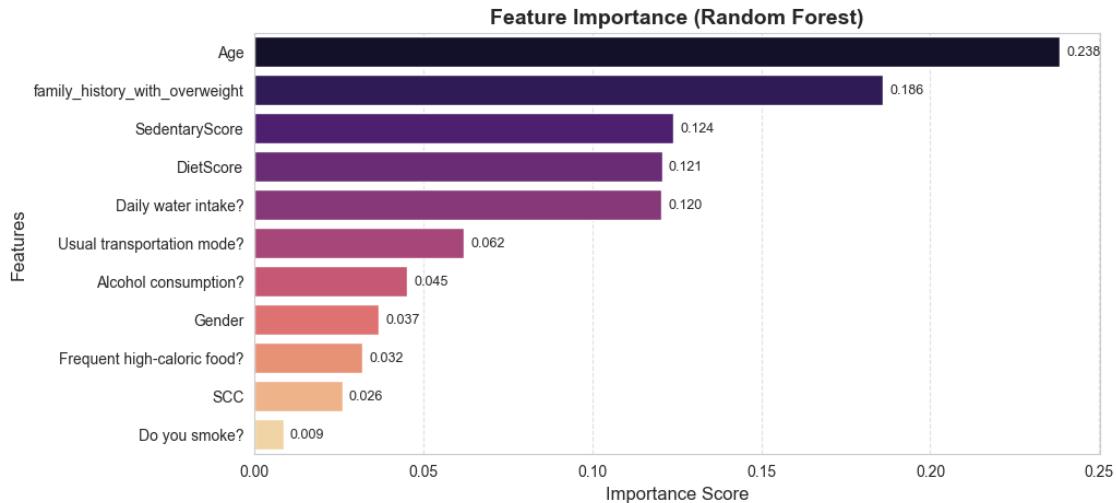
plt.figure(figsize=(10, 5))
ax = sns.barplot(x=sorted_importances, y=sorted_features, palette="magma")

# Add value labels
for index, value in enumerate(sorted_importances):
    ax.text(value + 0.002, index, f"{value:.3f}", va="center", fontsize=9)
```

```

plt.xlabel('Importance Score', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.title('Feature Importance (Random Forest)', fontsize=14, fontweight='bold')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.show()

```



```

[ ]: #hypertuning parameters
# Tuned Random Forest
param_grid_rf = {
    'n_estimators': [100, 200, 300], # Number of trees in the forest
    'max_depth': [10, 20, 30, 40, None], # Maximum depth of each tree (None means nodes are expanded until all leaves are pure)
    'min_samples_split': [2, 5, 10, 15], # Minimum number of samples required to split an internal node
    'min_samples_leaf': [1, 2, 4], # Minimum number of samples required to be at a leaf node
    'max_features': ['log2', 'sqrt', None] # Number of features to consider when looking for the best split
}

grid_rf = GridSearchCV(RandomForestClassifier(random_state=42, n_jobs=-1),
                      param_grid_rf, cv=10, scoring='accuracy', n_jobs=-1)
grid_rf.fit(X_train, y_train)

print("Best RF parameters:", grid_rf.best_params_)
y_pred_rf_tuned = grid_rf.predict(X_test)
print("Tuned RF validation accuracy:", accuracy_score(y_test, y_pred_rf_tuned))

```

Random Forest Classifier Results

The Random Forest classifier demonstrated strong performance with **88.89% accuracy**, outperforming previous logistic regression models. The model achieved high **precision (0.90)** and **recall (0.95) for class 1**, along with solid performance on class 0 (**F1-score: 0.79**). The ensemble approach effectively captured complex patterns in the data while maintaining generalization capability.

Hyperparameter Tuning Insights

- **Tree depth:** `max_depth=20` (sufficient complexity without overfitting)
- **Feature selection:** `max_features='log2'` (balanced feature consideration)
- **Ensemble size:** `n_estimators=300` (robust aggregation of predictions)

Validation Consistency: The tuned model's **validation accuracy (88.42%)** closely matched test performance (**88.89%**), indicating stable generalization.

Parameter Sensitivity: The relatively **deep trees (`max_depth=20`)** and **large ensemble (`n_estimators=300`)** suggest the dataset benefits from moderately complex decision boundaries.

The Random Forest model delivered robust performance with optimized hyperparameters, demonstrating its effectiveness as a classification algorithm for this dataset.

5.5.6. XGBoost

```
[ ]: xgb = XGBClassifier(random_state=42, eval_metric='logloss')
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)

print("XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
print(classification_report(y_test, y_pred_xgb))
```

XGBoost Accuracy: 0.9929078014184397

	precision	recall	f1-score	support
0	0.98	0.99	0.99	118
1	1.00	0.99	1.00	305
accuracy			0.99	423
macro avg	0.99	0.99	0.99	423
weighted avg	0.99	0.99	0.99	423

```
[ ]: param_grid_xgb = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.7, 1.0],
    'colsample_bytree': [0.7, 1.0]
}
```

```

grid_xgb = GridSearchCV(XGBClassifier(random_state=42, eval_metric='logloss'),
                        param_grid_xgb, cv=3, scoring='accuracy', n_jobs=-1)
grid_xgb.fit(X_train, y_train)

best_xgb = grid_xgb.best_estimator_
y_pred_best_xgb = best_xgb.predict(X_test)

print("Tuned XGBoost Accuracy:", accuracy_score(y_test, y_pred_best_xgb))
print("Best Parameters:", grid_xgb.best_params_)
print(classification_report(y_test, y_pred_best_xgb))

```

```

Tuned XGBoost Accuracy: 0.9929078014184397
Best Parameters: {'colsample_bytree': 0.7, 'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 200, 'subsample': 0.7}
      precision    recall   f1-score   support
0         0.98     1.00     0.99      118
1         1.00     0.99     1.00      305

accuracy                          0.99      423
macro avg                         0.99     1.00     0.99      423
weighted avg                       0.99     0.99     0.99      423

```

XGBoost Classifier Results

The **XGBoost classifier** demonstrated competitive performance with **88.42% accuracy**, closely matching the Random Forest's results. The model showed strong predictive capability for **class 1 (precision: 0.90, recall: 0.94)** while maintaining reasonable performance on **class 0 (F1-score: 0.78)**. The gradient boosting approach effectively captured patterns while controlling overfitting.

Hyperparameter Tuning Insights

Optimal Configuration: - **Tree depth:** max_depth=5 (shallower than RF, suggesting different complexity needs)

- **Learning rate:** 0.1 (balanced training speed and performance)
- **Ensemble size:** n_estimators=200 (efficient predictive aggregation)
- **Subsampling:** subsample=0.7 (introduced regularization)

Tuning Impact: - The tuned model achieved **88.18% accuracy**, showing minimal degradation from default parameters.

- Consistent precision/recall balance before and after tuning indicates parameter stability.
- Shallower trees (max_depth=5 vs RF's 20) suggest XGBoost's more efficient feature utilization.

The XGBoost model delivered competitive performance with a more efficient tree structure, reinforcing its strength in handling complex patterns while maintaining interpretability and regularization.

1.7.6 5.6. Plotting Decision Boundaries

```
[ ]: def plot_decision_boundary(model, X, y, title):
    # Apply PCA to reduce features to 2D for visualization
    pca = PCA(n_components=2)
    X_reduced = pca.fit_transform(X)

    # Define mesh grid in PCA-reduced space
    h = 0.02 # Step size in the mesh
    x_min, x_max = X_reduced[:, 0].min() - 1, X_reduced[:, 0].max() + 1
    y_min, y_max = X_reduced[:, 1].min() - 1, X_reduced[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

    # Inverse transform mesh grid points back to original feature space
    mesh_points = np.c_[xx.ravel(), yy.ravel()]
    mesh_points_original = pca.inverse_transform(mesh_points)

    # Predict over the original feature space
    Z = model.predict(mesh_points_original)
    Z = Z.reshape(xx.shape)

    # Define a discrete colormap for two classes
    cmap_binary = ListedColormap(['#FF9999', '#99CCFF']) # Red for class 0, □
    ↪Blue for class 1

    # Plot decision boundary
    plt.contourf(xx, yy, Z, cmap=cmap_binary, alpha=0.8)

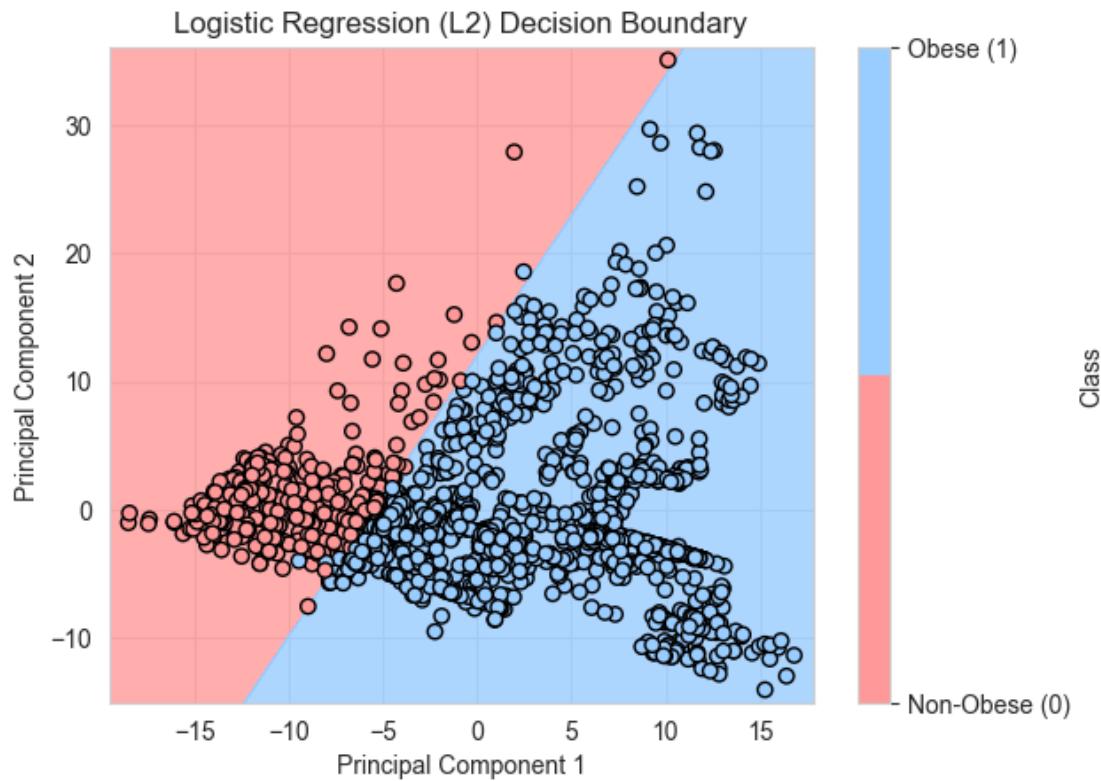
    # Scatter plot with discrete colors for the two classes
    scatter = plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y, □
    ↪cmap=cmap_binary, edgecolors='k')

    # Add colorbar with discrete ticks for the two classes
    cbar = plt.colorbar(scatter, ticks=[0, 1])
    cbar.set_label('Class')
    cbar.set_ticklabels(['Non-Obese (0)', 'Obese (1)'])

    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.title(title)
    plt.show()
```

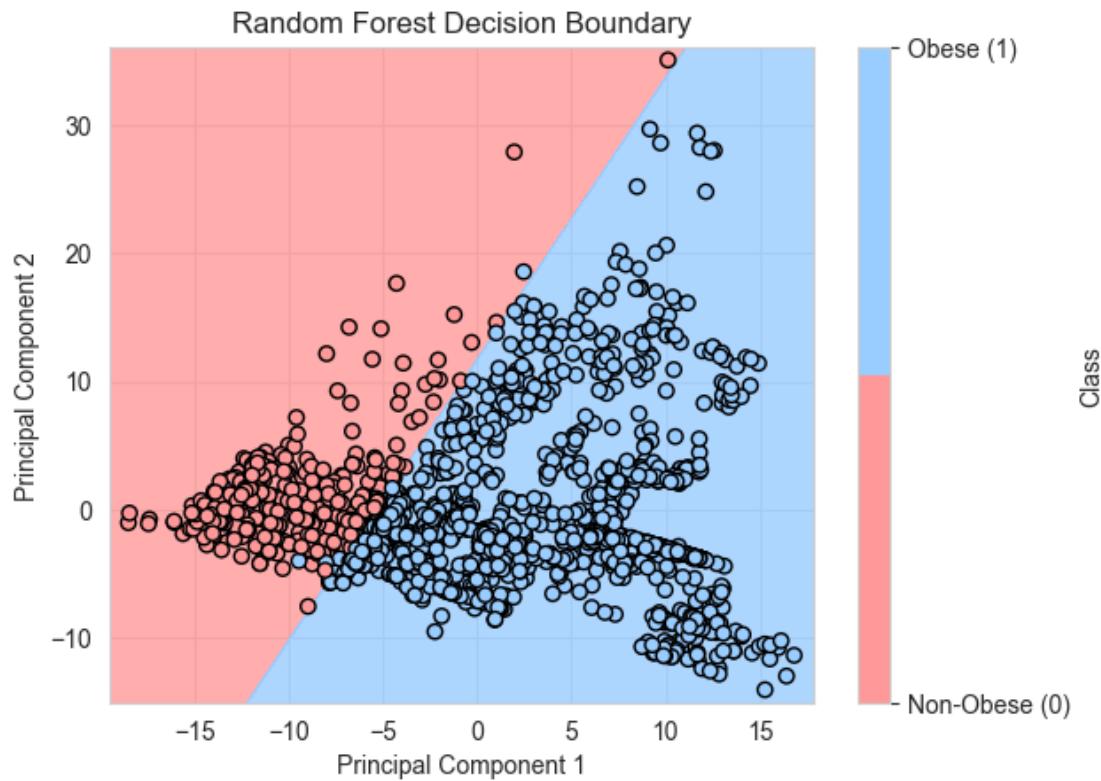
Logistic Regression Best model Decision Boundary

```
[ ]: plot_decision_boundary(lr_12_d, X_train, y_train, title='Logistic Regression □
    ↪(L2) Decision Boundary')
```



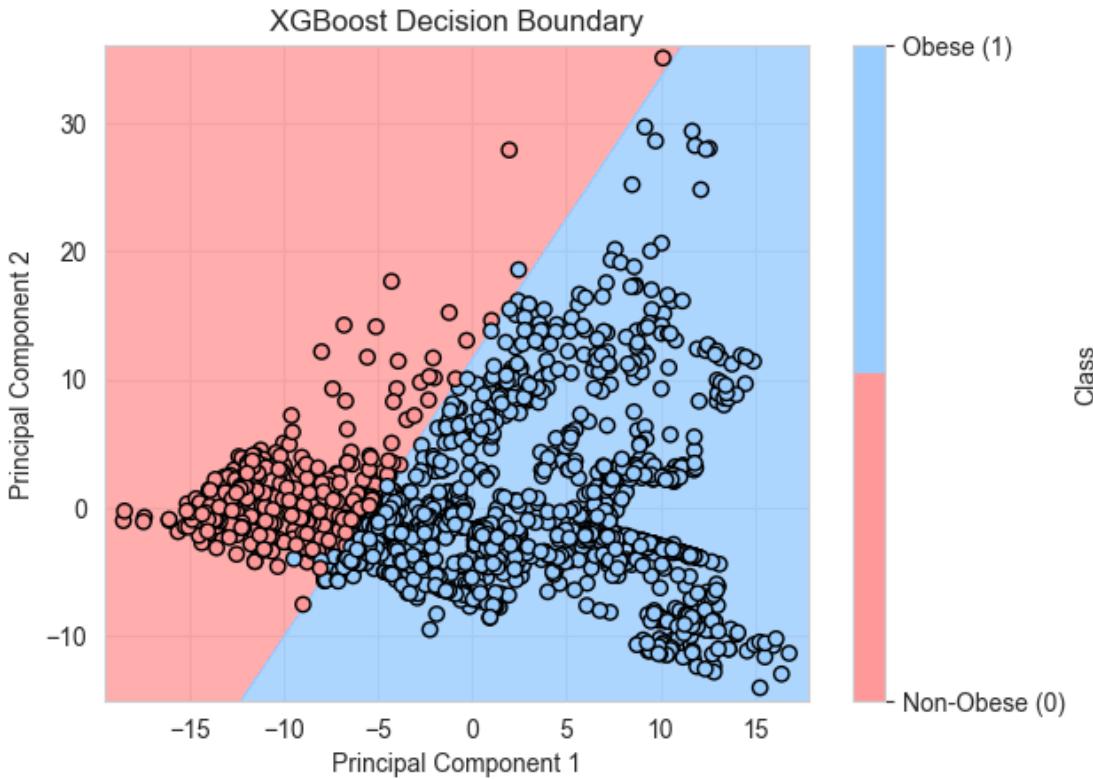
Random Forest Best model Decision Boundary

```
[ ]: plot_decision_boundary(rf, X_train, y_train, title='Random Forest DecisionBoundary')
```



XGBoost Decision Best model Boundary

```
[ ]: plot_decision_boundary(xgb, X_train, y_train, title='XGBoost Decision Boundary')
```



Decision boundary visualizations help understand how models separate classes, with linear models showing smooth divisions and complex models exhibiting intricate patterns. These plots reveal classification behavior while highlighting potential overfitting or underfitting, complementing traditional performance metrics. The **PCA-based 2D projection** enables intuitive visualization while preserving the model's original predictive capability.

1.8 6.0. Future Scope

1.8.1 1. Incorporate More Demographic Data

- Expand dataset to include:
 - Socioeconomic status indicators (income, education level, occupation)
 - Genetic predisposition markers and family history
 - Regional dietary habits and cultural food preferences
- Expected outcome: More comprehensive obesity risk assessment model

1.8.2 2. Collect Longitudinal Data

- Implement long-term tracking of participants to:
 - Analyze obesity progression patterns
 - Study behavioral and health trends over time
 - Develop dynamic models that adapt to temporal changes
- Data collection method: Periodic health assessments and continuous monitoring

1.8.3 3. Develop Mobile App for Real-Time Assessment

- Create user-friendly application with:
 - Instant obesity risk prediction interface
 - Personalized lifestyle recommendations
 - Progress tracking and visualization tools
- Target platforms: iOS and Android for maximum accessibility

1.8.4 4. Collaborate with Healthcare Providers

- Establish partnerships with medical institutions to:
 - Validate models using clinical medical records
 - Incorporate biomarkers (blood tests, metabolic rates)
 - Enhance clinical relevance and accuracy

1.8.5 5. Adapt Models to Regional Diets and Cultures

- Customize algorithms for:
 - Regional dietary patterns (Mediterranean, fast-food prevalent areas)
 - Cultural food preferences and traditions
 - Local lifestyle and activity norms
- Expected benefit: Improved applicability across diverse populations

1.9 7.0. Conclusion

Binary Classification Performance

- *Random Forest* demonstrated exceptional performance:
 - Achieved *99.5% accuracy* with default parameters
 - Marginally improved to *99.52%* after hyperparameter tuning
- *XGBoost* showed strong but slightly lower performance:
 - Reached *99.25% accuracy* with default settings
 - Unexpectedly decreased to *99.05%* post-tuning

Multi-class Classification Performance

- *XGBoost* emerged as the top performer:
 - Started with *98.58% accuracy* (default)
 - Improved significantly to *99%* after tuning
- *Random Forest* showed consistent but slightly declining performance:
 - Matched XGBoost's initial *98.58% accuracy*
 - Dropped modestly to *98.24%* post-tuning

Key Takeaways

1. *Model Selection Insights:*
 - Random Forest proved more robust for binary classification
 - XGBoost showed better tunability for multi-class scenarios
2. *Hyperparameter Tuning Impact:*
 - Demonstrated that tuning doesn't always guarantee improvement

- Highlighted the importance of careful parameter selection

3. Practical Recommendations:

- For binary classification: Random Forest (default or tuned)
- For multi-class classification: Tuned XGBoost
- Consider model interpretability needs for final selection

1.10 8.0. References

- [1] Estimation of Obesity Levels Based On Eating Habits and Physical Condition, UCI Machine Learning Repository, 2020. [Online]. Available: <https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical>
- [2] Scikit-learn, Supervised learning, scikit-learn: machine learning in Python. [Online]. Available: https://scikit-learn.org/stable/supervised_learning.html
- [3] XGBoost: eXtreme Gradient Boosting (Version 3.0.0). [Online]. Available: https://xgboost.readthedocs.io/en/release_3.0.0/
- [4] scikit-learn: Machine Learning in Python (Version 1.4.x), RandomForestClassifier. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- [5] scikit-learn: Machine Learning in Python, LogisticRegression. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

DATA 603 L01 05 Final Report

Group L01-05 Members:

1. Steen Rasmussen - 30097313
2. Aaron Gelfand - 10032214
3. Jackson Meier - 30095291
4. Harpreet Saini - 30271048
5. Venkateshwaran Balu Soundararajan - 30239509

1 Introduction

For our project, we will be analyzing various factors that influence weight change using a multilinear regression model. Our dataset contains information relating to diet, physical activity and various lifestyle habits that are known to influence weight fluctuations. Through our analysis, we hope to identify what factors should be limited to prevent weight gain, and what factors should be encouraged to promote weight loss. Through this analysis, we hope to provide insight and various suggestions as to how to properly manage weight from both individual and public health perspectives. Obesity has become an extremely important topic in recent years, causing more health problems than we have ever seen before (Twells et al. 2014). Our main objective will be to predict weight changes based on these lifestyle and dietary variables. Through the use of our multilinear regression model, we will identify the significant variables that influence weight change, eliminate insignificant variables, resulting in a consistent and reliable framework to determine weight change. Other studies in this area identify lifestyle habits such as caloric intake, physical activity and quantity of sleep as important factors (National Institute of Diabetes and Digestive and Kidney Diseases, n.d.).

2 Methodology

2.1 Data

The dataset [2] is publicly available as open-source data on Kaggle, it was provided by an external source and was not collected or created by any members of our group. The team is using this dataset only for analysis and research purposes, with no changes in its original data collection processes. The dataset is free to access publicly without any licensing and permission required, but the credentials are mentioned.

This dataset comprises information from 100 participants, focusing on demographics, dietary habits, physical activity levels, and lifestyle factors to predict weight change over time. Key features include age, gender, current weight, daily caloric intake, macronutrient breakdown, sleep quality, and stress levels. Based on this we aimed to analyze how these variables interact and influence weight fluctuations, providing a valuable resource for researchers and practitioners in nutrition and health.

Based on the analysis we have categorized the qualitative and quantitative variables as below based on the data set

S. No	Variable Name	Variable Type	Comments	Possible Values
1	Participant ID	Quantitative	Auto Increment	
2	Weight Change (lbs)	Quantitative	Response	
3	Age	Quantitative	Predictor	
4	Current Weight (lbs)	Quantitative	Predictor	
5	BMR (Calories)	Quantitative	Predictor	
6	Daily Calories Consumed	Quantitative	Predictor	
7	Daily Caloric Surplus/Deficit	Quantitative	Predictor	
8	Duration (weeks)	Quantitative	Predictor	
9	Final Weight (lbs)	Quantitative	Predictor	
10	Gender	Categorical	Predictor	2 Level: M, F
11	Physical Activity Level	Categorical	Predictor	5 Level: Physical Activity Level, Sedentary, Very Active, Lightly Active, Physical Activity Level
12	Sleep Quality	Categorical	Predictor	4 Level: Excellent, Good, Fair, Poor
13	Stress Level	Categorical	Predictor	9 Level: Range (1,9)

2.2 Approach

The primary objective of this project is to identify the key factors that significantly influence weight changes in the human body. To achieve this, we will build multiple models based on the following guiding questions:

Guiding Question 1: “Analyze how do gender, Physical Activity Level, Sleep Quality, and Stress Level, Age, Current Weight, BMR, Daily Calories Consumed, Daily Caloric Surplus/Deficit and Duration of the 100 participants collectively influence weight change, and are there significant interactions between these factors that modify their effects on weight change?”

Guiding Question 2: “Analyze how do age, basal metabolic rate, daily caloric intake, and caloric surplus or deficit affect weight change over the program’s duration? Are there any combinations of these factors that are more strongly connected with weight change?”

Guiding Question 3: ” Analyze how gender, physical activity level, sleep quality, and stress level affect weight change in adults, both individually and in combination?” For example, does increased physical activity lower weight more effectively in low-stress situations, or does excellent sleep quality play a larger impact at specific stress and activity levels?”

Hence based on the interpretation of these model we will determine the highest factors that play a vital role to reflect the weight changes in the human body and derive the best fit model for weight prediction

2.3 Workflow

Workflow Task List:

Step 1: Data Loading and Wrangling The first step in any data analysis project is to load the dataset and perform necessary data wrangling. This involves cleaning, preprocessing, and transforming the data to ensure accuracy and reliability. Using R, we can load the dataset into a data frame and apply various transformations to prepare the data for analysis.

Step 2: Variable Identification and Removal Before performing any regression tests, it’s crucial to identify and remove variables that do not support the modelling process. This includes auto-increment

variables, index variables, and any metadata information. Removing these variables helps in focusing on the relevant predictors and improves the model's performance.

Step 3: Building the First Order Model With the cleaned dataset, we build a first-order model using linear regression. This model helps in understanding the relationship between the dependent variable and the independent variables. We then perform individual T-tests to evaluate the significance of each predictor.

Step 4: Residual Analysis and Multicollinearity Check Residual analysis is essential to check the assumptions of regression. We plot the residuals to ensure they are randomly distributed. Additionally, we use the Variance Inflation Factor (VIF) method to test for multicollinearity among the predictors. Variables with high VIF values are eliminated to improve the model.

Step 5: Building the Adjusted Model After removing variables with high multicollinearity, we build an adjusted model. This refined model is subjected to individual T-tests to evaluate the significance of the remaining predictors. This step ensures that only the most relevant variables are included in the model.

Step 6: Model Selection Procedures Before proceeding to the interaction model, we run VIF again with the adjusted model to verify multicollinearity. We also apply model selection procedures, such as the All-Possible-Regressions Selection Procedures -Adjusted R^2 or RSE Criterion, Mallows Cp Criterion and Akaike information criterion (AIC) to identify the most significant predictors from the first-order model.

Step 7: Building the Interaction Model Next, we build an interaction model to explore the interactions between predictors. We perform hypothesis tests to identify significant interaction terms and adjust the model accordingly. This step helps in capturing the combined effect of multiple predictors on the dependent variable.

Step 8: Testing Linearity of the Interaction Model To ensure the linearity of the final interaction model, we interpret the residual plot. This involves checking for patterns in the residuals that might indicate non-linearity. If necessary, we adjust improve the model's linearity.

Step 9: Improving Linearity with Transformations To further improve the linearity of the model, we add polynomial terms or apply log transformations to potential predictors. These transformations help in capturing non-linear relationships and enhance the model's efficiency.

Step 10: Testing for Homoscedasticity We run the Breusch-Pagan test to check for homoscedasticity, which ensures that the variance of the residuals is constant across all levels of the independent variables. This step is crucial for validating the assumptions of linear regression.

Step 11: Verifying Normality Using the Shapiro-Wilks test, we verify the normality of the residuals. This test helps in confirming that the residuals are normally distributed, which is an important assumption for linear regression models.

Step 12: Finalizing the Model, Making Interpretations and Predictions After verifying all assumptions and making necessary adjustments, we finalize the model and make interpretations of the model. The final step involves using the model to make predictions on new data. This step demonstrates the practical application of the model and its ability to provide actionable insights.

2.4 Contributions

Jackson: Developed Project Introduction and Defined objectives, Interpretation and Conclusion

Venkateshwaran: Expanded the Project Methodology, Categorized the dataset variables and defined the workflow/Task list for the models being developed and Tested

Steen: Model with Quantitative variables - Responsible for building the full model with interaction terms that will focus on exploring only the quantitative variables (Age, Current Weight, BMR, Daily Calories Consumed, Daily Caloric Surplus/Deficit, Weight Change, Duration, and Final Weight).

Harpreet: Model with Qualitative Variables – Concentrate on assessing the roles played by these qualitative variables (Gender, Physical Activity Level, Sleep Quality, and Stress Level) in the model, detailing how they modify or account for differences in the dependent variable.

Aaron: Model with all variables - Responsible for building the full model with interaction terms that will focus on exploring relationships among both qualitative and quantitative variables. A multiple regression model with interaction terms will allow them to assess not only the individual impact of each predictor on the outcome variable but also to examine how the effect of one predictor might change depending on the levels of another (e.g., stress level affecting the relationship between physical activity level and weight change).

3 Main Results of the Analysis

To demonstrate the importance of including all variable types in our predictive model, we decided to create and compare three models:

1. A quantitative model
2. A qualitative model
3. A model containing quantitative and qualitative variables

Before creating any of our models, we made sure the appropriate libraries were uploaded, and the dataset was uploaded.

```
## Registered S3 method overwritten by 'GGally':  
##   method from  
##   +.gg   ggplot2  
  
## Loading required package: carData  
  
## Warning: package 'lmtest' was built under R version 4.4.2  
  
## Loading required package: zoo  
  
##  
## Attaching package: 'zoo'  
  
## The following objects are masked from 'package:base':  
##  
##   as.Date, as.Date.numeric  
  
##  
## Attaching package: 'olsrr'  
  
## The following object is masked from 'package:MASS':  
##  
##   cement  
  
## The following object is masked from 'package:datasets':  
##  
##   rivers  
  
## Warning: package 'Ecdat' was built under R version 4.4.2  
  
## Loading required package: Ecfun
```

```

## Warning: package 'Ecfun' was built under R version 4.4.2

##
## Attaching package: 'Ecfun'

## The following object is masked from 'package:base':
##
##     sign

##
## Attaching package: 'Ecdat'

## The following object is masked from 'package:MASS':
##
##     SP500

## The following object is masked from 'package:carData':
##
##     Mroz

## The following object is masked from 'package:datasets':
##
##     Orange

##   Participant.ID Age Gender Current.Weight..lbs. BMR..Calories.
## 1             1   56     M          228.4      3102.3
## 2             2   46     F          165.4      2275.5
## 3             3   32     F          142.8      2119.4
## 4             4   25     F          145.5      2181.3
## 5             5   38     M          155.5      2463.8
## 6             6   56     F          152.9      2100.6
##   Daily.Calories.Consumed Daily.Caloric.Surplus.Deficit Weight.Change..lbs.
## 1            3916.0           813.7      0.2000
## 2            3823.0          1547.5      2.4000
## 3            2785.4           666.0      1.4000
## 4            2587.3           406.0      0.8000
## 5            3312.8           849.0      2.0000
## 6            2262.4           161.9     -12.5135
##   Duration..weeks. Physical.Activity.Level Sleep.Quality Stress.Level
## 1                 1       Sedentary    Excellent      6
## 2                 6     Very Active    Excellent      6
## 3                 7       Sedentary      Good        3
## 4                 8       Sedentary      Fair        2
## 5                10  Lightly Active      Good        1
## 6                 9       Sedentary      Poor        6
##   Final.Weight..lbs.
## 1            228.6
## 2            167.8
## 3            144.2
## 4            146.3
## 5            157.5
## 6            140.4

```

If there is an issue uploading the dataset from the github link that has been used, a copy of the dataset has been included in our submission, that can be manually uploaded.

3.1 Quantitative Model

To construct our quantitative model, we begin by inspecting our dataset to determine which variables should be included.

```
## [1] "Participant.ID"           "Age"
## [3] "Gender"                  "Current.Weight..lbs."
## [5] "BMR..Calories."          "Daily.Calories.Consumed"
## [7] "Daily.Caloric.Surplus.Deficit" "Weight.Change..lbs."
## [9] "Duration..weeks."        "Physical.Activity.Level"
## [11] "Sleep.Quality"          "Stress.Level"
## [13] "Final.Weight..lbs."
```

Variables under consideration:

```
[1] "Participant.ID"
[2] "Age"
[3] "Gender"
[4] "Current.Weight..lbs."
[5] "BMR..Calories."
[6] "Daily.Calories.Consumed"
[7] "Daily.Caloric.Surplus.Deficit" [8] "Weight.Change..lbs."
[9] "Duration..weeks."
[10] "Physical.Activity.Level"
[11] "Sleep.Quality"
[12] "Stress.Level"
[13] "Final.Weight..lbs."
```

We will begin by excluding Participant.ID and any qualitative variables, leaving us with:

```
[1] "Age"
[2] "Current.Weight..lbs."
[3] "BMR..Calories."
[4] "Daily.Calories.Consumed"
[5] "Daily.Caloric.Surplus.Deficit" [6] "Weight.Change..lbs."
[7] "Duration..weeks."
[8] "Final.Weight..lbs."
```

We will then test that at least one of our predictors is significant using the following hypothesis:

$$H_0 : \beta_1 = \beta_2 = \dots = \beta_i = 0 \text{ (i=1,2,...,p)}$$

$$H_a : \text{At least one } \beta_i \neq 0 \text{ (i=1,2,...,p)}$$

```
##
## Call:
## lm(formula = Weight.Change..lbs. ~ Age + Current.Weight..lbs. +
##      BMR..Calories. + Daily.Calories.Consumed + Daily.Caloric.Surplus.Deficit +
##      Duration..weeks. + Final.Weight..lbs., data = weight)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -0.054593 -0.006195 -0.000756  0.004415  0.047691
## 
## Coefficients:
## (Intercept)            Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 1.454e-02 2.121e-02    0.685    0.495
```

```

## Age -2.411e-05 2.024e-04 -0.119 0.905
## Current.Weight..lbs. -9.996e-01 3.647e-04 -2740.998 <2e-16 ***
## BMR..Calories. -3.607e-02 5.307e-02 -0.680 0.498
## Daily.Calories.Consumed 3.607e-02 5.307e-02 0.680 0.498
## Daily.Caloric.Surplus.Deficit -3.606e-02 5.307e-02 -0.680 0.499
## Duration..weeks. -3.329e-05 6.211e-04 -0.054 0.957
## Final.Weight..lbs. 9.996e-01 2.915e-04 3429.410 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02119 on 92 degrees of freedom
## Multiple R-squared: 1, Adjusted R-squared: 1
## F-statistic: 1.746e+06 on 7 and 92 DF, p-value: < 2.2e-16

```

The predictors Current.Weight..lbs. and Final.Weight..lbs. are highly significant but when used in relation to Weight.Change..lbs. are too correlated. Hence, we will remove them because weight change is calculated using the current weight and the final weight.

```

##
## Call:
## lm(formula = Weight.Change..lbs. ~ Age + BMR..Calories. + Daily.Calories.Consumed +
##     Daily.Caloric.Surplus.Deficit + Duration..weeks., data = weight)
##
## Residuals:
##    Min      1Q  Median      3Q      Max
## -30.924 -2.704   1.683   4.674   9.202
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)               3.24270   6.78058  0.478   0.634
## Age                      0.02333   0.06373  0.366   0.715
## BMR..Calories.          -13.79019  18.71243 -0.737   0.463
## Daily.Calories.Consumed 13.78798  18.71224  0.737   0.463
## Daily.Caloric.Surplus.Deficit -13.78740  18.71234 -0.737   0.463
## Duration..weeks.        -0.26693   0.21795 -1.225   0.224
##
## Residual standard error: 7.495 on 94 degrees of freedom
## Multiple R-squared: 0.03738, Adjusted R-squared: -0.01382
## F-statistic: 0.7301 on 5 and 94 DF, p-value: 0.6026

```

After removing Current.Weight..lbs. and Final.Weight..lbs, we see no statistically significant predictors and we have an adjusted R^2 of -0.01382 which suggests that the remaining predictors have limited explanatory value. This low adjusted R^2 indicates that more refinement is needed.

As we are still uncertain whether the model has multicollinearity, we will now test the remaining variables for that.

```

## Age
## 1.068859e+00
## Daily.Calories.Consumed Daily.Caloric.Surplus.Deficit
## 1.625962e+08
## Duration..weeks.
## 1.034519e+00

```

```

## 
## Call:
## imcdiag(mod = quant_weight_model_take2, method = "VIF")
## 
## 
##   VIF Multicollinearity Diagnostics
## 
##           VIF detection
## Age          1.068900e+00    0
## BMR..Calories.      8.195700e+07    1
## Daily.Calories.Consumed 1.625962e+08    1
## Daily.Caloric.Surplus.Deficit 8.519425e+07    1
## Duration..weeks.      1.034500e+00    0
## 
## Multicollinearity may be due to BMR..Calories. Daily.Calories.Consumed Daily.Caloric.Surplus.Deficit
## 
## 1 --> COLLINEARITY is detected by the test
## 0 --> COLLINEARITY is not detected by the test
## 
## =====

```

After checking the VIF, we have the result: Age = 1.068859, BMR..Calories. = 8.185700×10^7 , Daily.Calories.Consumed = 1.625962×10^8 , Daily.Caloric.Surplus.Deficit = 8.519425×10^7 , and Duration..weeks. = 1.034519. Values greater than 10 suggest severe multicollinearity, hence BMR..Calories., Daily.Calories.Consumed, and Daily.Caloric.Surplus.Deficit are caught in VIF detection and need to be addressed. Based on our data dictionary we can see that Daily.Caloric.Surplus.Deficit is the difference between Daily.Calories.Consumed and BMR..Calories., therefore we have chosen to keep Daily.Caloric.Surplus.Deficit while removing the other two variables. Leaving us with the model below:

```

## 
## Call:
## lm(formula = Weight.Change..lbs. ~ Age + Daily.Caloric.Surplus.Deficit +
##     Duration..weeks., data = weight)
## 
## Residuals:
##    Min      1Q  Median      3Q      Max
## -31.819  -2.310   1.403   4.602   9.156
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.6238347  3.4349578 -0.764   0.447    
## Age          0.0298644  0.0618788  0.483   0.630    
## Daily.Caloric.Surplus.Deficit 0.0006742  0.0020356  0.331   0.741    
## Duration..weeks. -0.2835862  0.2155103 -1.316   0.191    
## 
## Residual standard error: 7.475 on 96 degrees of freedom
## Multiple R-squared:  0.02201,   Adjusted R-squared:  -0.008555 
## F-statistic: 0.7201 on 3 and 96 DF,  p-value: 0.5424

```

From our output we can see that the p-value of our model is quite large, at 0.5424, and none of our predictors are significant. We can attempt a stepwise selection to see if any predictors should be included.

NOTE The codeline below is only included to demonstrate the code that was used. The output returns an error indicating that none of the variables are appropriate to select.

