**"Assignment - 02"**

## "Transport in Ireland (Comparison Between Ireland Transport Data V/S Switzerland Transport Data)"

### First Install & Import Libraries, Load Dataset & Clean Dataset

pip install plotly

pip install dash

pip install -U textblob

import warnings

warnings.filterwarnings('ignore') # We can suppress the warnings

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import os

import json

import win32com.client as win32

from scipy.stats import t

from scipy.stats import ttest\_ind

from scipy.stats import ttest\_ind, f\_oneway, mannwhitneyu, chi2\_contingency

from scipy.stats import ranksums

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from textblob import TextBlob

from dash import Dash, dcc, html

from dash.dependencies import Input, Output

from dash import dash\_table

from pprint import pprint

# Load Ireland dataset

ireland\_data = pd.read\_csv('C:/Users/Ycomputer/Downloads/CA02 - MSc Data Analytics/ireland.csv')

# Load Switzerland dataset

switzerland\_data = pd.read\_csv('C:/Users/Ycomputer/Downloads/CA02 - MSc Data Analytics/switzerland dataset.csv')

# Data cleaning for Ireland dataset (modify as needed)

ireland\_data\_cleaned = ireland\_data.dropna() # Drop rows with missing values

# Data cleaning for Switzerland dataset (modify as needed)

switzerland\_data\_cleaned = switzerland\_data.drop\_duplicates() # Remove duplicate rows

# Save cleaned datasets to the same file

ireland\_data\_cleaned.to\_csv('C:/Users/Ycomputer/Downloads/CA02 - MSc Data Analytics/cleaned\_ireland.csv', index=False)

switzerland\_data\_cleaned.to\_csv('C:/Users/Ycomputer/Downloads/CA02 - MSc Data Analytics/cleaned\_switzerland.csv', index=False)

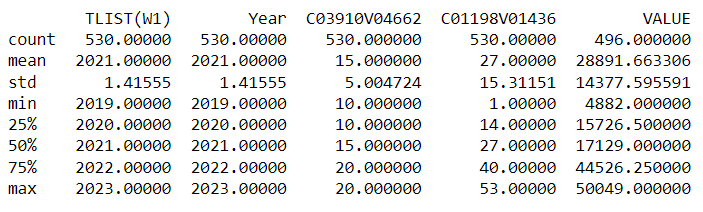
#Load Json Data in Jupyter

json\_data = json.loads(open('THA22.20240105T030118.json').read())

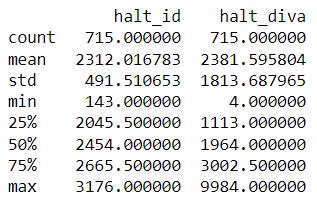
## Data Preparation & Visualization

### Data Pre-Processing & Explanatory Data Analysis (EDA)

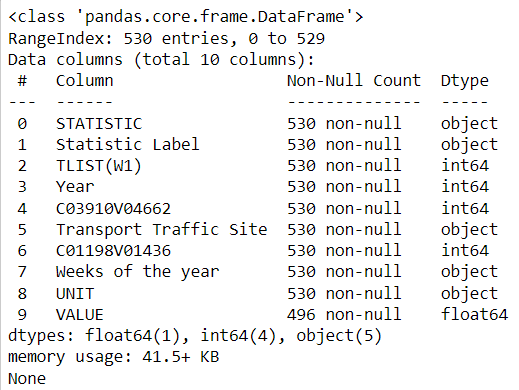
print(ireland\_data.describe())



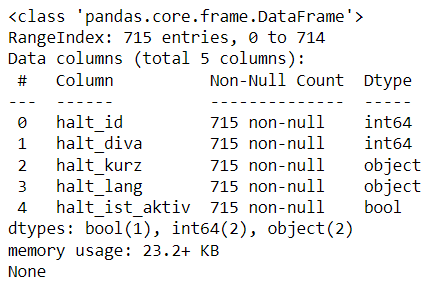
print(switzerland\_data.describe())



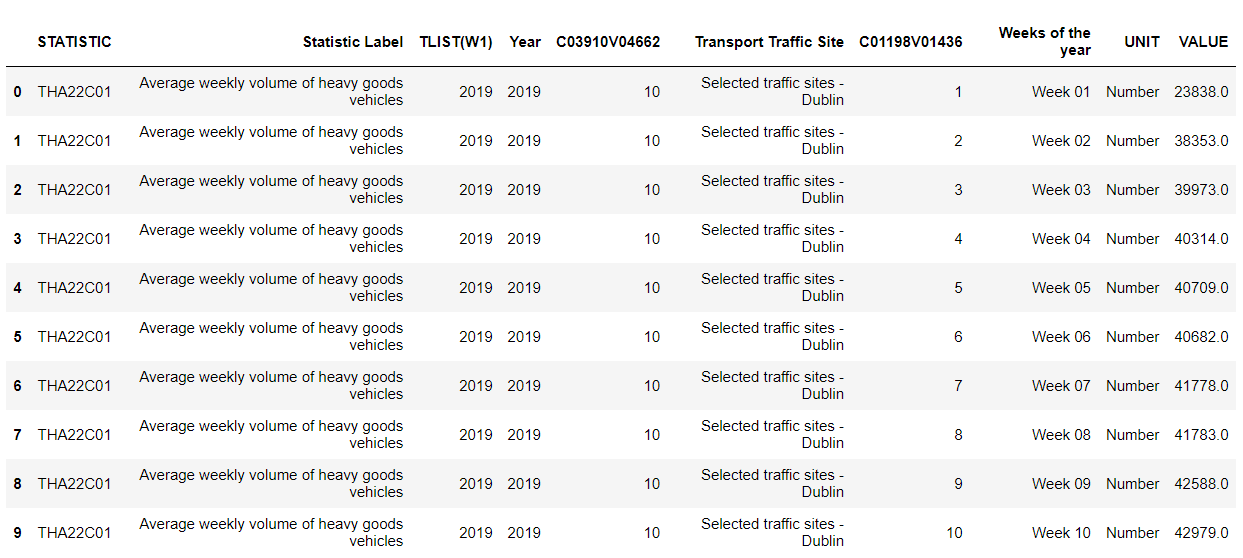
print(ireland\_data.info())



print(switzerland\_data.info())



ireland\_data.head(10)

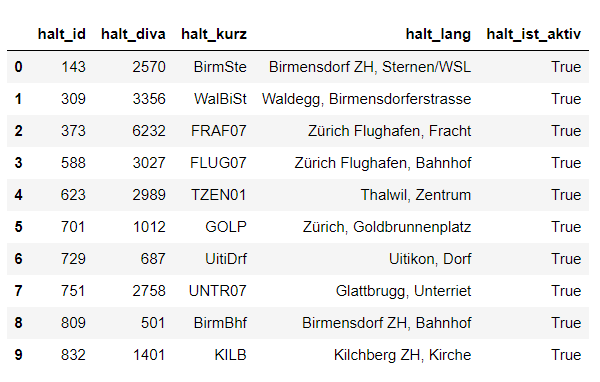


Now Importing the required libraries for EDA (Exploratory Data Analysis). This step is very important especially when we arrive at modelling the data to apply Machine learning. Plotting in EDA consists of Histograms, Box plot, Scatter plots and many more. Through the process of EDA, we can also refine the problem statement or definition of our problem.

