report

July 4, 2023

1 Investigating Bicycle Usage Patterns

AMSE 2023 course project by: Muhammetmyrat Yarmatov

The project "Investigating Bicycle Usage Patterns" aimed to gain insights into the factors influencing bicycle usage and understand the temporal patterns associated with it. The analysis focused on the impact of weather conditions and temporal variations on bicycle counts.

The project followed a systematic approach, starting with an Extract, Transform, Load (ETL) pipeline to acquire and preprocess the data. Two datasets were integrated and preprocessed, which included information such as date, time, weather conditions, and bicycle counts.

The analysis began with an exploration of the impact of weather conditions on bicycle usage. The project investigated the effects of different weather types, such as sunny, rainy, and cloudy, on bicycle counts. Additionally, the project examined whether specific temperature ranges were associated with higher bicycle usage. The influence of rainfall on bicycle counts was also assessed.

The second part of the analysis focused on temporal patterns in bicycle usage. The project explored seasonal and monthly variations in bicycle counts to identify any notable trends. Furthermore, the project examined how bicycle usage differed between weekdays and weekends, aiming to uncover variations in usage patterns during these time periods. The identification of distinct peak hours or periods of high bicycle usage on weekdays and weekends was another important aspect investigated.

Throughout the project, data visualization techniques, statistical analysis, and exploratory data analysis were employed to analyze and interpret the findings. Visualizations such as line plots, bar plots, and heatmaps were used to illustrate the relationships and patterns observed in the data.

The project's insights have the potential to inform various stakeholders, such as urban planners, policymakers, and bike-sharing program managers, in making data-driven decisions related to bicycle infrastructure, resource allocation, and marketing strategies.

In conclusion, "Investigating Bicycle Usage Patterns" shed light on the impact of weather conditions and temporal variations on bicycle usage. The project provided valuable insights into the factors influencing bicycle counts, allowing for a deeper understanding of usage patterns and facilitating evidence-based decision-making.

1.1 Install Dependencies

[ ]:
```
%pip install pandas
%pip install plotly
%pip install 'SQLAlchemy==1.4.46'
%pip install nbformat
%pip install matplotlib
%pip install seaborn
%pip install requests
```

2 Extract, Transform, and Load pipeline

The ETL (Extract, Transform, Load) pipeline is responsible for extracting data from multiple sources, performing necessary transformations or modifications on the extracted data, and finally loading the transformed data into a target destination, such as a database. It ensures that the data is in a consistent and usable format for subsequent analysis and processing. The ETL process is executed for two datasets. The data is extracted from the respective URLs, transformed using the defined ETL pipeline, and loaded into separate SQLite databases. The resulting pandas DataFrames (df1 and df2) serve as the in-memory representation of the transformed data, enabling further analysis and processing.

[2]:
```
# Import required libraries and define urls for datasources

import pandas as pd
import io
import requests
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

# Ignore the warning from pandas
warnings.filterwarnings('ignore')

url1 = "https://offenedaten-konstanz.de/sites/default/files/
 ↪Zaehlstelle_Herose_2019_stuendlich_Wetter.csv"
url2 = "https://bicycle.vlba.uni-oldenburg.de/csv?
 ↪region=Konstanz&table=zaehl&startDate=2019-01-01&endDate=2019-12-31&stations=Heroséparк&col
```

[3]:
```
def extract(url, withRequest=False):
    if withRequest == False:
        #For Datasource 1
        encoding = "ISO-8859-1"
        df = pd.read_csv(url, encoding=encoding, sep=';')

    else:
        #For Datasource 2
```

```
        encoding = "utf-8"
        response = requests.get(url)
        response.raise_for_status()
        content = response.content.decode(encoding)
        df = pd.read_csv(io.StringIO(content))

    return df
```

[4]:
```
def transform(df):
    # We already know the station's name and we got data from only that station␣
 ↪in Konstanz
    if 'station' in list(df.columns):
        df = df.drop('station', axis=1)

    return df
```

[5]:
```
def load(df, name):

    # Create a connection to the SQLite database using SQLAlchemy
    database_uri = 'sqlite:///' + name + '.sqlite'
    engine = create_engine(database_uri)

    # Save the DataFrame to the SQL table
    df.to_sql(name, engine, if_exists='replace', index=False)

    # Close the database connection
    engine.dispose()
```

[6]:
```
def etl(url, withRequest=False, name="random"):
    df = extract(url, withRequest=withRequest)
    df = transform(df)
    load(df, name)

    return df
```

[7]:
```
# We load the datasets into sqlites, and we use these pandas dataframes for␣
 ↪further use

df1 = etl(url1, withRequest=False, name="bikeCountKonstanz_ds1")
df2 = etl(url2, withRequest=True, name="bikeCountKonstanz_ds2")
```

3 Integrating and Pre-processing Datasets

In the "Integrating and Pre-processing datasets" part, two datasets were selected for analysis. The first dataset (Dataframe 1) contained hourly bicycle count numbers along with additional features like temperature, perceived temperature, weather type, and rainfall. The second dataset

(Dataframe 2) had fewer features but provided more frequent bicycle count data at a 15-minute interval. Recognizing that utilizing more features and higher frequency data can improve analysis and predictions, the datasets were compared, merged, and used together to gain better insights.

To accomplish this, necessary data pre-processing steps were performed. These steps involved handling missing values, aligning the datasets for merging purposes (including upsampling one dataset to match the frequency of the other), and ensuring consistency in values, data types, and variable names. By successfully merging the two datasets, a comprehensive dataset was created that could be utilized for further analysis and exploration.

[8]:
```
# Check NaN values in the datasets, and handle them properly:

df1_has_nan_values = df1.isna().any().any()
df2_has_nan_values = df2.isna().any().any()

if df1_has_nan_values:
    print("There are NaN values in the df1 DataFrame.")
else:
    print("There are no NaN values in the df1 DataFrame.")

print()

if df2_has_nan_values:
    print("There are NaN values in the df2 DataFrame.")
else:
    print("There are no NaN values in the df2 DataFrame.")
```

```
There are NaN values in the df1 DataFrame.

There are no NaN values in the df2 DataFrame.
```

[9]:
```
# Find the locations of nan values in df1 DataFrame

nan_locations = df1.isna()

# Iterate over columns
for column in nan_locations.columns:
    # Get row indices with NaN values in the current column
    rows_with_nan = nan_locations[column].index[nan_locations[column]].tolist()
    if rows_with_nan:
        print(f"NaN values found in column '{column}' at rows: {rows_with_nan}")
```

```
NaN values found in column 'Fahrradbruecke' at rows: [7203]
NaN values found in column 'Fahrradbruecke stadteinwaerts' at rows: [7203]
NaN values found in column 'Fahrradbruecke stadtauswaerts' at rows: [7203]
```

[10]:
```
df1.iloc[7203]
```

[10]:
```
Datum                                28/10/2019
Uhrzeit                                   04:00
Fahrradbruecke                              NaN
Fahrradbruecke stadteinwaerts               NaN
Fahrradbruecke stadtauswaerts               NaN
Symbol Wetter                    Leichter Nebel
Temperatur (°C)                             12
Gefühlte Temperatur (°C)                    12
Regen (mm)                                   0
Name: 7203, dtype: object
```

[11]:
```
# The data suggests that no bicycles passed during that time, so:

df1.at[7203, "Fahrradbruecke"] = 0
df1.at[7203, "Fahrradbruecke stadteinwaerts"] = 0
df1.at[7203, "Fahrradbruecke stadtauswaerts"] = 0
```