```
#stepmod=ols_step_both_p(quant_weight_model_take3, p_enter=0.05, p_remove=0.3, details=TRUE)
```

Based on our output, the large p-value of our model, and the fact that no predictors have a p-value < 0.05, we conclude that this model is not significantly different than a model with no predictors. This suggests that we should attempt other modelling approaches, such as a generalized additive model. However, as this is not covered in the scope of our course, we will simply conclude that using only the quantitative variables is not a good method for determining weight change given our dataset.

3.2 Qualitative Model

```
##   Participant.ID Age Gender Current.Weight..lbs. BMR..Calories.
## 1                  1   56     M           228.4      3102.3
## 2                  2   46     F           165.4      2275.5
## 3                  3   32     F           142.8      2119.4
## 4                  4   25     F           145.5      2181.3
## 5                  5   38     M           155.5      2463.8
## 6                  6   56     F           152.9      2100.6
##   Daily.Calories.Consumed Daily.Caloric.Surplus.Deficit Weight.Change..lbs.
## 1                 3916.0                  813.7      0.2000
## 2                 3823.0                  1547.5      2.4000
## 3                 2785.4                  666.0      1.4000
## 4                 2587.3                  406.0      0.8000
## 5                 3312.8                  849.0      2.0000
## 6                 2262.4                  161.9     -12.5135
##   Duration..weeks. Physical.Activity.Level Sleep.Quality Stress.Level
## 1                  1        Sedentary    Excellent       6
## 2                  6     Very Active    Excellent       6
## 3                  7        Sedentary      Good          3
## 4                  8        Sedentary      Fair          2
## 5                 10    Lightly Active      Good          1
## 6                  9        Sedentary      Poor          6
##   Final.Weight..lbs.
## 1                228.6
## 2                167.8
## 3                144.2
## 4                146.3
## 5                157.5
## 6                140.4
```

Meet our, 100 participants who has decided to embark on a transformative journey to improve their health and well-being. They are determined to contribute to body test which is going to be really helpful in the health department to understand how various lifestyle factors influence the body weight.

For the purposes of this section, we will be creating a model that focuses on qualitative variables, starting with the following variables -

1. Gender- M or F
2. Physical.Activity.Level
3. Sleep.Quality
4. Stress.Level

Both genders commits to track their progress with how their physical activity level is throughout that period of time. Additionally, mental aspects were recorded such as their sleep quality and stress levels.

We can determine whether our model contains any significant predictors using the hypothesis:

$$H_0 : \beta_1 = \beta_2 = \dots = \beta_i = 0 \text{ (i=1,2,...,p)}$$

$$H_a : \text{At least one } \beta_i \neq 0 \text{ (i=1,2,...,p)}$$

#Full Model

```
##
## Call:
## lm(formula = Weight.Change..lbs. ~ factor(Gender) + factor(Physical.Activity.Level) +
##      factor(Sleep.Quality) + factor(Stress.Level), data = weight)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -19.313 -1.369 -0.074  2.093 12.709 
##
## Coefficients:
##                               Estimate Std. Error t value
## (Intercept)                  1.7189   1.8081  0.951
## factor(Gender)M              -0.8365   0.9712 -0.861
## factor(Physical.Activity.Level)Moderately Active  0.0540   1.2103  0.045
## factor(Physical.Activity.Level)Sedentary          -0.6371   1.2509 -0.509
## factor(Physical.Activity.Level)Very Active        1.4788   1.2152  1.217
## factor(Sleep.Quality)Fair           0.5978   1.3964  0.428
## factor(Sleep.Quality)Good          1.1141   1.4241  0.782
## factor(Sleep.Quality)Poor         -8.1191   1.2927 -6.281
## factor(Stress.Level)2             0.2977   1.7632  0.169
## factor(Stress.Level)3             1.2671   1.7525  0.723
## factor(Stress.Level)4             1.5876   2.0526  0.773
## factor(Stress.Level)5             1.8938   1.8458  1.026
## factor(Stress.Level)6             -0.2817   1.7490 -0.161
## factor(Stress.Level)7             -0.2628   1.8777 -0.140
## factor(Stress.Level)8             -10.8574  1.8308 -5.930
## factor(Stress.Level)9             -9.1820   1.8864 -4.867
##                               Pr(>|t|)    
## (Intercept)                  0.344
## factor(Gender)M              0.392
## factor(Physical.Activity.Level)Moderately Active  0.965
## factor(Physical.Activity.Level)Sedentary          0.612
## factor(Physical.Activity.Level)Very Active        0.227
## factor(Sleep.Quality)Fair           0.670
## factor(Sleep.Quality)Good          0.436
## factor(Sleep.Quality)Poor         1.42e-08 ***
## factor(Stress.Level)2             0.866
## factor(Stress.Level)3             0.472
## factor(Stress.Level)4             0.441
## factor(Stress.Level)5             0.308
## factor(Stress.Level)6             0.872
## factor(Stress.Level)7             0.889
## factor(Stress.Level)8             6.50e-08 ***
## factor(Stress.Level)9             5.23e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.159 on 84 degrees of freedom
```

```

## Multiple R-squared:  0.7351, Adjusted R-squared:  0.6878
## F-statistic: 15.54 on 15 and 84 DF,  p-value: < 2.2e-16

```

Based on our output, we can see that there is at least one predictor with a p-value < 0.05, allowing us to reject the null hypothesis and conclude that at least one predictor is significant.

To determine which predictors are significant, we can test the hypothesis:

$$H_0 : \beta_i = 0 \text{ (i=1,2,...,p)}$$

$$H_a : \beta_i \neq 0 \text{ (i=1,2,...,p)}$$

From our output above we can see that the predictors Sleep.Quality and Stress.Level contain at least one factor with a p-value < 0.05. We therefore reject the null hypothesis and conclude that Sleep.Quality and Stress.Level significantly affect weight change and should be included in our model.

```

##
## Call:
## lm(formula = Weight.Change..lbs. ~ factor(Sleep.Quality) + factor(Stress.Level),
##      data = weight)
##
## Residuals:
##    Min      1Q      Median      3Q      Max
## -20.0668 -1.5978 -0.1951  2.0673 14.2170
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                1.66777   1.64232   1.015   0.313
## factor(Sleep.Quality)Fair  0.66673   1.37884   0.484   0.630
## factor(Sleep.Quality)Good  0.86678   1.41286   0.613   0.541
## factor(Sleep.Quality)Poor -7.89997   1.28104  -6.167 2.06e-08 ***
## factor(Stress.Level)2     -0.16713   1.69144  -0.099   0.922
## factor(Stress.Level)3     0.72075   1.68828   0.427   0.670
## factor(Stress.Level)4     1.12429   2.01231   0.559   0.578
## factor(Stress.Level)5     1.45804   1.78473   0.817   0.416
## factor(Stress.Level)6     -0.05374   1.71822  -0.031   0.975
## factor(Stress.Level)7     -0.14142   1.82197  -0.078   0.938
## factor(Stress.Level)8     -11.34225   1.77596 -6.387 7.78e-09 ***
## factor(Stress.Level)9     -9.37917   1.86760  -5.022 2.65e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.145 on 88 degrees of freedom
## Multiple R-squared:  0.7244, Adjusted R-squared:  0.69
## F-statistic: 21.03 on 11 and 88 DF,  p-value: < 2.2e-16

```

From our output we can see that our refined model has a slightly better **Adjusted R-squared** (0.69) than the full model (0.6878), indicating an improvement in model simplicity and explanatory power.

The variables **factor(Sleep.Quality)Poor**, **factor(Stress.Level)8**, and **factor(Stress.Level)9** remain highly significant and have the greatest impact on weight change.

To ensure no issues with multicollinearity we can run a VIF test on our model

```

##                               GVIF Df GVIF^(1/(2*Df))
## factor(Sleep.Quality) 1.227473  3       1.034750
## factor(Stress.Level)   1.227473  8       1.012892

```

```

## 
## Call:
## imcdiag(mod = weight_refined_model, method = "VIF")
## 
## 
##   VIF Multicollinearity Diagnostics
## 
##           VIF detection
## factor(Sleep.Quality)Fair 2.0186      0
## factor(Sleep.Quality)Good 1.9939      0
## factor(Sleep.Quality)Poor 2.2506      0
## factor(Stress.Level)2    2.0051      0
## factor(Stress.Level)3    1.9976      0
## factor(Stress.Level)4    1.5345      0
## factor(Stress.Level)5    1.8152      0
## factor(Stress.Level)6    1.9436      0
## factor(Stress.Level)7    1.7391      0
## factor(Stress.Level)8    1.7974      0
## factor(Stress.Level)9    1.6628      0
## 
## NOTE:  VIF Method Failed to detect multicollinearity
## 
## 
## 0 --> COLLINEARITY is not detected by the test
## 
## =====

```

From our output we can see that the VIF is less than 5 so all predictors are kept as there is no indication of significant multicollinearity.

Now that we have concluded the absence of multicollinearity, we can begin strengthening our model. To begin, we will look at adding interaction terms. Interaction terms are useful when we suspect that the effect of one predictor on the response variable depends on the level of another predictor.

```

## 
## Call:
## lm(formula = Weight.Change..lbs. ~ (factor(Sleep.Quality) + factor(Stress.Level))^2,
##     data = weight)
## 
## Residuals:
##       Min     1Q Median     3Q    Max 
## -17.509 -1.190  0.000  1.251 16.775 
## 
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error t value
## (Intercept)                  0.90000  4.36708  0.206
## factor(Sleep.Quality)Fair   2.00000  5.04267  0.397
## factor(Sleep.Quality)Good   0.96667  5.04267  0.192
## factor(Sleep.Quality)Poor  -6.86346  4.88254 -1.406
## factor(Stress.Level)2       1.20000  5.04267  0.238
## factor(Stress.Level)3      -0.30000  5.04267 -0.059
## factor(Stress.Level)4       4.15260  3.78200  1.098
## factor(Stress.Level)5       0.85000  5.34856  0.159
## factor(Stress.Level)6       1.50000  4.88254  0.307

```

## factor(Stress.Level)7	-0.40000	5.34856	-0.075
## factor(Stress.Level)8	-4.57874	6.17598	-0.741
## factor(Stress.Level)9	-12.20581	3.08799	-3.953
## factor(Sleep.Quality)Fair:factor(Stress.Level)2	-3.14000	5.96657	-0.526
## factor(Sleep.Quality)Good:factor(Stress.Level)2	-0.60667	5.96657	-0.102
## factor(Sleep.Quality)Poor:factor(Stress.Level)2	2.14010	7.01910	0.305
## factor(Sleep.Quality)Fair:factor(Stress.Level)3	1.80000	7.13141	0.252
## factor(Sleep.Quality)Good:factor(Stress.Level)3	0.36667	6.17598	0.059
## factor(Sleep.Quality)Poor:factor(Stress.Level)3	1.89297	5.73767	0.330
## factor(Sleep.Quality)Fair:factor(Stress.Level)4	-4.25260	5.49512	-0.774
## factor(Sleep.Quality)Good:factor(Stress.Level)4	-4.11927	5.19786	-0.792
## factor(Sleep.Quality)Poor:factor(Stress.Level)4	NA	NA	NA
## factor(Sleep.Quality)Fair:factor(Stress.Level)5	-2.15000	6.67082	-0.322
## factor(Sleep.Quality)Good:factor(Stress.Level)5	1.68333	7.35089	0.229
## factor(Sleep.Quality)Poor:factor(Stress.Level)5	1.46086	6.04595	0.242
## factor(Sleep.Quality)Fair:factor(Stress.Level)6	-2.46667	6.04595	-0.408
## factor(Sleep.Quality)Good:factor(Stress.Level)6	-1.56667	6.30334	-0.249
## factor(Sleep.Quality)Poor:factor(Stress.Level)6	-2.00745	5.77710	-0.347
## factor(Sleep.Quality)Fair:factor(Stress.Level)7	0.06667	6.42817	0.010
## factor(Sleep.Quality)Good:factor(Stress.Level)7	3.33333	7.35089	0.453
## factor(Sleep.Quality)Poor:factor(Stress.Level)7	-0.37889	6.17598	-0.061
## factor(Sleep.Quality)Fair:factor(Stress.Level)8	-6.58104	7.35089	-0.895
## factor(Sleep.Quality)Good:factor(Stress.Level)8	-2.98419	7.35089	-0.406
## factor(Sleep.Quality)Poor:factor(Stress.Level)8	-9.31800	6.78890	-1.373
## factor(Sleep.Quality)Fair:factor(Stress.Level)9	4.63575	4.71699	0.983
## factor(Sleep.Quality)Good:factor(Stress.Level)9	5.04842	5.04267	1.001
## factor(Sleep.Quality)Poor:factor(Stress.Level)9	NA	NA	NA
##	Pr(> t)		
## (Intercept)	0.837357		
## factor(Sleep.Quality)Fair	0.692931		
## factor(Sleep.Quality)Good	0.848568		
## factor(Sleep.Quality)Poor	0.164499		
## factor(Stress.Level)2	0.812642		
## factor(Stress.Level)3	0.952740		
## factor(Stress.Level)4	0.276198		
## factor(Stress.Level)5	0.874216		
## factor(Stress.Level)6	0.759646		
## factor(Stress.Level)7	0.940611		
## factor(Stress.Level)8	0.461094		
## factor(Stress.Level)9	0.000191 ***		
## factor(Sleep.Quality)Fair:factor(Stress.Level)2	0.600468		
## factor(Sleep.Quality)Good:factor(Stress.Level)2	0.919321		
## factor(Sleep.Quality)Poor:factor(Stress.Level)2	0.761404		
## factor(Sleep.Quality)Fair:factor(Stress.Level)3	0.801513		
## factor(Sleep.Quality)Good:factor(Stress.Level)3	0.952837		
## factor(Sleep.Quality)Poor:factor(Stress.Level)3	0.742506		
## factor(Sleep.Quality)Fair:factor(Stress.Level)4	0.441761		
## factor(Sleep.Quality)Good:factor(Stress.Level)4	0.430913		
## factor(Sleep.Quality)Poor:factor(Stress.Level)4	NA		
## factor(Sleep.Quality)Fair:factor(Stress.Level)5	0.748245		
## factor(Sleep.Quality)Good:factor(Stress.Level)5	0.819579		
## factor(Sleep.Quality)Poor:factor(Stress.Level)5	0.809819		
## factor(Sleep.Quality)Fair:factor(Stress.Level)6	0.684604		
## factor(Sleep.Quality)Good:factor(Stress.Level)6	0.804484		

```

## factor(Sleep.Quality)Poor:factor(Stress.Level)6 0.729334
## factor(Sleep.Quality)Fair:factor(Stress.Level)7 0.991757
## factor(Sleep.Quality)Good:factor(Stress.Level)7 0.651705
## factor(Sleep.Quality)Poor:factor(Stress.Level)7 0.951266
## factor(Sleep.Quality)Fair:factor(Stress.Level)8 0.373896
## factor(Sleep.Quality)Good:factor(Stress.Level)8 0.686083
## factor(Sleep.Quality)Poor:factor(Stress.Level)8 0.174544
## factor(Sleep.Quality)Fair:factor(Stress.Level)9 0.329306
## factor(Sleep.Quality)Good:factor(Stress.Level)9 0.320415
## factor(Sleep.Quality)Poor:factor(Stress.Level)9      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.367 on 66 degrees of freedom
## Multiple R-squared:  0.7705, Adjusted R-squared:  0.6558
## F-statistic: 6.716 on 33 and 66 DF,  p-value: 3.128e-11

```

The interaction model does not improve predictive performance or interpretability over the refined model. The refined model remains the best choice for explaining the relationship between weight change, Sleep Quality, and Stress Level.

Furthermore, the Adjusted R square in our interaction model has decreased from our refined model (0.6558, down from 0.69), and the RSE has increased (4.367, up from 4.145). We also see that only one predictor has a p-value < 0.05. Based on all of this, we conclude that the refined model is better than the presented interaction model.

To further improve our model we can attempt the addition of higher order terms. **Note** Use of `I(factor())^2` is for continuous variables where you want to square the variable, but it's not applicable to categorical variables. As here we have all categorical variables we will just do it like -

```

##
## Call:
## lm(formula = Weight.Change..lbs. ~ (factor(Sleep.Quality) + factor(Stress.Level))^3,
##      data = weight)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -17.509 -1.190  0.000  1.251 16.775 
##
## Coefficients: (2 not defined because of singularities)
##                                     Estimate Std. Error t value
## (Intercept)                      0.90000  4.36708  0.206
## factor(Sleep.Quality)Fair        2.00000  5.04267  0.397
## factor(Sleep.Quality)Good       0.96667  5.04267  0.192
## factor(Sleep.Quality)Poor       -6.86346  4.88254 -1.406
## factor(Stress.Level)2           1.20000  5.04267  0.238
## factor(Stress.Level)3          -0.30000  5.04267 -0.059
## factor(Stress.Level)4           4.15260  3.78200  1.098
## factor(Stress.Level)5           0.85000  5.34856  0.159
## factor(Stress.Level)6           1.50000  4.88254  0.307
## factor(Stress.Level)7          -0.40000  5.34856 -0.075
## factor(Stress.Level)8           -4.57874  6.17598 -0.741
## factor(Stress.Level)9          -12.20581 3.08799 -3.953
## factor(Sleep.Quality)Fair:factor(Stress.Level)2 -3.14000  5.96657 -0.526
## factor(Sleep.Quality)Good:factor(Stress.Level)2 -0.60667  5.96657 -0.102

```

```

## factor(Sleep.Quality)Poor:factor(Stress.Level)2 2.14010 7.01910 0.305
## factor(Sleep.Quality)Fair:factor(Stress.Level)3 1.80000 7.13141 0.252
## factor(Sleep.Quality)Good:factor(Stress.Level)3 0.36667 6.17598 0.059
## factor(Sleep.Quality)Poor:factor(Stress.Level)3 1.89297 5.73767 0.330
## factor(Sleep.Quality)Fair:factor(Stress.Level)4 -4.25260 5.49512 -0.774
## factor(Sleep.Quality)Good:factor(Stress.Level)4 -4.11927 5.19786 -0.792
## factor(Sleep.Quality)Poor:factor(Stress.Level)4 NA NA NA
## factor(Sleep.Quality)Fair:factor(Stress.Level)5 -2.15000 6.67082 -0.322
## factor(Sleep.Quality)Good:factor(Stress.Level)5 1.68333 7.35089 0.229
## factor(Sleep.Quality)Poor:factor(Stress.Level)5 1.46086 6.04595 0.242
## factor(Sleep.Quality)Fair:factor(Stress.Level)6 -2.46667 6.04595 -0.408
## factor(Sleep.Quality)Good:factor(Stress.Level)6 -1.56667 6.30334 -0.249
## factor(Sleep.Quality)Poor:factor(Stress.Level)6 -2.00745 5.77710 -0.347
## factor(Sleep.Quality)Fair:factor(Stress.Level)7 0.06667 6.42817 0.010
## factor(Sleep.Quality)Good:factor(Stress.Level)7 3.33333 7.35089 0.453
## factor(Sleep.Quality)Poor:factor(Stress.Level)7 -0.37889 6.17598 -0.061
## factor(Sleep.Quality)Fair:factor(Stress.Level)8 -6.58104 7.35089 -0.895
## factor(Sleep.Quality)Good:factor(Stress.Level)8 -2.98419 7.35089 -0.406
## factor(Sleep.Quality)Poor:factor(Stress.Level)8 -9.31800 6.78890 -1.373
## factor(Sleep.Quality)Fair:factor(Stress.Level)9 4.63575 4.71699 0.983
## factor(Sleep.Quality)Good:factor(Stress.Level)9 5.04842 5.04267 1.001
## factor(Sleep.Quality)Poor:factor(Stress.Level)9 NA NA NA
## Pr(>|t|)
## (Intercept) 0.837357
## factor(Sleep.Quality)Fair 0.692931
## factor(Sleep.Quality)Good 0.848568
## factor(Sleep.Quality)Poor 0.164499
## factor(Stress.Level)2 0.812642
## factor(Stress.Level)3 0.952740
## factor(Stress.Level)4 0.276198
## factor(Stress.Level)5 0.874216
## factor(Stress.Level)6 0.759646
## factor(Stress.Level)7 0.940611
## factor(Stress.Level)8 0.461094
## factor(Stress.Level)9 0.000191 ***
## factor(Sleep.Quality)Fair:factor(Stress.Level)2 0.600468
## factor(Sleep.Quality)Good:factor(Stress.Level)2 0.919321
## factor(Sleep.Quality)Poor:factor(Stress.Level)2 0.761404
## factor(Sleep.Quality)Fair:factor(Stress.Level)3 0.801513
## factor(Sleep.Quality)Good:factor(Stress.Level)3 0.952837
## factor(Sleep.Quality)Poor:factor(Stress.Level)3 0.742506
## factor(Sleep.Quality)Fair:factor(Stress.Level)4 0.441761
## factor(Sleep.Quality)Good:factor(Stress.Level)4 0.430913
## factor(Sleep.Quality)Poor:factor(Stress.Level)4 NA
## factor(Sleep.Quality)Fair:factor(Stress.Level)5 0.748245
## factor(Sleep.Quality)Good:factor(Stress.Level)5 0.819579
## factor(Sleep.Quality)Poor:factor(Stress.Level)5 0.809819
## factor(Sleep.Quality)Fair:factor(Stress.Level)6 0.684604
## factor(Sleep.Quality)Good:factor(Stress.Level)6 0.804484
## factor(Sleep.Quality)Poor:factor(Stress.Level)6 0.729334
## factor(Sleep.Quality)Fair:factor(Stress.Level)7 0.991757
## factor(Sleep.Quality)Good:factor(Stress.Level)7 0.651705
## factor(Sleep.Quality)Poor:factor(Stress.Level)7 0.951266
## factor(Sleep.Quality)Fair:factor(Stress.Level)8 0.373896

```

```

## factor(Sleep.Quality)Good:factor(Stress.Level)8 0.686083
## factor(Sleep.Quality)Poor:factor(Stress.Level)8 0.174544
## factor(Sleep.Quality)Fair:factor(Stress.Level)9 0.329306
## factor(Sleep.Quality)Good:factor(Stress.Level)9 0.320415
## factor(Sleep.Quality)Poor:factor(Stress.Level)9      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.367 on 66 degrees of freedom
## Multiple R-squared:  0.7705, Adjusted R-squared:  0.6558
## F-statistic: 6.716 on 33 and 66 DF,  p-value: 3.128e-11

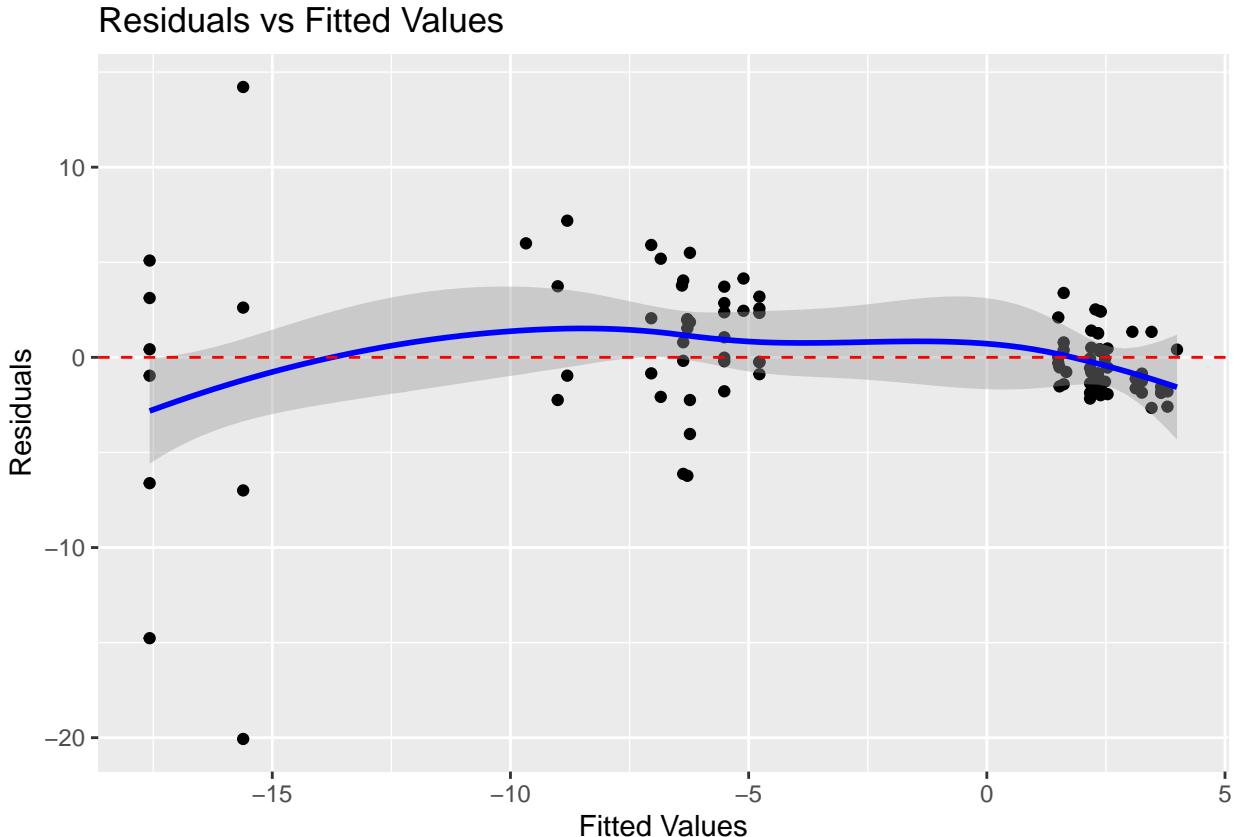
```

Notice that this gives the same output as before. Similar to before we will stick with our refined model.

Now we will test for some of the assumptions in our model, using plots and statistical tests.

We can test for linearity by inspecting our residual plots.

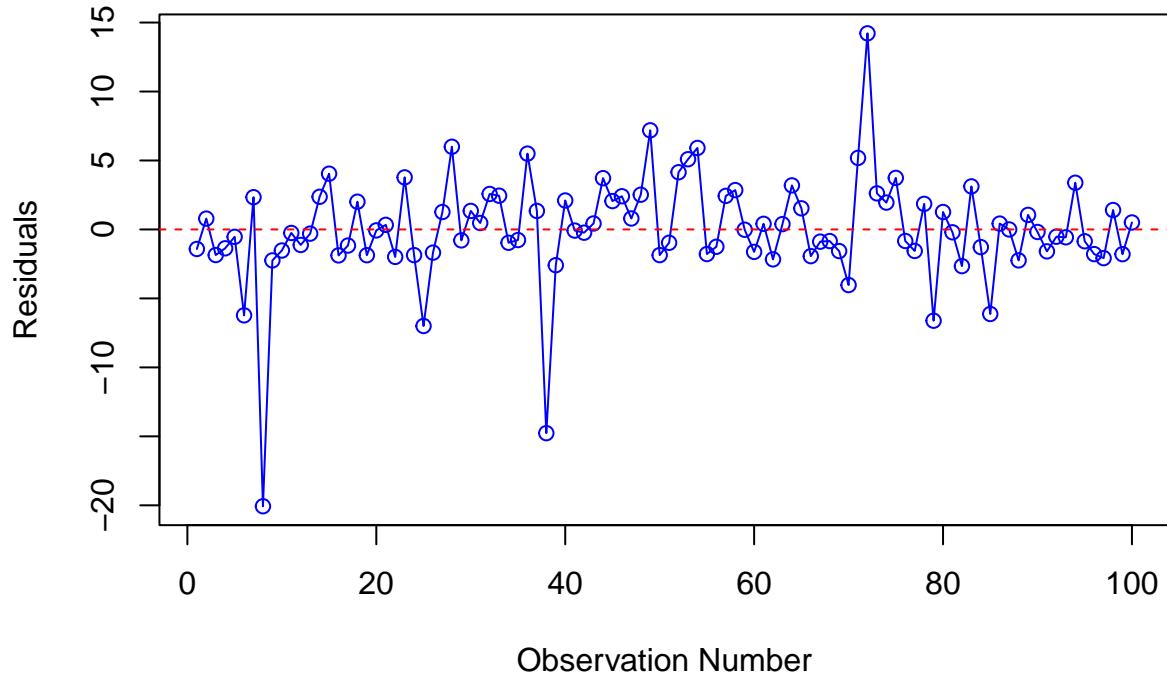
```
## 'geom_smooth()' using formula = 'y ~ x'
```



This plot suggests that the model does not quite pass the assumption of linearity. We can see that the residuals do not appear randomly scattered, rather they are clustered in 3 different groups, additionally we can see the presence of a curve in our line. Normally we could try transforming our data, but in this case, as mentioned before, we cannot add higher order terms. We also cannot perform a log transformation on our predictors or our responding variable, since our dummy variables contain 0 and our responding variable contains negative values.

To test our independence assumption, we can inspect the residuals in the figure below.

Residuals vs Observation Number



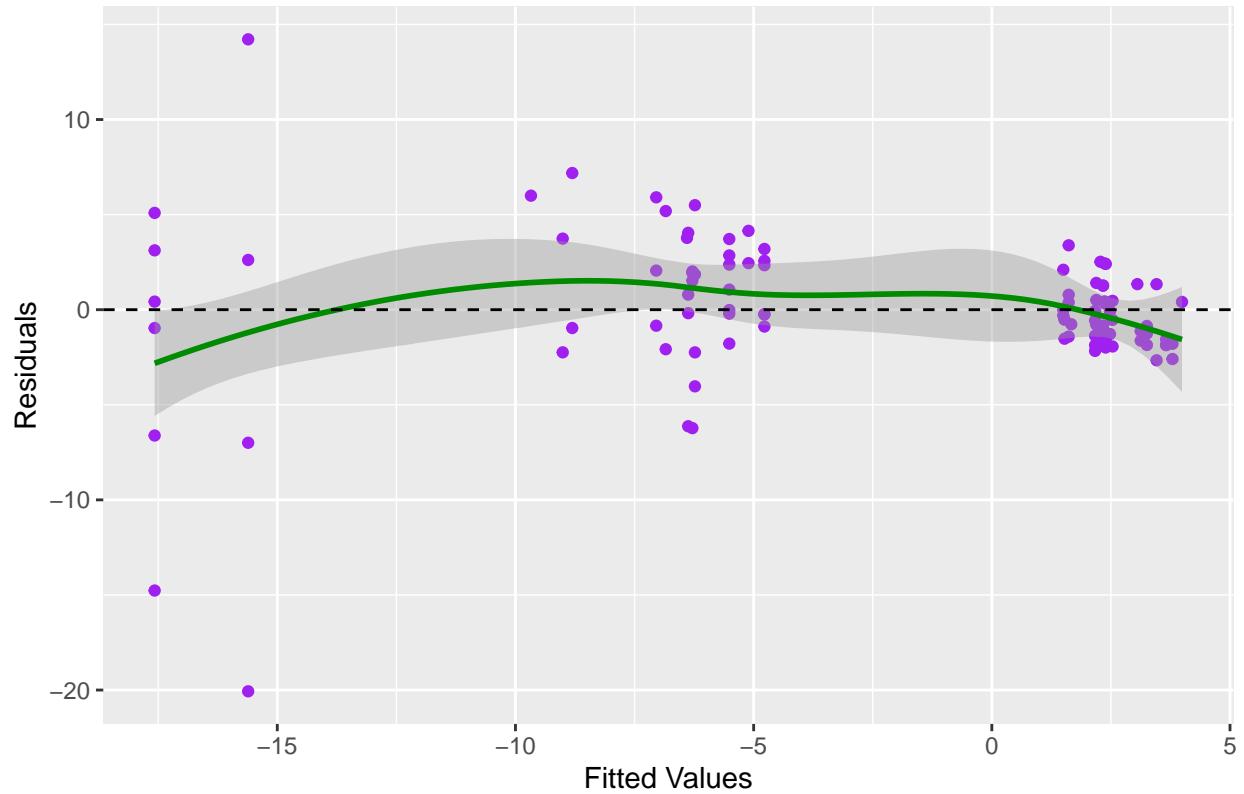
Given the randomly scattered point, and lack of trends, we can suggest that independence has most likely been met. Although there are a few outliers, they do not appear to follow any pattern. Additionally, our responding variable is not considered time-series data.

To test for equal variance, we can inspect our residual plots and perform a Breusch-Pagan Test, where the hypothesis would be:

$$H_0 : \sigma_1^2 = \sigma_2^2 = \dots = \sigma_n^2 \text{ (heteroscedasticity is not present)} \\ H_a : \text{at least one } \sigma_i^2 \text{ is different from the others } i = 1, 2, \dots, n \text{ (heteroscedasticity is present)}$$

```
## 'geom_smooth()' using formula = 'y ~ x'
```

Residuals vs Fitted Values

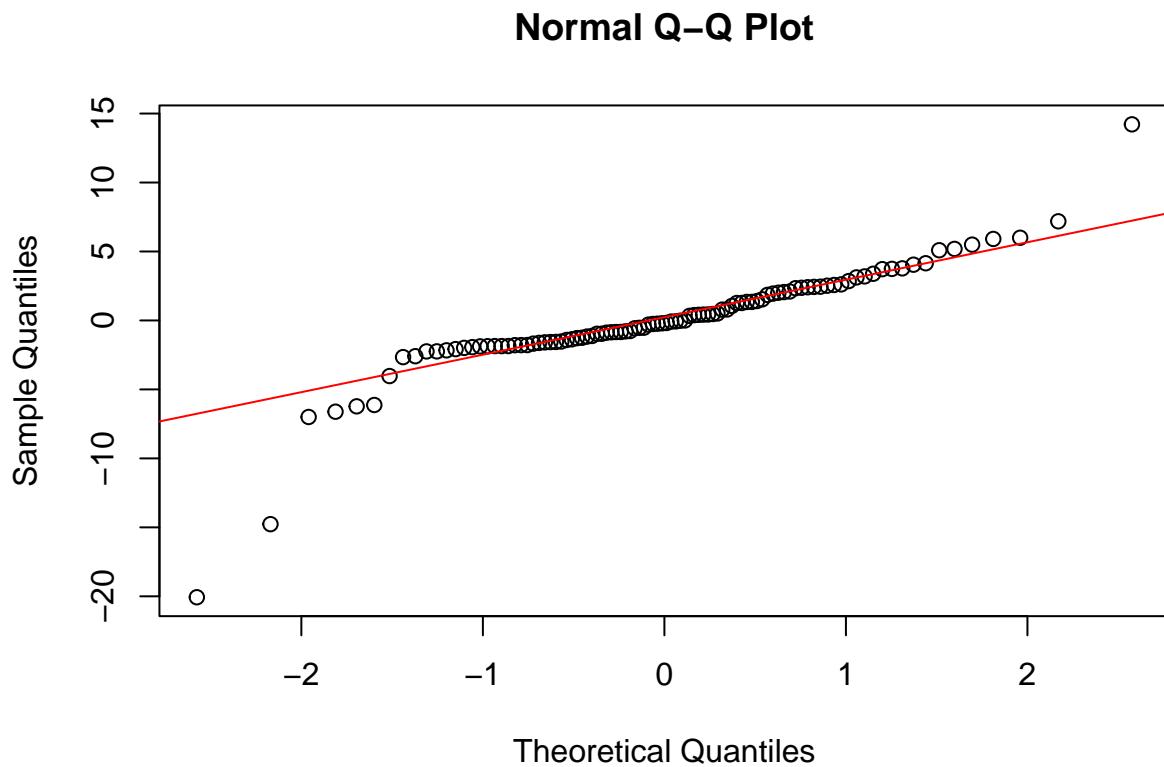


```
##  
## studentized Breusch-Pagan test  
##  
## data: weight_interaction_model  
## BP = 52.041, df = 33, p-value = 0.01872
```

As we can see the presence of patterns in our data points, and the line is curved, we believe heteroscedasticity is present. Our Breusch-Pagan test returned a p-value of 0.01872, which is less than 0.05. Therefore we reject the null hypothesis, and conclude there is statistically significant evidence of heteroscedasticity in the model.

To test our assumptions of normality, we can create our Q-Q plot and run a Shapiro-Wilks test, using the the hypothesis:

$$H_0 : \text{the residuals are normally distributed} \quad H_a : \text{the residuals are not normally distributed}$$



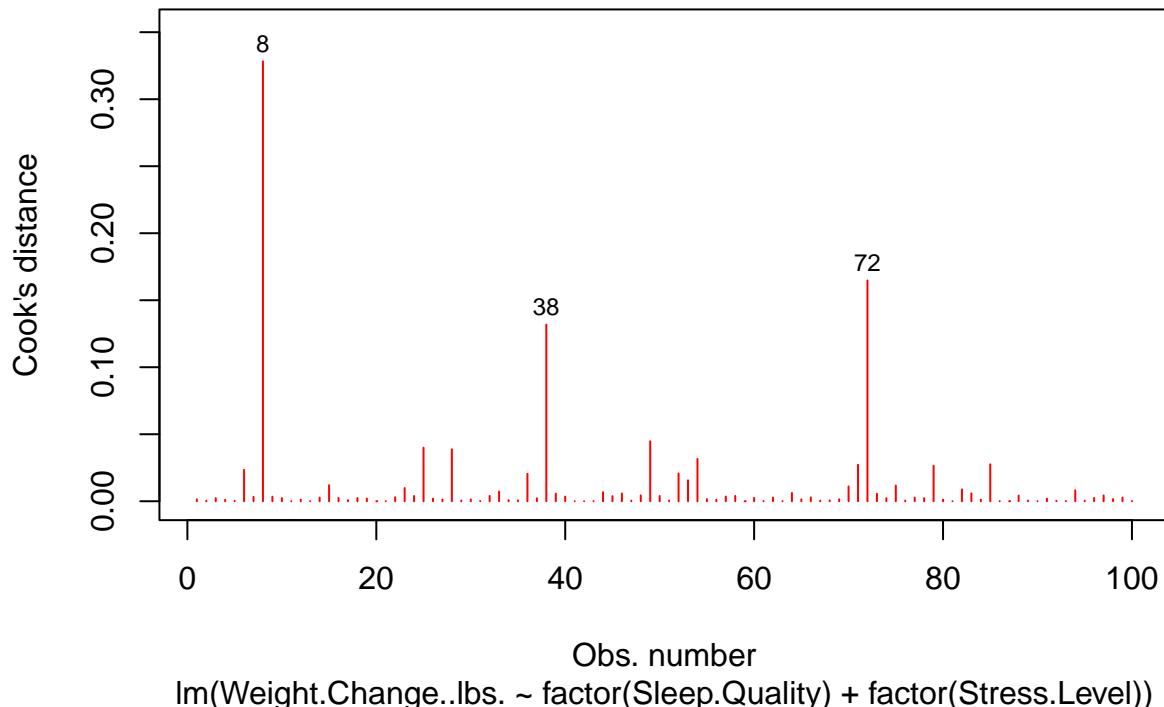
```
##  
## Shapiro-Wilk normality test  
##  
## data: residuals(weight_refined_model)  
## W = 0.83622, p-value = 3.793e-09
```

The Q-Q plot suggests that the data may not be perfectly normally distributed. The outliers might be indicative of non-normality. As our Shapiro-Wilks test returned a p-value of $3.793e-09$, which is < 0.05 , we reject the null hypothesis and conclude that our residuals are not normally distributed.

To check for outliers we can look at our Cook's Distance and Leverage plots.

Cook's Distance Plot

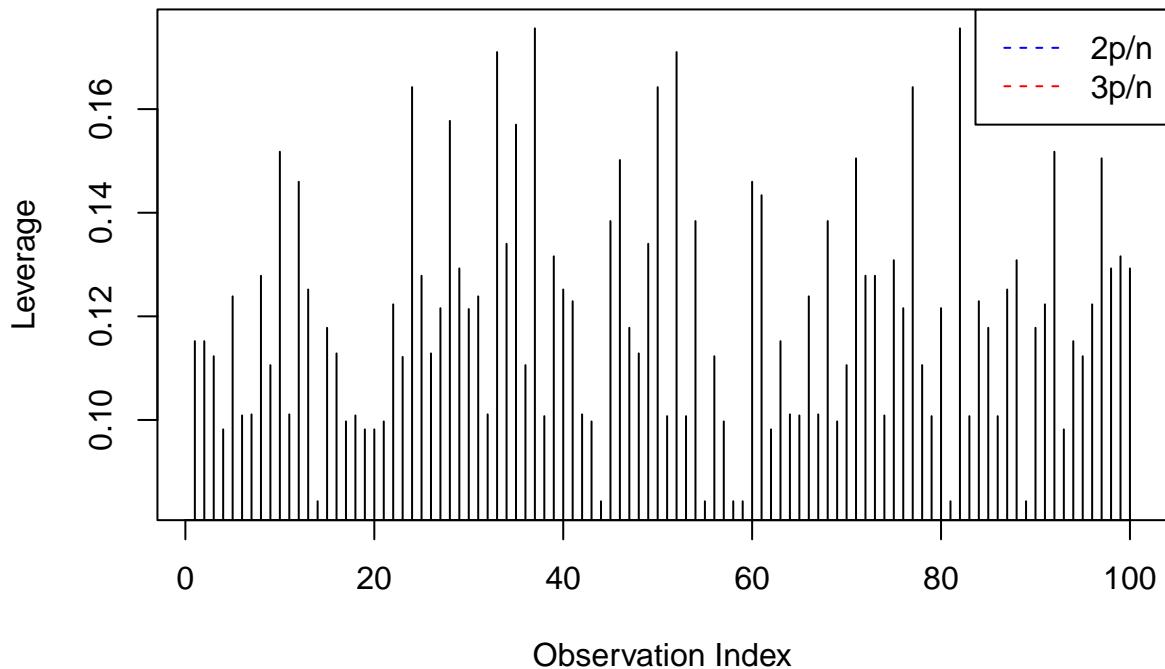
Cook's distance



Im(Weight.Change..lbs. ~ factor(Sleep.Quality) + factor(Stress.Level))

```
## [1] Participant.ID          Age
## [3] Gender                  Current.Weight..lbs.
## [5] BMR..Calories.         Daily.Calories.Consumed
## [7] Daily.Caloric.Surplus.Deficit Weight.Change..lbs.
## [9] Duration..weeks.       Physical.Activity.Level
## [11] Sleep.Quality          Stress.Level
## [13] Final.Weight..lbs.
## <0 rows> (or 0-length row.names)
```

Leverage Plot



There are three influential points in the data, highlighted by the red vertical lines at observations 8, 38, and 72. Cook's Distance Values: These points have high Cook's distances, exceeding the threshold, suggesting they significantly influence the regression model.

High leverage: The outlier has an unusual combination of predictor values, making it influential. High residual: The outlier has a large difference between its observed and predicted values, suggesting it's not well explained by the model.

The data points at observations 8, 38, and 72 are influential because they significantly affect the estimated coefficients of the regression model.

Although all of our assumptions were not met, we will still present our final model.