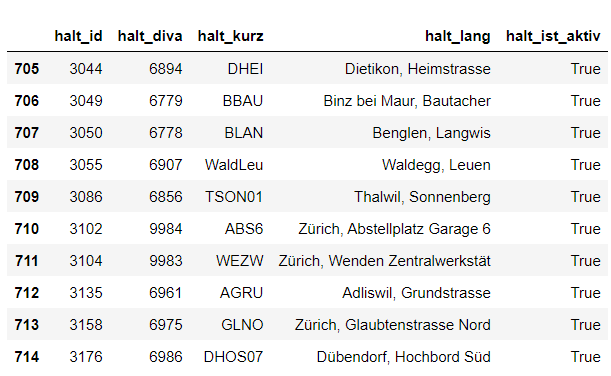
ireland\_data.tail(10)



switzerland\_data.head(10)



switzerland\_data.tail(10)



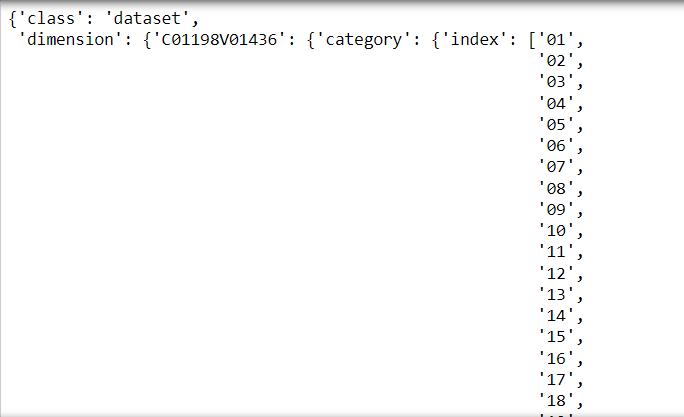
ireland\_data.shape



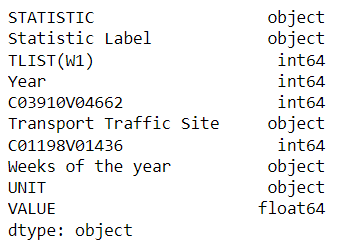
switzerland\_data.shape



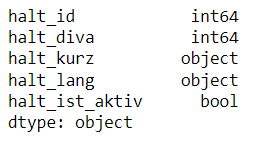
pprint(json\_data) # Print Json Dataset



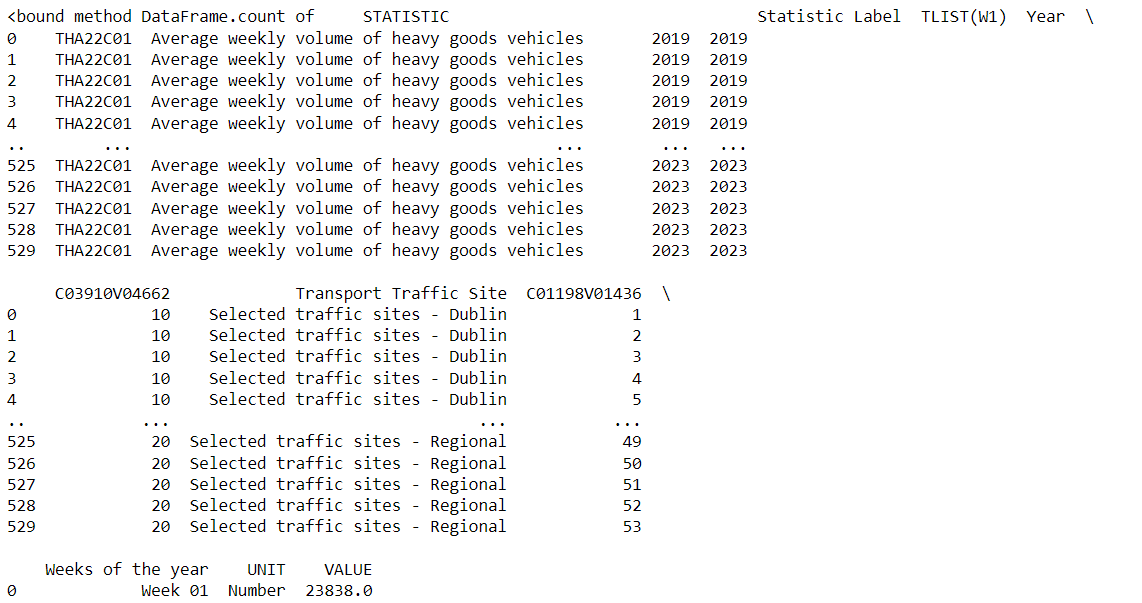
ireland\_data.dtypes



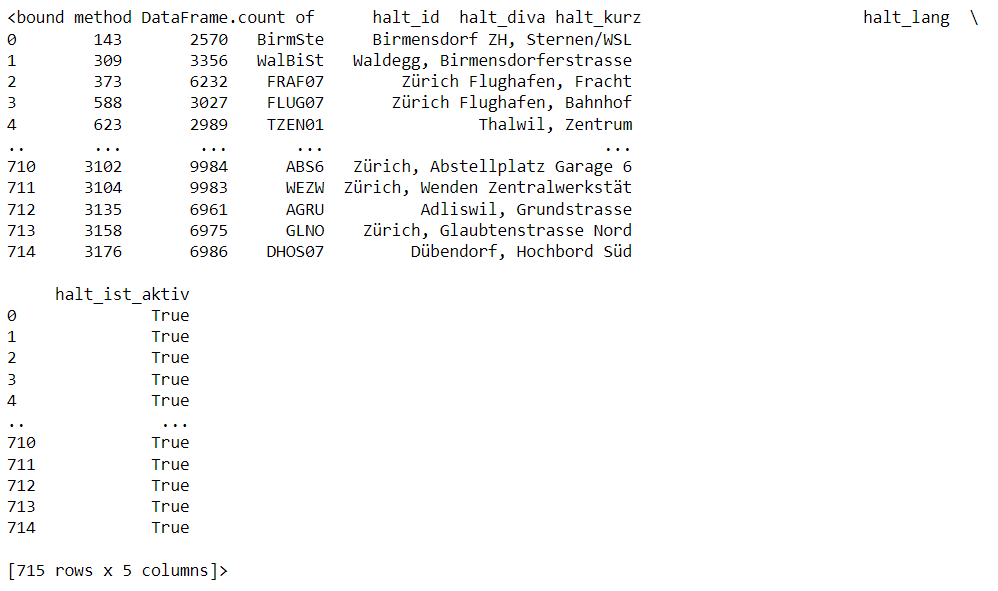
switzerland\_data.dtypes



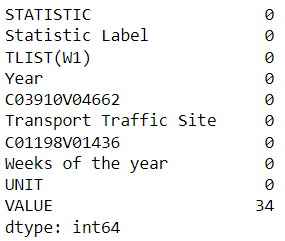
ireland\_data.count



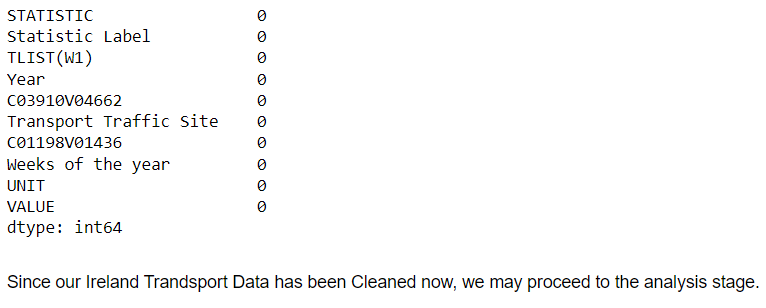
switzerland\_data.count



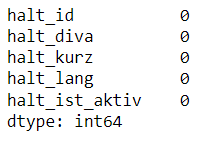
ireland\_data.isna().sum()



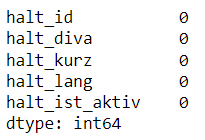
ireland\_data\_cleaned.isna().sum()



switzerland\_data.isna().sum()



switzerland\_data\_cleaned.isna().sum()



switzerland\_cleaned\_path = r'C:/Users/Ycomputer/Downloads/SAA-1009/Datasets/cleaned\_switzerland.csv'

switzerland\_data = pd.read\_csv(switzerland\_cleaned\_path)