[12]:
```
'''
To merge the datasets, it is essential to ensure that the bicycle count
detections align between the two datasets. Since one dataset records
bicycle counts every hour, the bicycle counts (inwards and outwards)
are summed up for each hour. These summed counts are then compared with
the dataset that has a 15-minute frequency to verify if they match each
other. This step ensures consistency and compatibility between the
datasets before proceeding with the merging process.
'''

totalTo = 0
totalFrom = 0
isCorrect = True

for i in range(len(df2) - 10): #len(df2)
    if i%4 == 0:
        if(i != 0):
            numTo = int(df1["Fahrradbruecke stadteinwaerts"][i//4 - 1])
            numFrom = int(df1["Fahrradbruecke stadtauswaerts"][i//4 - 1])

            if(numTo != totalTo or numFrom != totalFrom):
                print("From df1: ", numTo, numFrom)
                print("From df2: ", totalTo, totalFrom)
                print("Incorrect: Numbers do not match!")
                isCorrect = False
            elif i < 20:
                print("From df1: ", numTo, numFrom)
                print("From df2: ", totalTo, totalFrom)
                print("Numbers match!\n")
```

```
            totalTo = 0
            totalFrom = 0

    totalTo = totalTo + df2["countTo"][i]
    totalFrom = totalFrom + df2["countFrom"][i]

if isCorrect:
    print('\n'.join(['.' for _ in range(3)]))
    print("\nEverything matches perfectly!")
```

```
From df1:  25 36
From df2:  25 36
Numbers match!

From df1:  49 51
From df2:  49 51
Numbers match!

From df1:  73 70
From df2:  73 70
Numbers match!

From df1:  59 39
From df2:  59 39
Numbers match!

.
.
.

Everything matches perfectly!
```

[13]:
```
# As all numbers match perfectly, we will also match other variables,␣
 ↪datatypes, etc.
# to be able to merge datasets.

df1["start"] = df1["Datum"] + " " + df1["Uhrzeit"]
df1 = df1.drop(['Datum', 'Uhrzeit'], axis=1)
df1 = df1[['start'] + [col for col in df1.columns if col != 'start']]

df1['start'] = pd.to_datetime(df1['start'], format='%d/%m/%Y %H:%M')
df2['start'] = pd.to_datetime(df2['start'])

df1 = df1.set_index('start')
df1.index.name = 'Timestamp'

df2 = df2.set_index('start')
```

```
df2.index.name = 'Timestamp'
```

[14]:
```
df1.head()
```

[14]:

Timestamp	Fahrradbruecke	Fahrradbruecke stadteinwaerts	Fahrradbruecke stadtauswaerts	Symbol Wetter	Temperatur (°C)	Gefühlte Temperatur (°C)	Regen (mm)
2019-01-01 00:00:00	61.0	25.0	36.0	Leicht bewölkt	2	2	0
2019-01-01 01:00:00	100.0	49.0	51.0	Leicht bewölkt	2	2	0
2019-01-01 02:00:00	143.0	73.0	70.0	Leicht bewölkt	2	2	0
2019-01-01 03:00:00	98.0	59.0	39.0	Leicht bewölkt	3	3	0
2019-01-01 04:00:00	68.0	34.0	34.0	Leicht bewölkt	3	2	0

[15]:
```
df2.head()
```

[15]:

Timestamp	countTo	countFrom	classification
2019-01-01 00:00:00	3	0	Cold
2019-01-01 00:15:00	4	10	Light rain
2019-01-01 00:30:00	8	6	Cold
2019-01-01 00:45:00	10	20	Cold
2019-01-01 01:00:00	12	13	Cold

[16]:
```
# Upsample the second dataset to 15-minute intervals
df1_upsampled = df1.resample('15min').ffill()

# Merge the two datasets based on the index (Time)
merged_df = pd.merge(df2, df1_upsampled, left_index=True, right_index=True,␣
 ↪how='left')

# Drop unneccessary and already existed columns
merged_df = merged_df.drop(['Fahrradbruecke', 'Fahrradbruecke stadteinwaerts',
```

```
                          'Fahrradbruecke stadtauswaerts', 'Symbol␣
 ↪Wetter'],axis=1)

# Change names of variables
name_mapping = {
    'countTo': 'Count to',
    'countFrom': 'Count from',
    'classification': 'Weather type',
    'Temperatur (°C)': 'Temperature (°C)',
    'Gefühlte Temperatur (°C)': 'Perceived temperature (°C)',
    'Regen (mm)': 'Rain (mm)'
}

merged_df = merged_df.rename(columns=name_mapping)
df = merged_df
```

[17]:
```
# Last version of integrated and processed dataset

df.head()
```

[17]:
```
                     Count to  Count from Weather type  Temperature (°C)  \
Timestamp
2019-01-01 00:00:00         3           0         Cold                 2
2019-01-01 00:15:00         4          10   Light rain                 2
2019-01-01 00:30:00         8           6         Cold                 2
2019-01-01 00:45:00        10          20         Cold                 2
2019-01-01 01:00:00        12          13         Cold                 2

                     Perceived temperature (°C)  Rain (mm)
Timestamp
2019-01-01 00:00:00                           2          0
2019-01-01 00:15:00                           2          0
2019-01-01 00:30:00                           2          0
2019-01-01 00:45:00                           2          0
2019-01-01 01:00:00                           2          0
```

4 Exploratory Data Analysis

In the "Exploratory Data Analysis" part of the project, various analyses were conducted to gain insights into the dataset. The focus was on two main aspects: the impact of weather conditions on bicycle usage and the temporal patterns in bicycle usage.

For the impact of weather conditions, the analysis examined how different weather types (sunny, rainy, cloudy, etc.) affected bicycle counts. Additionally, the relationship between temperature and bicycle usage was explored to identify any specific temperature ranges associated with higher bicycle usage. The impact of rainfall on bicycle counts was also assessed.

Regarding temporal patterns, the analysis investigated seasonal and monthly variations in bicycle

counts to understand any noticeable trends throughout the year. The analysis also compared bicycle usage between weekdays and weekends to identify any differences in usage patterns during these time periods. Furthermore, distinct peak hours or periods of high bicycle usage were identified for weekdays and weekends.

Through data visualizations, statistical analysis, and exploratory techniques, these analyses aimed to uncover patterns, relationships, and trends within the dataset. The insights gained from this exploratory data analysis phase provide a foundation for further investigations and inform decision-making processes related to bicycle infrastructure, resource allocation, and marketing strategies.

4.1 1. Impact of Weather Conditions on Bicycle Usage:

4.1.1 How do different weather types (sunny, rainy, cloudy, etc.) affect bicycle counts?

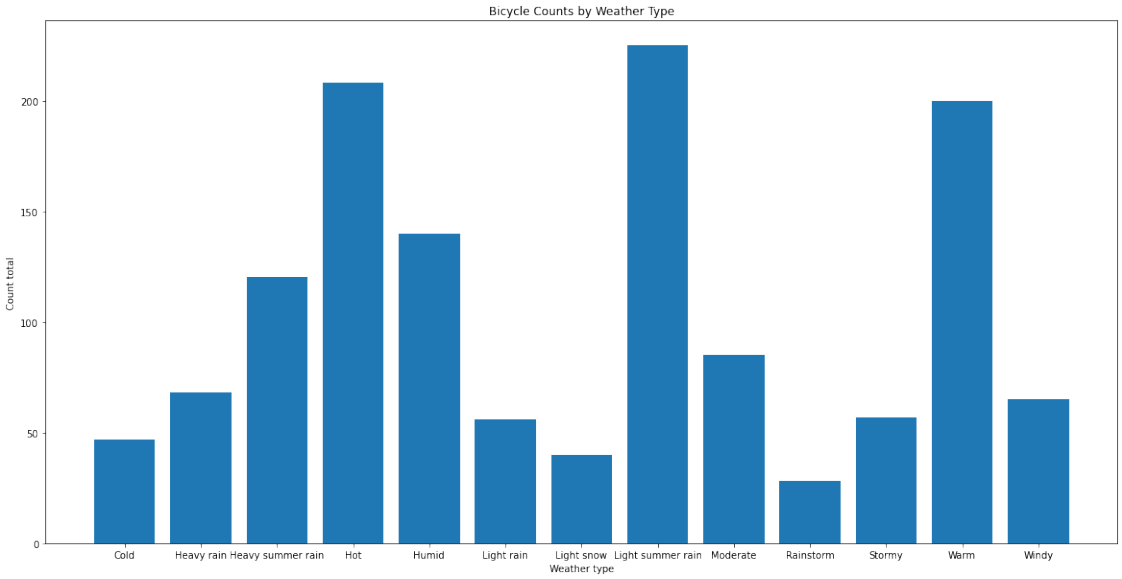
[18]:
```
# Calculate total counted bicycles from both directions and prioritize it in␣
↪columns
df["Count total"] = df["Count to"] + df["Count from"]
df = df[['Count total'] + [col for col in df.columns if col != 'Count total']]
```

[19]:
```
# Calculate descriptive statistics grouped by weather type
statistics = df.groupby('Weather type')['Count total'].agg(['mean', 'median',␣
↪'min', 'max', 'std'])
statistics
```

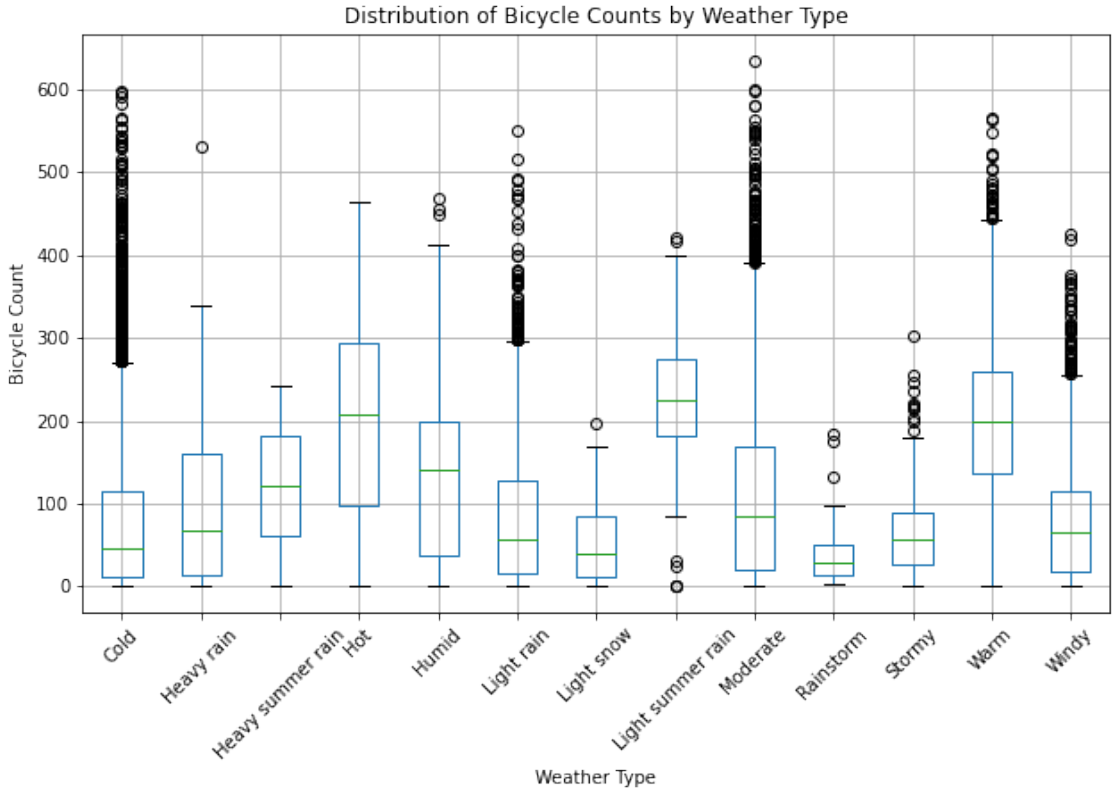
[19]:

Weather type	mean	median	min	max	std
Cold	74.167514	47.0	0	598	80.036989
Heavy rain	95.698413	68.0	0	531	93.768610
Heavy summer rain	120.500000	120.5	0	241	170.412734
Hot	194.019157	208.0	0	463	127.155279
Humid	137.796380	140.0	0	469	102.877726
Light rain	82.075810	56.0	0	549	82.113440
Light snow	53.341463	40.0	0	197	46.931382
Light summer rain	223.898438	225.0	0	420	81.293174
Moderate	104.856433	85.0	0	634	93.786068
Rainstorm	39.733333	28.0	3	183	40.254531
Stormy	70.103659	57.0	0	303	57.996575
Warm	196.271761	200.0	0	564	98.661005
Windy	79.703340	65.0	0	425	74.104734

[20]:
```
# Bar plot to compare bicycle counts for different weather types
plt.figure(figsize=(20, 10))
plt.bar(statistics["median"].index, statistics["median"].values)
plt.xlabel('Weather type')
plt.ylabel('Count total')
plt.title('Bicycle Counts by Weather Type')
plt.show()
```



[21]:
```
# Box plot to compare the distribution of bicycle counts for different weather types
df.boxplot('Count total', by='Weather type', figsize=(10, 6))

# Adjust the rotation of the x-axis labels
plt.xticks(rotation=45)

plt.xlabel('Weather Type')
plt.ylabel('Bicycle Count')
plt.title('Distribution of Bicycle Counts by Weather Type')
plt.suptitle('')  # Remove the automatically generated title
plt.show()
```



Through the analysis of "How do different weather types (sunny, rainy, cloudy, etc.) affect bicycle counts?", several key findings were identified. **Descriptive statistics and data visualizations** were utilized to explore the relationship between weather types and bicycle usage.

The analysis revealed that the weather types with the **highest bicycle usage** were identified as **light summer rain, hot, and warm**. These weather conditions seemed to be associated with increased bicycle usage. On the other hand, weather types such as **rainstorm, cold, and light snow** were found to have the least impact on bicycle counts, **indicating lower usage** during these conditions.