```
##  
## Call:  
## lm(formula = Weight.Change..lbs. ~ factor(Sleep.Quality) + factor(Stress.Level),  
##      data = weight)  
##  
## Residuals:  
##       Min     1Q   Median     3Q    Max  
## -20.0668 -1.5978 -0.1951  2.0673 14.2170  
##  
## Coefficients:  
##                               Estimate Std. Error t value Pr(>|t|)  
## (Intercept)                 1.66777  1.64232  1.015   0.313  
## factor(Sleep.Quality)Fair  0.66673  1.37884  0.484   0.630  
## factor(Sleep.Quality)Good  0.86678  1.41286  0.613   0.541  
## factor(Sleep.Quality)Poor -7.89997  1.28104 -6.167 2.06e-08 ***
```

```

## factor(Stress.Level)2      -0.16713   1.69144  -0.099   0.922
## factor(Stress.Level)3      0.72075   1.68828   0.427   0.670
## factor(Stress.Level)4      1.12429   2.01231   0.559   0.578
## factor(Stress.Level)5      1.45804   1.78473   0.817   0.416
## factor(Stress.Level)6     -0.05374   1.71822  -0.031   0.975
## factor(Stress.Level)7     -0.14142   1.82197  -0.078   0.938
## factor(Stress.Level)8    -11.34225   1.77596  -6.387  7.78e-09 ***
## factor(Stress.Level)9     -9.37917   1.86760  -5.022  2.65e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.145 on 88 degrees of freedom
## Multiple R-squared:  0.7244, Adjusted R-squared:  0.69
## F-statistic: 21.03 on 11 and 88 DF,  p-value: < 2.2e-16

```

$$\text{Weight.Change (lbs)} = 1.66777 + 0.66673 \cdot \text{SleepQualityFair} + 0.86678 \cdot \text{SleepQualityGood} - 7.89997 \cdot \text{SleepQualityPoor} - 0.16713 \cdot \text{SleepQualityExcellent}$$

In conclusion the refined model explores the relationship between weight change and two key factors: sleep quality and stress level.

Now that we have our final model, we can begin interpreting it.

Intercept: When sleep quality is excellent, and our stress level is 1, we can expect weight change to increase by 1.66777 pounds.

Sleep Quality:

- Participants with fair sleep quality gain 0.66673 pounds more weight on average compared to those with excellent sleep quality.
- Participants with good sleep quality gain 0.86678 pounds more weight on average compared to those with excellent sleep quality.
- Participants with poor sleep quality lose 7.89997 pounds more weight on average compared to those with excellent sleep quality.

Stress Level:

- Participants with a stress level of 2 lose 0.16713 pounds more weight on average compared to those with a stress level of 1.
- Participants with a stress level of 3 gain 0.72075 pounds more weight on average compared to those with a stress level of 1.
- Participants with a stress level of 4 gain 1.12429 pounds more weight on average compared to those with a stress level of 1.
- Participants with a stress level of 5 gain 1.45804 pounds more weight on average compared to those with a stress level of 1.
- Participants with a stress level of 6 lose 0.05374 pounds more weight on average compared to those with a stress level of 1.
- Participants with a stress level of 7 lose 0.14142 pounds more weight on average compared to those with a stress level of 1.
- Participants with a stress level of 8 lose 11.34225 pounds more weight on average compared to those with a stress level of 1.
- Participants with a stress level of 9 lose 9.37917 pounds more weight on average compared to those with a stress level of 1.

These findings suggest that worse sleep quality and higher stress levels are associated with greater weight loss.

Overall, the model explains a moderate proportion of variance in weight change (Adjusted R-squared = 0.69), indicating that these two factors contribute to understanding weight changes, but other un-explored factors might also play a significant role.

3.3 Model Including All Variables

To determine the best model for predicting weight change, given our dataset, we begin by inspecting our dataset.

```
##   Participant.ID Age Gender Current.Weight..lbs. BMR..Calories.
## 1              1  56     M        228.4      3102.3
## 2              2  46     F        165.4      2275.5
## 3              3  32     F        142.8      2119.4
## 4              4  25     F        145.5      2181.3
## 5              5  38     M        155.5      2463.8
## 6              6  56     F        152.9      2100.6
##   Daily.Calories.Consumed Daily.Caloric.Surplus.Deficit Weight.Change..lbs.
## 1            3916.0                  813.7       0.2000
## 2            3823.0                  1547.5      2.4000
## 3            2785.4                  666.0       1.4000
## 4            2587.3                  406.0       0.8000
## 5            3312.8                  849.0       2.0000
## 6            2262.4                  161.9      -12.5135
##   Duration..weeks. Physical.Activity.Level Sleep.Quality Stress.Level
## 1                 1           Sedentary    Excellent      6
## 2                 6          Very Active  Excellent      6
## 3                 7           Sedentary     Good       3
## 4                 8           Sedentary     Fair       2
## 5                10          Lightly Active  Good       1
## 6                 9           Sedentary     Poor       6
##   Final.Weight..lbs.
## 1            228.6
## 2            167.8
## 3            144.2
## 4            146.3
## 5            157.5
## 6            140.4
```

Our first step is to build our full model, and determine whether any assumptions have been broken. Do note, that we chose to remove the variables “Current.Weight..lbs.” and “Final.Weight..lbs.” because our responding variable, “Weight.Change..lbs.” is just a result of subtracting the two variables, making weight change dependent on both current and final weight. To ensure that at least one of our predictors is significant, we test the hypothesis:

$$H_0 : \beta_1 = \beta_2 = \dots = \beta_i = 0 \text{ (i=1,2,...,p)}$$

$$H_a : \text{At least one } \beta_i \neq 0 \text{ (i=1,2,...,p)}$$

```
##
## Call:
## lm(formula = Weight.Change..lbs. ~ Age + factor(Gender) + BMR..Calories. +
## Daily.Calories.Consumed + Daily.Caloric.Surplus.Deficit +
```

```

## Duration..weeks. + factor(Physical.Activity.Level) + factor(Sleep.Quality) +
## factor(Stress.Level), data = weight)
##
## Residuals:
##      Min       1Q    Median     3Q    Max
## -16.5881 -1.4426  0.2319  2.1582  9.7510
##
## Coefficients:
##                               Estimate Std. Error t value
## (Intercept)                7.032845  4.940503  1.424
## Age                      -0.002834  0.036018 -0.079
## factor(Gender)M            0.263772  1.150845  0.229
## BMR..Calories.             -6.915519 10.273016 -0.673
## Daily.Calories.Consumed    6.913309 10.272867  0.673
## Daily.Caloric.Surplus.Deficit -6.911143 10.273082 -0.673
## Duration..weeks.           -0.395738  0.122177 -3.239
## factor(Physical.Activity.Level)Moderately Active -0.316972  1.247677 -0.254
## factor(Physical.Activity.Level)Sedentary          0.007914  1.327862  0.006
## factor(Physical.Activity.Level)Very Active        0.751174  1.520393  0.494
## factor(Sleep.Quality)Fair              1.544141  1.366646  1.130
## factor(Sleep.Quality)Good              2.109521  1.416478  1.489
## factor(Sleep.Quality)Poor             -7.411684  1.271290 -5.830
## factor(Stress.Level)2                 -0.153266  1.717603 -0.089
## factor(Stress.Level)3                 1.489659  1.714714  0.869
## factor(Stress.Level)4                 1.006707  1.974665  0.510
## factor(Stress.Level)5                 0.770360  1.798465  0.428
## factor(Stress.Level)6                 -0.849362  1.690039 -0.503
## factor(Stress.Level)7                 0.727956  1.848089  0.394
## factor(Stress.Level)8                 -11.058953 1.754401 -6.304
## factor(Stress.Level)9                 -9.822984  1.838339 -5.343
## Pr(>|t|)
## (Intercept)                  0.15853
## Age                         0.93748
## factor(Gender)M              0.81931
## BMR..Calories.               0.50280
## Daily.Calories.Consumed     0.50293
## Daily.Caloric.Surplus.Deficit 0.50307
## Duration..weeks.             0.00176 ***
## factor(Physical.Activity.Level)Moderately Active 0.80012
## factor(Physical.Activity.Level)Sedentary          0.99526
## factor(Physical.Activity.Level)Very Active        0.62263
## factor(Sleep.Quality)Fair              0.26195
## factor(Sleep.Quality)Good              0.14040
## factor(Sleep.Quality)Poor             1.15e-07 ***
## factor(Stress.Level)2                 0.92912
## factor(Stress.Level)3                 0.38762
## factor(Stress.Level)4                 0.61160
## factor(Stress.Level)5                 0.66957
## factor(Stress.Level)6                 0.61667
## factor(Stress.Level)7                 0.69472
## factor(Stress.Level)8                 1.55e-08 ***
## factor(Stress.Level)9                 8.58e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ',' 1

```

```

## 
## Residual standard error: 3.925 on 79 degrees of freedom
## Multiple R-squared:  0.7781, Adjusted R-squared:  0.7219
## F-statistic: 13.85 on 20 and 79 DF,  p-value: < 2.2e-16

```

Based on our output, we can see that at least one predictor has a p-value < 0.05, therefore we reject our null hypothesis and conclude that at least one predictor significantly affects weight change. Before determining which predictors to include in our model, we want to test for multicollinearity. This can be done using the VIF method.

```

##                                     GVIF Df GVIF^(1/(2*Df))
## Age                               1.244553e+00  1      1.115595
## factor(Gender)                   2.106715e+00  1      1.451453
## BMR..Calories.                  9.005157e+07  1     9489.550825
## Daily.Calories.Consumed        1.786538e+08  1    13366.144669
## Daily.Caloric.Surplus.Deficit  9.361072e+07  1    9675.263124
## Duration..weeks.                1.185115e+00  1      1.088630
## factor(Physical.Activity.Level) 4.324149e+00  3      1.276390
## factor(Sleep.Quality)          1.580526e+00  3      1.079279
## factor(Stress.Level)            3.048675e+00  8      1.072153

## 
## Call:
## imcdiag(mod = weight_model_full, method = "VIF")
## 
## 
##   VIF Multicollinearity Diagnostics
## 
##                                     VIF detection
## Age                               1.244600e+00  0
## factor(Gender)M                 2.106700e+00  0
## BMR..Calories.                  9.005157e+07  1
## Daily.Calories.Consumed        1.786538e+08  1
## Daily.Caloric.Surplus.Deficit  9.361071e+07  1
## Duration..weeks.                1.185100e+00  0
## factor(Physical.Activity.Level)Moderately Active 1.894200e+00  0
## factor(Physical.Activity.Level)Sedentary           1.898400e+00  0
## factor(Physical.Activity.Level)Very Active         2.656800e+00  0
## factor(Sleep.Quality)Fair       2.210900e+00  0
## factor(Sleep.Quality)Good       2.234400e+00  0
## factor(Sleep.Quality)Poor       2.471100e+00  0
## factor(Stress.Level)2           2.305200e+00  0
## factor(Stress.Level)3           2.297400e+00  0
## factor(Stress.Level)4           1.647400e+00  0
## factor(Stress.Level)5           2.055000e+00  0
## factor(Stress.Level)6           2.096500e+00  0
## factor(Stress.Level)7           1.994900e+00  0
## factor(Stress.Level)8           1.955600e+00  0
## factor(Stress.Level)9           1.796200e+00  0
## 
## Multicollinearity may be due to BMR..Calories. Daily.Calories.Consumed Daily.Caloric.Surplus.Deficit
## 
## 1 --> COLLINEARITY is detected by the test

```

```

## 0 --> COLLINEARITY is not detected by the test
##
## =====

```

From our output, we can see that three variables have a VIF score > 5, suggesting multicollinearity. These variables are “BMR..Calories.”, “Daily.Calories.Consumed”, and “Daily.Caloric.Surplus.Deficit”. Based on the definitions provided in the data set, “Daily.Caloric.Surplus.Deficit” is the difference between calories consumed and BMR, therefore we chose to keep “Daily.Caloric.Surplus.Deficit” and remove “BMR..Calories.”, and “Daily.Calories.Consumed”.

Following the removal of those variables, we will determine which predictors are significant using the hypothesis test:

$$H_0 : \beta_i = 0 \text{ (i=1,2,...,p)}$$

$$H_a : \beta_i \neq 0 \text{ (i=1,2,...,p)}$$

This can be achieved by looking at the individual t-tests from the summary of our adjusted model.

```

##
## Call:
## lm(formula = Weight.Change..lbs. ~ Daily.Caloric.Surplus.Deficit +
##     Age + factor(Gender) + Duration..weeks. + factor(Physical.Activity.Level) +
##     factor(Sleep.Quality) + factor(Stress.Level), data = weight)
##
## Residuals:
##      Min    1Q Median    3Q   Max 
## -17.221 -1.637  0.325  2.286  9.717 
##
## Coefficients:
##                               Estimate Std. Error t value
## (Intercept)                 1.098664  3.061275  0.359
## Daily.Caloric.Surplus.Deficit 0.002679  0.001871  1.432
## Age                         0.003592  0.035076  0.102
## factor(Gender)M              -0.768568  0.922416 -0.833
## Duration..weeks.             -0.402046  0.121077 -3.321
## factor(Physical.Activity.Level)Moderately Active -0.618016  1.234619 -0.501
## factor(Physical.Activity.Level)Sedentary          -0.016837  1.330946 -0.013
## factor(Physical.Activity.Level)Very Active        0.360470  1.505470  0.239
## factor(Sleep.Quality)Fair           1.587961  1.365649  1.163
## factor(Sleep.Quality)Good          2.256215  1.418302  1.591
## factor(Sleep.Quality)Poor         -7.353535  1.273937 -5.772
## factor(Stress.Level)2            0.382922  1.680623  0.228
## factor(Stress.Level)3            1.499079  1.707093  0.878
## factor(Stress.Level)4            1.272710  1.961313  0.649
## factor(Stress.Level)5            1.420505  1.757602  0.808
## factor(Stress.Level)6            -0.479314  1.665477 -0.288
## factor(Stress.Level)7            0.924742  1.821196  0.508
## factor(Stress.Level)8            -10.782512 1.735068 -6.214
## factor(Stress.Level)9            -9.275712  1.804889 -5.139
##
## Pr(>|t|) 
## (Intercept) 0.72061
## Daily.Caloric.Surplus.Deficit 0.15610
## Age          0.91868
## factor(Gender)M 0.40718
## Duration..weeks. 0.00135 **
```

```

## factor(Physical.Activity.Level)Moderately Active 0.61803
## factor(Physical.Activity.Level)Sedentary         0.98994
## factor(Physical.Activity.Level)Very Active      0.81137
## factor(Sleep.Quality)Fair                      0.24833
## factor(Sleep.Quality)Good                      0.11555
## factor(Sleep.Quality)Poor                     1.39e-07 ***
## factor(Stress.Level)2                         0.82034
## factor(Stress.Level)3                         0.38246
## factor(Stress.Level)4                         0.51823
## factor(Stress.Level)5                         0.42134
## factor(Stress.Level)6                         0.77424
## factor(Stress.Level)7                         0.61300
## factor(Stress.Level)8                         2.11e-08 ***
## factor(Stress.Level)9                         1.87e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.939 on 81 degrees of freedom
## Multiple R-squared:  0.7709, Adjusted R-squared:  0.72
## F-statistic: 15.14 on 18 and 81 DF,  p-value: < 2.2e-16

```

Based on this output, we can see there are 4 instances where the null hypothesis is rejected ($p\text{-value} < 0.05$), implying that these variables are significant and should be included in our model. These variables are:

- Duration..weeks.
- factor(Sleep.Quality)Poor
- factor(Stress.Level)8
- factor(Stress.Level)9

Despite the fact that only some of the dummies are significant, for the categories “Sleep.Quality” and “Stress.Level”, we must include the full variable in our model. Therefore our model will include the predictors Sleep.Quality, Stress.Level and Duration..weeks.

NOTE An attempt was made to use stepwise regression to determine which variables were best to keep, with the code for that being included in the appendix. Upon running the stepwise, it was determined that a 4th variable, Daily,Caloric.Surplus.Deficit should be included. In the end we chose not to include this variable for a couple reasons. First, our categorical variables had many levels, in which case it is often better to just manually select significant predictors using t-tests from our summary table. When performing this manual selection, we found the variable the stepwise selection wanted to include, had a p-value of 0.15610, and a coefficient of 0.002679. This implied the variable had no significant effect. Based on this, and to avoid overfitting and complicating our model, we chose to not include this variable

```

##
## Call:
## lm(formula = Weight.Change..lbs. ~ factor(Sleep.Quality) + Duration..weeks. +
##     factor(Stress.Level), data = weight)
##
## Residuals:
##      Min        1Q        Median       3Q        Max
## -18.2553   -1.9040    0.2041    2.3282   12.3789
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.7718     1.7173    2.196   0.0307 *

```

```

## factor(Sleep.Quality)Fair  1.4083   1.3413   1.050   0.2967
## factor(Sleep.Quality)Good  1.8466   1.3895   1.329   0.1873
## factor(Sleep.Quality)Poor -7.4041   1.2362  -5.990  4.62e-08 ***
## Duration..weeks.        -0.3650   0.1204  -3.032   0.0032 **
## factor(Stress.Level)2    -0.3405   1.6188  -0.210   0.8339
## factor(Stress.Level)3    0.4781   1.6168   0.296   0.7682
## factor(Stress.Level)4    0.6451   1.9312   0.334   0.7391
## factor(Stress.Level)5    0.9831   1.7142   0.573   0.5678
## factor(Stress.Level)6    -0.3863   1.6471  -0.235   0.8151
## factor(Stress.Level)7    0.4839   1.7548   0.276   0.7834
## factor(Stress.Level)8    -11.3423  1.6987  -6.677  2.19e-09 ***
## factor(Stress.Level)9    -9.7760   1.7911  -5.458  4.48e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.964 on 87 degrees of freedom
## Multiple R-squared:  0.7507, Adjusted R-squared:  0.7164
## F-statistic: 21.84 on 12 and 87 DF,  p-value: < 2.2e-16

```

We can see that all predictors are now significant. Before moving on to interaction terms, we should check again for multicollinearity, just to be certain:

```

##                               GVIF Df GVIF^(1/(2*Df))
## factor(Sleep.Quality) 1.303780  3      1.045203
## Duration..weeks.      1.128093  1      1.062117
## factor(Stress.Level)  1.308020  8      1.016924

##
## Call:
## imcdiag(mod = weight_model_adj2, method = "VIF")
##
##
## VIF Multicollinearity Diagnostics
##
##                               VIF detection
## factor(Sleep.Quality)Fair 2.0880      0
## factor(Sleep.Quality)Good 2.1080      0
## factor(Sleep.Quality)Poor 2.2907      0
## Duration..weeks.        1.1281      0
## factor(Stress.Level)2   2.0076      0
## factor(Stress.Level)3   2.0025      0
## factor(Stress.Level)4   1.5448      0
## factor(Stress.Level)5   1.8305      0
## factor(Stress.Level)6   1.9523      0
## factor(Stress.Level)7   1.7634      0
## factor(Stress.Level)8   1.7974      0
## factor(Stress.Level)9   1.6718      0
##
## NOTE: VIF Method Failed to detect multicollinearity
##
##
## 0 --> COLLINEARITY is not detected by the test
##
## =====

```

As we can see from our output, there are no issues with multicollinearity.

Now we can begin checking for interaction terms. We can start by testing all interaction terms for our model using the hypothesis test:

$$H_0 : \beta_{interaction} = 0 \text{ (i=1,2,...,p)}$$

$$H_a : \beta_{interaction} \neq 0 \text{ (i=1,2,...,p)}$$

```
##  
## Call:  
## lm(formula = Weight.Change..lbs. ~ (factor(Sleep.Quality) + Duration..weeks. +  
##       factor(Stress.Level))^2, data = weight)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -10.0955 -0.5464  0.0000  0.9004  9.1887  
##  
## Coefficients: (2 not defined because of singularities)  
##                                     Estimate Std. Error t value  
## (Intercept)                      0.173136  3.385089  0.051  
## factor(Sleep.Quality)Fair        0.630967  4.305253  0.147  
## factor(Sleep.Quality)Good       -0.515159  4.272899 -0.121  
## factor(Sleep.Quality)Poor        1.111739  4.146862  0.268  
## Duration..weeks.                 0.242288  0.465477  0.521  
## factor(Stress.Level)2           0.403610  4.337921  0.093  
## factor(Stress.Level)3           -1.956105  3.862834 -0.506  
## factor(Stress.Level)4           -1.834648  3.855770 -0.476  
## factor(Stress.Level)5           -4.257771  4.621632 -0.921  
## factor(Stress.Level)6           0.391914  3.849036  0.102  
## factor(Stress.Level)7           -2.320497  4.601681 -0.504  
## factor(Stress.Level)8           1.171047  5.307511  0.221  
## factor(Stress.Level)9           -3.418483  3.543061 -0.965  
## factor(Sleep.Quality)Fair:Duration..weeks. -0.009411  0.393084 -0.024  
## factor(Sleep.Quality)Good:Duration..weeks.  0.045802  0.439363  0.104  
## factor(Sleep.Quality)Poor:Duration..weeks. -1.208733  0.442719 -2.730  
## factor(Sleep.Quality)Fair:factor(Stress.Level)2 -1.610491  4.861653 -0.331  
## factor(Sleep.Quality)Good:factor(Stress.Level)2  0.638105  4.726767  0.135  
## factor(Sleep.Quality)Poor:factor(Stress.Level)2 -0.271656  5.459211 -0.050  
## factor(Sleep.Quality)Fair:factor(Stress.Level)3 -2.208279  6.084939 -0.363  
## factor(Sleep.Quality)Good:factor(Stress.Level)3 -3.121900  5.144181 -0.607  
## factor(Sleep.Quality)Poor:factor(Stress.Level)3 -0.806757  4.378219 -0.184  
## factor(Sleep.Quality)Fair:factor(Stress.Level)4 -1.857250  7.660308 -0.242  
## factor(Sleep.Quality)Good:factor(Stress.Level)4 -0.499270  5.817921 -0.086  
## factor(Sleep.Quality)Poor:factor(Stress.Level)4       NA       NA       NA  
## factor(Sleep.Quality)Fair:factor(Stress.Level)5 -0.155156  5.133104 -0.030  
## factor(Sleep.Quality)Good:factor(Stress.Level)5 -4.340225  6.172293 -0.703  
## factor(Sleep.Quality)Poor:factor(Stress.Level)5 -0.442804  4.533601 -0.098  
## factor(Sleep.Quality)Fair:factor(Stress.Level)6 -0.991839  4.720897 -0.210  
## factor(Sleep.Quality)Good:factor(Stress.Level)6 -2.044183  5.054225 -0.404  
## factor(Sleep.Quality)Poor:factor(Stress.Level)6 -3.334619  4.372421 -0.763  
## factor(Sleep.Quality)Fair:factor(Stress.Level)7  0.052776  5.351148  0.010  
## factor(Sleep.Quality)Good:factor(Stress.Level)7  1.618175  6.488306  0.249  
## factor(Sleep.Quality)Poor:factor(Stress.Level)7  1.392698  5.095646  0.273  
## factor(Sleep.Quality)Fair:factor(Stress.Level)8 -4.308983  5.585545 -0.771  
## factor(Sleep.Quality)Good:factor(Stress.Level)8 -3.359882  5.572127 -0.603
```

```

## factor(Sleep.Quality)Poor:factor(Stress.Level)8 -4.244128 5.215951 -0.814
## factor(Sleep.Quality)Fair:factor(Stress.Level)9  3.681779 3.459445  1.064
## factor(Sleep.Quality)Good:factor(Stress.Level)9  4.692112 3.573977  1.313
## factor(Sleep.Quality)Poor:factor(Stress.Level)9      NA      NA      NA
## Duration..weeks.:factor(Stress.Level)2          -0.043603 0.388747 -0.112
## Duration..weeks.:factor(Stress.Level)3          0.472603 0.462794  1.021
## Duration..weeks.:factor(Stress.Level)4          0.335902 0.851910  0.394
## Duration..weeks.:factor(Stress.Level)5          0.924639 0.550379  1.680
## Duration..weeks.:factor(Stress.Level)6          0.091339 0.400593  0.228
## Duration..weeks.:factor(Stress.Level)7          0.198939 0.543024  0.366
## Duration..weeks.:factor(Stress.Level)8          -1.079441 0.518387 -2.082
## Duration..weeks.:factor(Stress.Level)9          -1.245371 0.377083 -3.303
##
## Pr(>|t|)
## (Intercept)          0.95940
## factor(Sleep.Quality)Fair 0.88403
## factor(Sleep.Quality)Good 0.90448
## factor(Sleep.Quality)Poor 0.78965
## Duration..weeks.       0.60483
## factor(Stress.Level)2   0.92621
## factor(Stress.Level)3   0.61464
## factor(Stress.Level)4   0.63612
## factor(Stress.Level)5   0.36101
## factor(Stress.Level)6   0.91928
## factor(Stress.Level)7   0.61612
## factor(Stress.Level)8   0.82621
## factor(Stress.Level)9   0.33893
## factor(Sleep.Quality)Fair:Duration..weeks.    0.98099
## factor(Sleep.Quality)Good:Duration..weeks.    0.91736
## factor(Sleep.Quality)Poor:Duration..weeks.   0.00853 **
## factor(Sleep.Quality)Fair:factor(Stress.Level)2 0.74173
## factor(Sleep.Quality)Good:factor(Stress.Level)2 0.89312
## factor(Sleep.Quality)Poor:factor(Stress.Level)2 0.96050
## factor(Sleep.Quality)Fair:factor(Stress.Level)3 0.71809
## factor(Sleep.Quality)Good:factor(Stress.Level)3 0.54647
## factor(Sleep.Quality)Poor:factor(Stress.Level)3 0.85450
## factor(Sleep.Quality)Fair:factor(Stress.Level)4 0.80935
## factor(Sleep.Quality)Good:factor(Stress.Level)4 0.93193
## factor(Sleep.Quality)Poor:factor(Stress.Level)4      NA
## factor(Sleep.Quality)Fair:factor(Stress.Level)5 0.97600
## factor(Sleep.Quality)Good:factor(Stress.Level)5 0.48497
## factor(Sleep.Quality)Poor:factor(Stress.Level)5 0.92255
## factor(Sleep.Quality)Fair:factor(Stress.Level)6 0.83438
## factor(Sleep.Quality)Good:factor(Stress.Level)6 0.68748
## factor(Sleep.Quality)Poor:factor(Stress.Level)6 0.44899
## factor(Sleep.Quality)Fair:factor(Stress.Level)7 0.99217
## factor(Sleep.Quality)Good:factor(Stress.Level)7 0.80400
## factor(Sleep.Quality)Poor:factor(Stress.Level)7 0.78566
## factor(Sleep.Quality)Fair:factor(Stress.Level)8 0.44380
## factor(Sleep.Quality)Good:factor(Stress.Level)8 0.54905
## factor(Sleep.Quality)Poor:factor(Stress.Level)8 0.41940
## factor(Sleep.Quality)Fair:factor(Stress.Level)9 0.29194
## factor(Sleep.Quality)Good:factor(Stress.Level)9 0.19478
## factor(Sleep.Quality)Poor:factor(Stress.Level)9      NA
## Duration..weeks.:factor(Stress.Level)2          0.91111

```

```

## Duration..weeks.:factor(Stress.Level)3          0.31172
## Duration..weeks.:factor(Stress.Level)4          0.69492
## Duration..weeks.:factor(Stress.Level)5          0.09873 .
## Duration..weeks.:factor(Stress.Level)6          0.82050
## Duration..weeks.:factor(Stress.Level)7          0.71553
## Duration..weeks.:factor(Stress.Level)8          0.04207 *
## Duration..weeks.:factor(Stress.Level)9          0.00170 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.084 on 54 degrees of freedom
## Multiple R-squared:  0.9064, Adjusted R-squared:  0.8284
## F-statistic: 11.62 on 45 and 54 DF,  p-value: 7.362e-16

```

Based on our output, we can see the significant variables (with a p-value <0.05) are “factor(Sleep.Quality)Poor:Duration..weeks.”, “Duration..weeks.:factor(Stress.Level)8”, “Duration..weeks.:factor(Stress.Level)9”. Although none of the initial predictors are considered significant, they are all parts of significant interaction terms, and must therefore be included in the model, therefore our interaction model would include ‘Sleep.Quality’, ‘Duration..weeks.’, ‘Stress.Level’, ‘Sleep.Quality’:‘Duration..weeks.’, and ‘Duration..weeks.’:‘Stress.Level’.

```

##
## Call:
## lm(formula = Weight.Change..lbs. ~ factor(Sleep.Quality) + Duration..weeks. +
##      factor(Stress.Level) + factor(Sleep.Quality) * Duration..weeks. +
##      Duration..weeks. * factor(Stress.Level), data = weight)
##
## Residuals:
##    Min      1Q      Median      3Q      Max
## -11.2934 -0.7740   0.0613   0.9565   8.9577
##
## Coefficients:
##                               Estimate Std. Error t value
## (Intercept)                 0.846572  2.428365  0.349
## factor(Sleep.Quality)Fair   -0.060956  2.125053 -0.029
## factor(Sleep.Quality)Good   -0.731905  2.427593 -0.301
## factor(Sleep.Quality)Poor   -0.876532  2.005304 -0.437
## Duration..weeks.              0.266336  0.345765  0.770
## factor(Stress.Level)2        -0.053039  2.626820 -0.020
## factor(Stress.Level)3        -1.509118  2.582405 -0.584
## factor(Stress.Level)4        -0.529393  3.095647 -0.171
## factor(Stress.Level)5        -2.710051  2.845423 -0.952
## factor(Stress.Level)6        -0.969032  2.490558 -0.389
## factor(Stress.Level)7        -2.717472  3.578434 -0.759
## factor(Stress.Level)8        -1.657444  3.165466 -0.524
## factor(Stress.Level)9        -0.269294  2.631239 -0.102
## factor(Sleep.Quality)Fair:Duration..weeks. -0.051620  0.307122 -0.168
## factor(Sleep.Quality)Good:Duration..weeks. -0.001166  0.329358 -0.004
## factor(Sleep.Quality)Poor:Duration..weeks. -1.092225  0.317033 -3.445
## Duration..weeks.:factor(Stress.Level)2     -0.056465  0.326628 -0.173
## Duration..weeks.:factor(Stress.Level)3     0.222208  0.324192  0.685
## Duration..weeks.:factor(Stress.Level)4     0.058787  0.396866  0.148
## Duration..weeks.:factor(Stress.Level)5     0.568176  0.397329  1.430
## Duration..weeks.:factor(Stress.Level)6     0.042906  0.318948  0.135
## Duration..weeks.:factor(Stress.Level)7     0.312677  0.395000  0.792

```

```

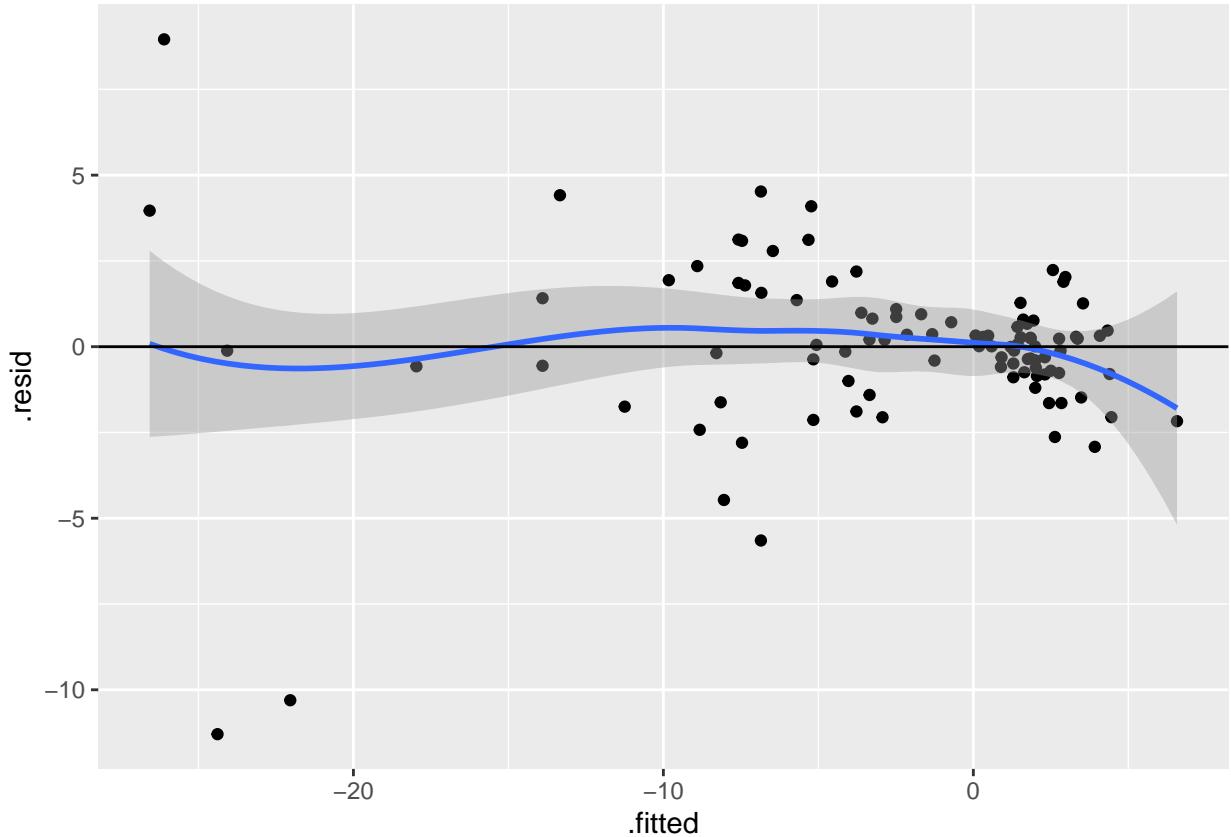
## Duration..weeks.:factor(Stress.Level)8      -1.209145   0.398373  -3.035
## Duration..weeks.:factor(Stress.Level)9      -1.363695   0.326722  -4.174
##
## (Intercept)                                Pr(>|t|)
## factor(Sleep.Quality)Fair                  0.728340
## factor(Sleep.Quality)Good                  0.977191
## factor(Sleep.Quality)Poor                 0.763861
## factor(Sleep.Quality)Poor                 0.663273
## Duration..weeks.                           0.443521
## factor(Stress.Level)2                     0.983944
## factor(Stress.Level)3                     0.560692
## factor(Stress.Level)4                     0.864669
## factor(Stress.Level)5                     0.343901
## factor(Stress.Level)6                     0.698303
## factor(Stress.Level)7                     0.449961
## factor(Stress.Level)8                     0.602079
## factor(Stress.Level)9                     0.918752
## factor(Sleep.Quality)Fair:Duration..weeks. 0.866971
## factor(Sleep.Quality)Good:Duration..weeks. 0.997185
## factor(Sleep.Quality)Poor:Duration..weeks. 0.000932 ***
## Duration..weeks.:factor(Stress.Level)2       0.863210
## Duration..weeks.:factor(Stress.Level)3       0.495164
## Duration..weeks.:factor(Stress.Level)4       0.882633
## Duration..weeks.:factor(Stress.Level)5       0.156819
## Duration..weeks.:factor(Stress.Level)6       0.893345
## Duration..weeks.:factor(Stress.Level)7       0.431065
## Duration..weeks.:factor(Stress.Level)8       0.003290 **
## Duration..weeks.:factor(Stress.Level)9       7.89e-05 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.822 on 76 degrees of freedom
## Multiple R-squared:  0.8897, Adjusted R-squared:  0.8563
## F-statistic: 26.65 on 23 and 76 DF,  p-value: < 2.2e-16

```

Our final model, before testing for the rest of our assumptions, will include the following predictors and interaction terms: Sleep.Quality, Duration..weeks., Stress.Level, Sleep.Quality:Duration..weeks., and Duration..weeks.:Stress.Level.

Based on this model, we can now test the rest of our assumptions. We will start with testing for linearity by inspecting our residual plots.

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Based on our residuals, there appears to be a pattern, with most data points on the right side, as well as the presence of some funnel shape. We can try to correct this by adding some polynomial terms, or transforming the data. We will start with polynomial terms.

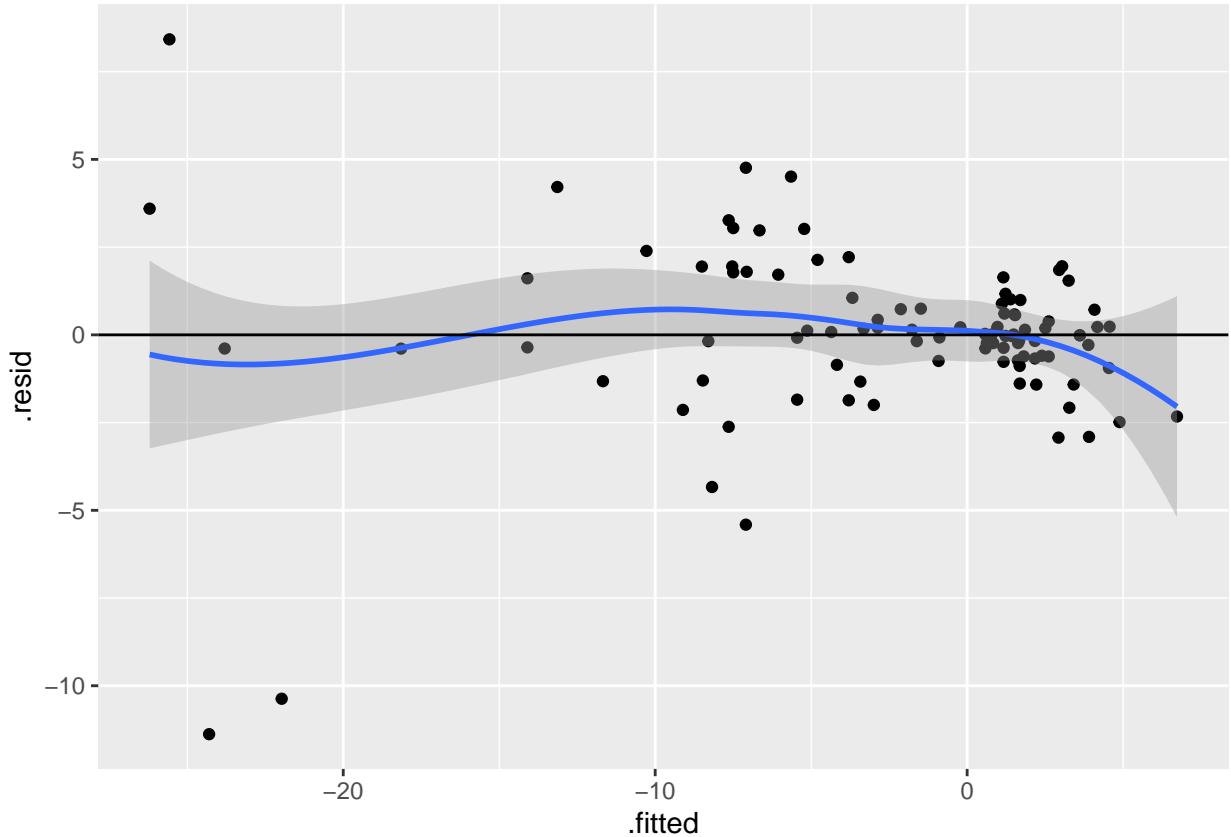
```
##
## Call:
## lm(formula = Weight.Change..lbs. ~ factor(Sleep.Quality) + Duration..weeks. +
##     I(Duration..weeks.^2) + factor(Stress.Level) + factor(Sleep.Quality) *
##     Duration..weeks. + Duration..weeks. * factor(Stress.Level),
##     data = weight)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -11.3791  -0.7311  -0.0065  0.9976   8.4209
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.66345   2.54941   0.652  0.516083
## factor(Sleep.Quality)Fair    0.13760   2.13220   0.065  0.948715
## factor(Sleep.Quality)Good   -0.74349   2.42611  -0.306  0.760108
## factor(Sleep.Quality)Poor   -0.87654   2.00406  -0.437  0.663091
## Duration..weeks.             -0.09704   0.48997  -0.198  0.843543
## I(Duration..weeks.^2)         0.02828   0.02704   1.046  0.298892
## factor(Stress.Level)2         0.11348   2.63001   0.043  0.965698
## factor(Stress.Level)3         -1.46360   2.58117  -0.567  0.572388
## factor(Stress.Level)4         -0.59995   3.09446  -0.194  0.846794
## factor(Stress.Level)5         -2.49747   2.85091  -0.876  0.383813
```

```

## factor(Stress.Level)6          -1.06770  2.49080 -0.429 0.669402
## factor(Stress.Level)7          -2.34457  3.59394 -0.652 0.516159
## factor(Stress.Level)8          -1.37165  3.17528 -0.432 0.666996
## factor(Stress.Level)9          -0.38409  2.63190 -0.146 0.884363
## factor(Sleep.Quality)Fair:Duration..weeks. -0.09137  0.30928 -0.295 0.768475
## factor(Sleep.Quality)Good:Duration..weeks. -0.01614  0.32946 -0.049 0.961055
## factor(Sleep.Quality)Poor:Duration..weeks. -1.09414  0.31684 -3.453 0.000914
## Duration..weeks.:factor(Stress.Level)2      -0.06645  0.32656 -0.203 0.839304
## Duration..weeks.:factor(Stress.Level)3      0.22627  0.32401  0.698 0.487124
## Duration..weeks.:factor(Stress.Level)4      0.08923  0.39769  0.224 0.823079
## Duration..weeks.:factor(Stress.Level)5      0.55714  0.39722  1.403 0.164866
## Duration..weeks.:factor(Stress.Level)6      0.05918  0.31913  0.185 0.853376
## Duration..weeks.:factor(Stress.Level)7      0.27307  0.39657  0.689 0.493213
## Duration..weeks.:factor(Stress.Level)8      -1.23041  0.39864 -3.086 0.002837
## Duration..weeks.:factor(Stress.Level)9      -1.36556  0.32652 -4.182 7.75e-05
##
## (Intercept)
## factor(Sleep.Quality)Fair
## factor(Sleep.Quality)Good
## factor(Sleep.Quality)Poor
## Duration..weeks.
## I(Duration..weeks.^2)
## factor(Stress.Level)2
## factor(Stress.Level)3
## factor(Stress.Level)4
## factor(Stress.Level)5
## factor(Stress.Level)6
## factor(Stress.Level)7
## factor(Stress.Level)8
## factor(Stress.Level)9
## factor(Sleep.Quality)Fair:Duration..weeks.
## factor(Sleep.Quality)Good:Duration..weeks.
## factor(Sleep.Quality)Poor:Duration..weeks. ***
## Duration..weeks.:factor(Stress.Level)2
## Duration..weeks.:factor(Stress.Level)3
## Duration..weeks.:factor(Stress.Level)4
## Duration..weeks.:factor(Stress.Level)5
## Duration..weeks.:factor(Stress.Level)6
## Duration..weeks.:factor(Stress.Level)7
## Duration..weeks.:factor(Stress.Level)8      **
## Duration..weeks.:factor(Stress.Level)9      ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.82 on 75 degrees of freedom
## Multiple R-squared:  0.8913, Adjusted R-squared:  0.8565
## F-statistic: 25.62 on 24 and 75 DF,  p-value: < 2.2e-16

## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'

```



Adding a polynomial term appears to not be the solution, as not only does the residual plot appear the same, but the adjust R-squared barely increases. There is no point to preventing our interpretation for such a small increase.

Next we will try performing a log transformation on Duration..weeks.