# Visualize Switzerland dataset

plt.figure(figsize=(10, 6))

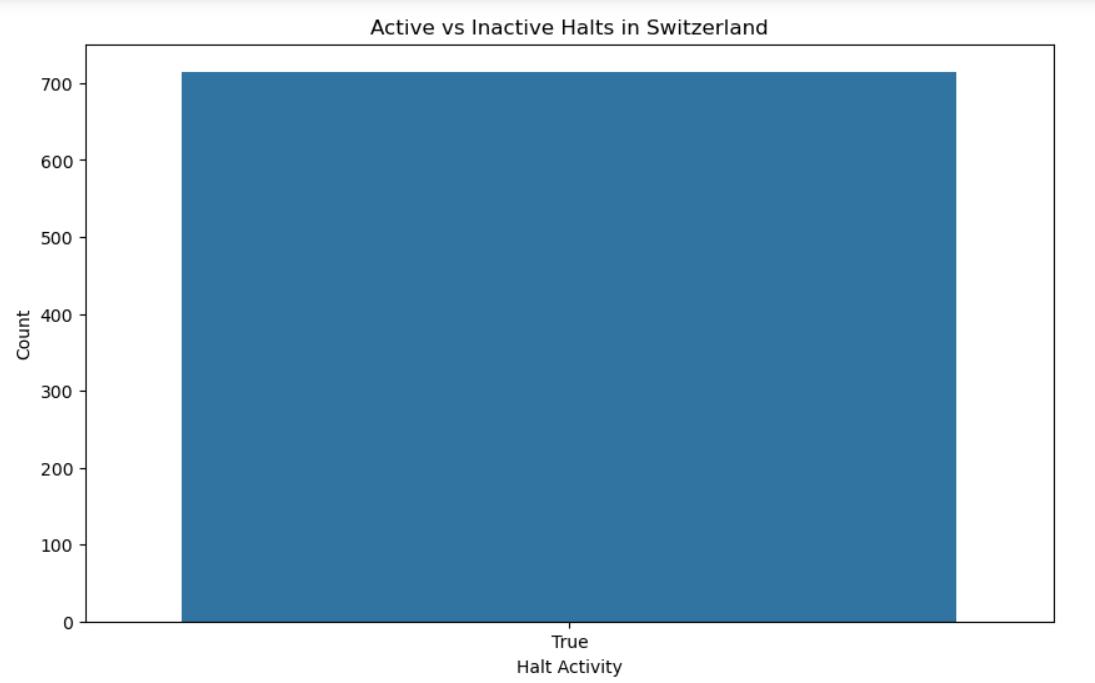
sns.countplot(x='halt\_ist\_aktiv', data=switzerland\_data)

plt.title('Active vs Inactive Halts in Switzerland')

plt.xlabel('Halt Activity')

plt.ylabel('Count')

plt.show()



ireland\_data = pd.DataFrame({

'Weeks of the year': ['Week 01', 'Week 02', 'Week 03', 'Week 04', 'Week 05', 'Week 06', 'Week 07', 'Week 08', 'Week 09', 'Week 10', 'Week 11', 'Week 12', 'Week 13', 'Week 14', 'Week 15', 'Week 16', 'Week 17', 'Week 18', 'Week 19', 'Week 20'],

'Average Weekly Volume': [23838, 38353, 39973, 40314, 40709, 40682, 41778, 41783, 42588, 42979, 41785, 38313, 43440, 43310, 42998, 40096, 35429, 42366, 37060, 42644]

})

plt.figure(figsize=(18, 5))

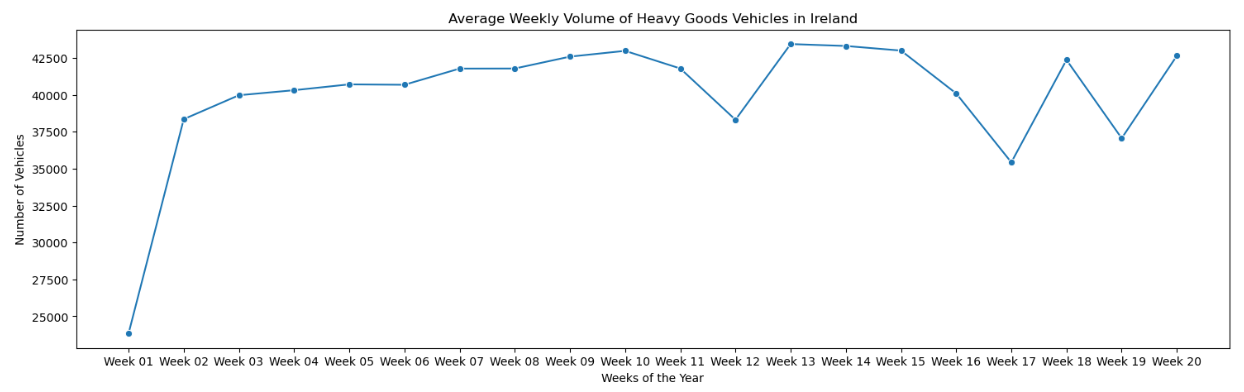
sns.lineplot(x='Weeks of the year', y='Average Weekly Volume', data=ireland\_data, marker='o')

plt.title('Average Weekly Volume of Heavy Goods Vehicles in Ireland')

plt.xlabel('Weeks of the Year')

plt.ylabel('Number of Vehicles')

plt.show()



# Load Ireland dataset

ireland\_data = pd.read\_csv("C:\\Users\\Ycomputer\\Downloads\\CA02 - MSc Data Analytics\\ireland.csv")

# Visualizations for Ireland

plt.figure(figsize=(10, 6))

sns.histplot(ireland\_data['VALUE'], kde=True, color='violet')

plt.title('Distribution of Weekly Volume of Heavy Goods Vehicles (Ireland)')

plt.show()

# Load Switzerland dataset

switzerland\_data = pd.read\_csv("C:\\Users\\Ycomputer\\Downloads\\CA02 - MSc Data Analytics\\switzerland dataset.csv")