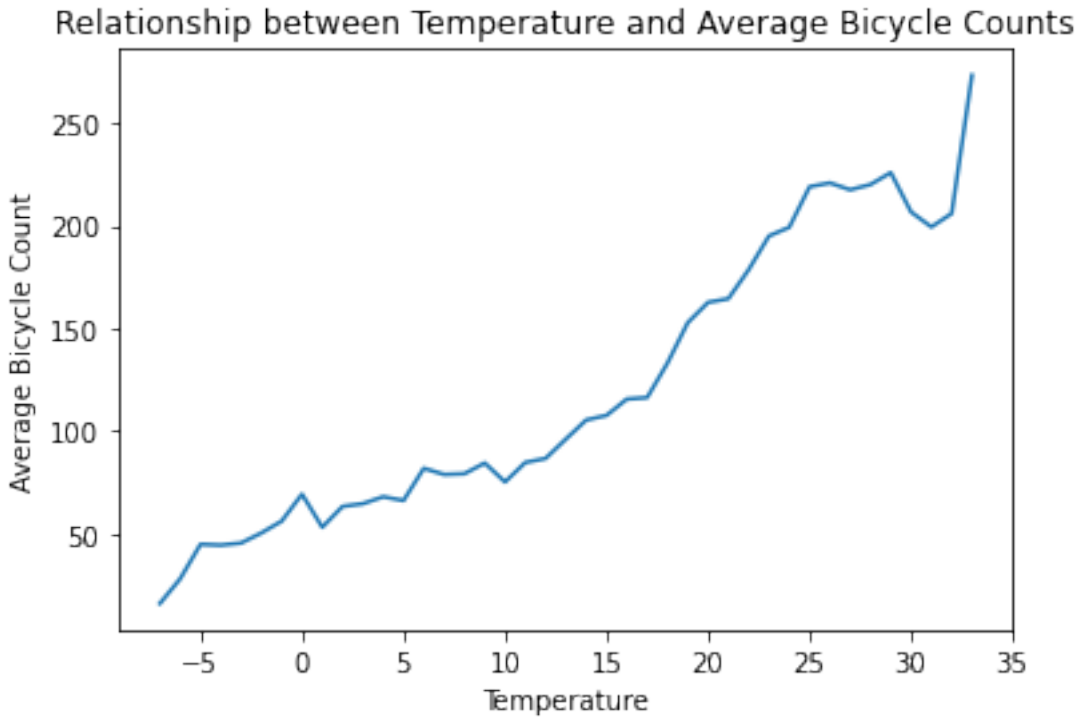
4.1.2 Is there a specific temperature range that correlates with higher bicycle usage?

[22]:
```
# Sort the dataset by temperature and calculate how many bicycles were
# used during those temperatures

dfSorted = df.sort_values('Temperature (°C)')

dictTemp = {}
for index, row in dfSorted.iterrows():
    myKey = row["Temperature (°C)"]
    myValue = row["Count total"]
    myCount = 1
```

```
    if myKey in dictTemp:
        dictTemp[myKey][0] += myValue
        dictTemp[myKey][1] += myCount

    else:
        dictTemp[myKey] = [myValue, myCount]
```

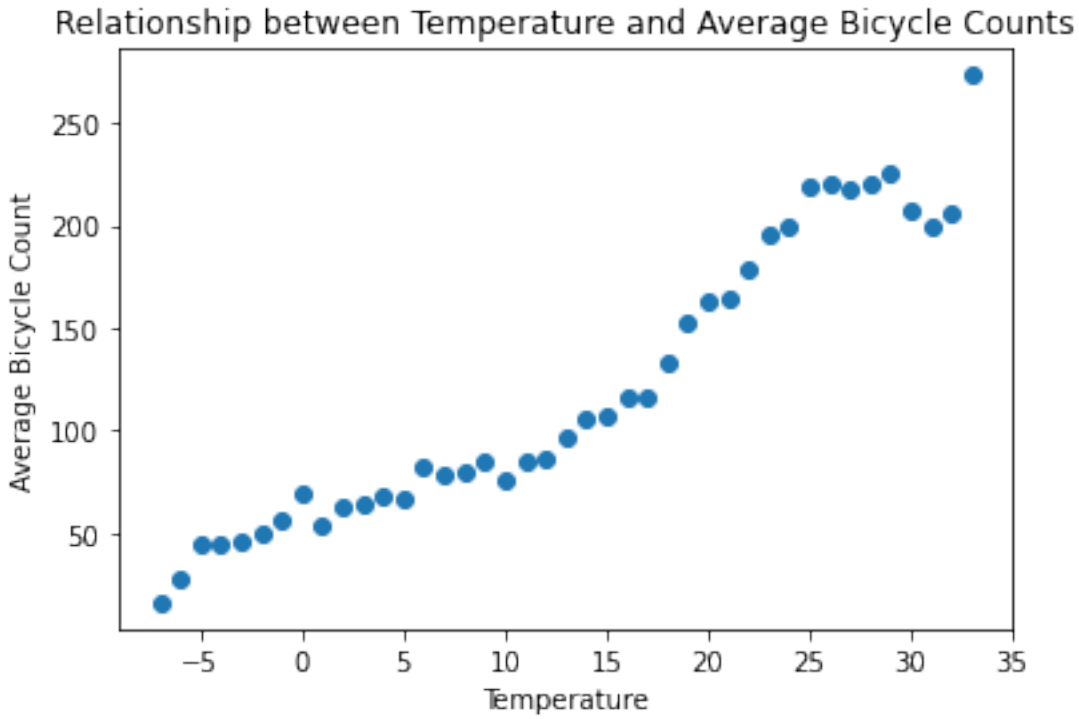
[23]:
```
# Calculate the average usage for each temperature
avg = []
for values in list(dictTemp.values()):
    avg.append(values[0]/values[1])
```

[24]:
```
# Line plot
plt.plot(list(dictTemp.keys()), avg)
plt.xlabel('Temperature')
plt.ylabel('Average Bicycle Count')
plt.title('Relationship between Temperature and Average Bicycle Counts')
plt.show()
```



[25]:
```
'''
Line plot shows that there is an increase of bicycle usage
from around 32 degrees but that does not make any sense.
```

```
Scatter plot helps us to see if it is just an outlier
and since we do not have any intermediate temperature,
it could have been just plotting the whole line for that.
'''

# Scatter plot
plt.scatter(list(dictTemp.keys()), avg)
plt.xlabel('Temperature')
plt.ylabel('Average Bicycle Count')
plt.title('Relationship between Temperature and Average Bicycle Counts')
plt.show()
```



Through the analysis of "Is there a specific temperature range that correlates with higher bicycle usage?", several insights were gained regarding the relationship between temperature and bicycle usage. The analysis revealed the following findings:

1. **Bicycle usage generally increases with higher temperatures**. As the temperature rises, there tends to be a corresponding increase in the number of bicycles being used.
2. There is a range of temperatures (around **25 to 30 degrees**) where the **bicycle usage remains relatively stable**. Within this temperature range, there is no significant variation in the number of bicycles being used.
3. **Beyond 30 degrees**, the **bicycle usage starts to decrease**. Very hot weather seems to have a negative impact on bicycle usage, potentially due to discomfort or other factors.

4. Notably, **at 34 degrees**, there is a **high number of bicycle usage**. This observation might be an outlier or have a specific explanation. I think the high bicycle usage at 34 degrees might not be solely attributed to the temperature itself but could be influenced by other factors such as peak usage hours coinciding with that temperature. For example, if the peak hour for bicycle usage is around 16:00 during the summer, more people may choose to ride bicycles regardless of the specific temperature.