```
##
## Call:
## lm(formula = Weight.Change..lbs. ~ factor(Sleep.Quality) + log(Duration..weeks.) +
##     factor(Stress.Level) + factor(Sleep.Quality) * log(Duration..weeks.) +
##     log(Duration..weeks.) * factor(Stress.Level), data = weight)
##
## Residuals:
##      Min        1Q        Median       3Q        Max 
## -12.5056  -0.9419   0.2044   1.2345   6.3533 
##
## Coefficients:
##             Estimate Std. Error t value
## (Intercept) 0.27355  3.42430  0.080
## factor(Sleep.Quality)Fair 0.11340  2.82176  0.040
## factor(Sleep.Quality)Good -2.57483  3.44092 -0.748
## factor(Sleep.Quality)Poor  1.58927  2.86099  0.555
## log(Duration..weeks.)    1.28194  1.93076  0.664
## factor(Stress.Level)2    0.81976  3.55617  0.231
## factor(Stress.Level)3   -2.72618  3.42733 -0.795
## factor(Stress.Level)4   -2.26130  3.85645 -0.586
## factor(Stress.Level)5   -4.49656  4.26194 -1.055
```

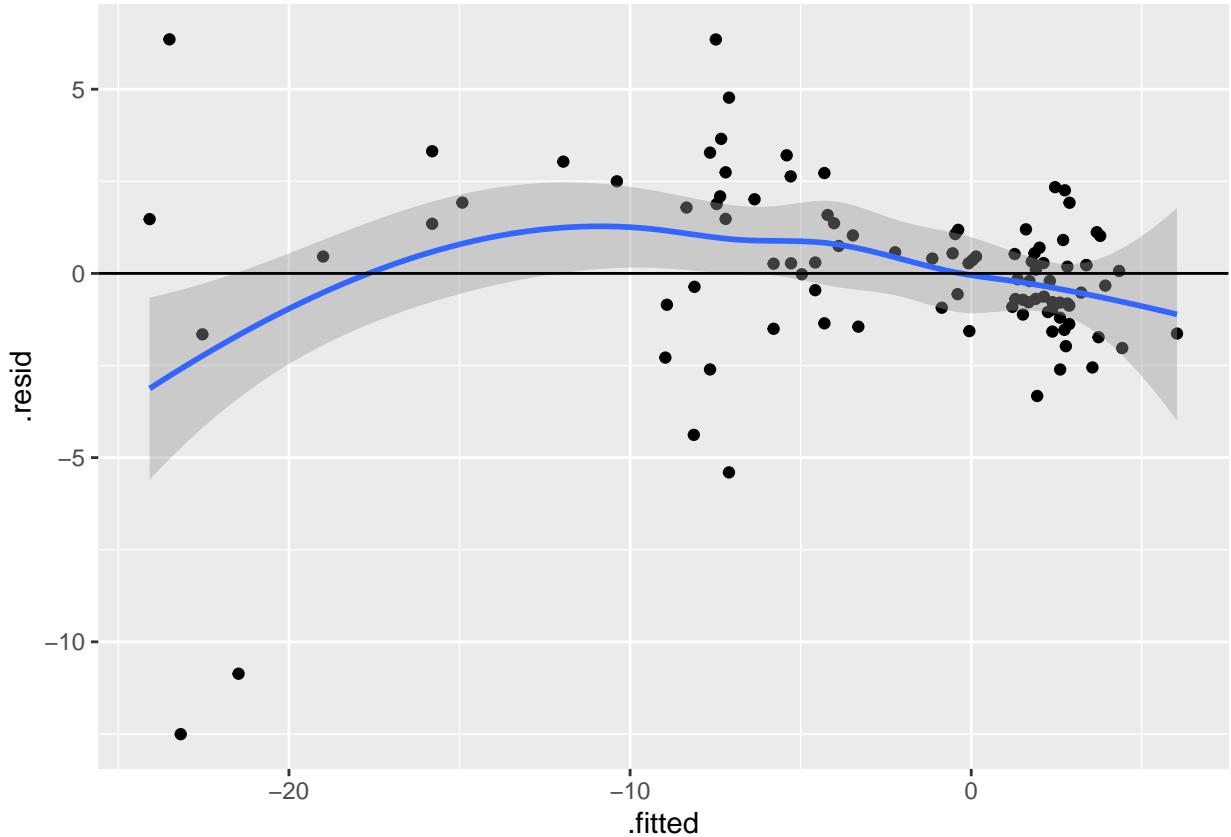
```

## factor(Stress.Level)6           -0.34971   3.35196  -0.104
## factor(Stress.Level)7           -2.58341   5.06097  -0.510
## factor(Stress.Level)8           2.24593   4.16139   0.540
## factor(Stress.Level)9           0.06974   3.32780   0.021
## factor(Sleep.Quality)Fair:log(Duration..weeks.) -0.08017   1.58629  -0.051
## factor(Sleep.Quality)Good:log(Duration..weeks.)  0.94134   1.84057   0.511
## factor(Sleep.Quality)Poor:log(Duration..weeks.) -5.61445   1.68822  -3.326
## log(Duration..weeks.):factor(Stress.Level)2      -0.63803   1.84765  -0.345
## log(Duration..weeks.):factor(Stress.Level)3      1.58105   1.79470   0.881
## log(Duration..weeks.):factor(Stress.Level)4      1.03392   2.02085   0.512
## log(Duration..weeks.):factor(Stress.Level)5      3.12719   2.36324   1.323
## log(Duration..weeks.):factor(Stress.Level)6      -0.05704   1.79414  -0.032
## log(Duration..weeks.):factor(Stress.Level)7      1.26398   2.41255   0.524
## log(Duration..weeks.):factor(Stress.Level)8      -6.77947   2.11771  -3.201
## log(Duration..weeks.):factor(Stress.Level)9      -6.13711   1.75701  -3.493
##
Pr(>|t|)

## (Intercept)          0.93654
## factor(Sleep.Quality)Fair 0.96805
## factor(Sleep.Quality)Good 0.45659
## factor(Sleep.Quality)Poor 0.58019
## log(Duration..weeks.)    0.50873
## factor(Stress.Level)2    0.81831
## factor(Stress.Level)3    0.42884
## factor(Stress.Level)4    0.55936
## factor(Stress.Level)5    0.29475
## factor(Stress.Level)6    0.91718
## factor(Stress.Level)7    0.61121
## factor(Stress.Level)8    0.59098
## factor(Stress.Level)9    0.98333
## factor(Sleep.Quality)Fair:log(Duration..weeks.) 0.95983
## factor(Sleep.Quality)Good:log(Duration..weeks.)  0.61053
## factor(Sleep.Quality)Poor:log(Duration..weeks.) 0.00136  **
## log(Duration..weeks.):factor(Stress.Level)2      0.73081
## log(Duration..weeks.):factor(Stress.Level)3      0.38112
## log(Duration..weeks.):factor(Stress.Level)4      0.61040
## log(Duration..weeks.):factor(Stress.Level)5      0.18971
## log(Duration..weeks.):factor(Stress.Level)6      0.97472
## log(Duration..weeks.):factor(Stress.Level)7      0.60186
## log(Duration..weeks.):factor(Stress.Level)8      0.00200  **
## log(Duration..weeks.):factor(Stress.Level)9      0.00080  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.917 on 76 degrees of freedom
## Multiple R-squared:  0.8821, Adjusted R-squared:  0.8464
## F-statistic: 24.72 on 23 and 76 DF,  p-value: < 2.2e-16

## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'

```



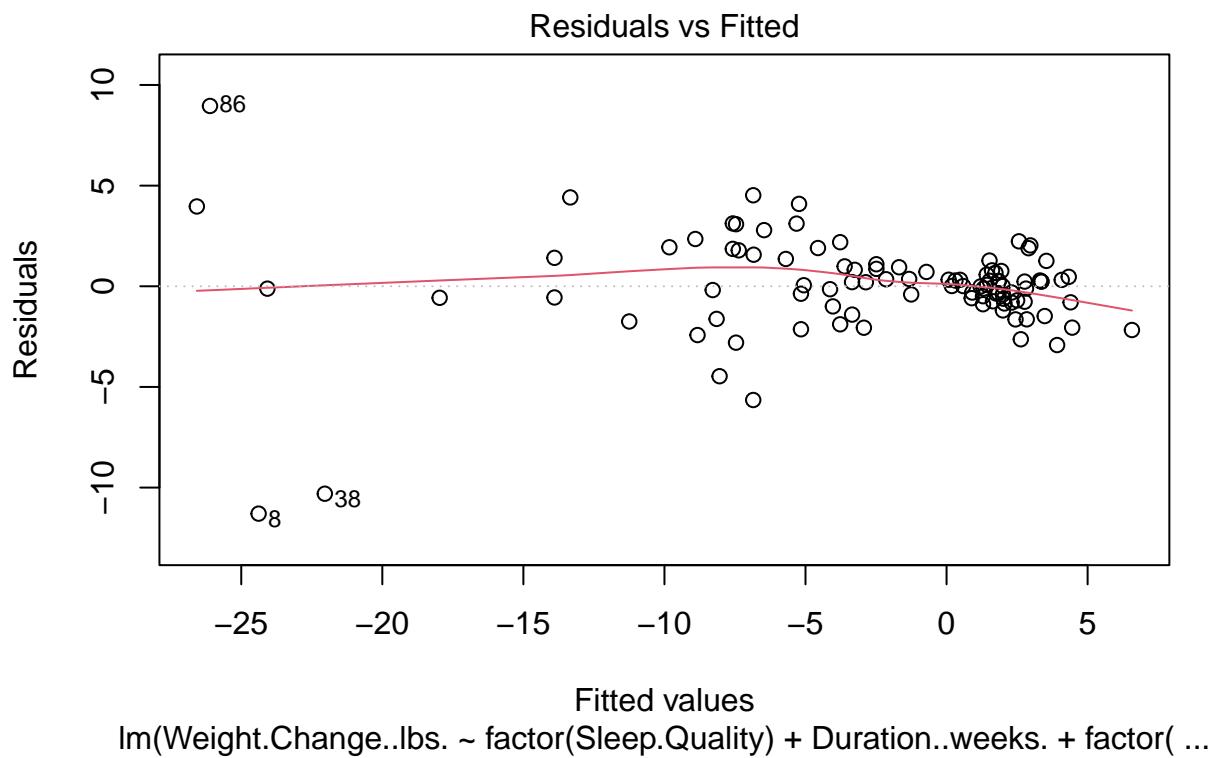
Transforming Duration..weeks. using a log transformation appears to not correct the linearity problem either. In fact the residual plot now looks worse than before.

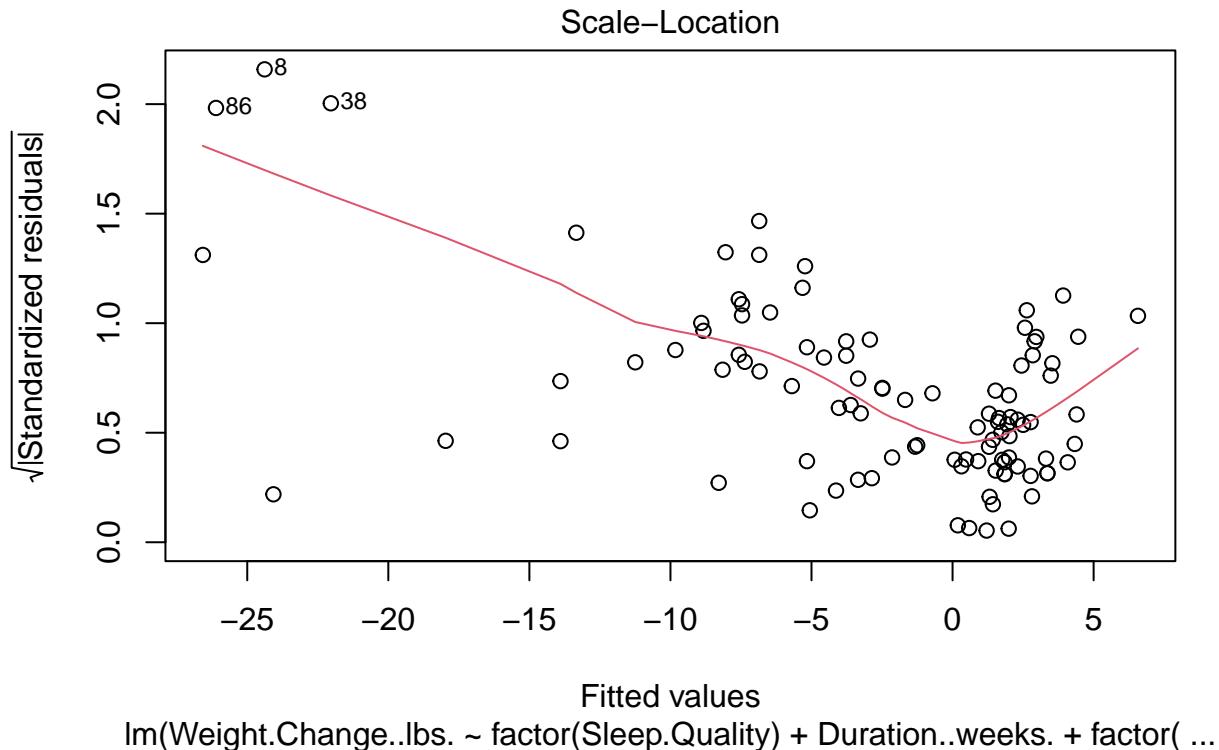
Based on these attempts, we may need to add more complex polynomial or transformations to our data. We may also need to attempt some other model types, but for the sake of this class we will move forward using our model before we attempting correcting linearity, and simply state that our model fails the linearity assumption.

The next assumption we will look at is independence. As our responding variable, Weight.Change..lbs. is not considered time-series data we can conclude this assumption has been passed.

To test for homoscedasticity we can look at our residual plots, scale-location plots, and run a Breusch-Pagan Test, using the following hypothesis:

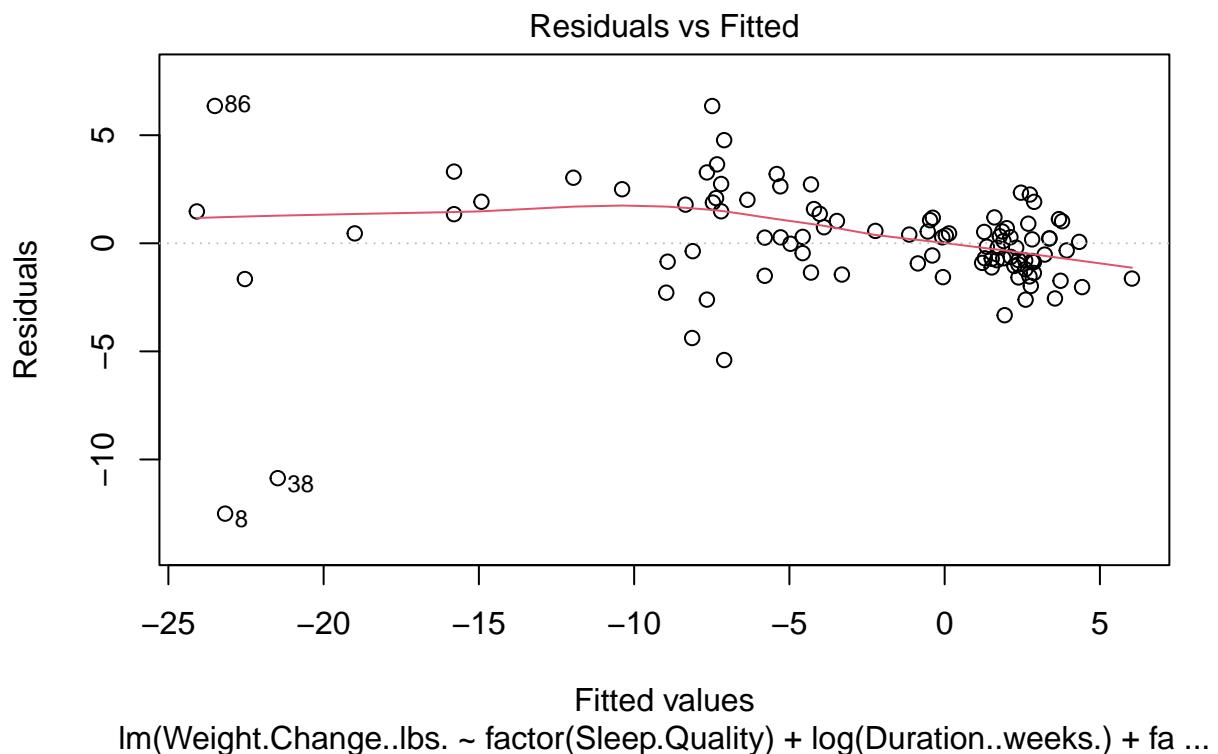
$$H_0 : \sigma_1^2 = \sigma_2^2 = \dots = \sigma_n^2 \text{ (heteroscedasticity is not present)} \\ H_a : \text{at least one } \sigma_i^2 \text{ is different from the others } i = 1, 2, \dots, n \text{ (heteroscedasticity is present)}$$

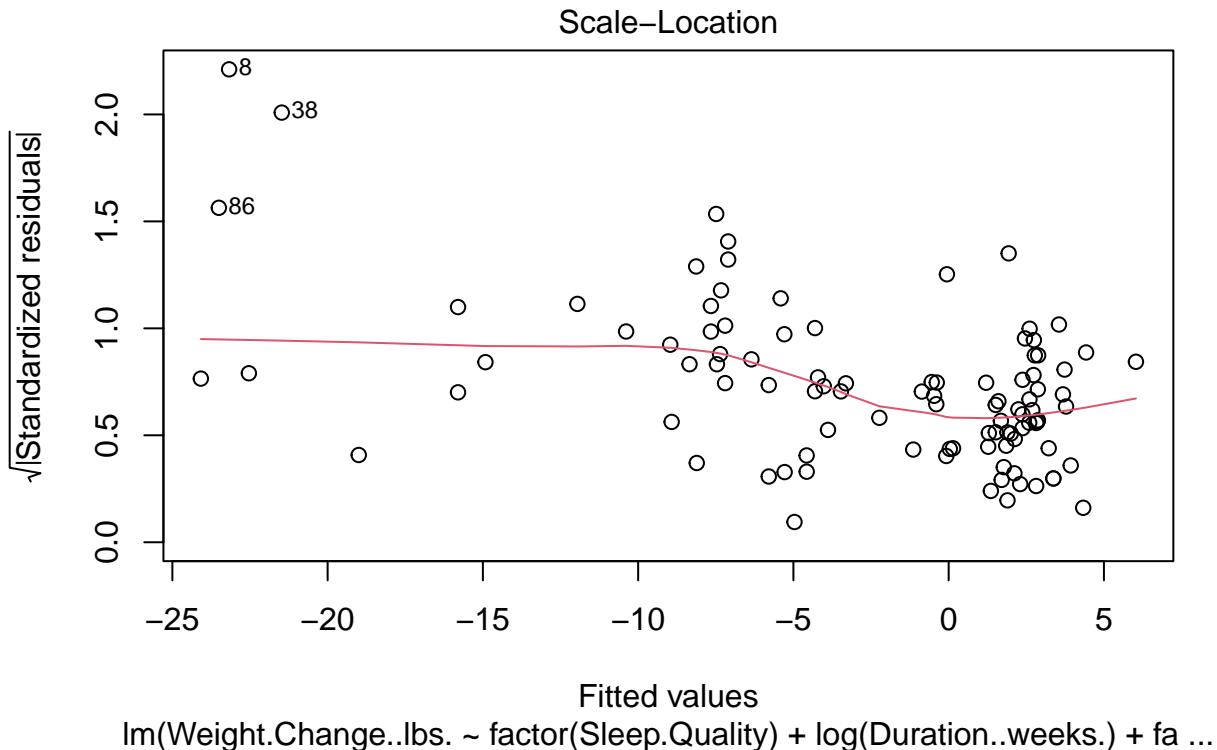




```
##  
## studentized Breusch-Pagan test  
##  
## data: weight_model_int_final  
## BP = 44.674, df = 23, p-value = 0.004356
```

Based on our plots and the fact that our p-value for our Breusch-Pagan Test (0.004356) is < 0.05 , we can reject our null hypothesis, implying the presence of heteroscedasticity. To try and correct this, we can attempt a log transformation.



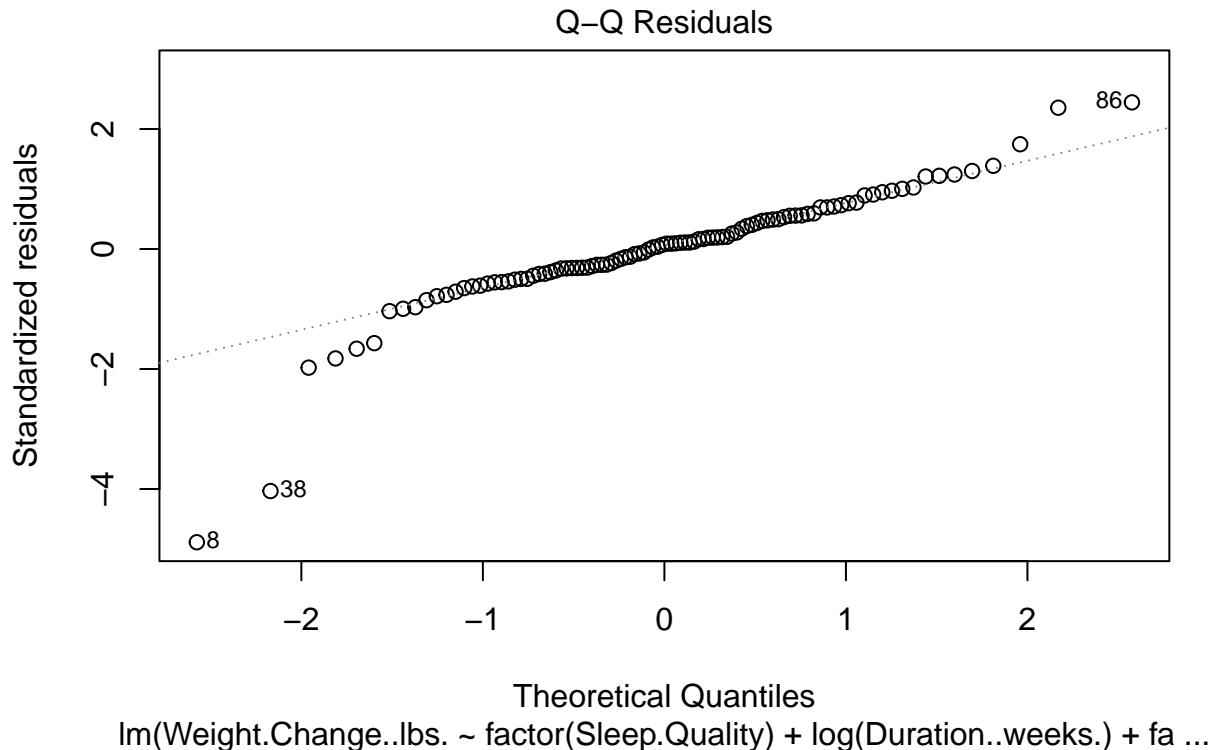


```
##  
## studentized Breusch-Pagan test  
##  
## data: weight_model_log  
## BP = 30.059, df = 23, p-value = 0.1477
```

Our plots look much better now than before, and we can see that our p-value for our Breusch-Pagan Test (0.1477) is > 0.05 . This means that we now fail to reject our null hypothesis, implying heteroscedasticity is not present. This suggests that our log transformation has improved our model, allowing it to now pass the homoscedasticity assumption. It should be noted that the adjusted R-square of our log model has decreased slightly from our previous model (0.8464 down from 0.8563). Despite this loss, we are comfortable moving forward with the log model, as it helps us pass one of our assumptions that was previously not met.

To test whether our residuals are normally distributed, we can inspect our Q-Q plot, as well as run a Shapiro-Wilk test, using the following hypothesis:

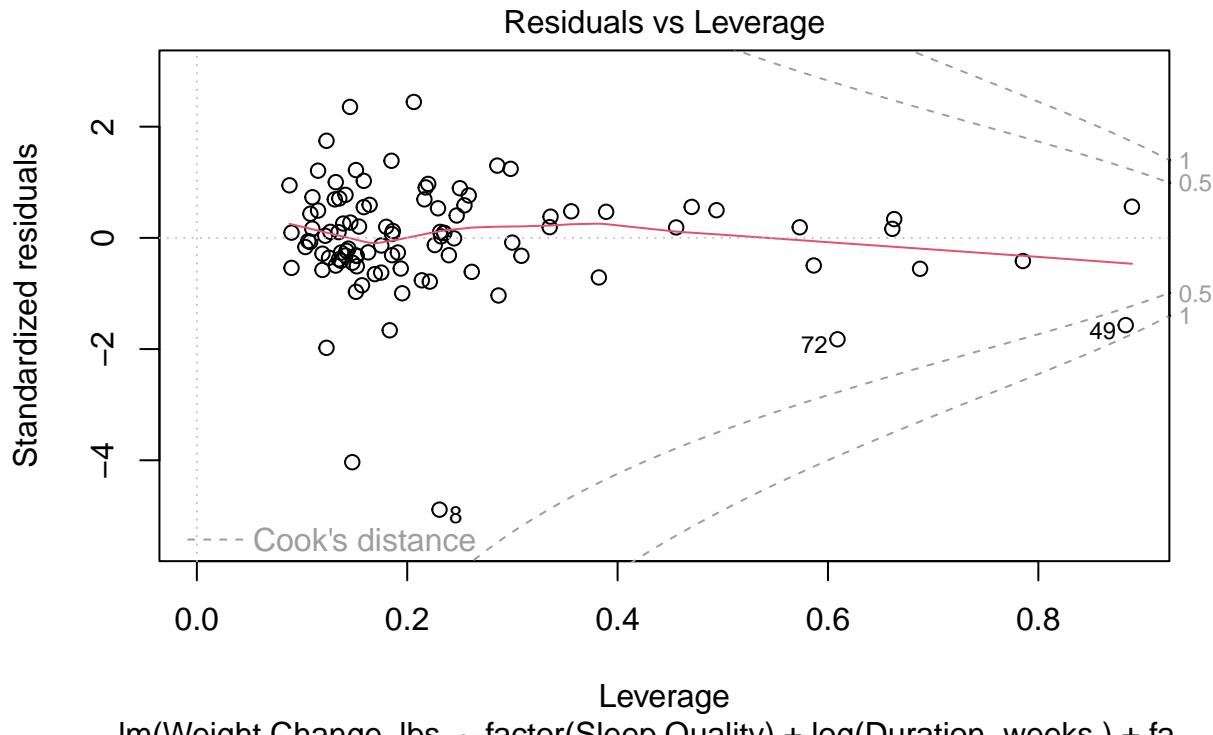
$$H_0 : \text{the residuals are normally distributed} \quad H_a : \text{the residuals are not normally distributed}$$



```
##  
## Shapiro-Wilk normality test  
##  
## data: residuals(weight_model_log)  
## W = 0.85478, p-value = 1.788e-08
```

Based on our Q-Q plot, we can see that the residuals are likely not normally distributed. This is further reinforced by our Shapiro Wilks test, which returned a p-value 1.788e-08. As this is < 0.05 , we reject our null hypothesis, implying that the residuals are not normally distributed. We can attempt to correct this using a Box-Cox transformation. Unfortunately, as our responding variable is weight change, there are values less than 0. Therefore a Box-Cox transformation cannot be done. For the time being we will simply conclude that our model has not passed the assumption of normality.

Finally, we will check for outliers, using our residuals vs. leverage plot.



Based on our plot, we can see that observation 49 is an outlier. We can attempt to remove the data point to see how it affects our model.

```
##
## Call:
## lm(formula = Weight.Change..lbs. ~ factor(Sleep.Quality) + log(Duration..weeks.) +
##     factor(Stress.Level) + factor(Sleep.Quality) * log(Duration..weeks.) +
##     log(Duration..weeks.) * factor(Stress.Level), data = weight_clean)
##
## Residuals:
##      Min        1Q    Median        3Q       Max 
## -12.7111 -0.8341  0.1531  1.1537  8.3484 
##
## Coefficients:
## (Intercept)          Estimate Std. Error t value
## factor(Sleep.Quality)Fair 0.21766  3.39093  0.064
## factor(Sleep.Quality)Good 0.18485  2.79447  0.066
## factor(Sleep.Quality)Poor -1.59689  3.46262 -0.461
## log(Duration..weeks.)    1.24320  2.84135  0.438
## factor(Stress.Level)2    1.21315  1.91233  0.634
## factor(Stress.Level)3    0.54944  3.52544  0.156
## factor(Stress.Level)4   -2.44803  3.39827 -0.720
## factor(Stress.Level)5   -1.93746  3.82412 -0.507
## factor(Stress.Level)6   -4.19952  4.22433 -0.994
## factor(Stress.Level)7   -0.32160  3.31916 -0.097
## factor(Stress.Level)8   -2.49840  5.01166 -0.499
## factor(Stress.Level)9  13.14533  8.01703  1.640
```

```

## factor(Stress.Level)9           -0.12103   3.29738  -0.037
## factor(Sleep.Quality)Fair:log(Duration..weeks.) -0.06075   1.57079  -0.039
## factor(Sleep.Quality)Good:log(Duration..weeks.)  0.55598   1.83868   0.302
## factor(Sleep.Quality)Poor:log(Duration..weeks.) -5.29987   1.68341  -3.148
## log(Duration..weeks.):factor(Stress.Level)2      -0.46614   1.83275  -0.254
## log(Duration..weeks.):factor(Stress.Level)3      1.45515   1.77888   0.818
## log(Duration..weeks.):factor(Stress.Level)4      0.89513   2.00296   0.447
## log(Duration..weeks.):factor(Stress.Level)5      2.98467   2.34180   1.275
## log(Duration..weeks.):factor(Stress.Level)6      -0.02623   1.77666  -0.015
## log(Duration..weeks.):factor(Stress.Level)7      1.23485   2.38898   0.517
## log(Duration..weeks.):factor(Stress.Level)8      -12.05261  3.93280  -3.065
## log(Duration..weeks.):factor(Stress.Level)9      -6.05003   1.74066  -3.476
##
## Pr(>|t|)
## (Intercept)          0.948991
## factor(Sleep.Quality)Fair 0.947436
## factor(Sleep.Quality)Good 0.646004
## factor(Sleep.Quality)Poor 0.662978
## log(Duration..weeks.) 0.527760
## factor(Stress.Level)2 0.876570
## factor(Stress.Level)3 0.473533
## factor(Stress.Level)4 0.613893
## factor(Stress.Level)5 0.323358
## factor(Stress.Level)6 0.923071
## factor(Stress.Level)7 0.619579
## factor(Stress.Level)8 0.105261
## factor(Stress.Level)9 0.970818
## factor(Sleep.Quality)Fair:log(Duration..weeks.) 0.969254
## factor(Sleep.Quality)Good:log(Duration..weeks.) 0.763198
## factor(Sleep.Quality)Poor:log(Duration..weeks.) 0.002357 **
## log(Duration..weeks.):factor(Stress.Level)2      0.799930
## log(Duration..weeks.):factor(Stress.Level)3      0.415939
## log(Duration..weeks.):factor(Stress.Level)4      0.656233
## log(Duration..weeks.):factor(Stress.Level)5      0.206415
## log(Duration..weeks.):factor(Stress.Level)6      0.988259
## log(Duration..weeks.):factor(Stress.Level)7      0.606752
## log(Duration..weeks.):factor(Stress.Level)8      0.003028 **
## log(Duration..weeks.):factor(Stress.Level)9      0.000851 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.889 on 75 degrees of freedom
## Multiple R-squared:  0.8859, Adjusted R-squared:  0.8509
## F-statistic: 25.32 on 23 and 75 DF,  p-value: < 2.2e-16

```

Based on the output of our model with and without the outlier, we can see that removing the outlier slightly improved our adjusted R-squared (0.8509 from 0.8464), while reducing our RSE (2.889 from 2.917), without changing the significance of any of our predictors. Based on this we feel comfortable removing the indicated outlier.

Although our model does fail the assumptions of linearity and normality, for the sake of this project we will go forward with our final model being:

$$X_{Weight.Change..lbs.} = 0.21766 + 0.18485 X_{Sleep.Quality_{Fair}} - 1.59689 X_{Sleep.Quality_{Good}} + 1.24320 X_{Sleep.Quality_{Poor}} + 1.21315 X_{log(L)}$$

Now that we have our final model, we can begin interpreting it.

Intercept: When sleep quality is excellent, stress level is 1, and log(duration) is 0 (which would be a duration of 1 week), the expected weight change is 0.21766 pounds.

Sleep Quality:

- If sleep quality is fair, the predicted weight change is $(0.18485 - 0.06075X_{log(Duration..weeks.)})$ pounds.
- If sleep quality is good, the predicted weight change is $(-1.59689 + 0.55598X_{log(Duration..weeks.)})$ pounds.
- If sleep quality is poor, the predicted weight change is $(1.24320 - 5.29987X_{log(Duration..weeks.)})$ pounds.
- If sleep quality is excellent, the predicted weight change would simply be our model equation excluding any $Sleep.Quality_i$ terms or interactions, where i = Fair, Good, or Poor, as all values would simply be 0 for those variables.

Duration in weeks:

- For every one-unit increase in $\log(\text{Duration..weeks.})$, weight change increases by:

$$(1.21315 - 0.06075X_{Sleep.Quality_{Fair}} + 0.55598X_{Sleep.Quality_{Good}} - 5.29987X_{Sleep.Quality_{Poor}} - 0.46614X_{Stress.Level_2} + 1.4551$$

Stress Level:

- If stress level is 2, the predicted weight change is $(0.54944 - 0.46614X_{log(Duration..weeks.)})$ pounds.
- If stress level is 3, the predicted weight change is $(-2.44803 + 1.45515X_{log(Duration..weeks.)})$ pounds.
- If stress level is 4, the predicted weight change is $(-1.93746 + 0.89513X_{log(Duration..weeks.)})$ pounds.
- If stress level is 5, the predicted weight change is $(-4.19952 + 2.98467X_{log(Duration..weeks.)})$ pounds.
- If stress level is 6, the predicted weight change is $(-0.32160 - 0.02623X_{log(Duration..weeks.)})$ pounds.
- If stress level is 7, the predicted weight change is $(-2.49840 + 1.23485X_{log(Duration..weeks.)})$ pounds.
- If stress level is 8, the predicted weight change is $(13.14533 - 12.05261X_{log(Duration..weeks.)})$ pounds.
- If stress level is 9, the predicted weight change is $(-0.12103 - 6.05003X_{log(Duration..weeks.)})$ pounds.
- If stress level is 1, the predicted weight change would simply be our model equation excluding any $Stress.Level_i$ terms or interactions, where i = 2 to 9, as all values would simply be 0 for those variables.

Lastly, our Adjusted R^2 for our model is 0.8509, implying that our 85.09% of the variability in weight change is explained by our model.

To demonstrate the predictiveness of our model, we have included an example. Assume a participant records their weight change over a period of 6 weeks, where they recorded poor sleep quality, and a stress level of 5. What would be there predicted weight change?