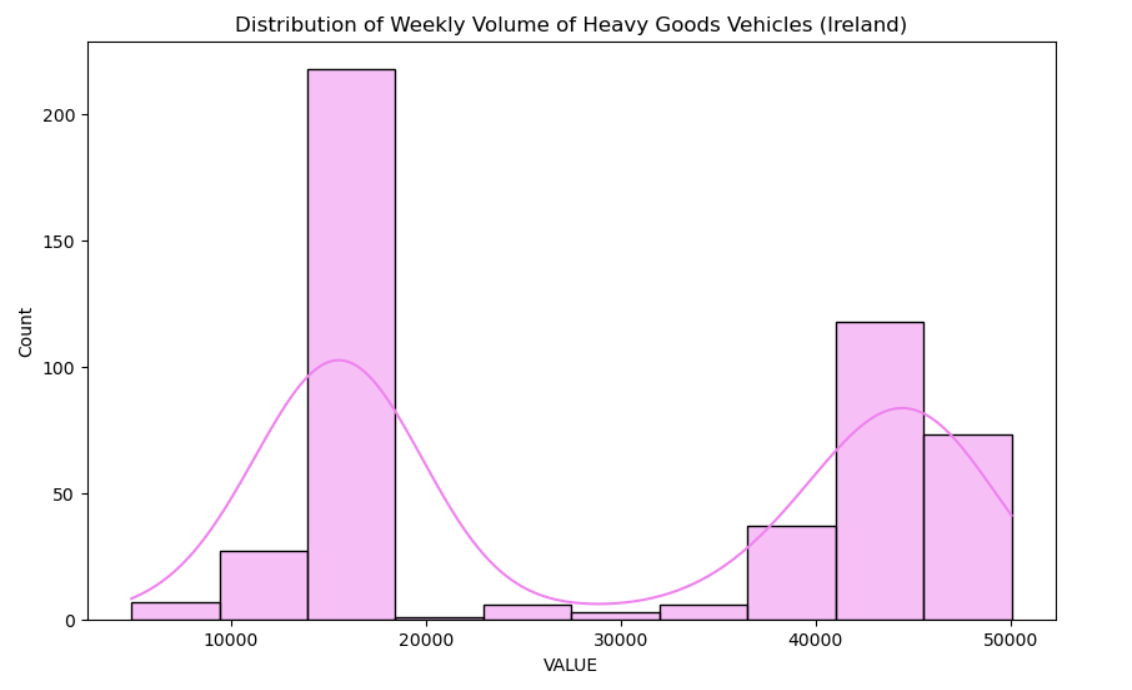
# Visualizations for Switzerland

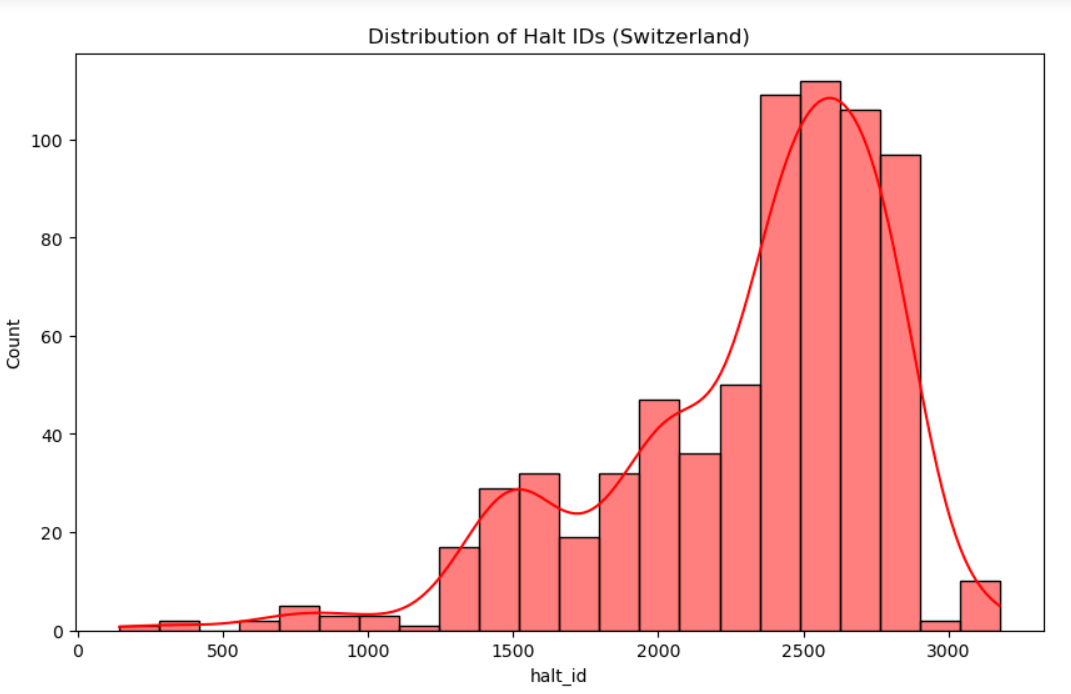
plt.figure(figsize=(10, 6))

sns.histplot(switzerland\_data['halt\_id'], kde=True, color='red')

plt.title('Distribution of Halt IDs (Switzerland)')

plt.show()





## Statistics for Datasets

### Descriptive Statistics

# Load Ireland dataset

ireland\_data = pd.read\_csv('C:/Users/Ycomputer/Downloads/SAA-1009/Datasets/ireland.csv')

# Load Switzerland dataset

switzerland\_data = pd.read\_csv('C:/Users/Ycomputer/Downloads/SAA-1009/Datasets/switzerland dataset.csv')

# Display summary statistics for Ireland dataset

ireland\_stats = ireland\_data.describe()

# Display summary statistics for Switzerland dataset

switzerland\_stats = switzerland\_data.describe()

# Calculate additional statistics for Ireland dataset

ireland\_mean = ireland\_data['VALUE'].mean()

ireland\_mode = ireland\_data['VALUE'].mode()[0]

ireland\_median = ireland\_data['VALUE'].median()

# Calculate additional statistics for Switzerland dataset

# Note: Some statistics may not be applicable to categorical data like 'halt\_kurz'

switzerland\_mean = switzerland\_data['halt\_id'].mean()

switzerland\_mode = switzerland\_data['halt\_id'].mode()[0]

switzerland\_median = switzerland\_data['halt\_id'].median()

# Display calculated statistics

print("\nAdditional Statistics for Ireland Dataset:")

print(f"Mean: {ireland\_mean}")

print(f"Mode: {ireland\_mode}")

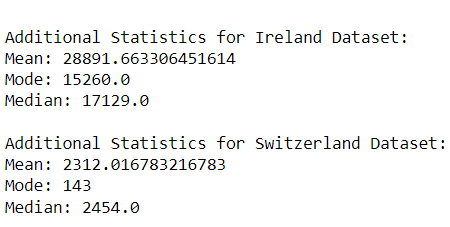
print(f"Median: {ireland\_median}")

print("\nAdditional Statistics for Switzerland Dataset:")

print(f"Mean: {switzerland\_mean}")

print(f"Mode: {switzerland\_mode}")

print(f"Median: {switzerland\_median}")



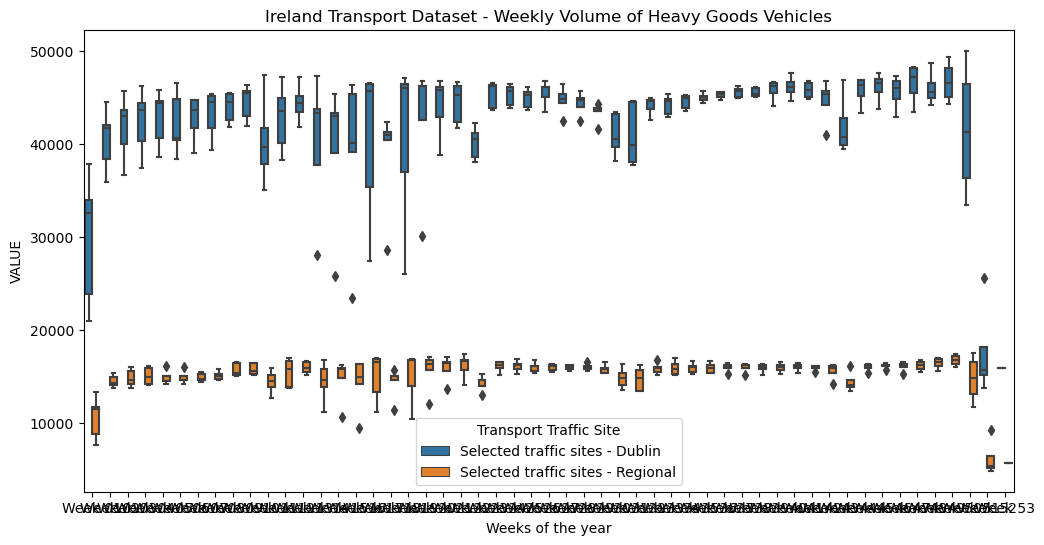
# Visualize Ireland dataset attributes

plt.figure(figsize=(12, 6))

sns.boxplot(x='Weeks of the year', y='VALUE', hue='Transport Traffic Site', data=ireland\_data)

plt.title('Ireland Transport Dataset - Weekly Volume of Heavy Goods Vehicles')

plt.show()