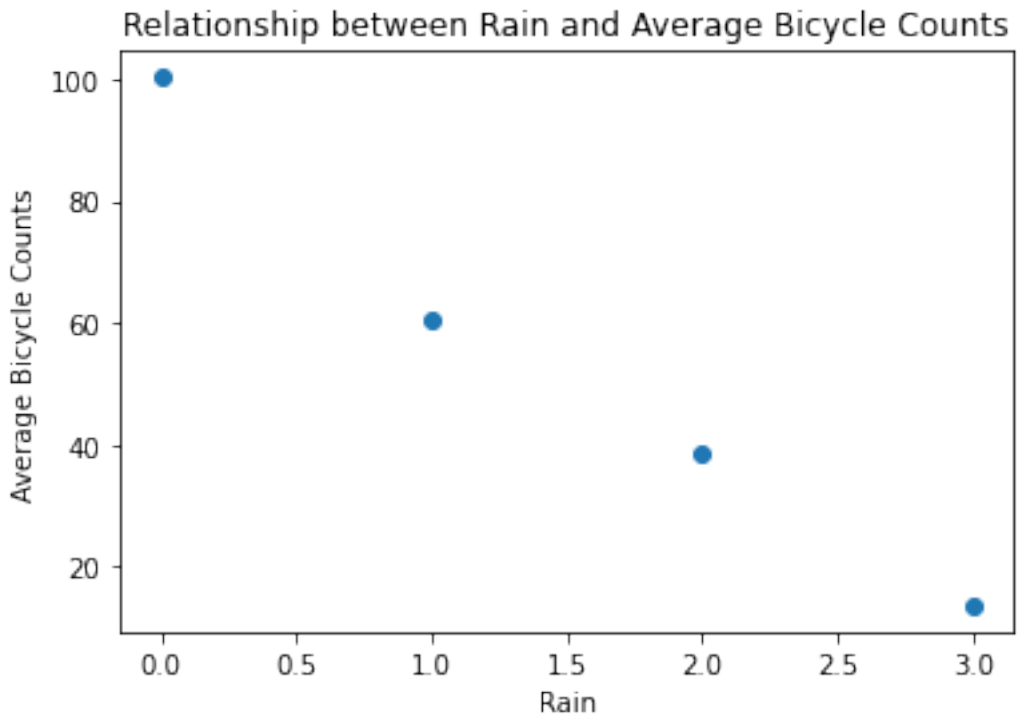
These findings provide valuable insights into the relationship between temperature and bicycle usage patterns, highlighting the impact of temperature on the number of bicycles being used and identifying temperature ranges where usage remains consistent or decreases.

4.1.3 Does rainfall significantly impact bicycle counts?

[26]:
```
# Sort dataframe by rain precipitation and create a dictionary
# to store number of bicycle counts for each rain values

dfSorted = df.sort_values('Rain (mm)')

dictRain = {}
for index, row in dfSorted.iterrows():
    myKey = row["Rain (mm)"]
    myValue = row["Count total"]
    myCount = 1

    if myKey in dictRain:
        dictRain[myKey][0] += myValue
        dictRain[myKey][1] += myCount

    else:
        dictRain[myKey] = [myValue, myCount]
```

[27]:
```
# Calculate the average number of bicycle counts for each rain values
avg = []
for key in dictRain.keys():
    avg.append([key, dictRain[key][0]/dictRain[key][1]])
```

[28]:
```
# Separate rainValues and countValues from the data
rainValues = [point[0] for point in avg]
countValues = [point[1] for point in avg]

# Create a scatter plot
plt.scatter(rainValues, countValues)
plt.xlabel('Rain')
plt.ylabel('Average Bicycle Counts')
plt.title('Relationship between Rain and Average Bicycle Counts')
plt.show()
```



Through the analysis of "Does rainfall significantly impact bicycle counts?", it was determined that rainfall does have a significant effect on bicycle usage. The findings were derived from a scatter plot visualization and indicate the following:

1. As the amount of **rain precipitation increases**, the average number of **bicycle usages notably decreases**.
2. When there is **no rain**, approximately **100 bicycles** pass every 15 minutes, indicating a higher level of bicycle usage under dry weather conditions.
3. With **1mm of rainfall**, the number of bicycles decreases to around **60**, suggesting a significant reduction in bicycle usage compared to dry weather.
4. As the rain precipitation increases to **2mm**, the number further declines to approximately **40 bicycles**, indicating a substantial decrease in usage.
5. When the rain reaches **3mm**, the number of bicycles decreases even further to **10**, highlighting a significant impact on bicycle usage during heavy rainfall.

These findings illustrate the clear correlation between rainfall and bicycle counts, indicating that higher levels of rain precipitation lead to a significant decrease in bicycle usage.

4.2 2. Temporal Patterns in Bicycle Usage:

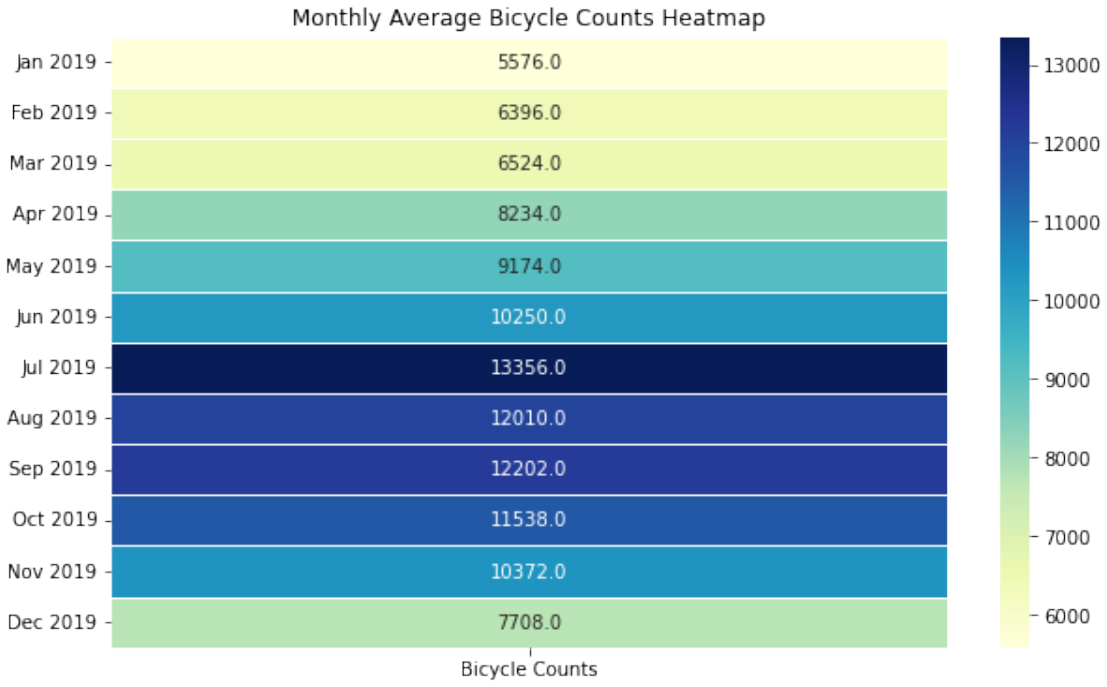
4.2.1 Are there any seasonal or monthly variations in bicycle counts?

[29]:
```
# Resample the data to monthly frequency and sum the bicycle counts
monthly_counts = df['Count total'].resample('M').sum()

# Calculate the number of days in each month
num_days = monthly_counts.index.days_in_month

# Calculate the average bicycle counts per day
average_counts_per_day = monthly_counts / num_days
average_counts_per_day = average_counts_per_day.astype(int)

# Convert the Series to a DataFrame with 'Month' as a column
df_heatmap = average_counts_per_day.to_frame(name='Bicycle Counts')

# Extract the 'Month' from the index and add it as a column
df_heatmap['Month'] = df_heatmap.index.strftime('%b %Y')

df_heatmap = df_heatmap.set_index("Month")
df_heatmap
```

[29]:

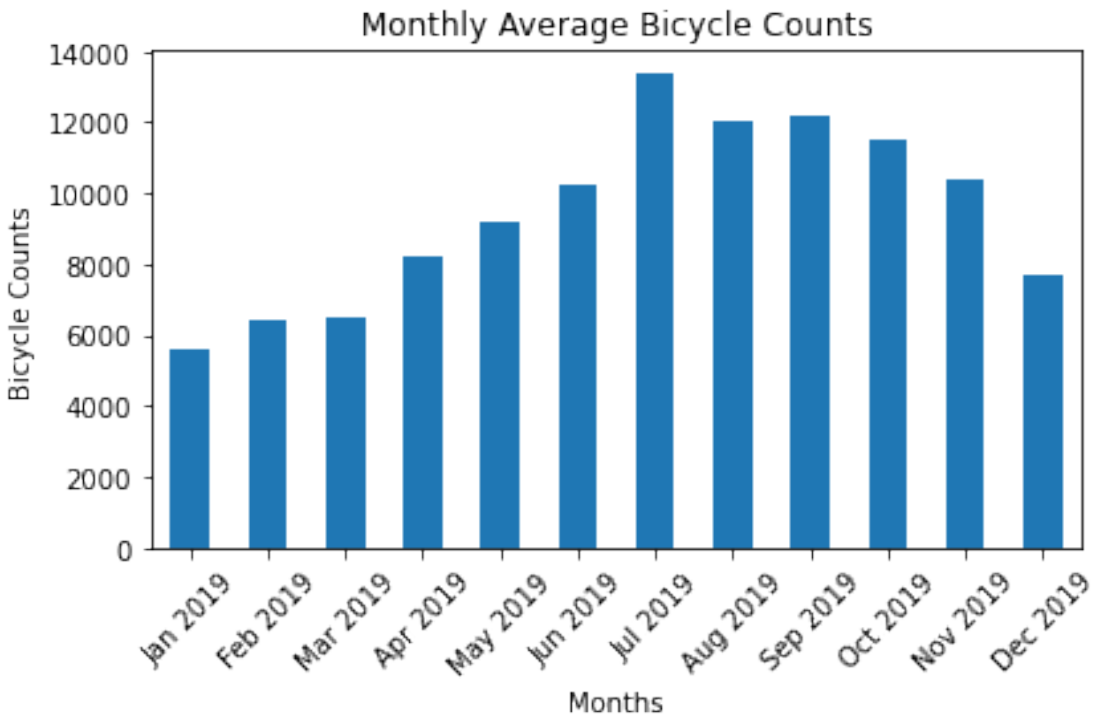
Month	Bicycle Counts
Jan 2019	5576
Feb 2019	6396
Mar 2019	6524
Apr 2019	8234
May 2019	9174
Jun 2019	10250
Jul 2019	13356
Aug 2019	12010
Sep 2019	12202
Oct 2019	11538
Nov 2019	10372
Dec 2019	7708

[30]:
```
# Create the heatmap plot
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, cmap='YlGnBu', annot=True, fmt=".1f", linewidths=0.5)
plt.title('Monthly Average Bicycle Counts Heatmap')
plt.xlabel('')
plt.ylabel('')

plt.show()
```



[31]:
```
# Create the bar plot
plt.figure(figsize=(10, 6))
df_heatmap.plot(kind='bar', legend=None)
plt.title('Monthly Average Bicycle Counts')
plt.xlabel('Months')
plt.ylabel('Bicycle Counts')

plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout for better spacing

plt.show()
```

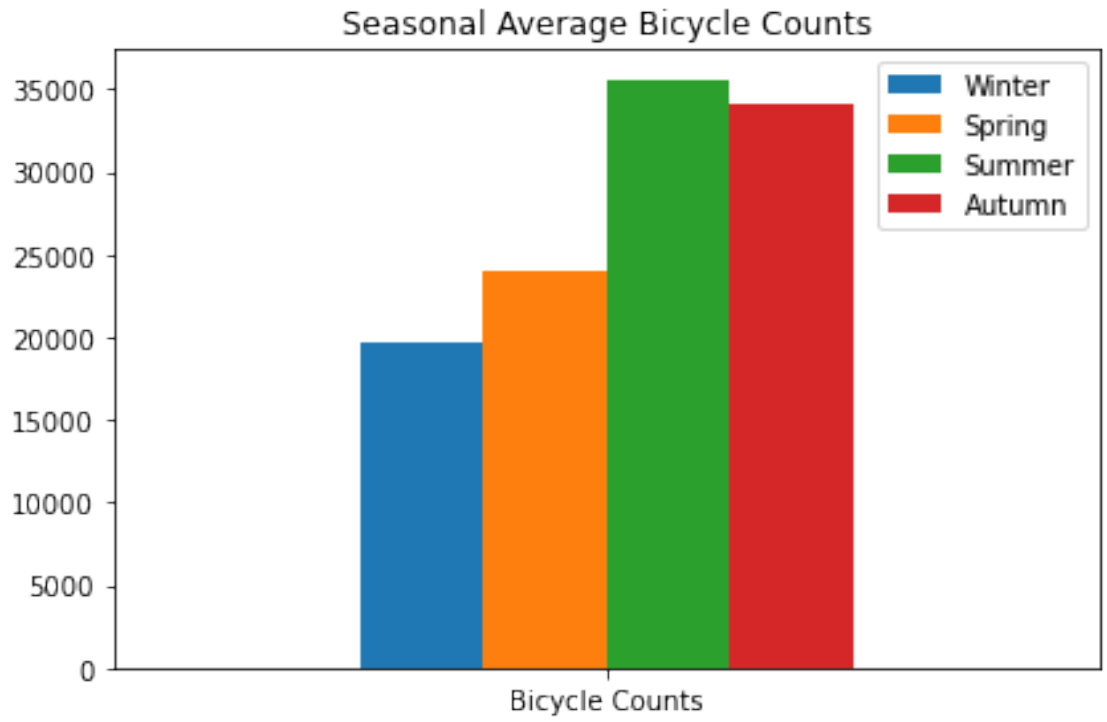
```
<Figure size 720x432 with 0 Axes>
```



[32]:
```python
# Define the seasons
seasons = {
    'Winter': ['Dec 2019', 'Jan 2019', 'Feb 2019'],
    'Spring': ['Mar 2019', 'Apr 2019', 'May 2019'],
    'Summer': ['Jun 2019', 'Jul 2019', 'Aug 2019'],
    'Autumn': ['Sep 2019', 'Oct 2019', 'Nov 2019']
}

# Create a new DataFrame with seasons
season_df = pd.DataFrame()
for season, months in seasons.items():
    season_counts = df_heatmap.loc[months].sum()
    season_df[season] = season_counts

# Create the bar plot for seasons
plt.figure(figsize=(10, 6))
season_df.plot(kind='bar')
plt.title('Seasonal Average Bicycle Counts')
plt.xlabel('')
plt.ylabel('')

plt.xticks(rotation=0)  # Rotate x-axis labels if needed
plt.tight_layout()  # Adjust layout for better spacing
```

```
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



Through the analysis of "Are there any seasonal or monthly variations in bicycle counts?", several observations were made regarding the seasonal and monthly patterns of bicycle usage. The findings can be summarized as follows:

1. Monthly variations: The analysis revealed that the months of **July, September, and August** exhibit the **highest** levels of bicycle usage. This trend suggests that there is a peak in bicycle usage during the summer months. The usage gradually increases from the beginning of the year, likely due to improving weather conditions. However, after a period of stability in high bicycle usage from August to October, the number of bicycle counts starts to decline towards the end of the year.
2. Seasonal variations: When examining the seasonal patterns, it was observed that the number of bicycle counts follows a decreasing trend from summer to winter. Specifically, the highest bicycle usage is observed during **summer, followed by autumn, spring, and winter**, in that order.

4.2.2 How does bicycle usage vary between weekdays and weekends?

[33]:
```
# Extract the weekday from the timestamp indexes
df['Weekday'] = df.index.dayofweek  # Monday is 0 and Sunday is 6