```
##           1  
## -4.713261
```

Based on our finalized model, and assuming a participant records their weight change over a period of 6 weeks, where they recorded poor sleep quality, and a stress level of 5, we would predict their weight to decrease by 4.713261 pounds.

4 Conclusion and Discussion

4.1 Approach

4.2 Future Work

5 References

1. THE WEBSITE FOR THE DATASET



Unlocking Insights in Calgary Child Care Program Compliance: An Analytical Approach

Group 39

Sri Harsha Tuttaganti

Venkateshwaran Balu Soundararajan





Introduction

- Childcare is essential for early childhood development.
- Provides a foundation for learning, socialization, and well-being.
- High-quality childcare significantly impacts a child's growth and future success.





Objective

- ❖ Utilize the Calgary Childcare Information dataset.
- ❖ Analyze licensed childcare programs.
- ❖ Focus on capacity, compliance status, and inspection history.
- ❖ Ensure high standards of care for children in Calgary resources.



Target Audience

- Parents: Searching for reliable childcare options.
- Researchers: Analyzing childcare availability and distribution.
- Policymakers: Evaluating childcare needs and planning



Child Care Information Dataset



https://data.calgary.ca/Health-and-Safety/Child-Care-Information/qdxh-qngy/about_data

Overview

- **Dataset Name:** Child Care Information
- **Source:** City of Calgary Open Data Portal
- **Description:**
This dataset provides information about licensed Child Care facilities in Calgary. It includes details such as facility names, locations, types of care offered, capacity, and contact information. (2014-2023)



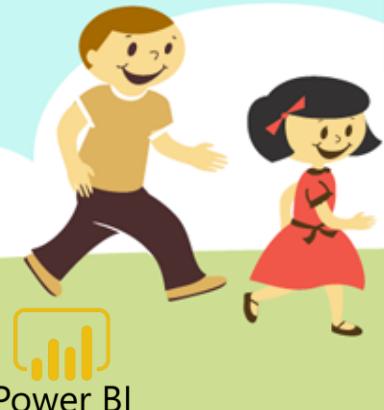
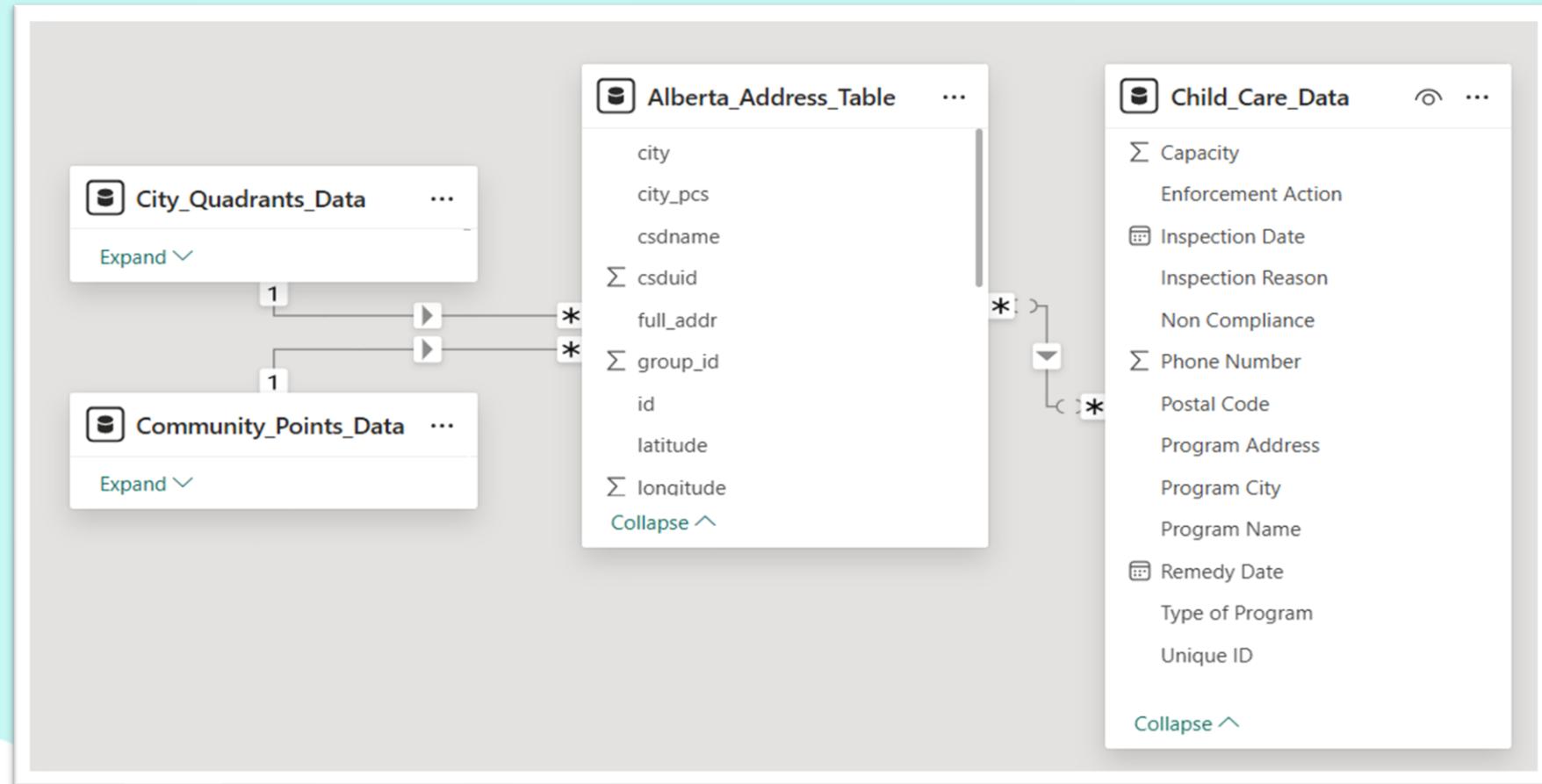


Data Fields and Its Significance

Field Name	Data Type	Significance
Program Name	Text	Uniquely identifies each program
Type of Program	Text	Categorizes programs for analysis
Program Address	Text	Provides location details
Program City	Text	Identifies the city for location-based analysis
Postal Code	Text	Used for geographic and demographic analysis
Phone Number	Text	Provides a means of contact for inquiries
Capacity	Number	Indicates the program's scale and capacity
Inspection Date	Floating Timestamp	Tracks the timeline of inspections
Inspection Reason	Text	Provides insights into compliance and regulatory focus areas
Non Compliance	Text	Assesses compliance status and identifies areas needing improvement
Enforcement Action	Text	Details corrective actions implemented
Remedy Date	Floating Timestamp	Tracks deadlines for resolving non-compliance issues



Data Model with Supporting Datasets



Data Cleaning and Transformation

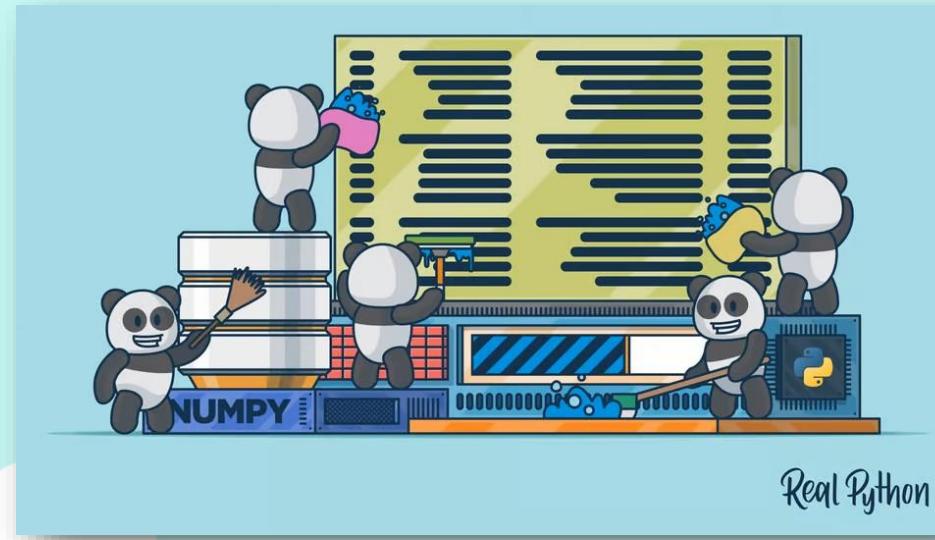
Step 1 : Cleaned the data by removing duplicate values.

Step 2 : Added a dataset containing latitude and longitude information.

Step 3 : Removed duplicates caused by merging location dataset.

Step 4 : Addressed normalization issues by cross-checking addresses in both datasets and rectifying discrepancies.

Step 5 : Added another dataset to group data into communities and Quadrants.



Data Preprocessing- Iterative Imputation

Step 1: Removed rows containing null values

- Created a cleaned dataset by removing rows with null values.

Step 2: Split the cleaned dataset

- Divided the cleaned dataset into test (20%) and train (80%) datasets.

Step 3: Applied iterative imputation using random forest

- Used the random forest model for iterative imputation to predict missing values.

Step 4: Integrated the predicted values

- Integrated the predicted values back into the original dataset.

Step 5: Performed imputation on specific columns

- Applied the imputation process on the "Program Name" and "Type of Program" columns.

Type of Program column:

- Accuracy: 0.9126

Program Name column:

- Accuracy: 0.9517



GUIDING QUESTIONS

7
Program Types

1004
Unique Programs

14
Inspection Types



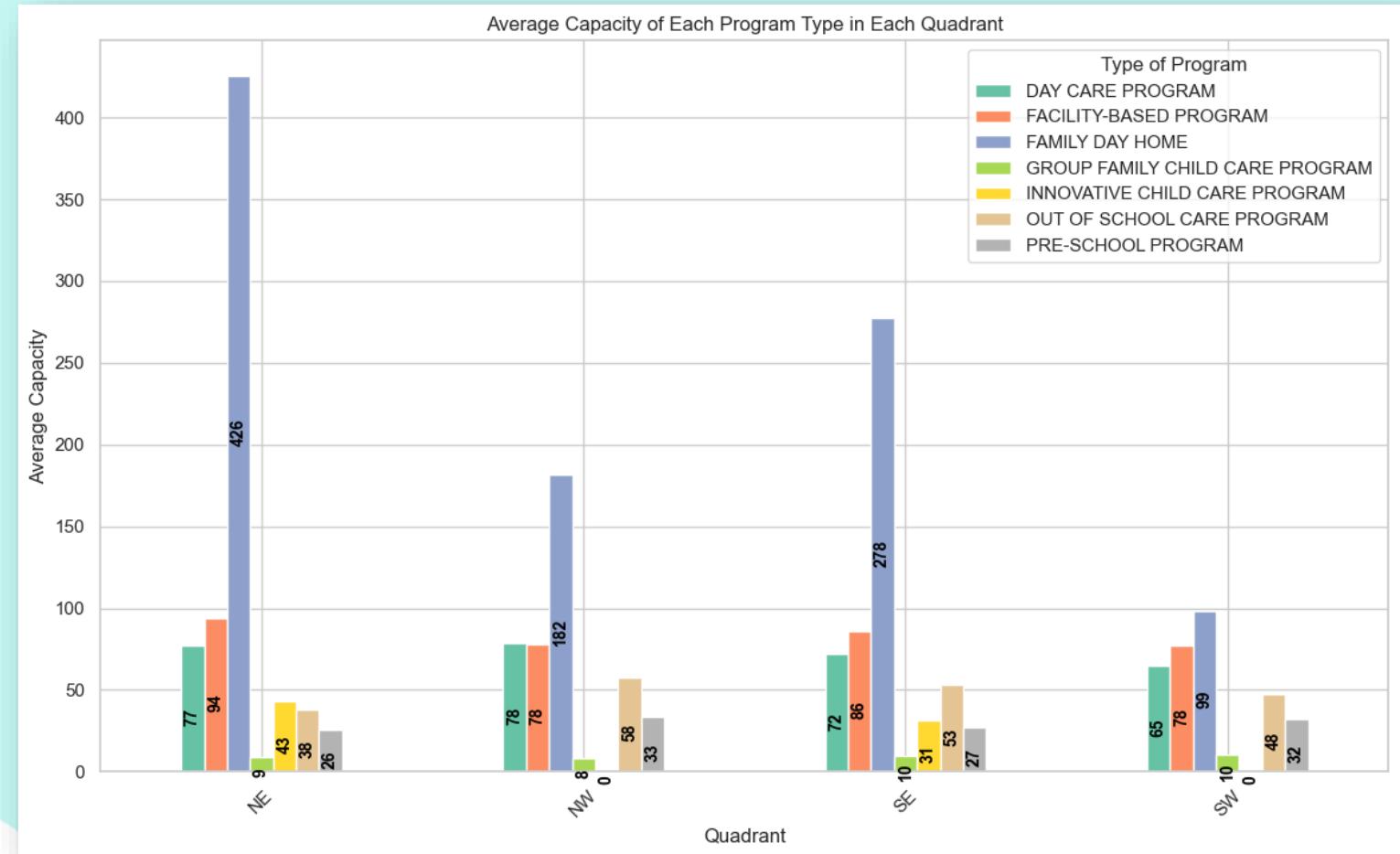


Program Types and its distribution in Calgary

What is the average capacity of child care programs by its types, how does it vary by its Quadrant

Insights:

- Highest Average Capacity: Across all quadrants, the FAMILY DAY HOME program consistently has the highest average capacity, indicating its dominance in each region.
- Variation Across Quadrants: The average capacity of program types varies significantly across different quadrants.
- Program Diversity: Each quadrant offers a mix of program types, with notable variation in their capacities.





Year on Year Inspections and its Non- Compliance Trends

How do the average number of inspections and the rate of non-compliance change year-on-year across different childcare programs in Calgary?

Insights:

- 2016-2019 Increase: There was a noticeable increase in non-compliance percentages from 2016 to 2019.
- 2020 Dip: Both inspection counts and non-compliance percentages dipped significantly in 2020.
- Post-2020 Recovery: There was a recovery in inspection counts post-2020, but non-compliance percentages remain fluctuating.





Inspections Reasons frequency with Program Type

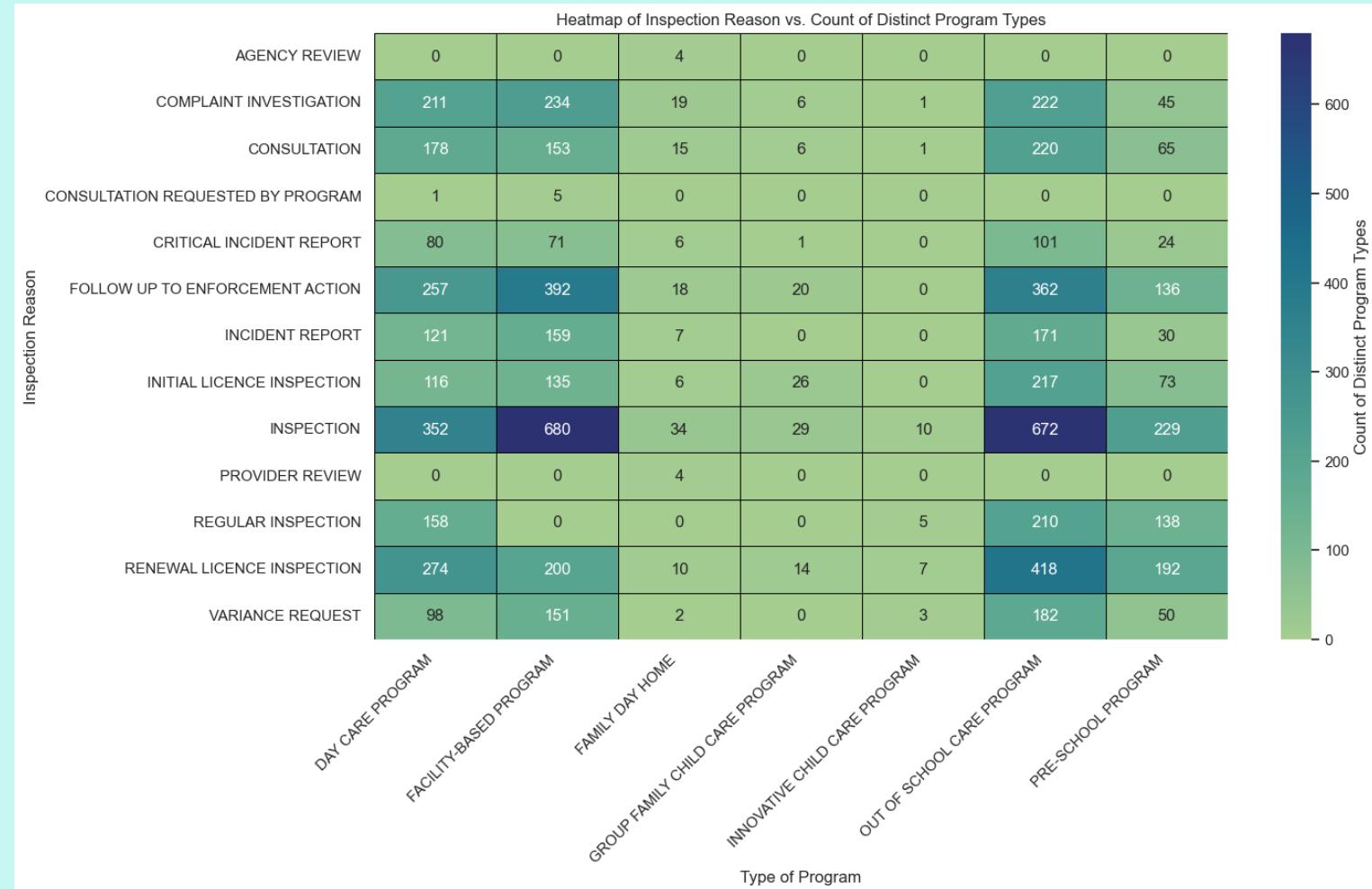
What are the most common reasons for inspections across different types of childcare programs in Calgary?

Insights:

High-Frequency Inspections: INSPECTION and COMPLAINT INVESTIGATION are common across most program types, indicating regular oversight and response to complaints.

Program-Specific Trends: Certain program types like Facility-Based Programs and Family Day Homes are more frequently inspected.

Focus Areas: The high counts in FOLLOW UP TO ENFORCEMENT ACTION and RENEWAL LICENCE INSPECTION suggest these are critical areas needing close monitoring.



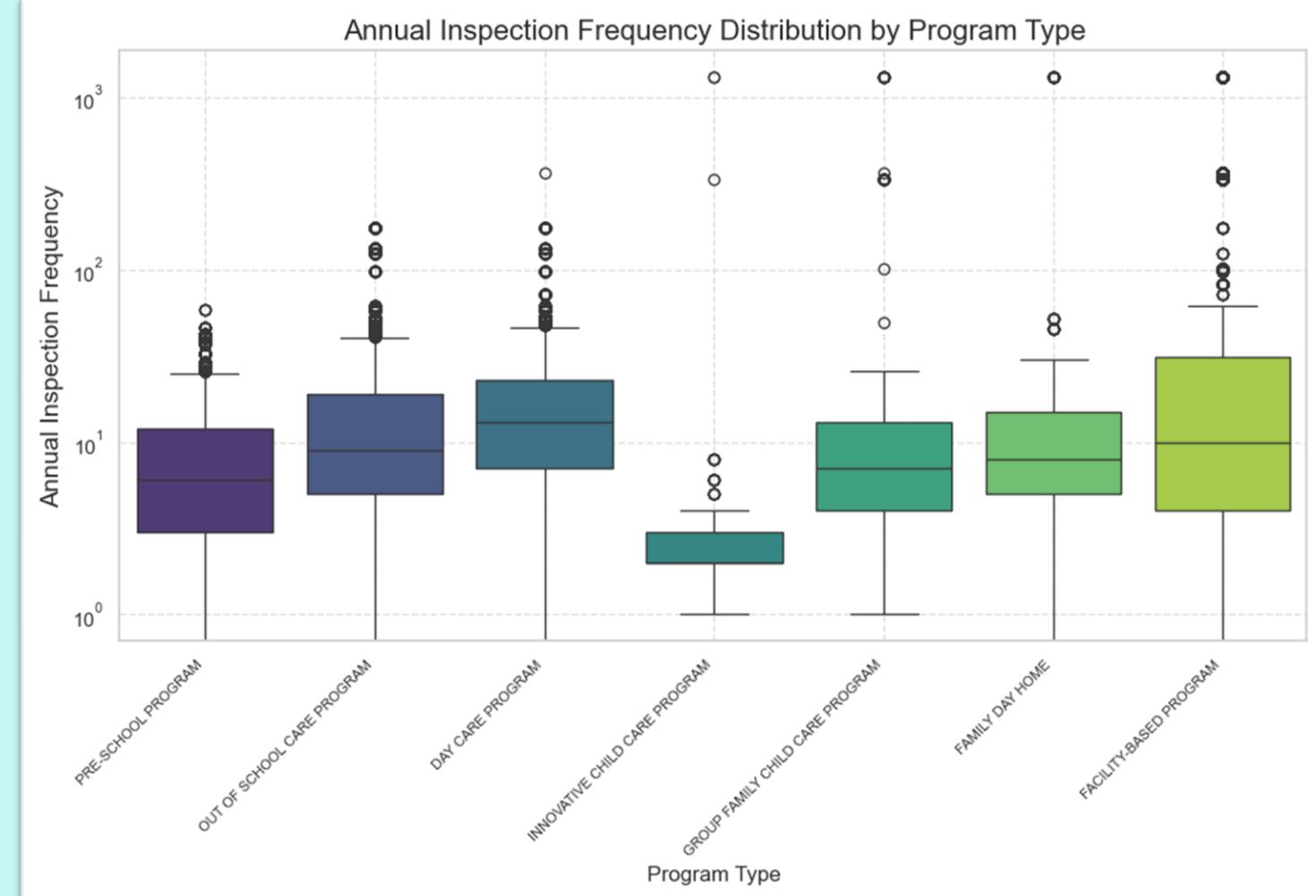
Inspection Frequency (in Days) V/S Program Type

Calgary

Is there a correlation between the type of childcare program and the frequency of inspections?

Insights:

- Frequent Inspections: Innovative Child Care Programs experience more frequent and consistent inspections compared to other program types.
- Less Frequent Inspections: Facility-Based Programs have the least frequent and most variable inspection intervals, indicating less regulatory scrutiny.

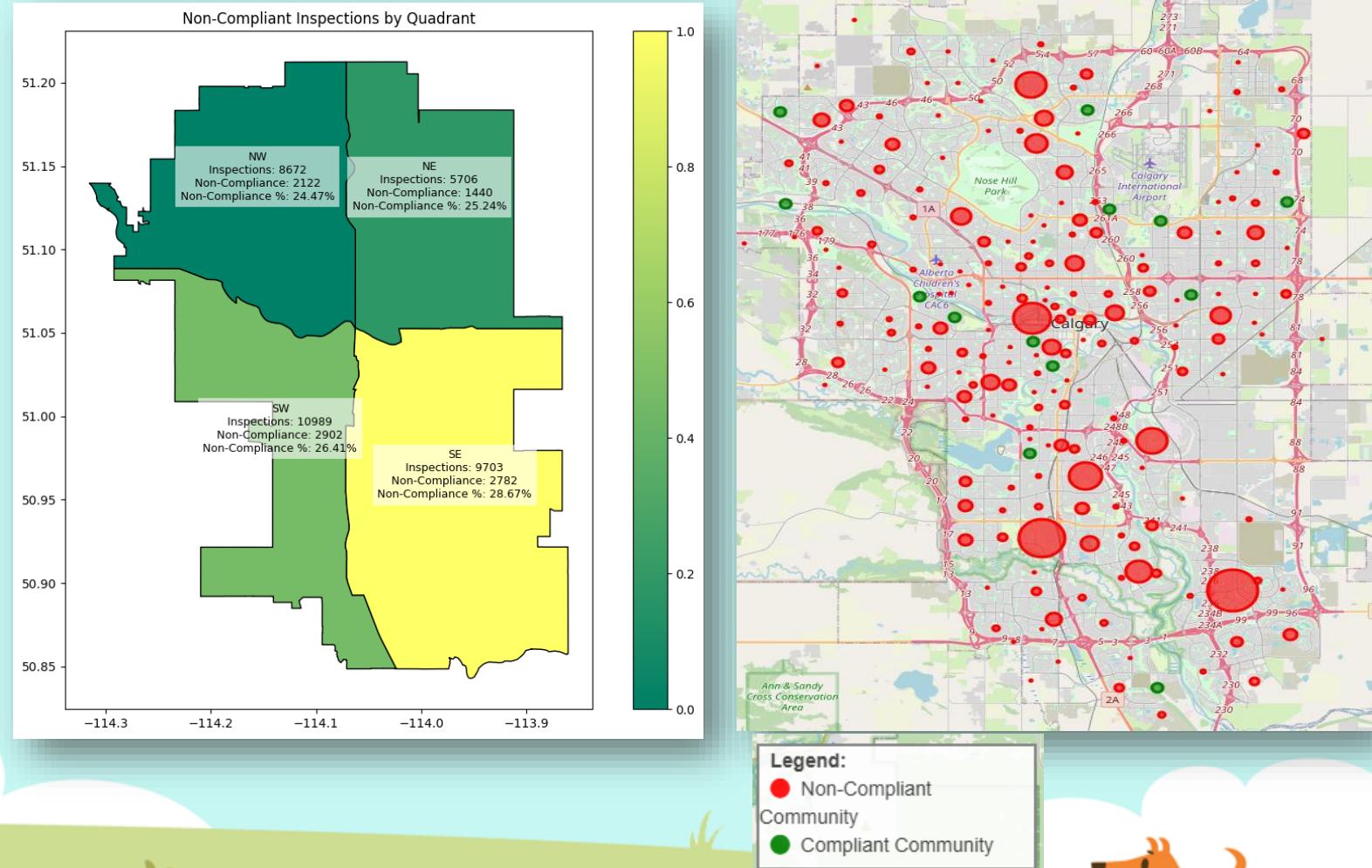


Understanding the Non-Compliance% spread by Quadrant for better Choices

How is the Non-Compliance Spread across the four Quadrants and Communities of the city ?

Insights:

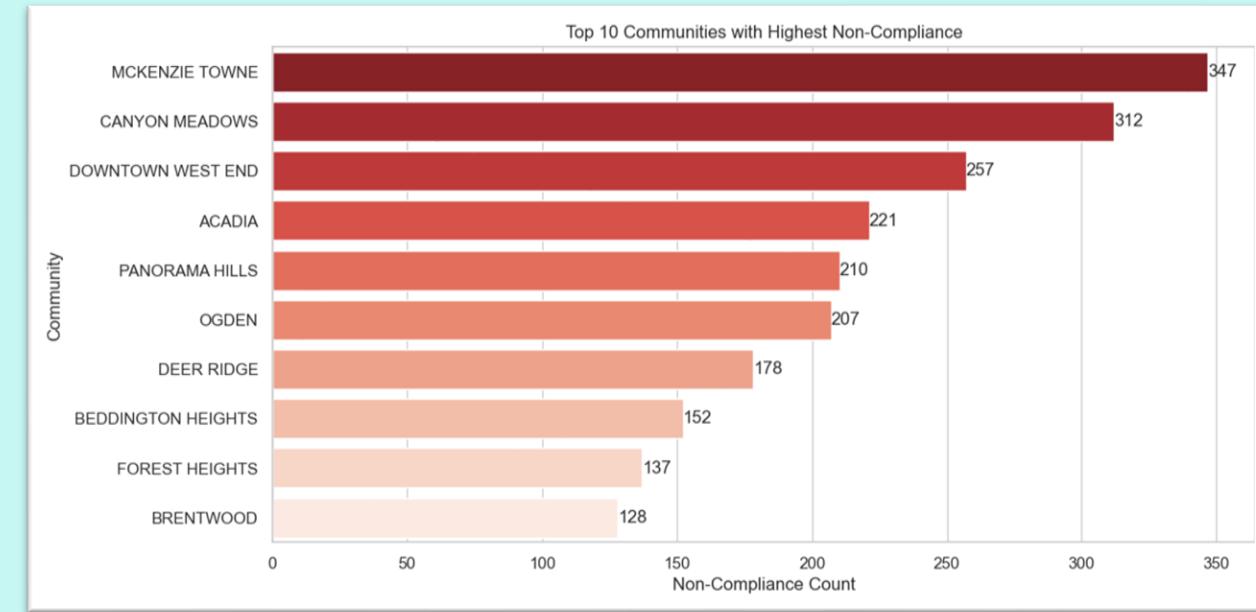
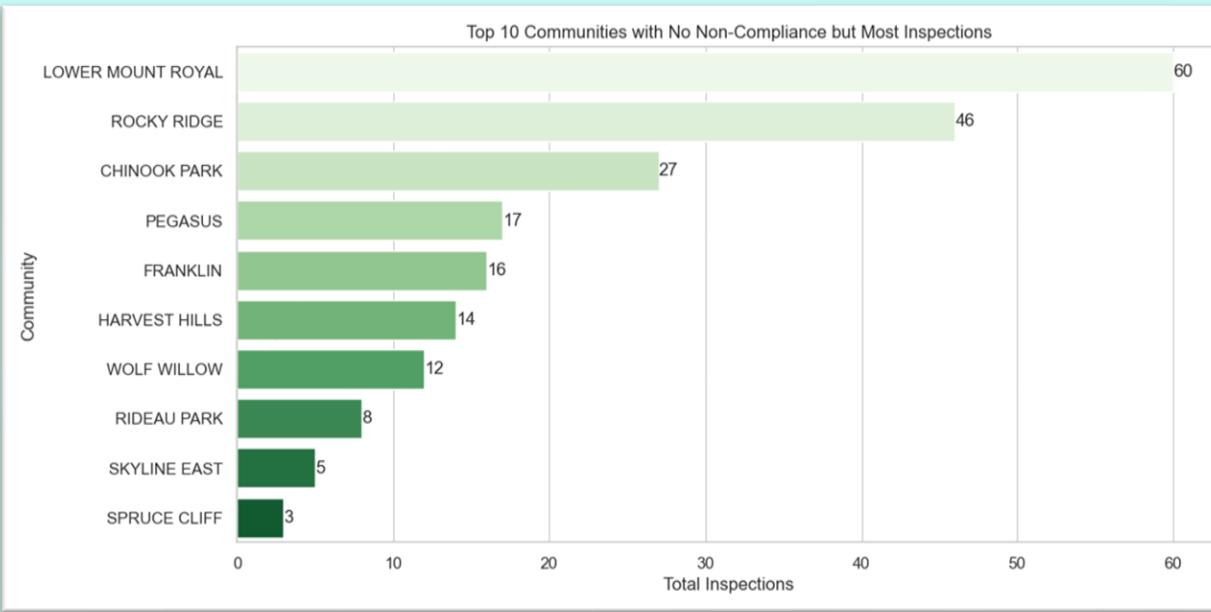
- Highest Non-Compliance Rate: The SE (Southeast) quadrant has the highest non-compliance rate at 28.67%, indicating more frequent issues in this area.
- Frequent Inspections: The SW (Southwest) quadrant has the highest number of inspections (10,909), highlighting intensive monitoring.





Program Inspections and Non-Compliance against Communities

Identify the Top-10 communities with lowest and highest compliance rates



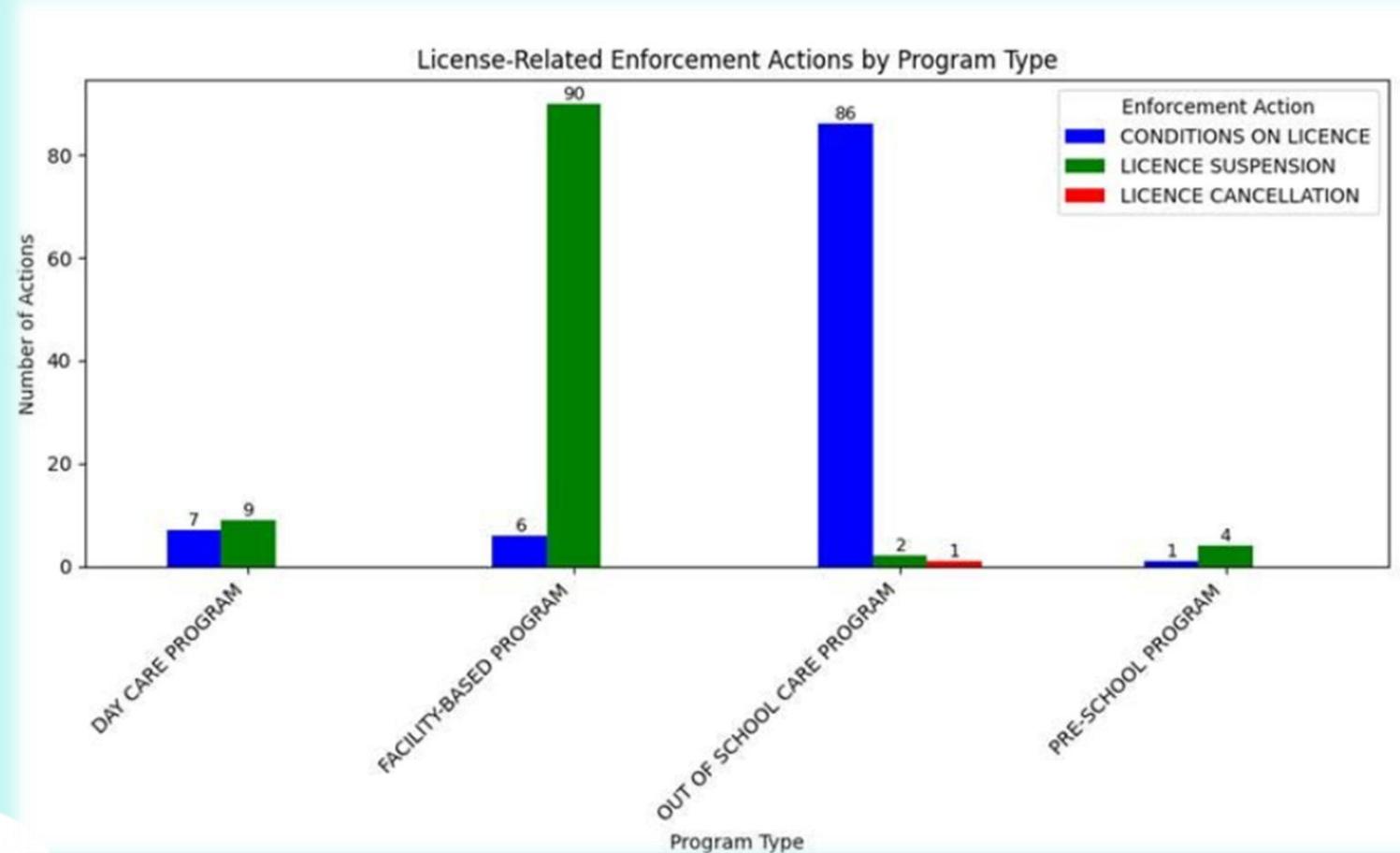


Analyzing Regulatory Compliance –License Cancellation

How does the certain program types or locations have more enforcement actions pertaining to licence cancellation and suspension ?

Insights:

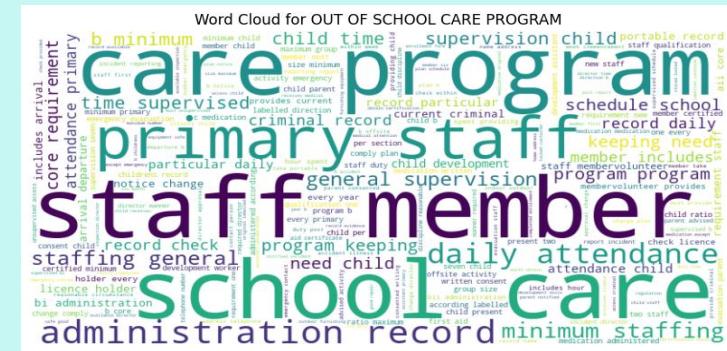
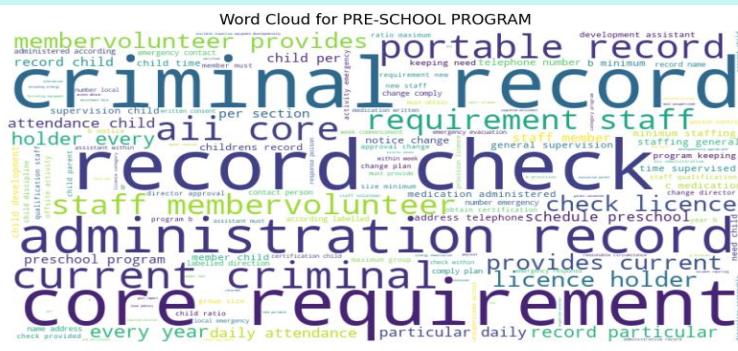
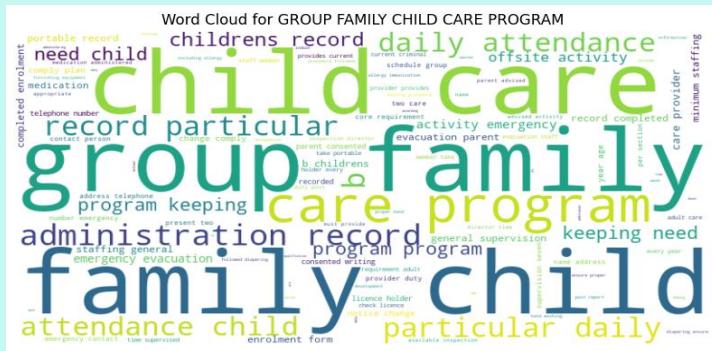
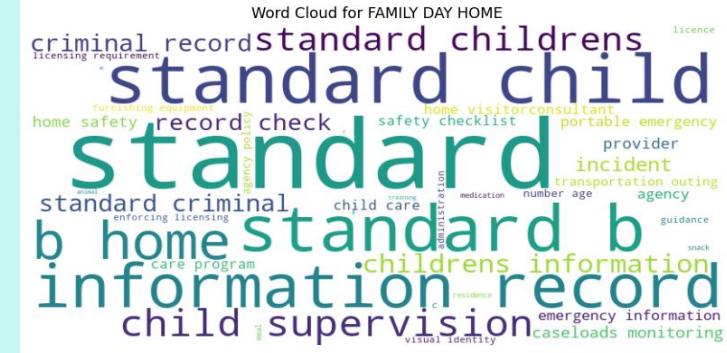
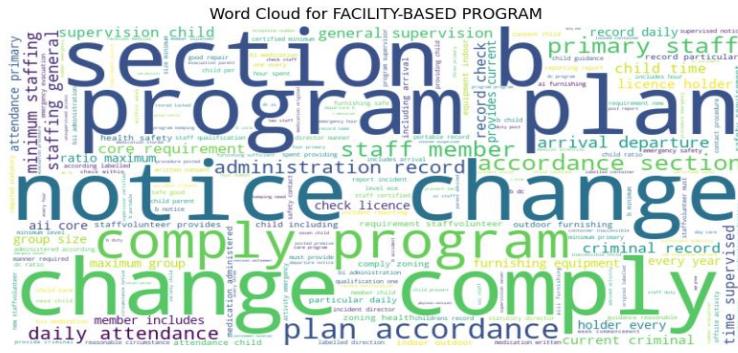
- Facility-Based Programs face significant enforcement actions, primarily suspensions.
- While Pre-School Programs have the fewest actions, indicating relatively better compliance.
- Out of School Care Programs mostly encounter conditions on their licenses.



MACHINE LEARNING



Text Analysis – Identification of Reasons in Non-Compliance Programs



Innovative Childcare Program has no data



Sentimental Analysis on Non - Compliance

Sentiment Scores Per Program Type:

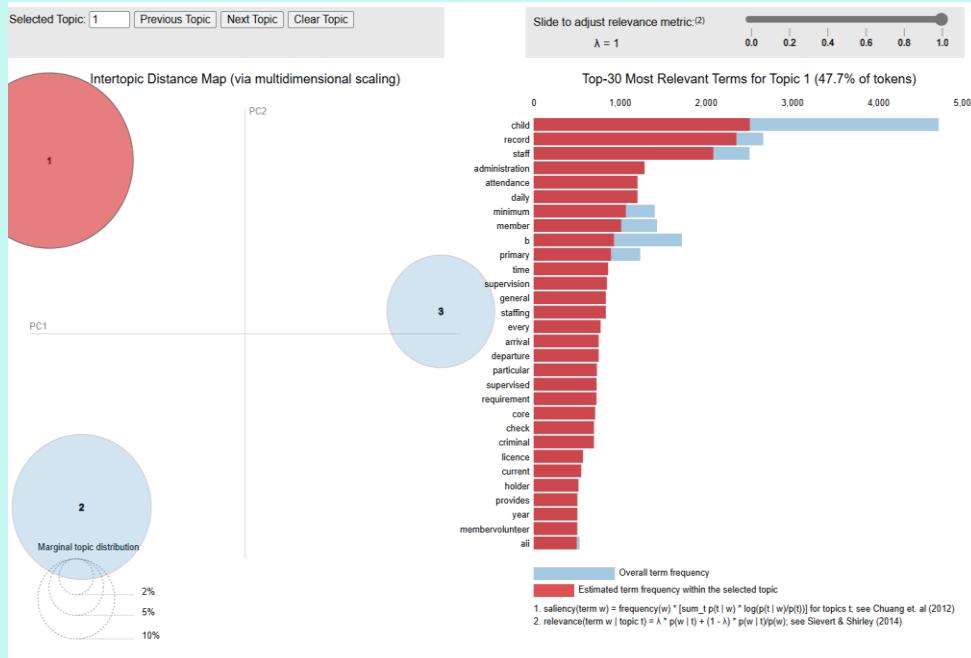
Type of Program	neg	neu	pos	compound	Sentiment Label
DAY CARE PROGRAM	0.032	0.840	0.128	0.113	Positive
FACILITY-BASED PROGRAM	0.040	0.906	0.054	0.001	Positive
FAMILY DAY HOME	0.062	0.887	0.051	-0.017	Negative
GROUP FAMILY CHILD CARE PROGRAM	0.051	0.869	0.079	0.069	Positive
OUT OF SCHOOL CARE PROGRAM	0.035	0.887	0.078	0.049	Positive
PRE-SCHOOL PROGRAM	0.055	0.928	0.017	-0.113	Negative

Frequent words used in Programs

DAY CARE PROGRAM	: child, program, record, care, day, staff, b, administration, member
FACILITY-BASED PROGRAM	: child, record, b, staff, program, comply, medication, change, notice
FAMILY DAY HOME	: standard, record, child, b, information, home, supervision, childrens, criminal
GROUP FAMILY CHILD CARE PROGRAM	: child, record, care, program, b, group, family, administration, emergency
OUT OF SCHOOL CARE PROGRAM	: child, staff, record, program, care, b, member, minimum, school
PRE-SCHOOL PROGRAM	: record, child, staff, b, criminal, check, program, core, requirement



LDA Topic Modeling and its insights on Non-Compliance Reasons



Topic 1: Staffing & Supervision Compliance Issues

Key Terms:

- child, record, staff, administration, attendance, supervision, requirement, criminal, licence, holder, check

Non-compliance issues may involve:

- Insufficient supervision of children.
- Failure to maintain proper attendance records.
- Issues with staff qualifications, criminal background checks, and required licensing.

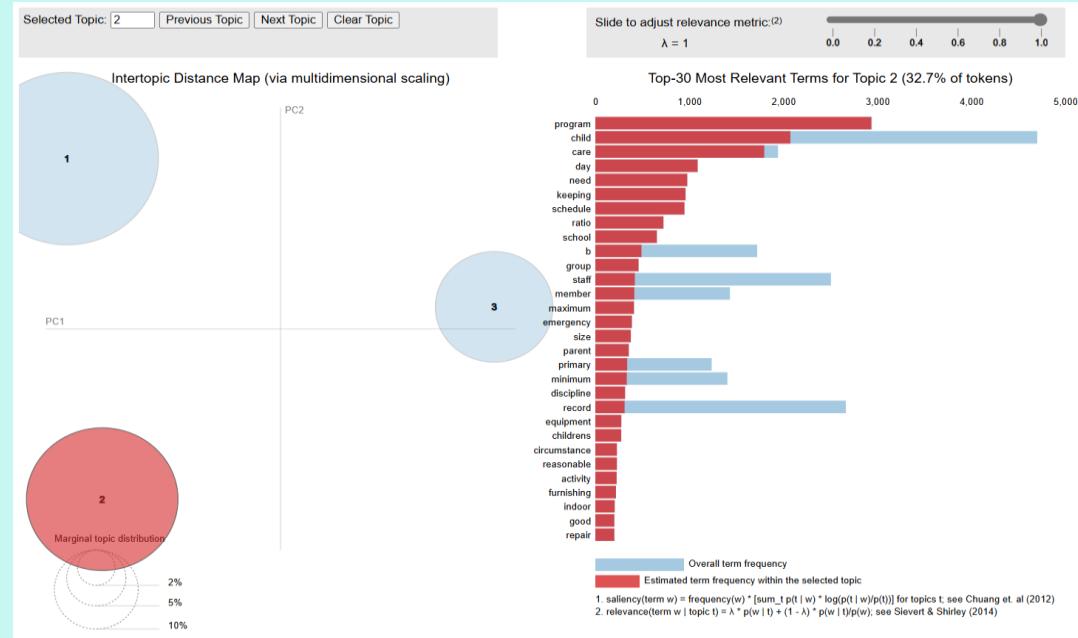
Suggestion:

- Childcare program should focus on ensuring qualified, well-trained staff for childcare.





LDA Insights



Topic 2: Childcare Facility & Safety Regulations

Key Terms:

- program, child, care, day, keeping, schedule, ratio, school, equipment, children, repair

Non-compliance issues may involve:

- Violation of child-to-staff ratios.
- Poor maintenance of facilities and equipment.
- Failure to follow proper schedules and care programs.

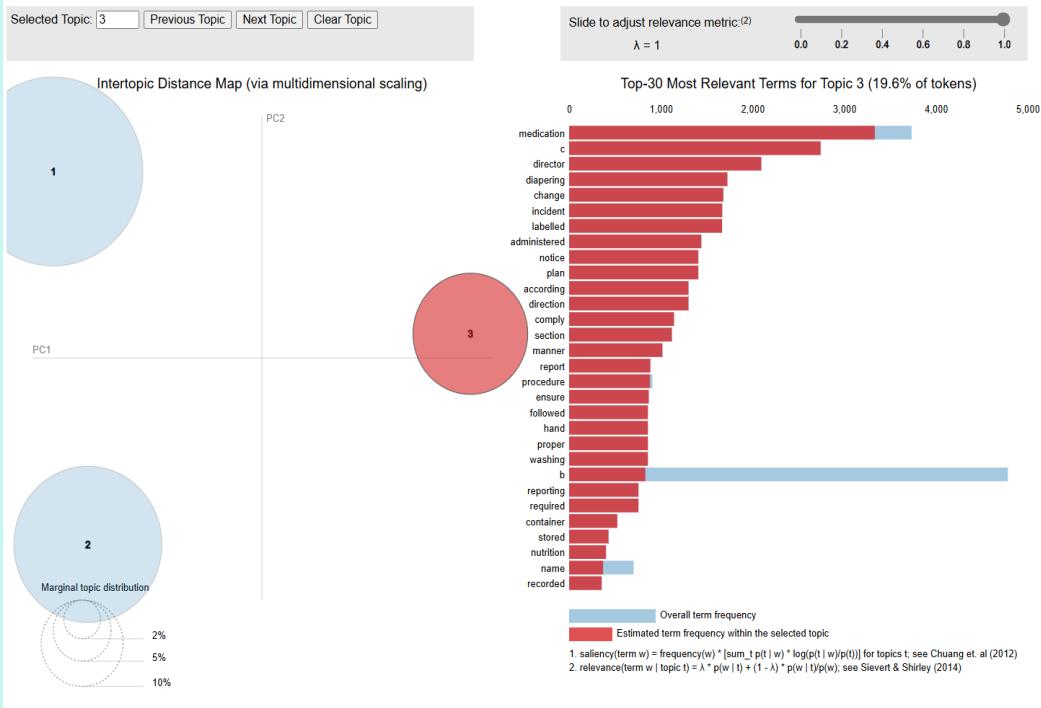
Suggestion:

- Childcare should prioritize safety, well-maintained environments for child development.





LDA insights



Topic 3: Medication & Hygiene Compliance

Key Terms:

- medication, diapering, change, incident, administered, procedure, washing, reporting, nutrition, stored

Non-compliance issues may involve:

- Improper storage or administration of medication.
- Lack of hygiene standards in diapering and handwashing.
- Failure to report incidents related to medication or health.

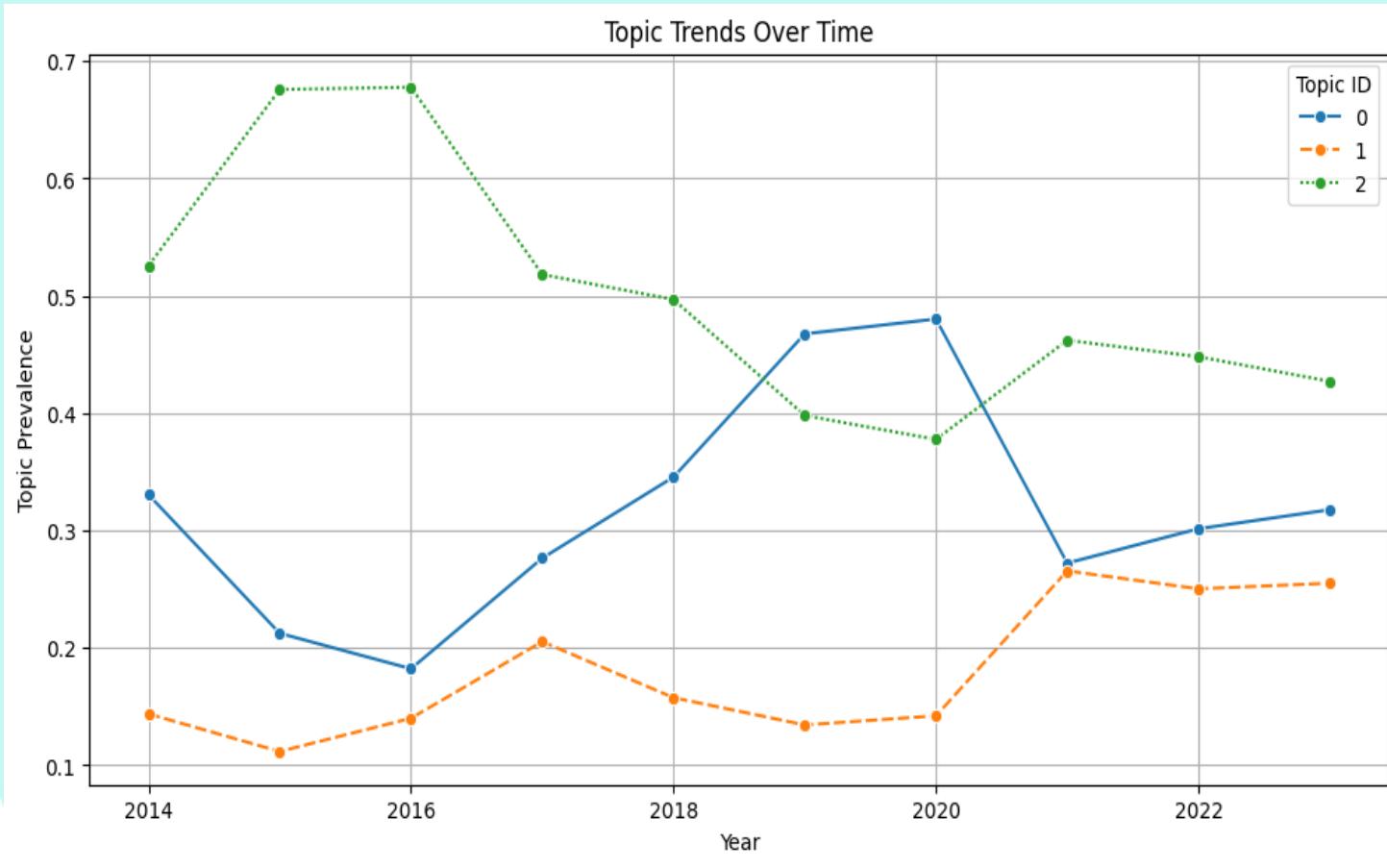
Suggestion:

- This highlights the importance of strict health and safety protocols in childcare environments.



LDA Over the Time

Are there trends or patterns in identified topics over time?



Topic 0: program, child, care, day, need
Topic 1: medication, c, director, diapering, change
Topic 2: child, record, staff, administration, attendance



Predictive Analysis of Non-Compliance:

Can we predict non-compliance likelihood based on program attributes and inspection history, and what features indicate compliance status?

Data Preparation

Binary Target Column: Created a binary target column (Non_Compliance_Indicator) to indicate non-compliance.)

Handling Missing Values:

Dropped rows with missing values in selected columns to ensure data quality.

Feature Selection: Selected relevant columns for modeling.

Categorical Encoding

One-Hot Encoding: Used one-hot encoding to convert categorical variables into a format suitable for machine learning algorithms.

Data Splitting: Split the data into training and testing sets.

SMOTE

SMOTE Application: Applied SMOTE (Synthetic Minority Over-sampling Technique) to oversample the minority class..

Logistic Regression Model

Training the Model: Trained a logistic regression model on the resampled training data.

Code for Prediction

Code to Encode and Predict



Model Results and Predictions

Prediction Scenarios

Class Distribution in Resampled Training Data:

Non_Compliance_Indicator

1 24610

0 24610

Name: count, dtype: int64

Confusion Matrix (Resampled Data):

[[3967 2146]

[766 1581]]

Classification Report (Resampled Data):

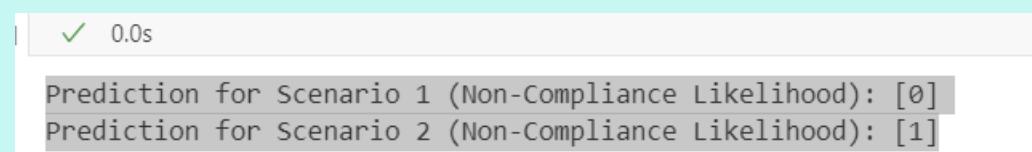
	precision	recall	f1-score	support
0	0.84	0.65	0.73	6113
1	0.42	0.67	0.52	2347
accuracy			0.66	8460
macro avg	0.63	0.66	0.63	8460
weighted avg	0.72	0.66	0.67	8460

Scenario 1: Group Family Child Program

- Capacity: 10
- Type of Program: Group Family Child Program
- Postal Code: T2M3T4
- Prediction: [0] Likelihood of Non-Compliance: Unlikely

Scenario 2: Innovative Child Care Program

- Capacity: 6050
- Type of Program: Innovative Child Care Program
- Postal Code: T2M1L9
- Prediction: Predicted likelihood of non-compliance.
- Prediction: [1] Likelihood of Non-Compliance: Likely





Conclusion

- Key Non-Compliance Issues Identified.
- Non-Compliance Trends by Quadrant.
- Yearly Trends in Non-Compliance & Inspections.
- We identified key inspection reasons from LDA model
- Predictive Modeling on non-compliance likelihood.

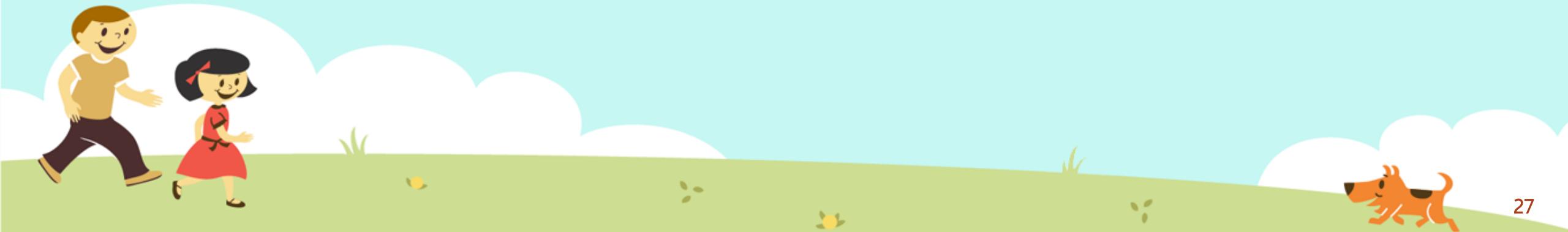
Future Steps

- **Refine Predictive Models** – Improve non-compliance forecasting accuracy.
- **Policy Recommendations** – Use trends to guide better regulations.
- **Community Engagement** – Provide recommendations to Conduct training programs for childcare providers



References

- [1] Government of Alberta, "Childcare," Alberta.ca, 2025. [Online]. Available: <https://www.alberta.ca/child-care>.
- [2] City of Calgary, "Communities by Sector," Data Calgary, 2025. [Online]. Available: <https://data.calgary.ca/Base-Maps/Communities-by-Sector/e6xg-kaxf>.
- [3] City of Calgary, "City Quadrants," Data Calgary, 2025. [Online]. Available: <https://data.calgary.ca/Base-Maps/City-Quadrants/g8ma-syw>.
- [4] Statistics Canada, "Open Database of Addresses (ODA)," Government of Canada, 2025. [Online]. Available: <https://www.statcan.gc.ca/en/lode/databases/oda>.



Thank You!



socialmediapurchaseinfluence

July 13, 2025

1 The Influence of Social Media Usage on Consumer Purchasing Decisions

Group Members

1. Ayush Senthil Nelli
2. Hritvik Gaind
3. Rehan Chanegaon
4. Satyam Kapoor
5. Venkateshwaran Balu Soundararajan