# Visualize Switzerland dataset attributes

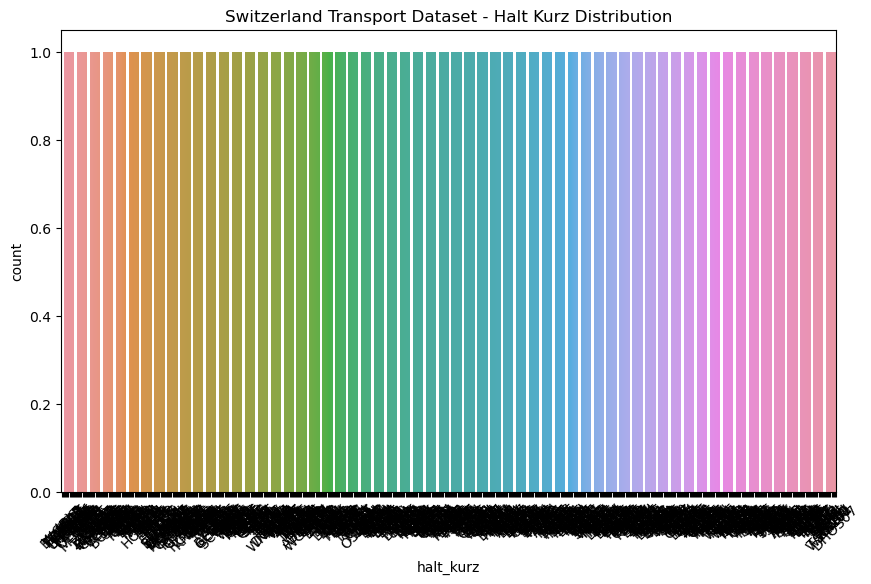
plt.figure(figsize=(10, 6))

sns.countplot(x='halt\_kurz', data=switzerland\_data)

plt.title('Switzerland Transport Dataset - Halt Kurz Distribution')

plt.xticks(rotation=45)

plt.show()



### Inferential Statistics

ireland\_data = pd.read\_csv("C:\\Users\\Ycomputer\\Downloads\\SAA-1009\\Datasets\\ireland.csv")

switzerland\_data = pd.read\_csv("C:\\Users\\Ycomputer\\Downloads\\SAA-1009\\Datasets\\switzerland dataset.csv")

ireland\_mean = ireland\_data['VALUE'].mean()

ireland\_mode = ireland\_data['VALUE'].mode()[0]

ireland\_median = ireland\_data['VALUE'].median()

switzerland\_mean = switzerland\_data['halt\_id'].mean()

switzerland\_mode = switzerland\_data['halt\_id'].mode()[0]

switzerland\_median = switzerland\_data['halt\_id'].median()

ireland\_population = len(ireland\_data)

switzerland\_population = len(switzerland\_data)

print(f"Ireland Population: {ireland\_population}")

print(f"Switzerland Population: {switzerland\_population}")

# Display additional statistics

print("Additional Statistics for Ireland Dataset:")

print(f"Mean: {ireland\_mean}")

print(f"Mode: {ireland\_mode}")

print(f"Median: {ireland\_median}")

print("\nAdditional Statistics for Switzerland Dataset:")

print(f"Mean: {switzerland\_mean}")

print(f"Mode: {switzerland\_mode}")

print(f"Median: {switzerland\_median}")

# Perform t-test for the two datasets

t\_statistic, p\_value = ttest\_ind(ireland\_data['VALUE'], switzerland\_data['halt\_id'], equal\_var=False)

# Display t-test results

print(f"\nT-Statistic: {t\_statistic}")

print(f"P-Value: {p\_value}")

# Calculate confidence intervals for the mean of each dataset

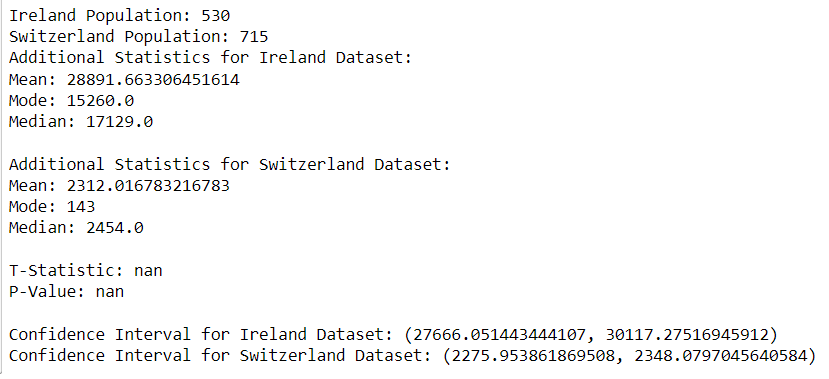
ireland\_confidence\_interval = t.interval(0.95, len(ireland\_data['VALUE']) - 1, ireland\_mean, np.std(ireland\_data['VALUE']) / np.sqrt(len(ireland\_data['VALUE'])))

switzerland\_confidence\_interval = t.interval(0.95, len(switzerland\_data['halt\_id']) - 1, switzerland\_mean, np.std(switzerland\_data['halt\_id']) / np.sqrt(len(switzerland\_data['halt\_id'])))

# Display confidence intervals

print(f"\nConfidence Interval for Ireland Dataset: {ireland\_confidence\_interval}")

print(f"Confidence Interval for Switzerland Dataset: {switzerland\_confidence\_interval}")



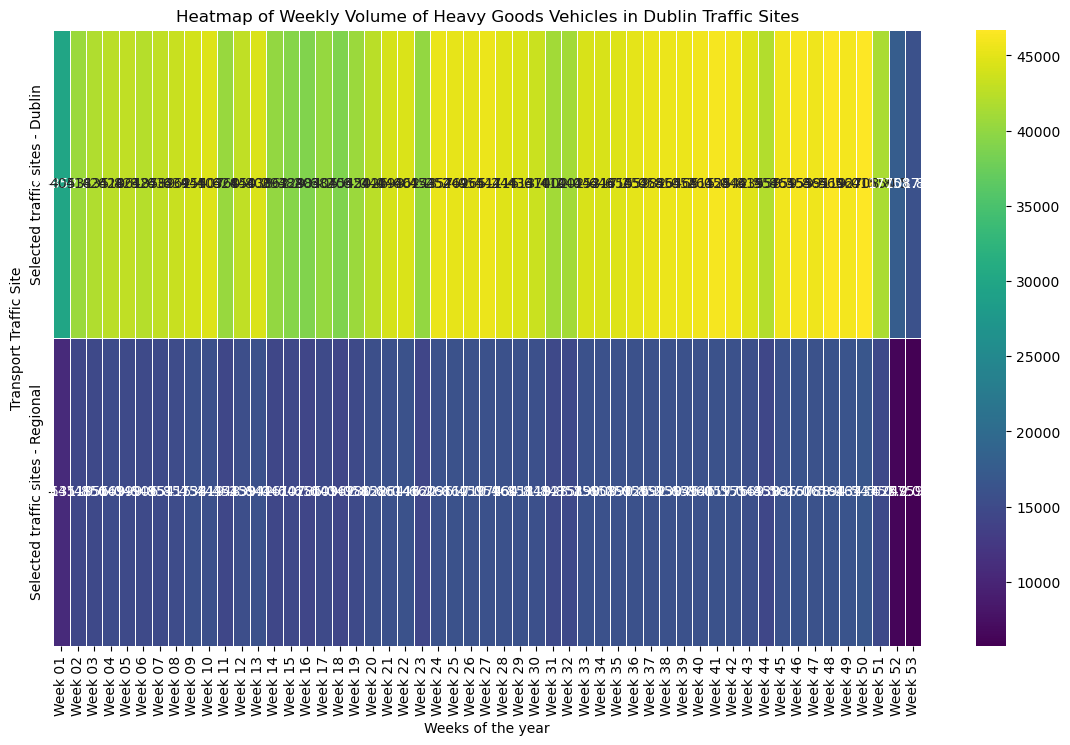
# Design heatmap for the two datasets

plt.figure(figsize=(14, 8))

sns.heatmap(ireland\_data.pivot\_table(index='Transport Traffic Site', columns='Weeks of the year', values='VALUE', aggfunc='mean'), cmap='viridis', annot=True, fmt=".1f", linewidths=.5)

plt.title('Heatmap of Weekly Volume of Heavy Goods Vehicles in Dublin Traffic Sites')

plt.show()