# Categorize weekdays as 'Weekday' and weekends as 'Weekend'
df['Day_Type'] = df['Weekday'].apply(lambda x: 'Weekday' if x < 5 else␣
↪'Weekend')

# Calculate descriptive statistics for bicycle counts by weekday type
stats = df.groupby('Day_Type')['Count total'].describe()
stats
```

[33]:

Day_Type	count	mean	std	min	25%	50%	75%	max
Weekday	24961.0	112.041104	101.462745	0.0	19.0	92.0	179.0	634.0
Weekend	9980.0	65.812725	64.091654	0.0	15.0	43.0	99.0	359.0

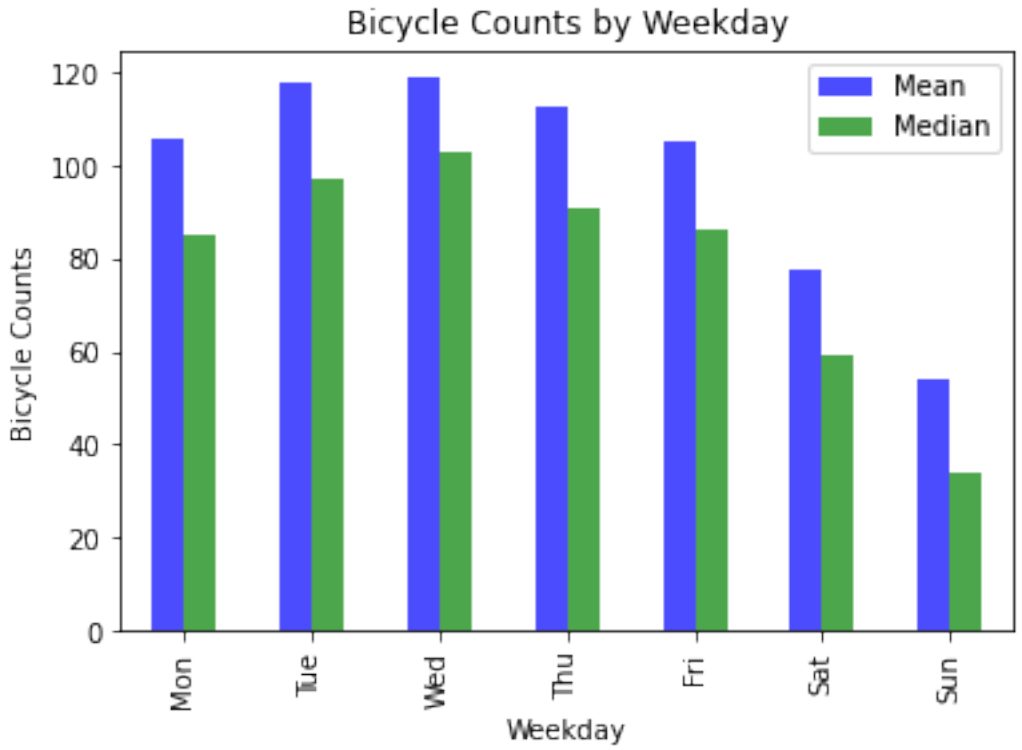
[34]:
```
# Calculate descriptive statistics for bicycle counts by weekday type
stats = df.groupby('Weekday')['Count total'].describe()

# Customize index labels to represent days of the week
stats = stats.drop('count', axis=1)
day_labels = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
↪'Saturday', 'Sunday']
stats.index = day_labels

stats
```

[34]:

	mean	std	min	25%	50%	75%	max
Monday	105.502003	99.473523	0.0	13.0	85.0	171.0	562.0
Tuesday	118.040857	108.078964	0.0	16.0	97.0	191.0	595.0
Wednesday	119.005208	105.092921	0.0	20.0	103.0	185.0	634.0
Thursday	112.602764	101.130967	0.0	22.0	91.0	178.0	598.0
Friday	105.053486	91.961606	0.0	21.0	86.0	168.0	564.0
Saturday	77.769231	69.684250	0.0	18.0	59.0	122.0	359.0
Sunday	53.846632	55.439082	0.0	13.0	34.0	77.0	300.0

[35]:
```
# Create a bar plot from the stats DataFrame
ax = stats[['mean', '50%']].plot(kind='bar', color=['blue', 'green'], alpha=0.7)

# Set the title and labels for the plot
plt.title('Bicycle Counts by Weekday')
plt.xlabel('Weekday')
plt.ylabel('Bicycle Counts')

# Set the x-axis tick labels
```

```
day_labels = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
plt.xticks(range(7), day_labels)

# Add a legend
ax.legend(['Mean', 'Median'], loc='upper right')

# Display the plot
plt.show()
```



Through the analysis of "How does bicycle usage vary between weekdays and weekends?", several key findings were identified:

1. Weekday vs. Weekend Usage: The analysis revealed that bicycle usage during **weekdays** is **significantly higher** compared to **weekends**. On average, the number of bicycles used during weekdays is approximately twice as much as during weekends. This suggests that bicycles are more frequently utilized for transportation purposes on weekdays, potentially for commuting or work-related activities.

2. Daily Usage Patterns: By analyzing the daily usage patterns, it was found that **Wednesday, Tuesday, and Thursday** exhibit the **highest levels** of bicycle usage. These days are followed by Monday and Friday, which still show relatively high usage but slightly lower than the mid-week peak. **Saturday and Sunday have the lowest bicycle usage**, indicating a decrease in demand over the weekend.

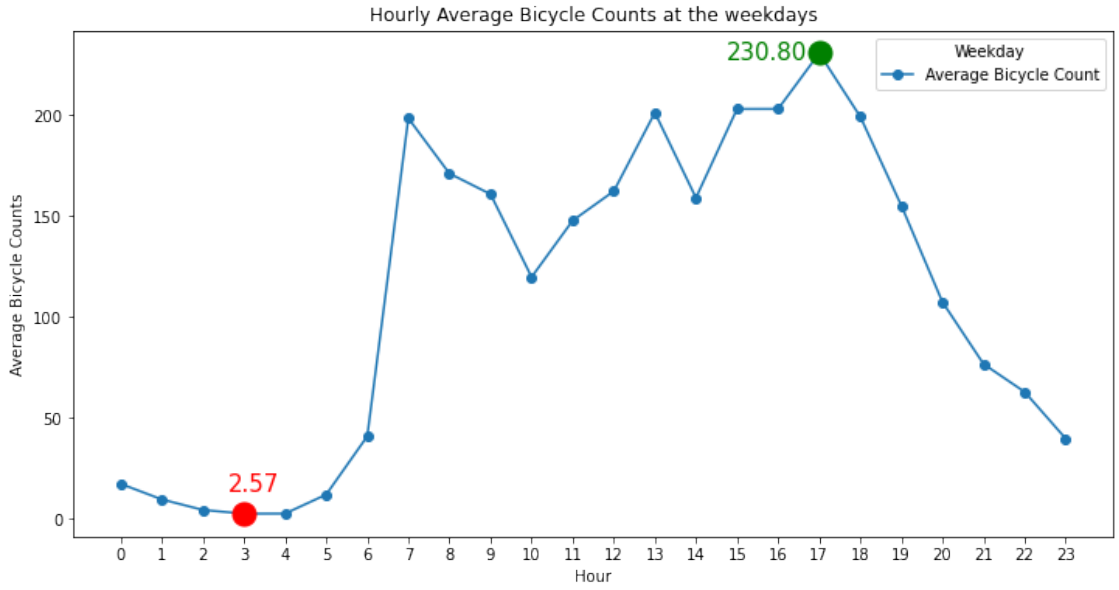
3. Median and Mean Analysis: The statistics and bar plot of median and mean values further support the findings. **Wednesday, Tuesday, and Thursday** consistently show the **highest median and mean bicycle** usage, indicating a consistent pattern of high usage throughout the weekdays. Monday and Friday exhibit slightly lower values, while **Saturday and Sunday have the lowest median and mean values**, reflecting reduced bicycle usage on weekends.

4.2.3 Are there any distinct peak hours or periods of high bicycle usage throughout the weekdays and weekends?

[36]:
```
# Extract daily hours from the timestamp indexes
df['Hour'] = df.index.hour
grouped = df.groupby(['Day_Type', 'Hour'])['Count total'].mean()
grouped.name = "Average Bicycle Count"
groupedWeekday = grouped.Weekday
groupedWeekend = grouped.Weekend
```

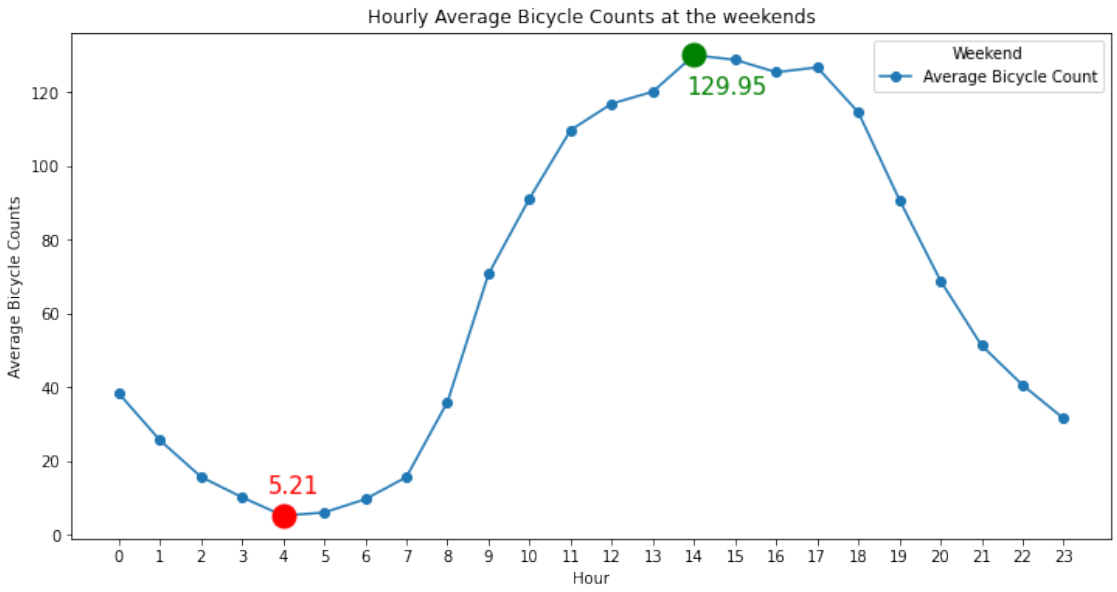
[37]:
```
# Plot the line with markers
line_plot_weekday = groupedWeekday.plot(kind='line', figsize=(12, 6),␣
 ↪marker='o')