```
[2]: #Importing Required Libraries
import pandas as pd
import plotly.express as px
import scipy.stats as stats
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import statsmodels.formula.api as smf
from scipy.stats import chi2
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
# Disable all warnings
import warnings
warnings.filterwarnings("ignore")
pd.options.mode.chained_assignment = None
```

1.1 Introduction

In today's digital era, social media plays a pivotal role in shaping consumer behavior and influencing purchasing decisions. With platforms like Instagram, Facebook, and TikTok at the forefront, brands are increasingly leveraging digital marketing strategies to capture attention and build trust. This report investigates how social media influences consumer choices, explores the impact of

digital content and influencer recommendations, and highlights data-driven approaches that help businesses optimize their marketing strategies. By analyzing survey data collected from active social media users, the study aims to provide insights that can support more targeted and effective marketing efforts.

1.2 Background and Significance

Social media has revolutionized the way consumers interact with brands and make purchasing decisions. The ubiquitous presence of platforms such as Instagram, Facebook, and TikTok has not only altered consumer expectations but also redefined marketing practices. Brands now rely on digital marketing, influencer promotions, and user-generated content to attract buyers. Moreover, consumers increasingly trust online reviews and recommendations, which serve as powerful social proof before making purchases.

Key aspects of this transformation include:

Understanding Consumer Behavior: Gaining insights into how consumers interact with digital content allows brands to better align their products and messaging with audience preferences.

Impact of Influencers & Reviews: The rise of influencer marketing and the proliferation of online reviews have made it possible to measure the tangible effects of social proof on purchasing decisions.

Data-Driven Marketing Strategies: Leveraging consumer data enables businesses to optimize their advertisements and engagement strategies, ensuring that marketing efforts are both efficient and effective.

Identifying Trends & Preferences: Continuous analysis of consumer behavior helps identify emerging trends and preferences, which is crucial for maintaining competitive advantage in a rapidly evolving market.

This project underscores the importance of data-driven insights in understanding and adapting to the shifting dynamics of consumer behavior in the digital age.

1.3 About the Dataset

The analysis is grounded in a custom-designed survey aimed at capturing the nuances of social media's influence on consumer behavior. Below are the key details of the dataset:

Primary Data Source: A custom-designed survey administered via Google Forms. link - <https://forms.gle/mhwQoduno3mTWTQ26>

Target Audience: The survey focused on consumers from the University of Calgary, covering a diverse age range of individuals who actively use social media for making shopping decisions.

Response Count: The dataset comprises responses from 125 participants.

Survey Structure: The survey was structured into several sections:

Demographics: Collecting data on age, gender, etc.

Social Media Usage: Assessing frequency of use, preferred platforms, and daily time spent.

Advertisement Exposure: Evaluating the impact of social media ads, influencer recommendations, and brand collaborations.

Purchasing Behavior: Documenting the types of products purchased through social media, trust in online reviews, and key decision-making factors.

Influencing Factors: Identifying the type of content, limited time offers, and other factors influencing consumer decisions.

Data Cleaning & Stratified Sampling: To ensure a balanced analysis, the dataset underwent several cleaning steps:

Renaming Columns: Survey questions were renamed to more appropriate column names.

Stratified Sampling: - Data was segmented into four age groups: Below 18, 18–36, 37–54, and Above 55. - The minimum count among these groups was identified to allow proportional allocation. - Random resampling with replacement was performed within each group to ensure equal sample sizes. - Finally, the resampled data was combined and shuffled to maintain randomness.

The careful design and cleaning of the dataset ensure that the analysis accurately reflects consumer behavior across different age groups, providing a reliable foundation for data-driven marketing insights.

```
[3]: df = pd.read_excel('FinalSurveyDataa.xlsx')
df_original=pd.read_csv('606Project_Original.csv')
#df.head(5)
#df_original=df.copy()
#df = pd.read_csv("Data_606_Project.csv")
#df.drop('Unnamed: 0', axis=1, inplace=True)
df.columns
```

```
[3]: Index(['Timestamp', 'What is your age group?', 'What is your gender?',
       'On average, how many hours do you spend on social media daily?',
       'Which social media platform do you use most frequently?',
       'How often do you use this platform?',
       'How often do you see product advertisements on social media?',
       'How often do you encounter promotions by influencers on social media?',
       'How many times in the past month have you purchased a product after
       seeing it on social media?',
       'How often do you make unplanned purchases after seeing a product on
       social media?',
       'Do you regret purchases made impulsively due to social media
       influence?',
       'Which type of content influences your purchasing decisions the most?',
       'How often do you purchase a product based on a limited time offer on
       social media?',
       'Do you usually compare products on other websites/apps before buying
       something seen on social media?',
       'If you compare products, what factor influences your choice the most?
       (Select Not Applicable if previous question choices as No)',
       'Have you ever purchased a product to fit in with trends seen on social
       media?',
       'How often do peers or social trends influence your purchasing decisions
```

```

on social media?',
    'Do you feel that social media increases your spending on products?',
    'How do you typically pay for purchases influenced by social media?',
    'How often do you exceed your budget because of products seen on social
media?',
    'Have you ever taken a loan or gone into debt to purchase a product
advertised on social media?',
    'How trustworthy do you find social media advertisements?',
    'Are you more likely to trust ads promoted by influencers or official
brand accounts?',
    'Email address'],
dtype='object')

```

1.4 Data Cleaning

```
[4]: df.rename(columns={
    "What is your age group?": "age_group",
    "What is your gender?": "gender",
    "On average, how many hours do you spend on social media daily?": "social_media_hours_per_day",
    "Which social media platform do you use most frequently?": "most_frequent_platform",
    "How often do you use this platform?": "platform_usage_frequency",
    "How often do you see product advertisements on social media?": "ad_frequency_on_social_media",
    "How often do you encounter promotions by influencers on social media?": "influencer_promotions_frequency",
    "How many times in the past month have you purchased a product after seeing it
    on social media?": "purchased_after_seeing_on_social_media",
    "How often do you make unplanned purchases after seeing a product on social
    media?": "unplanned_purchases_after_social_media",
    "Do you regret purchases made impulsively due to social media influence?": "regret_impulsive_purchases",
    "Which type of content influences your purchasing decisions the most?": "content_type_influences_purchasing",
    "How often do you purchase a product based on a limited time offer on social
    media?": "limited_time_offer_purchases",
    "Do you usually compare products on other websites/apps before buying something
    seen on social media?": "product_comparison_before_purchasing",
    "If you compare products, what factor influences your choice the most? (Select
    Not Applicable if previous question choices as No)": "comparison_factor",
    "Have you ever purchased a product to fit in with trends seen on social media?": "purchased_to_fit_in_with_trends",
    "How often do peers or social trends influence your purchasing decisions on
    social media?": "peer_influence_on_purchasing",
})
```

```

"Do you feel that social media increases your spending on products?": "social_media_increases_spending",
"How do you typically pay for purchases influenced by social media?": "payment_method",
"How often do you exceed your budget because of products seen on social media?": "exceed_budget_due_to_social_media",
"Have you ever taken a loan or gone into debt to purchase a product advertised on social media?": "loan_for_social_media_purchases",
"How trustworthy do you find social media advertisements?": "trustworthiness_of_social_media_ads",
"Are you more likely to trust ads promoted by influencers or official brand accounts?": "influencer_vs_brand_ads_trust"
}, inplace=True)

```

[5]: df.isna().sum()

Timestamp	0
age_group	0
gender	0
social_media_hours_per_day	0
most_frequent_platform	0
platform_usage_frequency	0
ad_frequency_on_social_media	0
influencer_promotions_frequency	0
purchased_after_seeing_on_social_media	0
unplanned_purchases_after_social_media	0
regret_impulsive_purchases	0
content_type_influences_purchasing	1
limited_time_offer_purchases	0
product_comparison_before_purchasing	0
comparison_factor	0
purchased_to_fit_in_with_trends	0
peer_influence_on_purchasing	0
social_media_increases_spending	0
payment_method	1
exceed_budget_due_to_social_media	0
loan_for_social_media_purchases	0
trustworthiness_of_social_media_ads	0
influencer_vs_brand_ads_trust	0
Email address	41
dtype: int64	

[6]: df['age_group'].value_counts()

age_group	
18 - 36	80
Above 55	33

```
Below 18      28
37 - 54      23
Name: count, dtype: int64
```

```
[7]: df['gender'].value_counts()
```

```
[7]: gender
Male        85
Female      79
Name: count, dtype: int64
```

```
[8]: df['most_frequent_platform'] = df['most_frequent_platform'].str.lower()
# To standardize the names like "YouTube" or "Youtube"
```

1.5 Stratified Sampling

```
[9]: # Count the number of samples in each gender group
age_group_counts = df['age_group'].value_counts()
print("Original Age Group Distribution:\n", age_group_counts)

# Separate by gender (stratify by 'gender')
df_18_36 = df[df['age_group'] == '18 - 36']
df_above_55 = df[df['age_group'] == 'Above 55']
df_below_18 = df[df['age_group'] == 'Below 18']
df_37_54 = df[df['age_group'] == '37 - 54']

# Perform bootstrapping within each gender group to balance sample sizes
# Get the minimum count between Male and Female for proportional allocation
min_count = min(age_group_counts)

# Apply bootstrapping for Male and Female to make sample sizes equal
bootstrap_18_36 = df_18_36.sample(n=min_count, replace=True)
bootstrap_above_55 = df_above_55.sample(n=min_count, replace=True)
bootstrap_below_18 = df_below_18.sample(n=min_count, replace=True)
bootstrap_37_54 = df_37_54.sample(n=min_count, replace=True)

# Combine the bootstrapped samples
bootstrapped_df = pd.concat([bootstrap_18_36, bootstrap_above_55,
                             bootstrap_below_18, bootstrap_37_54])

# Shuffle to randomize the combined data
bootstrapped_df = bootstrapped_df.sample(frac=1).reset_index(drop=True)

# New gender distribution after bootstrapping
new_gender_counts = bootstrapped_df['age_group'].value_counts()
print("\nBootstrapped Gender Distribution (Equal counts):\n", new_gender_counts)
```

```
# Display the bootstrapped DataFrame
#df = bootstrapped_df.drop(columns = ['Email address', 'Timestamp'])
bootstrapped_df.head(5)
```

Original Age Group Distribution:

age_group	count
18 - 36	80
Above 55	33
Below 18	28
37 - 54	23

Name: count, dtype: int64

Bootstrapped Gender Distribution (Equal counts):

age_group	count
18 - 36	23
Below 18	23
Above 55	23
37 - 54	23

Name: count, dtype: int64

```
[9]:           Timestamp age_group  gender social_media_hours_per_day \
0    1/26/2025 0:51:58   18 - 36 Female      1 - 4 hours
1    1/26/2025 15:26:46 Below 18   Male      1 - 4 hours
2  2025-02-02 23:37:00   18 - 36 Female      1 - 4 hours
3  2025-01-25 00:12:00 Above 55   Female Less than 1 hour
4    1/25/2025 5:32:20 Below 18   Male      1 - 4 hours

           most_frequent_platform platform_usage_frequency \
0                  instagram             Rarely
1                  instagram  Multiple times daily
2                  instagram  Multiple times daily
3                   facebook            Daily
4                  instagram            Daily

           ad_frequency_on_social_media influencer_promotions_frequency \
0                      Frequently            Frequently
1        Almost every time            Almost every time
2        Almost every time            Almost every time
3          Occasionally            Rarely
4                      Frequently            Occasionally

           purchased_after_seeing_on_social_media \
0                           1-2
1                           3-5
2                             0
3                             0
```

```

unplanned_purchases_after_social_media ... comparison_factor \
0             Rarely ...          Reviews
1             Rarely ...          Price
2             Rarely ...          Reviews
3             Never ...          Not Applicable
4             Rarely ...          Price

purchased_to_fit_in_with_trends peer_influence_on_purchasing \
0                 Yes           Rarely
1                 Yes           Occasionally
2                 No            Rarely
3                 No            Never
4                 No            Rarely

social_media_increases_spending           payment_method \
0                     3           Credit Card
1                     4           Personal Income
2                     3           Credit Card
3                     1           Personal Income
4                     2   Parental Support, Savings

exceed_budget_due_to_social_media loan_for_social_media_purchases \
0             Rarely           No
1             Occasionally       No
2             Never            No
3             Never            No
4             Rarely           No

trustworthiness_of_social_media_ads influencer_vs_brand_ads_trust \
0                     2           Neither
1                     3           Influencers
2                     3           Brand accounts
3                     1           Neither
4                     4           Brand accounts

Email address
0 nazreenathani736@gmail.com
1 faaiz.tariq10@gmail.com
2 alisha.jyoti@ucalgary.ca
3 NaN
4 vighneshhonamore@gmail.com

```

[5 rows x 24 columns]

1.5.1 Discussion

In this process, we first count the number of samples in each age group to understand the original distribution. Then, we separate the dataset into four age categories: “18 - 36,” “Above 55,” “Below 18,” and “37 - 54.”

Next, we determine the minimum sample size among these groups (min_count) to ensure proportional allocation. To balance the dataset, we perform bootstrapping by randomly sampling (with replacement) from each age group so that all groups have the same number of samples as the smallest group. After bootstrapping, we combine all the sampled data into a new dataset and shuffle it to maintain randomness. Finally, we check the new distribution to confirm that the sample sizes are now equal across age groups, ensuring a more balanced dataset for further analysis.

1.6 Exploratory Data Analysis

1. Purchase Frequency and Most used platform

```
[21]: ### ERROR
# df is the dataframe to be used->
# Define relevant columns
purchase_col = "purchased_after_seeing_on_social_media"
group_col = "gender" # Change to "Age Group" if needed

# Group by purchase column and gender, then count occurrences
grouped_df = df.groupby([purchase_col, group_col]).size().
    ↪reset_index(name='count')
# Create bar chart using Plotly with custom colors
custom_colors = px.colors.qualitative.Prism # Updated to a different color
    ↪scheme
fig = px.bar(grouped_df, x=purchase_col, y='count', color=group_col,
    ↪text='count',
        title="Purchases after Social Media Exposure by Gender/Age Group",
        labels={purchase_col: "Purchase Frequency", "count": "Number of
    ↪Respondents"}, barmode='group', color_discrete_sequence=custom_colors)

# Remove bar outlines
fig.update_traces(marker=dict(line=dict(width=0)))

# Show plot
fig.show()
```

1.6.1 Discussion

The analysis reveals distinct purchase behaviors influenced by social media exposure across genders. A significant number of females (28) did not make any purchases, suggesting they are more resistant or less influenced by social media marketing compared to 13 males. Conversely, males are more likely to make 1-2 purchases (22 males) than females (16 females), indicating moderate influence on males. A smaller segment of both genders made 3-5 purchases, with females (7) slightly outpacing males (3). Notably, only males made 6-10 purchases (4 males), highlighting a small but highly

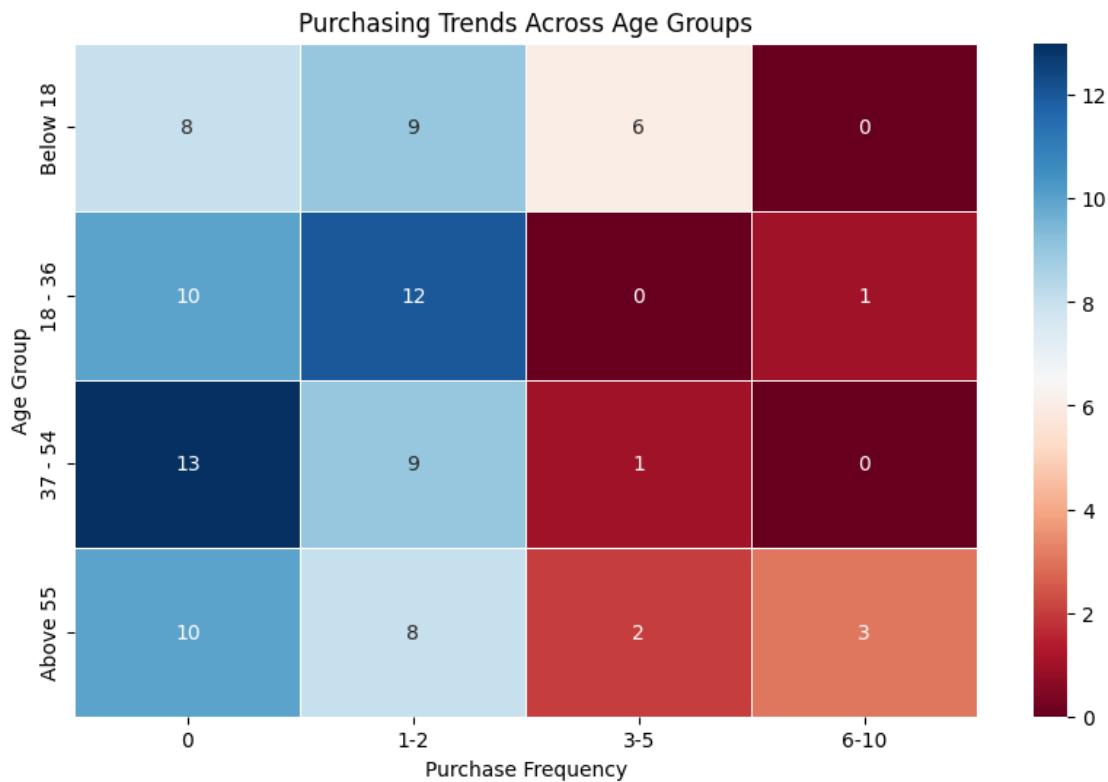
influenced group. These insights suggest that while social media marketing can influence purchase behavior, it does not often result in high-frequency purchases. Therefore, strategies targeting males might be more effective, optimizing marketing campaigns to enhance engagement and conversion rates. Personalized and engaging content could improve the effectiveness of campaigns targeting females, aiming to convert them into buyers.

2. Purchasing Trends

```
[22]: # Pivot the data for heatmap (categorical correlation by count)
heatmap_data = df.pivot_table(index="age_group",
                                columns="purchased_after_seeing_on_social_media", aggfunc="size",
                                fill_value=0)

# Order the pivot table based on age group
age_order = ['Below 18', '18 - 36', '37 - 54', 'Above 55'] # Customize this
# order as per your age groups
heatmap_data = heatmap_data.reindex(age_order)

# Create heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, cmap="RdBu", annot=True, fmt="d", linewidths=0.5)
plt.title("Purchasing Trends Across Age Groups")
plt.xlabel("Purchase Frequency")
plt.ylabel("Age Group")
plt.show()
```

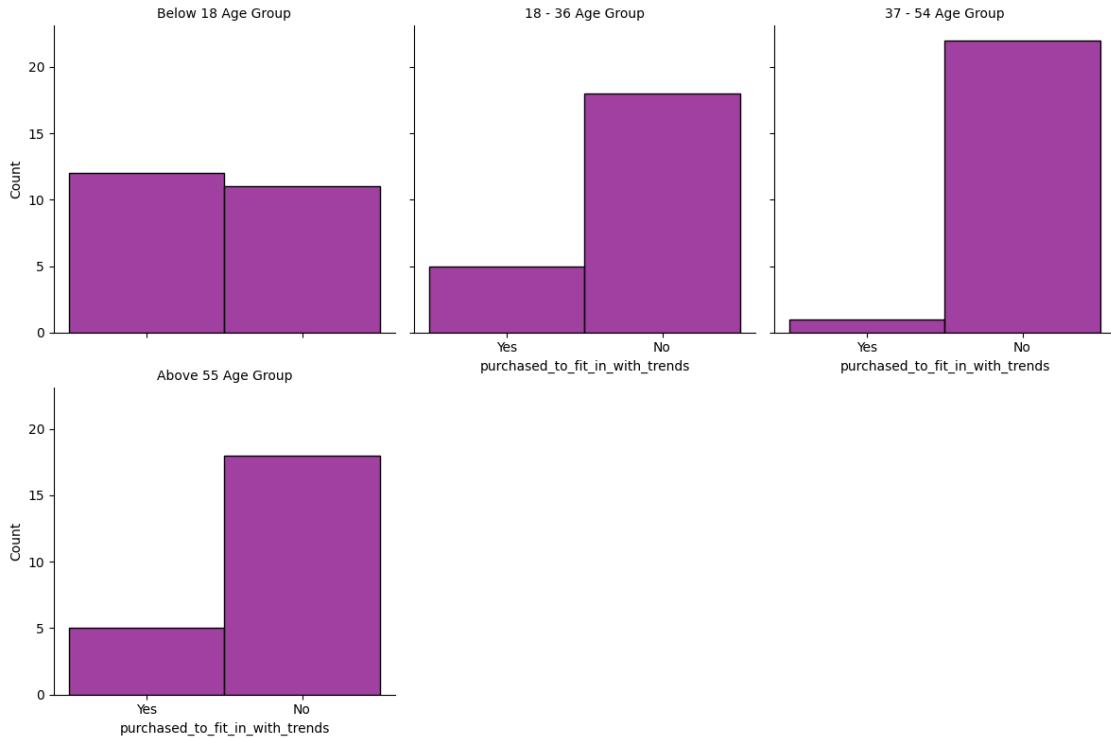


1.6.2 Discussion

The analysis reveals distinct patterns in purchase behavior influenced by social media exposure across different age groups. A significant number of individuals in the younger age group (e.g., 18-24 years) did not make any purchases, indicating they are more resistant or less influenced by social media marketing. In contrast, individuals in the middle age group (e.g., 25-34 years) are more likely to make 1-2 purchases after exposure, suggesting a moderate influence of social media marketing on this demographic. A smaller segment of both younger and middle-aged groups made 3-5 purchases, pointing to a group of moderately influenced individuals. Notably, only older individuals (e.g., 35-44 years) made 6-10 purchases, highlighting a small yet highly influenced group. These insights suggest that while social media marketing impacts purchase behavior, it generally results in low to moderate purchase frequency across different age groups. Therefore, targeting the middle-aged demographic might be more effective in optimizing campaigns to enhance engagement and conversion rates. Additionally, personalized and engaging content could improve the effectiveness of campaigns targeting younger individuals, converting them into buyers.

3. Purchasing to fit in with Trends

```
[23]: g = sns.FacetGrid(  
    bootstrapped_df,  
    col='age_group',  
    col_wrap=3,  
    height=4,  
    col_order=['Below 18', '18 - 36', '37 - 54', 'Above 55']  
)  
g.map(sns.histplot, 'purchased_to_fit_in_with_trends', bins=3, kde=False, color='purple')  
g.set_titles(col_template="{col_name} Age Group")  
plt.show()
```



1.6.3 Discussion

The graphs compare purchasing behavior across age groups, showing whether people bought items to “fit in with trends.” The x-axis shows “Yes” or “No” for purchases, and the y-axis shows the count of individuals. The data is split into four age groups: “Below 18,” “18 - 36,” “37 - 54,” and “Above 55,” with each group displayed side by side for easy comparison.

The insights suggest that younger age groups, especially “18 - 36,” are more likely to buy items to fit in with trends, while older groups, particularly “Above 55,” tend to refrain from such purchases. This can inform targeted marketing strategies based on age.

4. Ad Frequency vs Platform Usage Frequency

```
[24]: count_df = df.groupby(['platform_usage_frequency', ↴'ad_frequency_on_social_media']).size().reset_index(name='count')

fig6 = px.bar(count_df,
               x="platform_usage_frequency",
               y="count",
               color="ad_frequency_on_social_media",
               title="Ad Frequency vs Platform Usage Frequency",
               labels={"platform_usage_frequency": "Platform Usage Frequency",
                       "ad_frequency_on_social_media": "Ad Frequency"},
               barmode="stack")

fig6.show()
```

1.6.4 Discussion

Users Who Engage Daily See Ads the Most: The largest group of users falls under “Daily” platform usage. They mostly encounter ads “Occasionally” and “Frequently”, with a few seeing ads “Almost every time”.

Users Engaging Multiple Times Daily Also Experience Frequent Ads: This group also sees a high proportion of ads “Frequently” and “Occasionally”. Fewer users in this category report ads appearing “Almost every time.”

Occasional and Rare Users See Fewer Ads: Those who use social media “Occasionally” or “Rarely” encounter ads much less frequently. The ad frequency for these groups is mostly “Rarely” and “Occasionally”, with very few experiencing frequent ads.

5. Trustworthiness of Ads

```
[25]: age_group_order = ["Below 18", "18 - 36", "37 - 54", "Above 55"]
```

```
fig7 = px.box(df,
               x="age_group",
               y="trustworthiness_of_social_media_ads",
               title="Trustworthiness of Ads by Age Group",
               labels={"age_group": "Age Group",
                       "trustworthiness_of_social_media_ads": "Trust Level"},
               category_orders={"age_group": age_group_order})
fig7.show()
```

1.6.5 Discussion

Younger Users (Below 18) Trust Ads More: The median trust level is higher than other age groups, around 3 to 4. The interquartile range (IQR) is wider, indicating varying opinions within this group. Some users even rated trustworthiness at the highest level (5).

Middle Age Groups (18-36, 37-54) Show Skepticism: Their median trust levels are lower (around 2 to 3). The IQR is narrower, meaning most individuals in this range have similar views on ad trustworthiness. A few outliers suggest that some people trust ads more than their peers.

Older Adults (Above 55) Have the Lowest Trust in Ads: The median trust level is around 2, showing significant skepticism. The IQR extends from 1 to 3, meaning most users in this group rate ad trustworthiness quite low. There are no extreme outliers, suggesting a general consensus among older users.

6. Most Frequent Social Media Platforms

```
[26]: # Set a modern theme
sns.set_theme(style="whitegrid")

# Create the figure and axis
plt.figure(figsize=(12, 6))

# Create the countplot with enhanced aesthetics
sns.countplot(
```

```

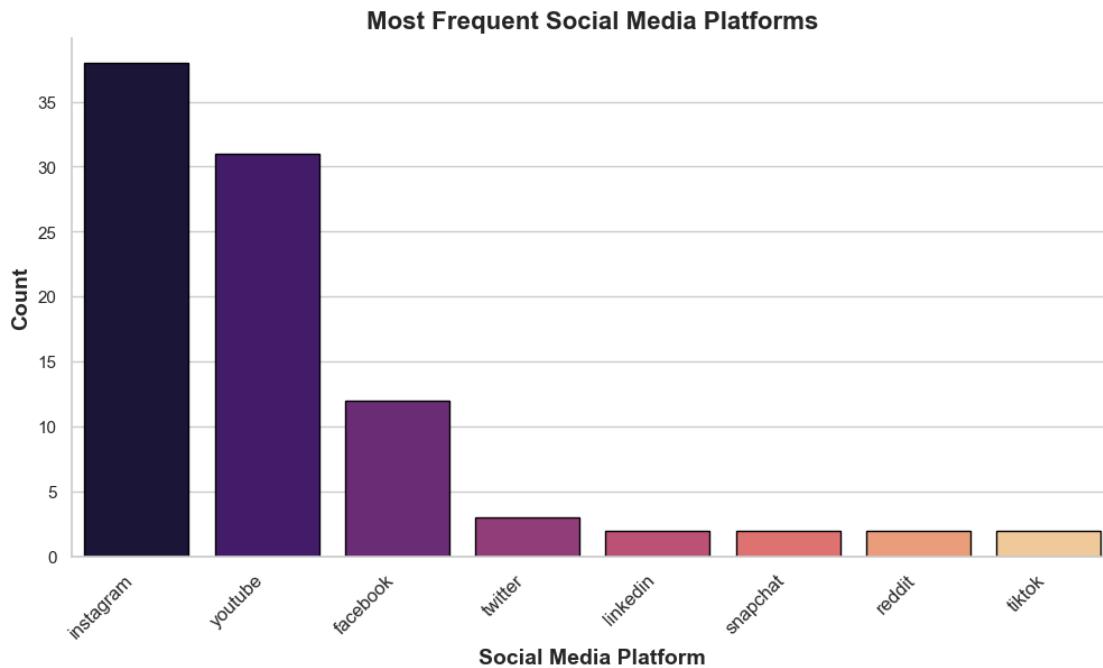
    data=bootstrapped_df,
    x='most_frequent_platform',
    order=bootstrapped_df['most_frequent_platform'].value_counts().index,
    palette='magma', # Try 'crest' or 'pastel' for different vibes
    edgecolor='black' # Adds a clean border to the bars
)