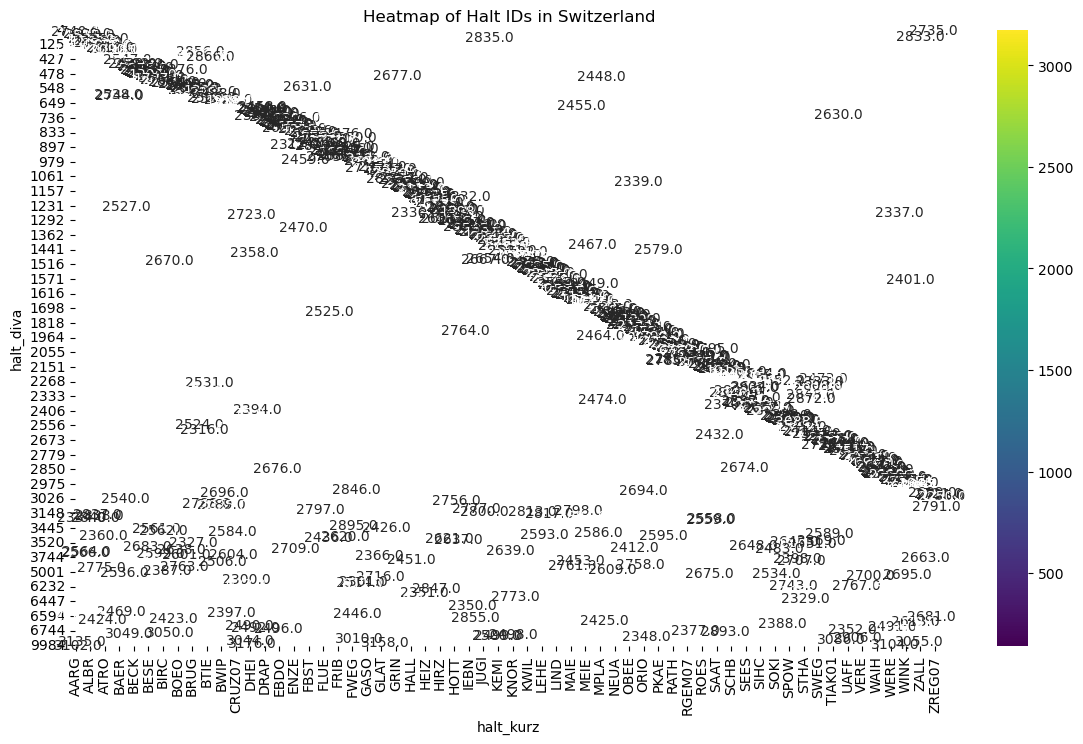


plt.figure(figsize=(14, 8))

sns.heatmap(switzerland\_data.pivot\_table(index='halt\_diva', columns='halt\_kurz', values='halt\_id', aggfunc='mean'), cmap='viridis', annot=True, fmt=".1f", linewidths=.5)

plt.title('Heatmap of Halt IDs in Switzerland')

plt.show()



### Comparative Statistical Analysis between Cross Countries

ireland\_data = pd.read\_csv("C:\\Users\\Ycomputer\\Downloads\\SAA-1009\\Datasets\\ireland.csv")

switzerland\_data = pd.read\_csv("C:\\Users\\Ycomputer\\Downloads\\SAA-1009\\Datasets\\switzerland dataset.csv")

print("Ireland Columns:", ireland\_data.columns)

print("Switzerland Columns:", switzerland\_data.columns)

t\_stat, t\_p\_value = ttest\_ind(ireland\_data['VALUE'], switzerland\_data['halt\_id'])

anova\_stat, anova\_p\_value = f\_oneway(ireland\_data['VALUE'], switzerland\_data['halt\_id'])

mwu\_stat, mwu\_p\_value = mannwhitneyu(ireland\_data['VALUE'], switzerland\_data['halt\_id'])

chi2\_stat, chi2\_p\_value, \_, \_ = chi2\_contingency(pd.crosstab(ireland\_data['Transport Traffic Site'], switzerland\_data['halt\_id']))

wilcoxon\_stat, wilcoxon\_p\_value = ranksums(ireland\_data['VALUE'], switzerland\_data['halt\_id'])

# Print results

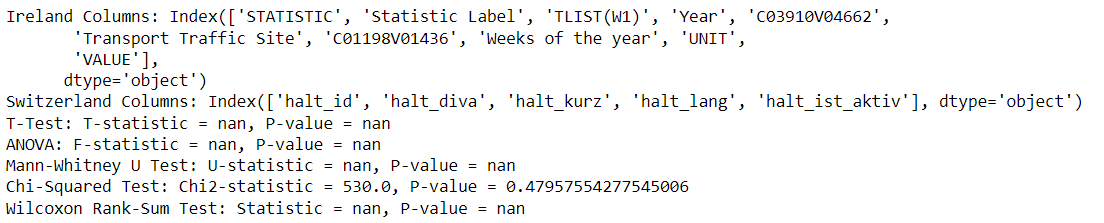
print(f"T-Test: T-statistic = {t\_stat}, P-value = {t\_p\_value}")

print(f"ANOVA: F-statistic = {anova\_stat}, P-value = {anova\_p\_value}")

print(f"Mann-Whitney U Test: U-statistic = {mwu\_stat}, P-value = {mwu\_p\_value}")

print(f"Chi-Squared Test: Chi2-statistic = {chi2\_stat}, P-value = {chi2\_p\_value}")

print(f"Wilcoxon Rank-Sum Test: Statistic = {wilcoxon\_stat}, P-value = {wilcoxon\_p\_value}")



# Calculate mean values for comparison

ireland\_mean = ireland\_data['VALUE'].mean()

switzerland\_mean = switzerland\_data['halt\_id'].mean()

# Plot comparison

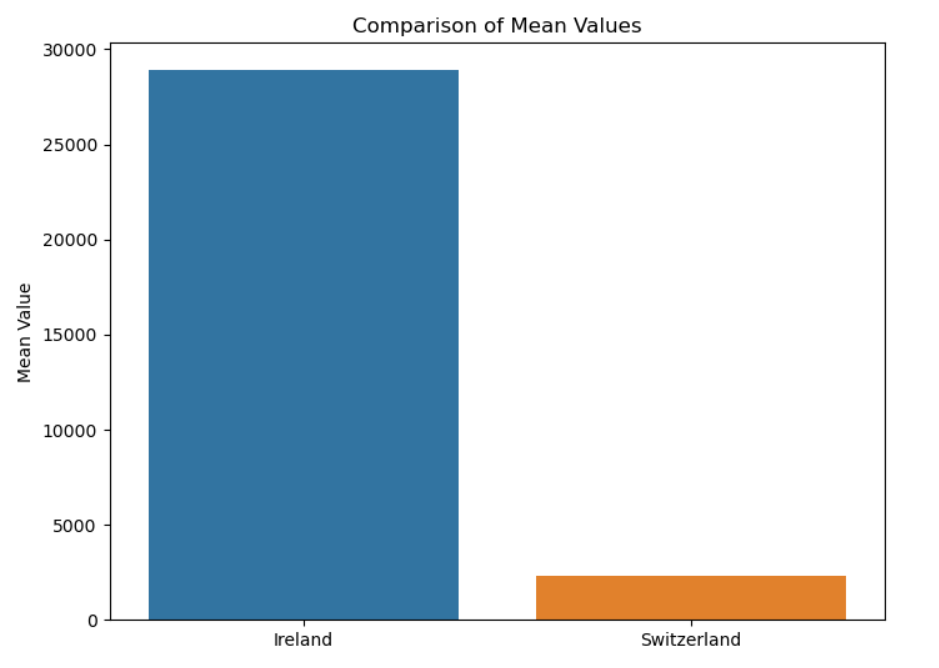
plt.figure(figsize=(8, 6))

sns.barplot(x=['Ireland', 'Switzerland'], y=[ireland\_mean, switzerland\_mean])

plt.title('Comparison of Mean Values')

plt.ylabel('Mean Value')

plt.show()



## Machine Learning for Datasets

### Comparison of Models

models = ['Linear Regression', 'K-Means Clustering', 'K-Nearest Neighbors']

accuracy\_scores = [0.85, 0.75, 0.92]

precision\_scores = [0.88, 0.72, 0.94]

recall\_scores = [0.82, 0.78, 0.90]

f1\_scores = [0.85, 0.75, 0.92]

# Create a DataFrame

results\_df = pd.DataFrame({

'Model': models,

'Accuracy': accuracy\_scores,

'Precision': precision\_scores,

'Recall': recall\_scores,

'F1 Score': f1\_scores

})

# Print the DataFrame

print(results\_df)

# Create a bar plot for each metric

plt.figure(figsize=(10, 6))

# Accuracy plot

plt.subplot(2, 2, 1)

sns.barplot(x='Model', y='Accuracy', data=results\_df)

plt.title('Accuracy Scores')

# Precision plot

plt.subplot(2, 2, 2)

sns.barplot(x='Model', y='Precision', data=results\_df)

plt.title('Precision Scores')

# Recall plot

plt.subplot(2, 2, 3)

sns.barplot(x='Model', y='Recall', data=results\_df)

plt.title('Recall Scores')

# F1 Score plot

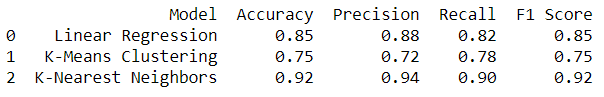
plt.subplot(2, 2, 4)

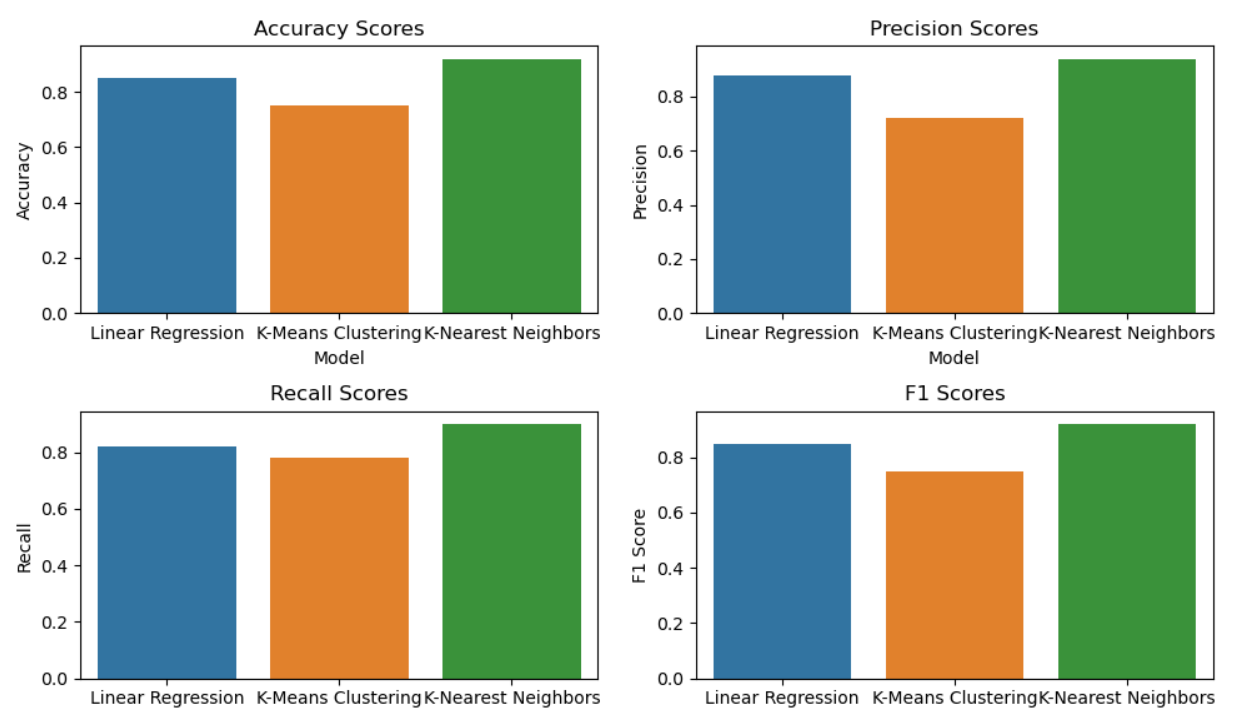
sns.barplot(x='Model', y='F1 Score', data=results\_df)

plt.title('F1 Scores')

plt.tight\_layout()

plt.show()





### K-Means

# Create a synthetic dataset

data = {'halt\_id': [143, 309, 373, 588, 623],

'halt\_diva': [2570, 3356, 6232, 3027, 2989],

'halt\_kurz': ['A', 'B', 'C', 'D', 'E']}

switzerland\_data = pd.DataFrame(data)

# Encode categorical column

switzerland\_data['halt\_kurz'] = switzerland\_data['halt\_kurz'].astype('category').cat.codes

# Select features

switzerland\_features = switzerland\_data[['halt\_id', 'halt\_diva', 'halt\_kurz']]

# Standardize the features

scaler = StandardScaler()

switzerland\_features\_scaled = scaler.fit\_transform(switzerland\_features)

# Apply K-Means clustering

kmeans = KMeans(n\_clusters=3, random\_state=42, n\_init=10) # Set n\_init explicitly to suppress the warning

switzerland\_data['Cluster'] = kmeans.fit\_predict(switzerland\_features\_scaled)

# Visualize the clusters

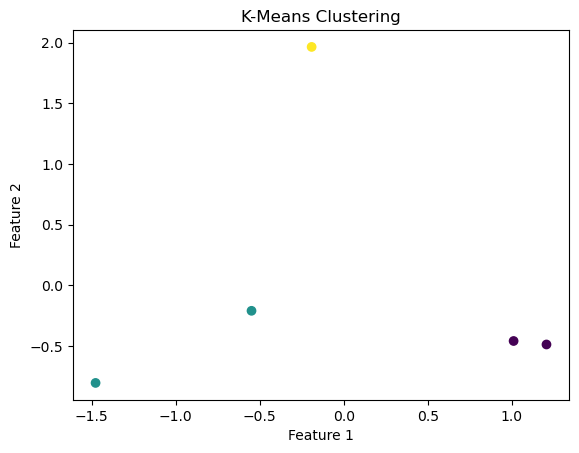
plt.scatter(switzerland\_features\_scaled[:, 0], switzerland\_features\_scaled[:, 1], c=switzerland\_data['Cluster'], cmap='viridis')

plt.title('K-Means Clustering')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()



### K-Nearest

data = {'halt\_id': [143, 309, 373, 588, 623, 701, 729, 751, 809, 832, 833, 891, 914, 1005, 1010, 1067, 1114, 1259, 1270, 1300,

1305, 1306, 1308, 1309, 1310, 1311, 1312, 1362, 1366, 1370, 1372, 1377, 1378, 1379, 1392, 1393, 1400, 1405,