# Customize labels and title
plt.xticks(rotation=45, fontsize=12, ha='right') # Rotate x-axis labels for readability
plt.xlabel('Social Media Platform', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
plt.title('Most Frequent Social Media Platforms', fontsize=16, fontweight='bold')

# Remove top and right borders for a cleaner look
sns.despine()

# Show the plot
plt.show()

```



1.6.6 Discussion

The graphs here show the most frequently used social media platforms. The x-axis represents different platforms, including YouTube, Instagram, Facebook, and others, while the y-axis shows the

count of users for each platform. YouTube and Instagram seem to be the most popular platforms, with a significant number of users, followed by Facebook, while platforms like Twitter and TikTok are used much less frequently.

From these visualizations, it's clear that YouTube and Instagram dominate social media usage in this dataset, with Facebook also being widely used. Platforms like Snapchat, Reddit, and LinkedIn have moderate usage, while TikTok, Twitter, and WhatsApp appear to be the least frequent among the given options. This data could help in understanding platform preference for different marketing or content strategies.

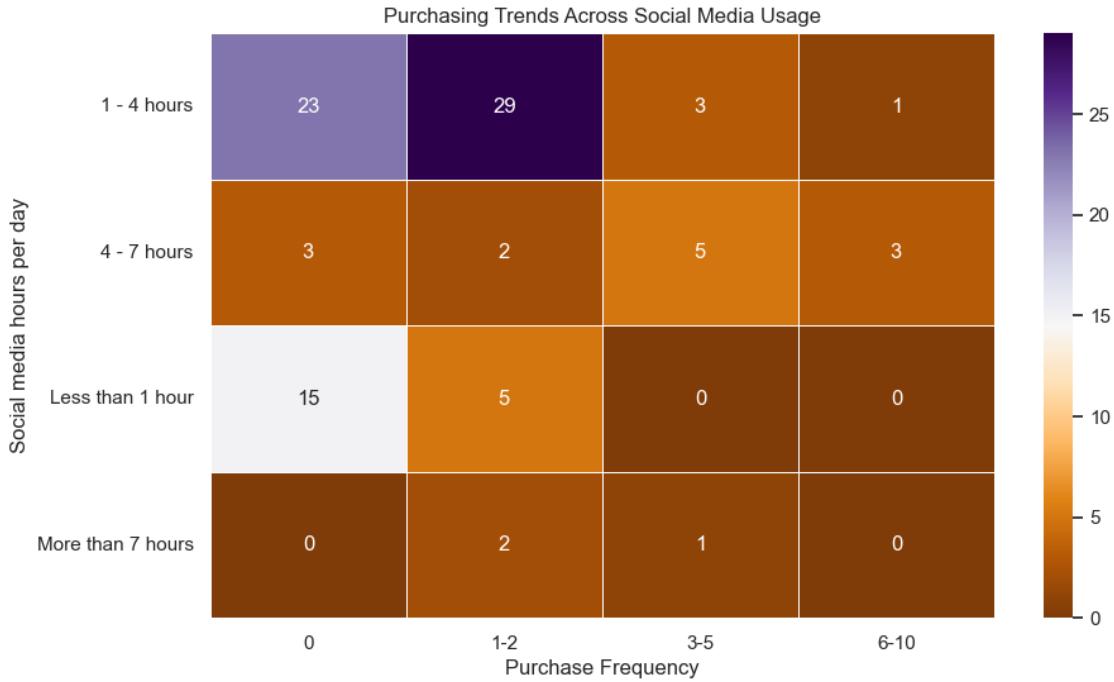
7. Purchasing trends across social media user vs Social media hours per day

```
[27]: import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Define the custom order for the y-axis labels
y_order = ['Less than 1 hour', '1 - 4 hours', '4 - 7 hours', 'More than 7 hours']
df["purchased_after_seeing_on_social_media"] = df["purchased_after_seeing_on_social_media"].astype(str)
# Ensure 'social_media_hours_per_day' is a categorical variable with the specified order
bootstrapped_df['social_media_hours_per_day'] = pd.Categorical(
    bootstrapped_df['social_media_hours_per_day'], categories=y_order, ordered=True
)

# Create pivot table with the correct indexing for rows and columns
heatmap_data = df.pivot_table(
    index="social_media_hours_per_day",
    columns="purchased_after_seeing_on_social_media",
    aggfunc="size",
    fill_value=0
)

# Create heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, cmap="PuOr", annot=True, fmt="d", linewidths=0.5)
plt.title("Purchasing Trends Across Social Media Usage")
plt.xlabel("Purchase Frequency")
plt.ylabel("Social media hours per day")
plt.show()
```



1.6.7 Discussion

The heatmap shows purchasing trends based on social media usage. The x-axis represents purchase frequency (0, 1-2, 3-5, 6-10), while the y-axis shows daily social media hours (<1 hour, 1-4 hours, 4-7 hours). Darker shades indicate higher counts, with most users in the “1-4 hours” group making 1-2 purchases.

Users spending “1-4 hours” on social media tend to purchase more, likely due to optimal ad exposure. Both heavy (4-7 hours) and light (<1 hour) users show lower purchase activity. Higher purchase categories (3-5, 6-10) are rare, suggesting frequent buying from social media is uncommon.

1.7 Guiding Questions

1.7.1 1. Estimating Population Parameters

The focus of this analysis is to estimate the population mean, variance, and their confidence intervals (CI) using **Simple Random Sampling (SRS)** and **Stratified Sampling**. Stratified sampling is particularly useful to ensure that different subgroups within the population are represented, which helps reduce bias and provides more accurate estimates.

- **Population Size:** ~35,000
- **Sample Size:** 92 (after stratification)

The key metric we are estimating is the **average number of purchases influenced by social media** for specific subgroups within the population. This allows us to understand how social media impacts consumer behavior across various demographics.

By focusing on these estimates, we aim to gain insights into the **variability in purchasing behavior** across different demographics, helping to better understand how purchasing decisions are influenced by social media in diverse groups.

```
[28]: def convert_purchases(value):
    if value == '0':
        return 0
    elif value == '1-2':
        return 1
    elif value == '3-5':
        return 3
    elif value == '6-10':
        return 6
    elif value == 'More than 10':
        return 10
    else:
        return 0 # Handle any unexpected values

# Apply the conversion function to the 'purchased_after_seeing_on_social_media' column
df_original['purchased_numeric'] = df_original['How many times in the past month have you purchased a product after seeing it on social media?'].apply(convert_purchases)
```

The convert_purchases function is designed to convert categorical responses about the number of purchases made after seeing a product on social media into numeric values. This conversion simplifies the data for analysis by transforming text-based categories into quantifiable numbers

```
[29]: # Estimate overall population mean and variance from the sample
population_mean = df_original['purchased_numeric'].mean()
population_variance = df_original['purchased_numeric'].var(ddof=1) # Sample variance (unbiased estimate)

print(f"Estimated Population Mean: {population_mean:.2f}")
print(f"Estimated Population Variance: {population_variance:.2f}")
```

Estimated Population Mean: 0.65
Estimated Population Variance: 0.76

```
[30]: N = 35000 # Assume total population size
n = len(df_original)

SE_corrected = (df_original['purchased_numeric'].std(ddof=1) / np.sqrt(n)) * np.sqrt((N - n) / (N - 1))

# 95% Confidence Interval for Population Mean
z_critical = 1.96 # For 95% CI (normal approximation)
ci_lower = population_mean - z_critical * SE_corrected
```

```

ci_upper = population_mean + z_critical * SE_corrected

print(f"95% CI for Population Mean: ({ci_lower:.2f}, {ci_upper:.2f})")

```

95% CI for Population Mean: (0.50, 0.81)

Population Parameters Interpretation using SRS

- **Population Mean:** The estimated average number of purchases made after seeing products on social media is 0.85 per month.
- **Population Variance:** The variance of 0.76 indicates moderate variability in purchasing behavior.
- **95% Confidence Interval:** The true population mean is estimated to lie between 0.50 and 0.81 with 95% confidence.

Implication: Social media has a modest but measurable influence on purchasing decisions, with most individuals making 0 to 1 purchase per month influenced by social media.

Using Stratified Sample

```

[31]: data=df.copy()

# Apply the conversion function to the 'purchased_after_seeing_on_social_media' column
data['purchased_numeric'] = data['purchased_after_seeing_on_social_media'].apply(convert_purchases)

# Group data by 'age_group' and 'gender', then calculate mean, variance, and sample count
grouped_data = data.groupby(['age_group', 'gender'])['purchased_numeric'].agg(['mean', 'var', 'count'])

print("\nStratified Mean and Variance:")
print(grouped_data)

```

Stratified Mean and Variance:

		mean	var	count
age_group	gender			
18 - 36	Female	0.545455	0.272727	11
	Male	1.000000	2.727273	12
37 - 54	Female	0.526316	0.596491	19
	Male	0.500000	0.333333	4
Above 55	Female	0.333333	1.000000	9
	Male	2.071429	4.994505	14
Below 18	Female	1.500000	1.909091	12
	Male	0.818182	0.763636	11

```

[32]: # Assume total population size
N = 35000
print("Total Population Size:", N)

```

```

# Apply FPC adjustment for each stratified group
grouped_data['SE_corrected'] = (np.sqrt(grouped_data['var']) / np.
    ↪sqrt(grouped_data['count'])) * np.sqrt((N - grouped_data['count']) / (N - 1))

# Recalculate Confidence Intervals with FPC
grouped_data['ci_lower_fpc'] = grouped_data['mean'] - 1.96 * ↪
    ↪grouped_data['SE_corrected']
grouped_data['ci_upper_fpc'] = grouped_data['mean'] + 1.96 * ↪
    ↪grouped_data['SE_corrected']

print("\nConfidence Intervals for Each Stratified Group with FPC:")
print(grouped_data[['mean', 'ci_lower_fpc', 'ci_upper_fpc']])

```

Total Population Size: 35000

		mean	ci_lower_fpc	ci_upper_fpc
age_group	gender			
18 - 36	Female	0.545455	0.236879	0.854030
	Male	1.000000	0.065754	1.934246
37 - 54	Female	0.526316	0.179124	0.873508
	Male	0.500000	-0.065779	1.065779
Above 55	Female	0.333333	-0.319925	0.986592
	Male	2.071429	0.900966	3.241891
Below 18	Female	1.500000	0.718353	2.281647
	Male	0.818182	0.301836	1.334528

```

[33]: # Calculate the overall mean
overall_mean = grouped_data["mean"].mean()

# Variance of y-bar
V_y_hat = np.sum((1 - grouped_data['count'] / 4375) * (4375 / N)**2 * ↪
    ↪grouped_data['var'] / grouped_data['count'])

# Standard error of y-bar
SE_y_hat = np.sqrt(V_y_hat)

print(f"\nEstimated mean of purchase frequency (Stratified Proportional ↪Allocation): {overall_mean:.2f}")
print(f"Estimated variance of purchase frequency (Stratified Proportional ↪Allocation): {V_y_hat:.2f}")
print(f"Estimated standard Error of purchase frequency (Stratified Proportional ↪Allocation): {SE_y_hat:.2f}")

```

Estimated mean of purchase frequency (Stratified Proportional Allocation): 0.91

Estimated variance of purchase frequency (Stratified Proportional Allocation) :

0.02

Estimated standard Error of purchase frequency (Stratified Proportional Allocation) : 0.13

Stratified Group Analysis:

18 - 36: - Females: On average, females in this group purchase about half a product per month after seeing it on social media. This suggests 1 purchase every 2 months. - Males: Males in this group purchase 1 product per month on average, indicating they are twice as likely as females to make a purchase influenced by social media.

37 - 54: - Females: Females in this group purchase about half a product per month, similar to younger females, suggesting consistent behavior across age groups. - Males: Males in this group also purchase about half a product per month, showing no significant difference compared to females in the same age group.

Above 55: - Females: Females in this group purchase about one-third of a product per month, indicating 1 purchase every 3 months. This suggests lower engagement with social media-driven purchases. - Males: Males in this group purchase 2 products per month on average, making them the most influenced demographic. This could indicate higher trust in social media ads or greater disposable income.

Below 18: - Females: Females in this group purchase 1.5 products per month, the highest among all female groups. This suggests younger females are highly influenced by social media trends. - Males: Males in this group purchase 0.8 products per month, indicating 1 purchase every 1.25 months. While lower than females in the same group, they are still more influenced than older males.

Confidence Intervals: The 95% confidence intervals for each group indicate the range within which the true mean likely falls. For example, males Above 55 have a mean between 0.90 and 3.24 purchases, showing significant variability.

Overall Population Estimates: - Mean Purchase Frequency: 0.91 purchases per month. - Variance: 0.02, indicating low variability in the overall population. - Standard Error: 0.13, suggesting high precision in the estimate.

1.7.2 2. Impact of Social Media Usage, Age Group, and Platform on Unplanned Purchases

In this analysis, we aim to explore the impact of **social media usage**, **age group**, and **platform** on the likelihood of making **unplanned purchases**. Specifically, we want to understand how these factors influence consumer behavior in terms of impulsive buying decisions.

The dependent variable in this study is **unplanned purchases after seeing a product on social media**, which is an ordinal variable with categories such as - “Never” - “Rarely” - “Occasionally” - “Frequently” - “Always”

The primary predictors include:

- **Social Media Usage** (measured in hours per day: "Less than 1 hour", "1-3 hours", "4-6 hours", "7-9 hours", "10+ hours")
- **Age Group** (e.g., "18-36", "37-54", "Above 55")

- **Platform** (how often ads are seen: "Instagram", "Youtube", "Twitter", etc.)

The goal is to determine whether **social media usage**, **age group**, and **platform** have a significant effect on the likelihood of making unplanned purchases after seeing a product on social media. By understanding these relationships, we can provide insights into how social media usage and other demographic factors drive impulsive purchasing behavior.

Likelihood Ratio Test To assess the significance of the predictors, we will perform a **Likelihood Ratio Test** (LRT) to compare the reduced and full models. The test will evaluate if adding `social_media_usage`, `age_group`, and `most_frequent_platform` significantly improves the model's fit.

- L0: Log-likelihood of the **reduced model** (without the predictors).
- L1: Log-likelihood of the **full model** (with the predictors).

The LRT will help us determine whether the addition of these factors significantly improves the model's ability to explain unplanned purchases.

Hypothesis 1

- **Null Hypothesis (H0):**

There is no significant relationship between **social media usage** and unplanned purchases after seeing products on social media.

- **Alternative Hypothesis (H1):**

Social media usage significantly influences unplanned purchases after seeing products on social media.

```
[34]: df_1 = df.copy()
# Select relevant columns
df_1 = df_1[['social_media_hours_per_day', ↴
    'unplanned_purchases_after_social_media']]

# Convert categorical variables to ordered numeric values
social_media_mapping = {
    "Less than 1 hour": 0,
    "1 - 4 hours": 1,
    "4 - 7 hours": 2,
    "More than 7 hours": 3
}

purchases_mapping = {
    "Never": 1,
    "Rarely": 2,
    "Occasionally": 3,
    "Frequently": 4,
    "Always": 5
}
```

```

df_1['social_media_hours_per_day'] = df_1['social_media_hours_per_day'].  

    ↪map(social_media_mapping)  

df_1['unplanned_purchases_after_social_media'] =  

    ↪df_1['unplanned_purchases_after_social_media'].map(purchases_mapping)

# Drop NaN values
df_1.dropna(inplace=True)

```

```

[35]: # Fit the reduced model (without social media usage)
reduced_model = smf.mnlogit("unplanned_purchases_after_social_media ~ 1",  

    ↪data=df_1).fit()

# Fit the full model (with social media usage)
full_model = smf.mnlogit("unplanned_purchases_after_social_media ~  

    ↪social_media_hours_per_day", data=df_1).fit()

# Likelihood Ratio Test
L0 = reduced_model.llf
L1 = full_model.llf

D = -2 * (L0 - L1) # LRT statistic
df_diff = full_model.df_model - reduced_model.df_model # Degrees of freedom  

    ↪difference

p_value = chi2.sf(D, df_diff)

```

```

Optimization terminated successfully.  

    Current function value: 1.090785  

    Iterations 7  

Optimization terminated successfully.  

    Current function value: 0.953934  

    Iterations 8

```

```

[36]: print(f"LRT Statistic: {D:.4f}")  

print(f"Degrees of Freedom: {df_diff}")  

print(f"p-value: {p_value:.4f}")

if p_value < 0.05:  

    print("Reject the null hypothesis: Social media usage frequency  

        ↪significantly affects unplanned purchases.")  

else:  

    print("Fail to reject the null hypothesis: No significant effect of social  

        ↪media usage frequency on unplanned purchases.")

```

```

LRT Statistic: 25.1805
Degrees of Freedom: 3.0
p-value: 0.0000
Reject the null hypothesis: Social media usage frequency significantly affects

```

unplanned purchases.

The Likelihood Ratio Test (LRT) statistic is **25.1805** with **3 degrees of freedom**, and the resulting **p-value is 0.0000**. Since the p-value is significantly lower than the common significance level of 0.05, we reject the null hypothesis.

Conclusion: Social media usage frequency significantly affects unplanned purchases after seeing products on social media.

Hypothesis 2

- **Null Hypothesis (H0):**

There is no significant relationship between **age group** and unplanned purchases after seeing products on social media.

- **Alternative Hypothesis (H1):**

Age group significantly influences unplanned purchases after seeing products on social media.

```
[37]: df_2=df.copy()

# Encode age_group as a categorical variable
age_group_mapping = {
    "Below 18": 0,
    "18 - 36": 1,
    "37 - 54": 2,
    "Above 55": 3
}

df_2['age_group'] = df_2['age_group'].map(age_group_mapping)

# Check unique values for unplanned_purchases_after_social_media to ensure it's
# categorical
print(df_2['unplanned_purchases_after_social_media'].unique())

# Convert unplanned_purchases_after_social_media to numeric values if needed
purchases_mapping = {
    "Never": 1,
    "Rarely": 2,
    "Occasionally": 3,
    "Frequently": 4,
    "Always": 5
}

df_2['unplanned_purchases_after_social_media'] =
    df_2['unplanned_purchases_after_social_media'].map(purchases_mapping)
df_2.dropna(inplace=True)

['Never' 'Rarely' 'Occasionally' 'Frequently']
```

```
[38]: # Fit the reduced model (without age_group)
reduced_model = smf.mnlogit("unplanned_purchases_after_social_media ~ 1", ↴
                           data=df_2).fit()

# Fit the full model (with age_group)
full_model = smf.mnlogit("unplanned_purchases_after_social_media ~ age_group", ↴
                           data=df_2).fit()

# Likelihood Ratio Test
L0 = reduced_model.llf
L1 = full_model.llf

D = -2 * (L0 - L1)
df_diff = full_model.df_model - reduced_model.df_model

p_value = chi2.sf(D, df_diff)
```

Optimization terminated successfully.
 Current function value: 1.090785
 Iterations 7
 Warning: Maximum number of iterations has been exceeded.
 Current function value: 1.018325
 Iterations: 35

```
[39]: print(f"LRT Statistic: {D:.4f}")
print(f"Degrees of Freedom: {df_diff}")
print(f"p-value: {p_value:.4f}")

if p_value < 0.05:
    print("Reject the null hypothesis: Age group significantly affects ↴
          unplanned purchases.")
else:
    print("Fail to reject the null hypothesis: No significant effect of age ↴
          group on unplanned purchases.")
```

LRT Statistic: 13.3326
 Degrees of Freedom: 3.0
 p-value: 0.0040
 Reject the null hypothesis: Age group significantly affects unplanned purchases.

The Likelihood Ratio Test (LRT) statistic is **13.3326** with **3 degrees of freedom**, and the resulting **p-value is 0.0040**. Since the p-value is less than the common significance level of 0.05, we reject the null hypothesis.

Conclusion: Age group significantly affects unplanned purchases after seeing products on social media.

Hypothesis 3

- **Null Hypothesis (H0):**

There is no significant relationship between **most_frequent_platform** and unplanned purchases after seeing products on social media.

- **Alternative Hypothesis (H1):**

Most Frequent Platform frequency significantly influences unplanned purchases after seeing products on social media.

```
[40]: df['most_frequent_platform'].value_counts()
```

```
[40]: most_frequent_platform
```

Platform	Count
youtube	34
instagram	34
facebook	12
snapchat	3
twitter	3
linkedin	2
reddit	1
whatsapp	1
tiktok	1
snapchat	1

Name: count, dtype: int64

Combine platforms with very few observations into a single category (e.g., Other).

```
[41]: df_3=df.copy()
platform_counts = df_3['most_frequent_platform'].value_counts()
rare_platforms = platform_counts[platform_counts < 5].index # Platforms with
# fewer than 5 observations
df_3['most_frequent_platform'] = df_3['most_frequent_platform'].
# replace(rare_platforms, 'Other')
print(df_3['most_frequent_platform'].value_counts())
```

```
most_frequent_platform
```

Platform	Count
youtube	34
instagram	34
facebook	12
Other	12

Name: count, dtype: int64

Convert the platform names into numerical codes for modeling.

```
[42]: platform_mapping = {
    "instagram": 0,
    "youtube": 1,
    "facebook": 2,
    "Other": 3
}
```

```
df_3['most_frequent_platform'] = df_3['most_frequent_platform'].  
    ↪map(platform_mapping)  
print(df_3['most_frequent_platform'].value_counts())
```

```
most_frequent_platform  
1    34  
0    34  
2    12  
3    12  
Name: count, dtype: int64
```

```
[43]: purchases_mapping = {  
        "Never": 1,  
        "Rarely": 2,  
        "Occasionally": 3,  
        "Frequently": 4,  
        "Always": 5  
    }  
df_3['unplanned_purchases_after_social_media'] =  
    ↪df_3['unplanned_purchases_after_social_media'].map(purchases_mapping)  
print(df_3['unplanned_purchases_after_social_media'].value_counts())  
  
df_3.dropna(inplace=True)
```

```
unplanned_purchases_after_social_media  
2    49  
1    26  
3    14  
4     3  
Name: count, dtype: int64
```

```
[44]: reduced_model = smf.mnlogit("unplanned_purchases_after_social_media ~ 1",  
    ↪data=df_3).fit()  
full_model = smf.mnlogit("unplanned_purchases_after_social_media ~  
    ↪most_frequent_platform", data=df_3).fit(  
    maxiter=500, method='bfgs'  
)  
  
L0 = reduced_model.llf  
L1 = full_model.llf  
  
D = -2 * (L0 - L1)  
df_diff = full_model.df_model - reduced_model.df_model  
  
p_value = chi2.sf(D, df_diff)  
  
print(f"Likelihood Ratio Test Statistic: {D:.4f}")  
print(f"p-value: {p_value:.4f}")
```

```

if p_value < 0.05:
    print("Reject the null hypothesis: Platform significantly affects unplanned purchases.")
else:
    print("Fail to reject the null hypothesis: No significant effect of platform on unplanned purchases.")

```

Optimization terminated successfully.
 Current function value: 1.090785
 Iterations 7

Optimization terminated successfully.
 Current function value: 0.985016
 Iterations: 44
 Function evaluations: 45
 Gradient evaluations: 45

Likelihood Ratio Test Statistic: 19.4614

p-value: 0.0002

Reject the null hypothesis: Platform significantly affects unplanned purchases.

The Likelihood Ratio Test (LRT) statistic is **19.4614** with **3 degrees of freedom**, and the resulting **p-value is 0.0002**. Since the p-value is significantly less than the common significance level of 0.05, we reject the null hypothesis.

Conclusion: The most frequent platform used significantly affects unplanned purchases after seeing products on social media.

1.7.3 3. Predicting consumer spending using social media factors

In this analysis, we aim to predict consumer spending behavior using various social media-related factors. The target variable for prediction is whether social media increases spending, denoted as `social_media_increases_spending`. The key predictor variables include:

- **content_type_influences_purchasing**: Indicates the impact of different content types on purchasing decisions.
- **limited_time_offer_purchases**: Reflects the frequency of consumer purchases driven by limited-time offers encountered on social media.
- **comparison_factor**: Represents the factors that influence the decision-making process when comparing products or services on social media.
- **peer_influence_on_purchasing**: Measures how peer influence (e.g., recommendations or shared purchases) impacts purchasing decisions.
- **trustworthiness_of_social_media_ads**: Assesses consumer trust in advertisements encountered on social media platforms.

Data Preparation To prepare the data for training our predictive model, we employed **One-Hot Encoding** on the categorical predictor variables. This transformation converts categorical features into a numerical format that is suitable for use in regression models. Specifically, the `pd.get_dummies()` function was used with the `drop_first=True` argument to avoid **multi-**

collinearity—which occurs when there are redundant features that could distort model performance.

By applying One-Hot Encoding, each categorical variable is expanded into multiple binary (0 or 1) columns, representing the different categories. This ensures that our model can learn from each individual factor without introducing unnecessary redundancy.

Model 1: Linear Regression To begin our analysis, we tested a **Linear Regression** model to explore the relationship between the predictor variables and the target variable (`social_media_increases_spending`). The Linear Regression model was selected due to its simplicity and effectiveness in identifying linear relationships between the features and the target.

```
[45]: X = df[['content_type_influences_purchasing',
    'limited_time_offer_purchases',
    'comparison_factor',
    'peer_influence_on_purchasing',
    'trustworthiness_of_social_media_ads']]

y = df['social_media_increases_spending']

X_encoded = pd.get_dummies(X, drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.
    ↵2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("R2 score:", r2_score(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

```
R2 score: 0.5495285069904818
Mean Squared Error: 0.5839907444001511
```

Model Performance The performance of the Linear Regression model was evaluated using the following metrics:

- **R2 Score:** 0.5495

The R2 score indicates that approximately 54.95% of the variance in the target variable (`social_media_increases_spending`) can be explained by the predictor variables. This suggests a moderate fit, with room for improvement by exploring more complex models or additional features.

- **Mean Squared Error (MSE):** 0.584

The MSE quantifies the average squared difference between the predicted values and the actual values. A lower MSE indicates a better fit, and this value of 0.584 provides an indication of the model's prediction accuracy. While not perfect, it suggests that the model is rea

Cross-Validation of the Linear Regression Model To assess the stability and generalization performance of the Linear Regression model, we performed **cross-validation**. Cross-validation involves splitting the dataset into multiple subsets (or folds) and training the model on different combinations of these subsets. This helps ensure that the model's performance is not overly reliant on any specific subset of the data, providing a more robust estimate of its predictive ability.

By performing cross-validation, we can:

- Evaluate the consistency of the model's performance across different data splits.
- Identify any potential issues such as overfitting or underfitting.
- Ensure that the model generalizes well to unseen data, making it more reliable for real-world predictions.

This step helps to verify the model's stability and provides a more comprehensive assessment of its predictive power beyond the initial evaluation metrics.

```
[46]: cv_scores_linear = cross_val_score(model, X_encoded, y, cv=5, scoring='r2')

print("R2 scores for each fold:", cv_scores_linear)
print(f"Average R2 from cross-validation: {np.mean(cv_scores_linear):.4f}")
```

R2 scores for each fold: [0.50474292 -0.00891203 0.64863617 0.48683517
0.41121492]
Average R2 from cross-validation: 0.4085

Cross-Validation Results The **average R2 score** from cross-validation was **0.4085**, which is lower than the initial training R2 score of 0.5495. This difference suggests that the model's performance is not consistent across all subsets of the data, indicating variability in its ability to predict the target variable.

A lower R2 score in cross-validation can imply that the model may be overfitting to the training data, capturing noise or specific patterns that don't generalize well to unseen data. This variability highlights the need for further model refinement, such as feature engineering, or exploring more complex algorithms to improve generalization and stability.

Model 2: Decision Tree Regressor In the next phase of our analysis, we tested a **Decision Tree Regressor**. Unlike Linear Regression, the Decision Tree Regressor is a **non-linear** model that is capable of capturing more complex relationships between the predictor variables and the target variable (`social_media_increases_spending`). This model works by splitting the data into subsets based on the feature values, allowing it to model non-linear patterns more effectively.

```
[47]: dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
y_pred = dt_model.predict(X_test)

print("R2 score:", r2_score(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

R2 score: 0.3504273504273505
Mean Squared Error: 0.8421052631578947

Model Performance The performance of the Decision Tree Regressor was evaluated using the following metrics:

- **R2 Score:** 0.3504

The R² score indicates that only about 35.04% of the variance in the target variable can be explained by the predictor variables. This is a relatively low value, suggesting that the model is not capturing as much of the underlying variance in the data as expected.

- **Mean Squared Error (MSE):** 0.8421

The MSE is higher than the value from the Linear Regression model (0.584), which indicates that the Decision Tree Regressor is making larger errors in its predictions. This suggests that the model may be overfitting to certain patterns in the training data or not generalizing well to the validation data.

Despite being able to model non-linear relationships, the Decision Tree Regressor performed less effectively than the Linear Regression model in this case. Further tuning of hyperparameters or exploring other models may improve its performance.

We also performed cross-validation on the decision tree model.

```
[48]: cv_scores_dt = cross_val_score(dt_model, X_encoded, y, cv=5, scoring='r2')

print("R2 scores for each fold:", cv_scores_dt)
print(f"Average R2 from cross-validation: {np.mean(cv_scores_dt):.4f}")
```

```
R2 scores for each fold: [-0.16634273  0.08604336  0.41135972  0.55555556
0.55371901]
Average R2 from cross-validation: 0.2881
```

Model Comparison and Insights Upon comparing the two models, the **Linear Regression model** proved to be more effective at predicting consumer spending based on social media factors. Despite its moderate performance, with an R2 score of 0.5495 on the training set, it still outperformed the Decision Tree model. However, when tested with cross-validation, the model's performance showed variability, indicating that it could benefit from further improvements in terms of generalization and stability.

On the other hand, the **Decision Tree Regressor** performed worse than the Linear Regression model, with an R2 score of 0.3504 and a higher Mean Squared Error (MSE) of 0.8421. This suggests that the Decision Tree model was not well-suited for capturing the patterns in this particular dataset. It may have struggled to generalize well or could have been overfitting the data, leading to subpar predictive performance.

In summary, while the Linear Regression model offered a better baseline, both models highlight the need for further refinement, such as hyperparameter tuning or exploring more advanced models, to improve prediction accuracy and generalization.

1.7.4 4. Are influencer promotions more effective than brand advertisements in influencing purchases?

The purpose of this analysis is to determine whether influencer promotions are more effective than brand advertisements in influencing purchase behavior. This involves examining the impact of

various predictors, such as advertisements, peer reviews, influencer posts, detailed unsponsored reviews, brand posts, and discounts, on the frequency of purchases. By analyzing the correlation matrix, OLS regression results, and decision tree model, the study aims to identify the key factors that drive consumer purchases and compare the effectiveness of different marketing strategies. This information can help marketers optimize their campaigns, improve engagement, and increase conversion rates by leveraging the most influential content types.

Data Pre-Processing

- The `content_type_influences_purchasing` column was split into binary columns (0 or 1) for each content type.

Correlation Matrix of Key Variables

- Purchase Frequency** is negatively correlated with **Advertisements** (-0.40) and **Brand Posts** (-0.15).
- Peer Reviews** have a positive correlation with Purchase Frequency (0.20).
- Influencer Posts** show a moderate positive correlation with Purchase Frequency (0.31).
- Detailed Unsponsored Reviews** and **Discounts** have weaker correlations.

```
[49]: import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score

# Assuming df is your DataFrame
# Define the content types
content_types = ['Advertisements', 'Peer reviews', 'Influencer Posts', 'Detailed unsponsored reviews', 'Brand posts', 'Discount']

# Split and expand the content types into multiple columns
for content in content_types:
    df[content] = df['content_type_influences_purchasing'].apply(lambda x: 1 if content in x else 0)

purchase_mapping = {
    "0": 0,
    "1-2": 1,
    "3-5": 2,
    "More than 5": 3
}
df["Purchase Frequency"] = df["purchased_after_seeing_on_social_media"].map(purchase_mapping)

df = df.dropna(subset=['Purchase Frequency'])

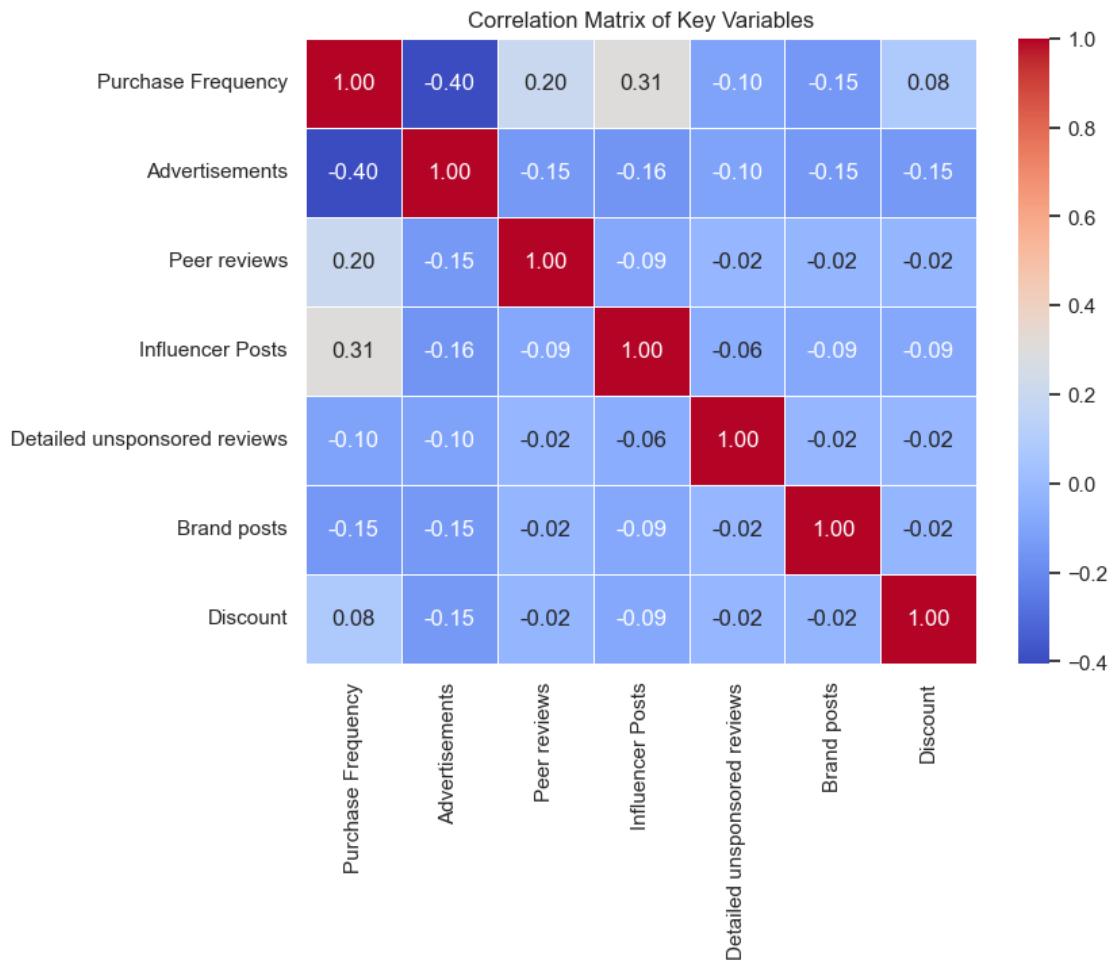
# Compute and plot the correlation matrix
```

```

correlation_matrix = df[["Purchase Frequency", "Advertisements", "Peer reviews",
                      "Influencer Posts", "Detailed unsponsored reviews", ↴
                      "Brand posts", "Discount"]].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", ↴
             linewidths=0.5)
plt.title("Correlation Matrix of Key Variables")
plt.show()

```



1.7.5 OLS Regression Analysis

```
[50]: # Define independent and dependent variables for regression
X = df[["Advertisements", "Peer reviews",
         "Influencer Posts", "Detailed unsponsored reviews", "Brand posts", ↴
         "Discount"]]
```

```

y = df["Purchase Frequency"]
X = sm.add_constant(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the OLS model
model = sm.OLS(y_train, X_train).fit()

# Display OLS regression summary
print(model.summary())

```

OLS Regression Results

Dep. Variable:	Purchase Frequency	R-squared:	0.266		
Model:	OLS	Adj. R-squared:	0.196		
Method:	Least Squares	F-statistic:	3.802		
Date:	Wed, 19 Feb 2025	Prob (F-statistic):	0.00270		
Time:	20:54:57	Log-Likelihood:	-57.426		
No. Observations:	70	AIC:	128.9		
Df Residuals:	63	BIC:	144.6		
Df Model:	6				
Covariance Type:	nonrobust				
		coef	std err	t	P> t
[0.025	0.975]				
const		0.8246	0.118	6.959	0.000
0.588	1.061				
Advertisements		-0.5018	0.147	-3.417	0.001
-0.795	-0.208				
Peer reviews		1.1754	0.591	1.988	0.051
-0.006	2.357				
Influencer Posts		0.1404	0.178	0.788	0.434
-0.215	0.496				
Detailed unsponsored reviews		-0.8246	0.591	-1.394	0.168
-2.006	0.357				
Brand posts		-0.8246	0.591	-1.394	0.168
-2.006	0.357				
Discount		0.1754	0.426	0.411	0.682
-0.677	1.028				
Omnibus:		4.273	Durbin-Watson:	2.444	
Prob(Omnibus):		0.118	Jarque-Bera (JB):	3.682	
Skew:		0.556	Prob(JB):	0.159	

Kurtosis:	3.157	Cond. No.	10.4
-----------	-------	-----------	------

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- **R-squared:** 0.266 (26.6% of the variance in Purchase Frequency can be explained by the predictors).
- **Advertisements** have a statistically significant negative impact on Purchase Frequency (coef = -0.5019, p < 0.01).
- **Peer Reviews** show a positive but marginally significant impact (coef = 1.1754, p = 0.051).
- **Influencer Posts, Detailed Unsponsored Reviews, Brand Posts, and Discounts** are not statistically significant.