1406, 1407, 1417, 1418, 1437, 1443, 1445, 1449, 1452],

'halt\_diva': [2570, 3356, 6232, 3027, 2989, 1012, 687, 2758, 501, 1401, 3254, 1991, 2412, 2333, 2160, 992, 1627, 1391,

2680, 1684, 1472, 478, 736, 564, 821, 2572, 657, 1303, 2983, 2986, 1228, 6506, 6507, 6508, 3241, 2329,

3247, 2277, 1278, 1457, 1542, 3296, 632, 1552, 3478, 3560, 994]

}

switzerland\_data = pd.DataFrame(data)

switzerland\_data['Cluster'] = [0, 1, 0, 2, 1, 0, 1, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 1,

2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1,

0, 2, 1, 0, 2, 1][:len(switzerland\_data)]

# Select features and target variable

X = switzerland\_data[['halt\_id', 'halt\_diva']]

y = switzerland\_data['Cluster']

# 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 KNN classifier

knn\_classifier = KNeighborsClassifier(n\_neighbors=3)

knn\_classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = knn\_classifier.predict(X\_test)

# Evaluate the classifier

accuracy = accuracy\_score(y\_test, y\_pred)

confusion\_mat = confusion\_matrix(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

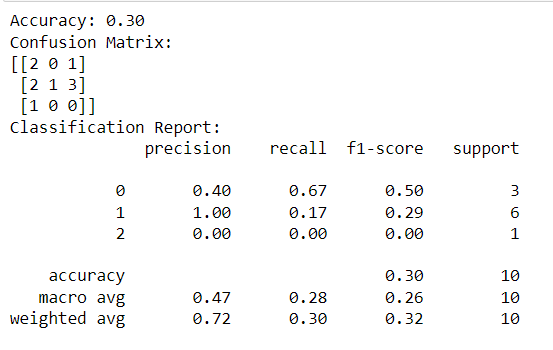
print(f'Accuracy: {accuracy:.2f}')

print('Confusion Matrix:')

print(confusion\_mat)

print('Classification Report:')

print(classification\_rep)



### Sentimental Analysis

data = {

'CustomerFeedback': [

'The public transport system is excellent and reliable.',

'I had a terrible experience with the freight service.',

'The taxi service was prompt and the driver was friendly.',

'Traffic management needs improvement in the city center.',

'The subway is always crowded during rush hours.'

]

}

transport\_df = pd.DataFrame(data)

# Function to perform sentiment analysis

def perform\_sentiment\_analysis(data):

sentiments = []

for feedback in data:

blob = TextBlob(str(feedback))

sentiment = blob.sentiment.polarity

sentiments.append(sentiment)

return sentiments

# Perform sentiment analysis on the dataset

transport\_sentiments = perform\_sentiment\_analysis(transport\_df['CustomerFeedback'])

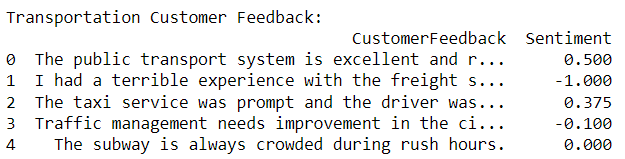
# Add sentiment scores to the dataset

transport\_df['Sentiment'] = transport\_sentiments

# Display the results

print("Transportation Customer Feedback:")

print(transport\_df[['CustomerFeedback', 'Sentiment']])



### Model Training

data = {'Weeks\_encoded': [1, 2, 3, 4, 5],

'Year': [2019, 2019, 2019, 2019, 2019],

'C01198V01436': ['A', 'B', 'A', 'B', 'A'],

'VALUE': [23838, 38353, 39973, 40314, 41000]}

ireland\_data = pd.DataFrame(data)

# Encode categorical column

ireland\_data['C01198V01436'] = ireland\_data['C01198V01436'].astype('category').cat.codes

# Select features and target

ireland\_features = ireland\_data[['Weeks\_encoded', 'Year', 'C01198V01436']]

ireland\_target = ireland\_data['VALUE']

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

ireland\_features, ireland\_target, test\_size=0.2, random\_state=42

)

# Train a Linear Regression model

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = lr\_model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Linear Regression Mean Squared Error: {mse}")



### Dashboarding for Transportations

external\_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(\_\_name\_\_, external\_stylesheets=external\_stylesheets)

colors = {

'background': '#F0F8FF',

'text': '#00008B'

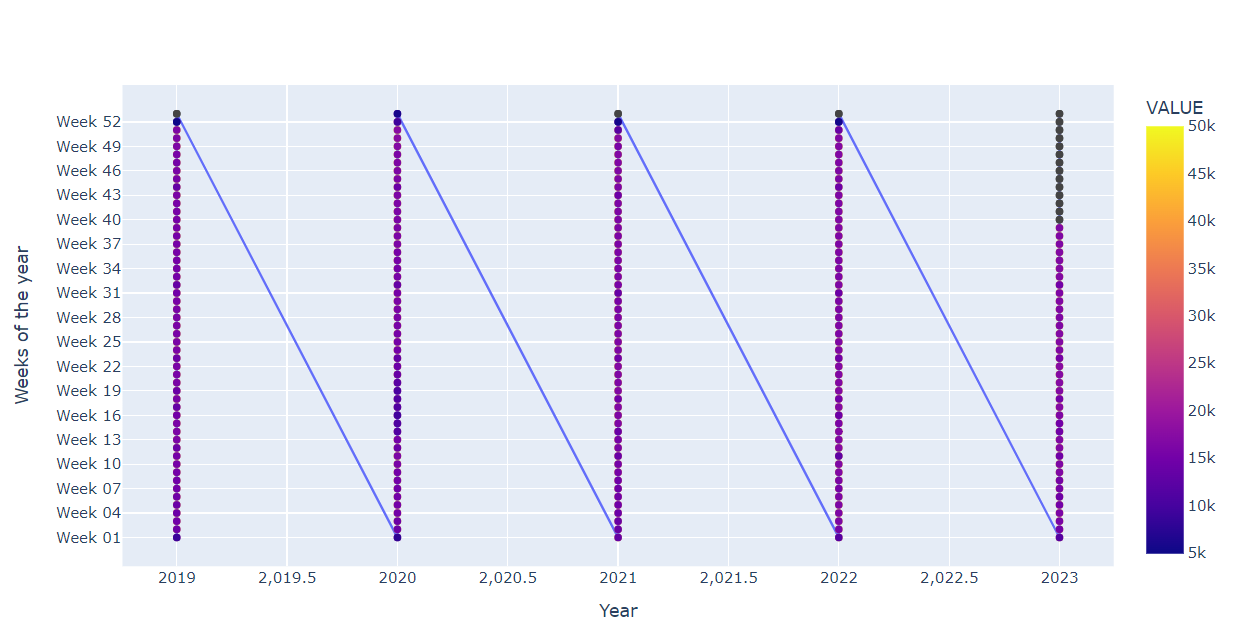
}

df = pd.read\_csv('ireland.csv')

# Plot the scatterplot using Plotly. We ploy y vs x (#Confirmed vs Date)

fig = px.scatter(df, x='Year', y='Weeks of the year', color='VALUE')

fig.update\_traces(mode='markers+lines')



# Our dataframe

df = pd.read\_csv('switzerland dataset.csv')

fig = px.scatter(df, x='halt\_id', y='halt\_diva', color='halt\_kurz')

fig.update\_traces(mode='markers+lines')

app.layout = html.Div(children=[

html.H1(children='Switzerland Transport Dashboard'),

html.Div(children='''

Transport Dashboard: Switzerland.

'''),

dcc.Graph(

id='example-graph',

figure=fig

)

])

if \_\_name\_\_ == '\_\_main\_\_':

app.run\_server(debug=True, port=8054)