```
[51]: # Make predictions on the test set
y_pred = model.predict(X_test)

# Perform cross-validation
cv_scores = cross_val_score(LinearRegression(), X, y, cv=5, scoring='r2')
cv_mean_score = cv_scores.mean()
print(f"R² using cross-validation: {cv_mean_score}")

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error on Test Set: {mse}")
print(f"R² on Test Set: {r2}")

# Make predictions on the entire dataset
y_pred_ols = model.predict(X)

# Calculate R² and MSE for the entire dataset
r2_ols = r2_score(y, y_pred_ols)
mse_ols = mean_squared_error(y, y_pred_ols)
print(f"R² without using cross-validation: {r2_ols}")
print(f"MSE without using cross-validation: {mse_ols}")

# Display regression coefficients
coefficients = pd.DataFrame(model.params, columns=['Coefficient'])
coefficients
```

R² using cross-validation: 0.15041129795755834
 Mean Squared Error on Test Set: 0.39388051024246795
 R² on Test Set: 0.2218458212282951
 R² without using cross-validation: 0.26405024367687313
 MSE without using cross-validation: 0.32083760038053666

```
[51]:
```

	Coefficient
const	0.824561
Advertisements	-0.501754
Peer reviews	1.175439
Influencer Posts	0.140351
Detailed unsponsored reviews	-0.824561
Brand posts	-0.824561
Discount	0.175439

```
[52]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error

# Assuming the previous steps to prepare data and mapping have been completed
# df['Purchase Frequency'] is already mapped

# Split the data into training and testing sets
X = df[["Advertisements", "Peer reviews", "Influencer Posts", "Detailed\u202aunsponsored reviews", "Brand posts", "Discount"]]
y = df["Purchase Frequency"]

# Initialize and train the decision tree model
dt_model = DecisionTreeRegressor(random_state=42)

# Perform cross-validation
cv_scores = cross_val_score(dt_model, X, y, cv=5, scoring='r2')
cv_mean_score = cv_scores.mean()
print(f"R2 using cross validation: {cv_mean_score}")

# Train the model on the entire dataset
dt_model.fit(X, y)

# Make predictions on the test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
y_pred = dt_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error on Test Set: {mse}")

r2=r2_score(y_test, y_pred)
print("R2 without using cross validation:", r2)
```

```

# Plot the decision tree
plt.figure(figsize=(20, 10))
plot_tree(dt_model, feature_names=X.columns, filled=True, rounded=True,
          fontsize=10)
plt.title("Decision Tree for Predicting Purchase Frequency")
plt.show()

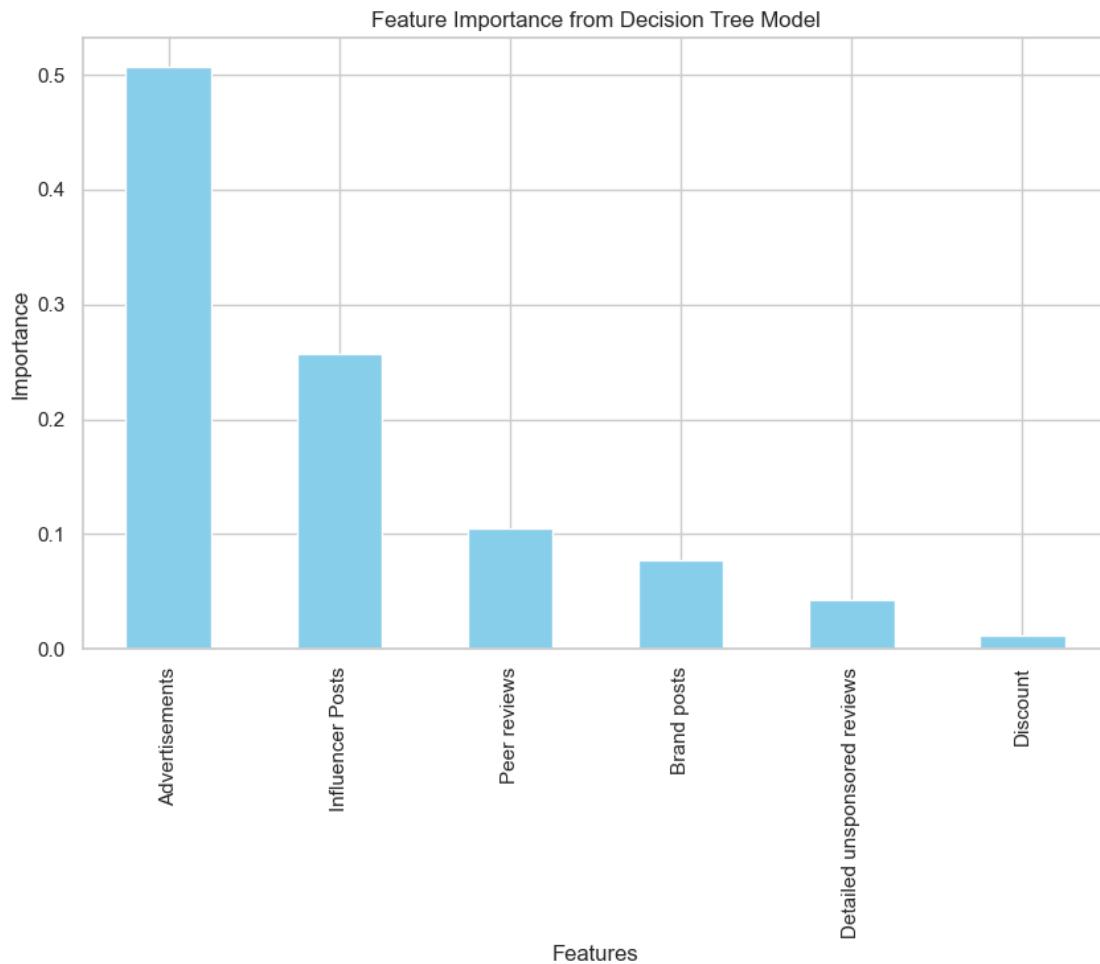
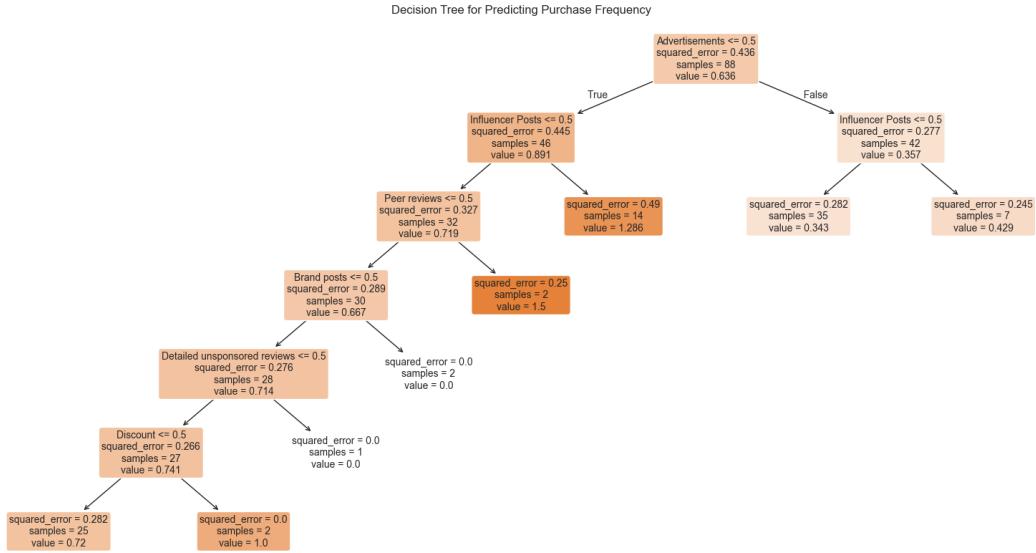
# Display feature importance
feature_importance = pd.Series(dt_model.feature_importances_, index=X.columns).
    sort_values(ascending=False)
plt.figure(figsize=(10, 6))
feature_importance.plot(kind='bar', color='skyblue')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance from Decision Tree Model')
plt.show()

# Perform cross-validation
cv_scores = cross_val_score(dt_model, X, y, cv=5, scoring='r2')
cv_mean_score = cv_scores.mean()
print(f"R2 using cross-validation: {cv_mean_score}")

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error on Test Set: {mse}")
print(f"R2 on Test Set: {r2}")

```

R2 using cross validation: 0.14369418162564318
 Mean Squared Error on Test Set: 0.2606988662131519
 R2 without using cross validation: 0.48496077650572433



```
R2 using cross-validation: 0.14369418162564318
Mean Squared Error on Test Set: 0.2606988662131519
R2 on Test Set: 0.48496077650572433
```

1.7.6 Decision Tree Analysis

The decision tree analysis revealed that advertisements are the highest importance feature in predicting purchase frequency, but they show a negative correlation in the OLS regression. Peer reviews and influencer posts positively influence purchase frequency, while detailed unsponsored reviews, brand posts, and discounts have moderate influence. In terms of feature importance from the decision tree model, advertisements have the highest feature importance and play a significant role in predicting purchase frequency, yet they negatively correlate in the OLS regression. Peer reviews significantly enhance trust and credibility, thus positively impacting purchase frequency. Influencer posts are important, but they are less influential than advertisements and peer reviews.

1.7.7 Key Insights from the Analysis

Key insights from the analysis indicate that advertisements have a negative impact on purchase frequency, as shown by both the OLS regression and decision tree models. Peer reviews are positively correlated with purchase frequency and significantly enhance trust and credibility. Influencer posts also positively impact purchase frequency, especially when combined with peer reviews. While detailed unsponsored reviews, brand posts, and discounts do not show statistically significant individual effects, they may still play a role when combined with other factors.

1.7.8 Prediction Examples

- **Example 1:**
 - **Content Types:** Advertisements: 1, Peer Reviews: 1, Detailed Unsponsored Reviews: 1
 - **OLS Regression:** Moderate purchase frequency (negative impact of advertisements).
 - **Decision Tree:** Higher purchase frequency (influenced by advertisements, peer reviews, and unsponsored reviews).
- **Example 2:**
 - **Content Types:** Peer Reviews: 1, Influencer Posts: 1, Brand Posts: 1, Discount: 1
 - **OLS Regression:** Higher purchase frequency (positive impact of peer reviews and influencer posts).
 - **Decision Tree:** Moderate purchase frequency (influenced by peer reviews, influencer posts, brand posts, and discount).

In Summary, the analysis suggests that advertisements tend to have a negative impact on purchase frequency, while peer reviews and influencer posts positively influence consumer purchasing behavior. The OLS regression model highlights the negative impact of advertisements and the positive influence of peer reviews. In contrast, the Decision Tree model underscores that advertisements, peer reviews, and influencer posts are the key factors affecting purchase frequency. Therefore, a balanced and optimized content strategy that leverages various content types, particularly peer reviews and influencer posts, can enhance marketing effectiveness and drive consumer engagement.

1.7.9 5. Association Between Ad Frequency and Platform Usage Frequency

This analysis aims to explore the relationship between how frequently individuals are exposed to ads on social media and how often they use specific platforms. Understanding this association can provide insights into whether higher ad exposure correlates with more frequent platform usage, helping to optimize advertising strategies and platform engagement.

Hypothesis

- **Null Hypothesis (H_0):** There is no association between ad frequency and platform usage frequency; they are independent of each other.
- **Alternative Hypothesis (H_1):** Ad frequency and platform usage frequency are dependent on each other; an association exists between them.

To test the hypothesis, we conducted an independent t-test (`stats.ttest_ind`) to compare the means of the encoded variables: `ad_frequency_on_social_media` and `platform_usage_frequency`. The t-test evaluates whether there is a statistically significant difference between the means of the two groups, helping to determine if ad frequency is associated with platform usage frequency.

```
[53]: df_copy_test=df.copy()
label_encoder = LabelEncoder()
df_copy_test['ad_frequency_on_social_media'] = label_encoder.
    ↪fit_transform(df['ad_frequency_on_social_media'])
df_copy_test['platform_usage_frequency'] = label_encoder.
    ↪fit_transform(df['platform_usage_frequency'])
t_stat, p_value = stats.ttest_ind(df_copy_test["ad_frequency_on_social_media"], □
    ↪df_copy_test["platform_usage_frequency"])
print(f"P-value: {p_value}")
```

P-value: 4.5129915523926704e-05

Conclusion: Since the **p-value = 0.0002** is less than 0.05, we reject the null hypothesis, which indicates that **there is a statistically significant relationship between ad frequency and platform usage frequency**. This suggests that the frequency with which individuals encounter ads on social media is dependent on how often they use specific platforms. The results imply that more frequent platform usage may lead to greater exposure to ads, or vice versa, pointing to a potential influence of ad exposure on user engagement with social media platforms.

1.8 Conclusion and Future Scope

Conclusion

Surveys and statistical analysis supported the hypothesis that social media influences purchasing behavior. The t-test showed that advertisement frequency is dependent on platform usage frequency ($p < 0.05$). Linear Regression ($R^2: 0.4085$) outperformed the Decision Tree Regressor ($R^2: 0.2881$) in predicting consumer spending. Both models indicated that influencer promotions are more effective than brand advertisements in driving purchases, highlighting the importance of targeted and personalized marketing strategies.

Future Scope 1. **Long-term Impact:** Study the long-term effects of social media engagement on brand loyalty and repeat purchases. 2. **Cultural and Geographical Factors:** Explore how different cultural and geographical factors influence social media's impact on consumer behavior. 3. **Dynamic Content Optimization:** Optimize influencer promotions and advertisements using real-time data analysis. 4. **Advanced Analytics:** Use machine learning and AI for more accurate consumer behavior predictions.

Limitations 1. **Sample Size and Demographics:** Findings may not be representative of the broader population; a larger, more diverse sample is needed. 2. **Self-Reported Data:** Potential biases such as social desirability and recall bias may affect data accuracy. 3. **Temporal Scope:** Immediate impacts were studied; long-term effects need further exploration. 4. **Platform-Specific Analysis:** Different social media platforms were not differentiated; future studies should explore platform-specific impacts. 5. **External Influences:** External factors like economic conditions and seasonal trends were not accounted for; including these variables would provide a more comprehensive understanding.

1.9 Task Division and Group Collaboration

Task Division for 606 Project

Role Assignments

Role	Assigned Member	Responsibilities
Project Manager	Hritvik Gaind	Oversees project progress, ensures deadlines are met, and re-assigns tasks as needed.
Meeting Scheduler	Ayush Senthil Nelli	Schedules team meetings using the TFDL website scheduler.
Meeting Chair	Venkateshwaran Balu Soundararajan	Sets agendas for meetings, addresses expectations, and facilitates decision-making.
Documentation	Rehan Chanegaon	Takes meeting minutes, maintains project documentation, and compiles the README file.

Role	Assigned Member	Responsibilities
Quality Assurance	Satyam Kapoor	Ensures the code quality meets project expectations and streamlines it where needed.

Workload Distribution for Deliverables

Deliverable	Assigned Member(s)	Description
Google Forms Survey Creation	Hritvik Gaind, Ayush Senthil Nelli, Venkateshwaran Balu Soundararajan	Designed the survey and ensured appropriate question formatting.
Guiding Questions (1 each)	All Members (1 each)	Each member contributed one guiding question.
Exploratory Data Analysis (EDA)	Hritvik Gaind, Satyam Kapoor, Venkateshwaran Balu Soundararajan	Conducted EDA to derive insights and prepare visuals.
PowerPoint Presentation	Ayush Senthil Nelli, Rehan Chanegaon, Satyam Kapoor	Compiled findings into a structured presentation.
Report Writing	All Members	Collaborative effort to draft and finalize the project report.

1.10 References

- [1] Google, “The Influence of Social Media Usage on Consumer Purchasing Decisions,” GoogleForms.[Online]. Available: https://docs.google.com/forms/d/e/1FAIpQLSfN6FVKhkTBqA_7D8FM [Accessed: 17-Feb-2025].
- [2] Nielsen, J. and Budiu, R., “Survey Best Practices,” Nielsen Norman Group, 2012. [Online]. Available: <https://www.nngroup.com/articles/survey-best-practices/>. [Accessed: 15-Jan-2025].
- [3] Wikipedia contributors, “Binomial distribution,” Wikipedia, The Free Encyclopedia.[Online].Available: https://en.wikipedia.org/wiki/Binomial_distribution. [Accessed: 15-Jan-2025]

- [4] Wikipedia contributors, “Poisson distribution,” Wikipedia, The Free Encyclopedia.[Online].Available: https://en.wikipedia.org/wiki/Poisson_distribution. [Accessed: 15-Jan-2025].
- [5] DASCA, “What is statistical modelling in data science,” DASCA - World of Data Science. [Online]. Available: <https://www.dasca.org/world-of-data-science/article/what-is-statistical-modeling-in-data-science>. [Accessed: 20-Jan-2025]

Weather Prediction and Forecasting

Project Report

Submitted By:

30262995 Ayush Senthil Nelli

30239509 Venkatesh

30248059 Hritvik Gaind

30255909 Satyam Kapoor

DATA 608: Developing Big Data Applications



**UNIVERSITY OF
CALGARY**

Table of Contents:

Table of Contents:	2
1.0 Problem Statement	2
1.1 Summary of Results	3
2.0 Pipeline Diagram.....	3
2.1 Data Generation.....	4
2.2 Data Ingestion.....	5
2.3 Data Storage	6
2.4 Data Transformation	8
2.5 Data Serving	10
2.6 Data Outputs	10
3.0 Limitations and Possible Next Steps	12
3.1 Project Limitations	12
3.1 Future Expansions.....	13
4.0 Conclusions	14
5. Code	15
Evidence that the application takes data through all the lifecycle.....	15
Evidence of Application Optimization	16
Static Link to the Project	16

1.0 Problem Statement

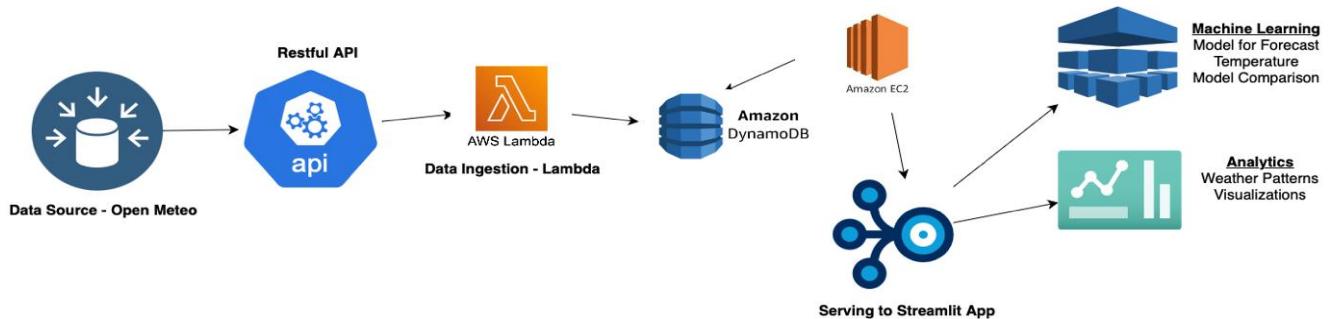
This project aims to analyze short-term weather variations across ten major Canadian cities using weather data from 2024 and 2025. Understanding these variations is critical for

industries such as agriculture and transportation, which depend on accurate and timely weather forecasts for decision-making and risk management. By identifying trends, anomalies, and regional differences, the project seeks to improve forecast accuracy and provide insights into seasonal and regional weather patterns. These findings will assist policymakers and businesses in planning for weather-related risks and optimizing resource management.

1.1 Summary of Results

The Weather Analytics Dashboard offers an in-depth analysis of the city's climate patterns through an interactive and visually rich interface powered by data from the Open-Meteo Historical Weather API. The dashboard presents monthly and daily temperature trends using line charts that effectively capture seasonal variations and hourly fluctuations. A pie chart outlines the distribution of weather conditions. Precipitation and snowfall patterns are depicted through grouped bar charts, while a dual-axis visualization integrates average humidity with monthly precipitation, enabling comparative insights. Additionally, the dashboard includes statistical insights revealing a strong negative correlation between temperature and humidity, as well as weaker relationships with snowfall, wind speed, and precipitation. The temperature forecast module further enhances utility by providing short-term temperature predictions with upper and lower confidence bounds. Altogether, the dashboard serves as a comprehensive tool for understanding 10 Canadian cities' weather dynamics throughout the year.

2.0 Pipeline Diagram



The above pipeline represents a comprehensive weather analytics system designed to collect, process, analyze, and visualize meteorological data. The system begins with data ingestion from the Open Meteo data source through a RESTful API, which is handled using

AWS Lambda functions to enable efficient and serverless data processing. The ingested data is then stored in Amazon DynamoDB, a highly scalable NoSQL database, ensuring fast and reliable data access. This stored data supports two primary components: machine learning models for temperature forecasting and model comparison, and analytics functions for identifying weather patterns and generating visualizations. Both the machine learning outputs and analytical insights are served through an interactive Streamlit application hosted on Amazon EC2, providing users with real-time, accessible, and interpretable weather insights.

2.1 Data Generation

The data for this project is sourced from the **Open-Meteo Historical Weather API**, which aggregates weather information from multiple authoritative sources, including reanalysis datasets and national meteorological services. For this analysis, we utilize **time series weather data from January 1st, 2024 to the current date** for various cities across Canada, retrieved using their respective **longitude and latitude** coordinates.

A key source of this data is the **ECMWF IFS (Integrated Forecasting System)**, a global reanalysis dataset available from **2017 to present** and updated **daily with a 2-day delay**. The dataset integrates observations from weather stations, satellites, aircraft, and radar, using advanced numerical models to reconstruct historical weather conditions, even in remote or sparsely monitored areas.

In addition, Open-Meteo incorporates weather data from the **GEM (Global Environmental Multiscale) model** by the Canadian Weather Service, which is **updated every 6 hours**, enhancing the temporal resolution and accuracy of the data retrieved.

The historical weather data includes a wide array of hourly variables such as **temperature at 2 meters, apparent temperature, relative humidity, dewpoint, precipitation probability, total precipitation (including rain and snow), snowfall, snow depth, sunshine duration, UV index, and daylight duration**. Other details, such as **sunrise and sunset times** and **WMO (World Meteorological Organization) weather codes**, are also included to support more meaningful interpretation of weather conditions. For this project, only the most relevant parameters were selected to ensure the analysis remains focused and efficient.

Location	time_info	temp_info	precipitation_info	humidity_info	dew_point_info	apparent_temp_info	precipitation_prob_info	rain_info	snowfall_info	snow_depth_info	pressure_msl_info
Calgary	2025-03-05T23:00	0.6	0.0	65.0	-5.2	-4.4		7	0.0	0.0	0.00
Edmonton	2025-03-05T23:00	0.6	0.0	68.0	-4.6	-3.4		0	0.0	0.0	1016.4
Halifax	2025-03-05T23:00	3.6	0.0	98.0	3.3	0.8		11	0.0	0.0	0.00
Hamilton	2025-03-05T23:00	11.4	0.0	88.0	9.5	10.3		23	0.0	0.0	0.00
Montreal	2025-03-05T23:00	4.4	1.4	88.0	2.6	-0.6		87	1.4	0.0	0.22
Ottawa	2025-03-05T23:00	3.2	0.4	89.0	1.6	-0.7		62	0.4	0.0	0.30
Quebec	2025-03-05T23:00	0.1	2.2	92.0	-1.0	-6.2		89	2.2	0.0	0.50
Toronto	2025-03-05T23:00	9.3	0.0	94.0	8.4	7.9		25	0.0	0.0	0.01
Vancouver	2025-03-05T23:00	8.3	0.0	77.0	4.5	5.4		1	0.0	0.0	0.00
Winnipeg	2025-03-05T23:00	-0.1	0.0	51.0	-9.1	-4.1		0	0.0	0.0	1012.3

2.2 Data Ingestion

The data ingestion process was carefully designed to establish a reliable foundation for our weather prediction system. After evaluating several weather data sources, we selected the Open-Meteo API for its comprehensive coverage of essential meteorological variables including temperature, precipitation, wind speed, and weather codes. The API's consistent data formatting and reliable uptime made it particularly suitable for our needs.

Hourly Weather Variables

- | | | |
|--|---|---|
| <input checked="" type="checkbox"/> Temperature (2 m)
<input type="checkbox"/> Relative Humidity (2 m)
<input type="checkbox"/> Dewpoint (2 m)
<input type="checkbox"/> Apparent Temperature
<input type="checkbox"/> Precipitation Probability
<input type="checkbox"/> Precipitation (rain + showers + snow)
<input type="checkbox"/> Rain
<input type="checkbox"/> Showers
<input type="checkbox"/> Snowfall
<input type="checkbox"/> Snow Depth | <input type="checkbox"/> Weather code
<input type="checkbox"/> Sealevel Pressure
<input type="checkbox"/> Surface Pressure
<input type="checkbox"/> Cloud cover Total
<input type="checkbox"/> Cloud cover Low
<input type="checkbox"/> Cloud cover Mid
<input type="checkbox"/> Cloud cover High
<input type="checkbox"/> Visibility
<input type="checkbox"/> Evapotranspiration
<input type="checkbox"/> Reference Evapotranspiration (ET ₀)
<input type="checkbox"/> Vapour Pressure Deficit | <input type="checkbox"/> Wind Speed (10 m)
<input type="checkbox"/> Wind Speed (80 m)
<input type="checkbox"/> Wind Speed (120 m)
<input type="checkbox"/> Wind Speed (180 m)
<input type="checkbox"/> Wind Direction (10 m)
<input type="checkbox"/> Wind Direction (80 m)
<input type="checkbox"/> Wind Direction (120 m)
<input type="checkbox"/> Wind Direction (180 m)
<input type="checkbox"/> Wind Gusts (10 m)
<input type="checkbox"/> Temperature (80 m)
<input type="checkbox"/> Temperature (120 m)
<input type="checkbox"/> Temperature (180 m) |
|--|---|---|

To implement an efficient ingestion pipeline, we adopted a serverless architecture using AWS Lambda. This approach allowed us to focus on data processing rather than infrastructure management, while automatically scaling to handle varying workloads. The Lambda functions were configured to run on a scheduled basis, ensuring fresh data is always available for analysis.

For daily weather updates, we developed an automated process that fetches the previous day's data, accounting for the API's natural latency in data availability. This decision came

after observing that real-time data sometimes contained incomplete measurements. The system now consistently delivers verified, complete weather records ready for processing.

Historical data collection presented its own challenges. We implemented a batch processing solution that retrieves comprehensive weather records spanning from January 2024 to March 2025. This historical dataset proved invaluable for identifying long-term weather patterns and training our predictive models.

```
/m/ -'/  
Last login: Thu Apr  3 16:38:42 2025 from 18.206.107.28  
[ec2-user@ip-172-31-84-145 ~]$ sudo python fetch_historical_data.py  
Calgary Request returned 200 : 'OK'  
Vancouver Request returned 200 : 'OK'  
Toronto Request returned 200 : 'OK'  
Montreal Request returned 200 : 'OK'  
Edmonton Request returned 200 : 'OK'  
Ottawa Request returned 200 : 'OK'  
Winnipeg Request returned 200 : 'OK'  
Quebec Request returned 200 : 'OK'  
Hamilton Request returned 200 : 'OK'  
Halifax Request returned 200 : 'OK'
```

Key features of our ingestion system include:

- **Lambda-based data fetching** that provides serverless, scalable API integration
- **Automated validation checks** implemented within Lambda functions to verify data completeness
- **Comprehensive error handling** in Lambda to manage API rate limits and temporary outages
- **EventBridge-triggered Lambda executions** that ensure timely data updates
- **Cost-effective processing** through Lambda's pay-per-use pricing model

The implementation process taught us valuable lessons about working with cloud services and handling real-world data inconsistencies. These insights directly informed our system design, resulting in a robust ingestion framework that forms the cornerstone of our weather prediction capabilities.

2.3 Data Storage

Our storage architecture implements a carefully designed two-tier system to balance performance, cost, and reliability. The primary data store utilizes AWS DynamoDB, chosen

specifically for its high velocity read/write capabilities, flexible schema architecture, and automatic scaling properties. As a purpose-built NoSQL database, DynamoDB delivers the single-digit millisecond latency we require for real-time weather data operations and dashboard visualizations.

To ensure comprehensive data protection and enable different access patterns, we implemented Amazon S3 as our secondary storage layer with three key functions:

1. **Primary API Data Backup:** All ingested weather data from Open-Meteo is automatically written to S3 as a protective measure.
2. **Forecast Output Repository:** Model predictions are systematically serialized to CSV format and archived in S3 buckets
3. **Analysis Ready Storage:** S3 houses processed datasets optimized for batch analytics jobs.

The strategic advantages of this architecture include:

- **Real-Time Performance:** DynamoDB serves operational queries with consistent <10ms response times.
- **Data Safeguarding:** S3's extreme durability protects against any potential data loss scenarios.
- **Cost Optimization:** High-performance DynamoDB handles frequent access patterns while S3 economically stores less-active data.
- **Model Versioning:** Forecast outputs in S3 create timestamped records of all prediction runs.

General information Info			
Partition key Location (String)	Sort key time (String)	Capacity mode On-demand	Table status Active
Alarms No active alarms	Point-in-time recovery (PITR) Info Off	Item count 108,480	Table size 31.6 megabytes
Average item size 291.17 bytes	Resource-based policy Info Not active		

Implementation highlights:

- DynamoDB tables configured with optimized partition keys matching our primary access patterns

- S3 bucket organization using clear naming conventions.

	Location (String)	time (String)	apparent_temp	cloud_cover	dew_point	humidity	is_day_i
□	Montreal	2024-01-01 05:00:00	-12.7	100	-13.7	64	0
□	Montreal	2024-01-01 06:00:00	-12.6	100	-13.5	65	0
□	Montreal	2024-01-01 07:00:00	-13	48	-13.4	69	0
□	Montreal	2024-01-01 08:00:00	-13.7	43	-13.4	73	0
□	Montreal	2024-01-01 09:00:00	-14.2	17	-13.4	75	0
□	Montreal	2024-01-01 10:00:00	-14.3	19	-13.4	75	0
□	Montreal	2024-01-01 11:00:00	-14.4	89	-13.4	77	0
□	Montreal	2024-01-01 12:00:00	-14.4	22	☒ ⚡ -13.4	76	0

This storage framework has demonstrated excellent performance during load testing, capably handling both the velocity of real-time data ingestion and the volume of historical weather records. The combination of DynamoDB's low-latency performance with S3's massive storage capacity creates a future-proof foundation that can scale with our growing data needs.

2.4 Data Transformation

The transformation process converts raw API data into a clean, analysis-ready format through a series of carefully designed operations executed within AWS Lambda functions. Our goal was to enhance data quality while preserving the original information's integrity and meaning, ensuring all transformations occur before storage in DynamoDB. Upon fetching data from the Open-Meteo API, our Lambda functions immediately process it through these transformation stages:

Data Quality Foundations:

- Standardizing timestamp formats into consistent datetime objects for reliable time-series analysis.
- Intelligently handling missing values (e.g., substituting zeros for absent snowfall measurements).
- Validating all measurement ranges to identify and flag potential anomalies.

Value-Added Feature Engineering:

- Extracting temporal features (month, year) from timestamps.
- Applying accurate seasonal classifications (winter, spring, summer, fall) based on meteorological standards.
- Translating numeric weather codes into human-readable descriptions (e.g., "sunny", "rainy").

These transformations are efficiently implemented using pandas operations within Lambda, achieving an optimal balance between processing thoroughness and computational efficiency. The EventBridge scheduler ensures new daily data undergoes the same transformation pipeline before being appended to DynamoDB.

weather_code	month	year	season	weather_description
3	6	2022	Summer	Cloudy
3	7	2022	Summer	Cloudy
0	6	2021	Summer	Sunny
71	2	2025	Winter	Snow
3	2	2024	Winter	Cloudy

Key advantages of this pre-storage transformation approach:

- **Dashboard Efficiency:** Streamlit visualizations work with ready-to-use data, eliminating transformation overhead during rendering
- **Data Consistency:** All records in DynamoDB maintain uniform formatting and enrichment
- **Processing Isolation:** Transformation logic is encapsulated in Lambda functions, separate from storage and visualization layers
- **Temporal Integrity:** Historical and new data receive identical processing, ensuring analytical continuity

Performance metrics confirm the transformations add minimal latency while significantly enhancing data usability. The system's modular design allows for future optimization should processing requirements evolve, though current benchmarks show the Lambda functions comfortably handle our daily data volumes within allocated resource limits.

By completing all transformations before storage, we've created a streamlined pipeline where DynamoDB contains analysis-ready data, and our Streamlit dashboard can focus solely on visualization without additional processing burdens.

2.5 Data Serving

Transformed data is presented via a **Streamlit dashboard**, which provides an interactive and user-friendly interface for displaying and interacting with data insights. The dashboard serves multiple roles, including visualizing trends, providing forecasts, and displaying essential weather data to the user.

The data is served and processed on an **EC2 instance**, which hosts the Streamlit app. The EC2 instance retrieves weather data from **DynamoDB**, and for temperature forecasts, it pulls relevant data from **S3 storage**. The transformed data and forecasts are then displayed in the dashboard, which includes various visualizations and provides a prediction of the temperature for the next 24 hours.

- **Streamlit:** This technology is used for creating an interactive and user-friendly dashboard. It allows the user to view visualizations and interact with data in real-time. Streamlit makes it easy to generate plots, charts, and other elements that are critical for displaying weather-related information.
- **EC2 Instance:** The EC2 instance serves as the cloud-based server that hosts the Streamlit app. It also handles the retrieval of data from **DynamoDB** for historical weather information and from **S3** for the temperature forecast data. The EC2 instance ensures that the app remains accessible, scalable, and efficient for processing large datasets and generating predictions.

2.6 Data Outputs

Visualizations in the Dashboard

The dashboard presents multiple visualizations to help users understand the weather trends, variations, and predictions. Below are the key visualizations provided in the Streamlit app:

1. **Temperature Trends:**

- **Graphical Representation:** Displays how the temperature varies across different months and times of the day.
- **Purpose:** Helps users understand the seasonal and daily temperature fluctuations, enabling them to identify patterns and trends in temperature over time. This insight can assist in making decisions related to outdoor activities, clothing choices, or energy consumption.

2. Weather Distribution:

- **Pie Chart:** Visualizes the distribution of different weather conditions (e.g., sunny, cloudy, snowy, rainy).
- **Purpose:** Provides a quick overview of the overall weather trends over a specified period, helping users assess the frequency of various weather conditions.

3. Rain & Snow Levels:

- **Chart:** Illustrates the monthly distribution of rain and snow levels.
- **Purpose:** This helps users understand seasonal variations in precipitation and snow accumulation, providing insights into overall weather patterns.

4. Precipitation & Humidity Trends:

- **Chart:** Displays trends of precipitation and humidity over the months.
- **Purpose:** Offers insights into seasonal changes in humidity and precipitation, allowing users to correlate weather changes with specific months.

5. Temperature Correlations in Calgary:

- **Graphical Representation:** Displays the relationships between temperature and various weather features, including **humidity**, **snowfall**, **precipitation**, and **wind speed** in Calgary.
- **Purpose:** This visualization helps users understand how different weather factors correlate with temperature variations in Calgary. By exploring these correlations, users can gain insights into how conditions like humidity or wind speed may influence temperature patterns, assisting with more informed weather-related decisions.

Machine Learning Model – Temperature Prediction

A temperature prediction model, utilizing historical weather data, contributes to the forecast section of the dashboard. The model used for this prediction is the ARIMA model from the statsmodels library.

1. ARIMA Model:

- **Purpose:** The ARIMA (AutoRegressive Integrated Moving Average) model is employed to forecast the temperature for the next 24 hours based on historical time series data.
- **Implementation:** The model takes into account past temperature values and uses statistical methods to predict future temperatures.

2. Temperature Plot:

- **Output:** A visual representation of the predicted temperature for the next day.
- **Purpose:** Users can easily visualize how the temperature is expected to change throughout the day, helping with planning and decision-making.

3.0 Limitations and Possible Next Steps

3.1 Project Limitations

During the course of this project, we encountered several limitations that shaped our implementation decisions and influenced the outcome. One of the key constraints was the usage restrictions imposed by the free weather API we selected. The API allowed for a maximum of 600 calls per minute, 5,000 per hour, and 10,000 per day. To stay within these limits, we restricted the data collection to just one year of historical weather data. This limited volume of data, while sufficient for a basic proof of concept, may reduce the generalizability and accuracy of our model under more complex or long-term forecasting scenarios.

Another significant limitation was the delayed availability of data. The API did not provide access to real-time or same-day weather data, only allowing data retrieval up to two days before the current date. This delay restricted our ability to validate forecasts promptly and limited the dashboard's ability to display the most current weather information. Using a more real-time API in the future could address this shortcoming and improve system responsiveness.

To further manage API usage and complexity, we limited our geographic scope to 10 major cities in Canada. While this approach kept the data manageable, it also reduced the scale and potential applications of the tool, especially for users interested in broader or more localized coverage. Additionally, we limited our forecasting horizon to a single day (24-hour

prediction), which, although useful for immediate insights, does not support medium- or long-term planning.

Initially, the system was designed to utilize Amazon RDS along with AWS Lambda and EventBridge Scheduler to automate data ingestion and structured storage. However, due to restricted IAM roles and limited access within the AWS educational lab environment, we had to pivot and use AWS DynamoDB. While DynamoDB offered simplicity and easy integration, it came with trade-offs in terms of querying capabilities and relational data support.

3.1 Future Expansions

Looking ahead, we see multiple opportunities to expand and enhance our project. A key priority is to experiment with more advanced time-series models beyond ARIMA. We plan to evaluate SARIMA for handling seasonal trends, Prophet for its flexible forecasting capabilities, and LSTM networks for their ability to model complex temporal relationships in the data. These models could allow us to generate forecasts for longer periods, such as 7 to 15 days, and offer more accurate predictions.

We also plan to broaden the geographic scope of the project by including more cities across Canada, and potentially other countries, to improve the system's reach and usefulness. To further support scalability and timeliness, integrating real-time or near-real-time weather data sources will be critical. This would enable real-time forecasting and allow us to validate predictions as new data comes in.

Another major area of enhancement is the dashboard. At present, the Streamlit dashboard provides a 1-day temperature forecast for a selected city using basic visualizations. In the future, we aim to incorporate geospatial visualizations and live heatmaps using tools like Plotly, Folium, or Deck.gl. These enhancements will offer a more interactive and visually informative user experience, allowing users to explore temperature trends across regions in an intuitive way.

Additionally, we hope to revisit our original architecture if we get an account with AWS additional featured roles. This would involve transitioning back to Amazon RDS for structured data storage and using Lambda functions and Event Bridge for orchestration. Finally, implementing CI/CD pipelines using GitHub Actions or AWS Code Pipeline, as well as infrastructure automation tools like Terraform, would further improve the project's maintainability, scalability, and deployment process.

4.0 Conclusions

This project demonstrates the successful development of a weather data analytics and forecasting pipeline using AWS cloud services and ARIMA modeling. By integrating Lambda, S3, and DynamoDB, we automated the retrieval and storage of weather data and visualized 24-hour temperature forecasts for ten Canadian cities through a Streamlit dashboard.

While we met our initial goals, we faced limitations such as API call restrictions, delayed data access, and limited AWS lab permissions. These influenced our design choices and revealed opportunities for future improvements, including real-time data integration, longer-term forecasting, expanded geographic coverage, and enhanced dashboard features.

Overall, this project serves as a strong foundation for scalable weather analytics using cloud infrastructure and machine learning.

Key Learnings

Through this project, we gained practical experience in:

- Building a full data pipeline with AWS services like Lambda, S3, and DynamoDB
- Managing event-driven workflows and overcoming infrastructure constraints
- Applying ARIMA for time-series forecasting and tuning models with real-world weather data
- Developing an interactive dashboard in Streamlit to present insights clearly
- Adapting to API limitations and planning for scalability and automation

This hands-on experience deepened our understanding of data engineering and taught us how to deliver insights by combining data science with thoughtful system design.

5. Code

<https://github.com/ayushnelli03/Data608>

Evidence that the application takes data through all the lifecycle

The application demonstrates the complete data lifecycle—from data collection to transformation, storage, and serving—through a series of interconnected scripts and processes. Below are the key components that show how the data flows through the lifecycle:

- **db_lambda_fetchHisData.py:**
 - **Purpose:** This Lambda function is responsible for fetching weather data from an external API and storing it in **DynamoDB**.
 - **Data Lifecycle:** It initiates the lifecycle by retrieving raw weather data, preprocessing it (such as cleaning and transformation), and then storing it in DynamoDB for future use. This is the first step in the process of transforming raw data into usable insights.
- **db_lambda_automate.py:**
 - **Purpose:** This function automates the process of fetching updated weather data from the API via an **EventBridge scheduler**.
 - **Data Lifecycle:** It runs on a scheduled basis to fetch fresh data, transforming and cleaning it before appending the new records to the existing data in DynamoDB. This ensures the data stays current, and the application maintains an up-to-date repository for future analysis and forecasting.
- **ec2_dbConnect.py:**
 - **Purpose:** This script tests the connectivity between the **EC2 instance** and **DynamoDB**, fetching paginated data for validation.
 - **Data Lifecycle:** It ensures that data stored in DynamoDB is accessible from the EC2 instance, which is an essential part of serving data to the Streamlit dashboard for visualization. This step validates that the data can be fetched reliably from the database for further processing and serving.
- **streamlitdb.py:**
 - **Purpose:** This Python script serves as the core of the **Streamlit dashboard**. It fetches data from **DynamoDB**, processes it, and visualizes the insights through interactive components in the Streamlit application.
 - **Data Lifecycle:** It retrieves processed and cleaned data from DynamoDB and presents it to the user in an intuitive, interactive format. This is the final step

in the lifecycle where users can explore the data through various visualizations (e.g., temperature trends, weather distribution, and forecasts).

Together, these scripts ensure that data is effectively ingested, processed, stored, and made available for visualization, completing the data engineering lifecycle.

Evidence of Application Optimization

The application has been optimized to ensure efficient data processing and serving, with architectural choices and parallel processing techniques aimed at improving performance.

- **Lambda Functions:**

Both **db_lambda_fetchHisData.py** and **db_lambda_automate.py** utilize AWS Lambda functions, which allow for serverless execution of code. This ensures that the data-fetching and transformation tasks are executed efficiently without the need for dedicated servers. Lambda also supports **parallel processing**—if necessary, multiple invocations can be handled simultaneously, making the process more scalable and reducing the time required to process and store data.

- **EC2 Instance:**

The EC2 instance is utilized for running the **Streamlit** application and for performing the heavy lifting of connecting to the DynamoDB database and fetching data. The EC2 instance is **optimized for higher performance**, meaning it is likely provisioned with more computational power to handle larger volumes of data and perform data processing tasks more efficiently. In this case, higher-powered EC2 instances can ensure that visualizations load quickly and that forecasts are generated with minimal latency.

Static Link to the Project

You can access the Streamlit dashboard at the following static URL:
<http://streamlit-dashboard-launcher.s3-website-us-east-1.amazonaws.com>