# Get the indices of the minimum and maximum values
min_idx = [groupedWeekday.idxmin()]
max_idx = [groupedWeekday.idxmax()]

# Add markers for the minimum and maximum points with larger size
line_plot_weekday.plot(min_idx, groupedWeekday.loc[min_idx], marker='o',␣
 ↪markersize=15, color='red')
line_plot_weekday.plot(max_idx, groupedWeekday.loc[max_idx], marker='o',␣
 ↪markersize=15, color='green')

# Annotate the minimum and maximum values near the points
for idx, min_val in zip(min_idx, groupedWeekday.loc[min_idx]):
    line_plot_weekday.annotate(f'{min_val:.2f}', xy=(idx, min_val),␣
 ↪xytext=(-10, 15),
                        textcoords='offset points', color='red', fontsize = 15)

for idx, max_val in zip(max_idx, groupedWeekday.loc[max_idx]):
    line_plot_weekday.annotate(f'{max_val:.2f}', xy=(idx, max_val),␣
 ↪xytext=(-60, -4),
                        textcoords='offset points', color='green', fontsize = 15)

plt.title('Hourly Average Bicycle Counts at the weekdays')
plt.xlabel('Hour')
plt.ylabel('Average Bicycle Counts')
plt.legend(title='Weekday')

plt.xticks(range(24))
```

```
plt.show()
```



[38]:
```
# Plot the line with markers
line_plot_weekend = groupedWeekend.plot(kind='line', figsize=(12, 6),␣
 ↪marker='o')

# Get the indices of the minimum and maximum values
min_idx = [groupedWeekend.idxmin()]
max_idx = [groupedWeekend.idxmax()]

# Add markers for the minimum and maximum points with larger size
line_plot_weekend.plot(min_idx, groupedWeekend.loc[min_idx], marker='o',␣
 ↪markersize=15, color='red')
line_plot_weekend.plot(max_idx, groupedWeekend.loc[max_idx], marker='o',␣
 ↪markersize=15, color='green')

# Annotate the minimum and maximum values near the points
for idx, min_val in zip(min_idx, groupedWeekend.loc[min_idx]):
    line_plot_weekend.annotate(f'{min_val:.2f}', xy=(idx, min_val),␣
 ↪xytext=(-10, 15),
                       textcoords='offset points', color='red', fontsize = 15)

for idx, max_val in zip(max_idx, groupedWeekend.loc[max_idx]):
    line_plot_weekend.annotate(f'{max_val:.2f}', xy=(idx, max_val), xytext=(-5,␣
 ↪-25),
                       textcoords='offset points', color='green', fontsize = 15)
```

```
plt.title('Hourly Average Bicycle Counts at the weekends')
plt.xlabel('Hour')
plt.ylabel('Average Bicycle Counts')
plt.legend(title='Weekend')

plt.xticks(range(24))

plt.show()
```



The analysis of "Are there any distinct peak hours or periods of high bicycle usage throughout the weekdays and weekends?" yielded the following findings

Weekdays Analysis:

1. Peak Hours: The analysis of hourly average bicycle counts on weekdays revealed distinct peak hours of high bicycle usage. The **minimum** average bicycle usage occurred **at 03:00** in the morning, with a value of **2.57**. The **maximum** average bicycle usage was observed **at 17:00**, with a value of **230.80**.
2. Morning Commute: The usage **gradually increased from 06:00 to 08:00**, indicating that people likely use bicycles for commuting to their jobs and schools during this time.
3. Lunchtime and Midday Usage: There was a **slight decrease** in bicycle usage **from 08:00 to 11:00**, followed by an **increase from 11:00 to 13:00**. This suggests the presence of lunchtime riders or individuals using bicycles for leisure or errands during the day.
4. Evening Peak: Bicycle usage continued to **fluctuate and gradually increased** until reaching the maximum value at **17:00**. This peak hour may be attributed to people returning home from work or engaging in recreational activities with their bicycles.

5. Nighttime Decline: **After 17:00, bicycle usage decreased continuously** throughout the night, gradually reaching its lowest point at 03:00 the next morning.

Weekends Analysis:

1. Peak Hours: On weekends, the analysis showed a different pattern compared to weekdays. The **minimum** average bicycle usage occurred **at 04:00** in the morning, with a value of **5.21**. The usage **gradually increased** until reaching the maximum point at **14:00**, where bicycle usage was at its highest with the value **129.95**.
2. Stable and Smooth Usage: Unlike weekdays, the plot depicting **hourly average bicycle counts on weekends exhibited a smoother and more stable pattern**. The usage remained relatively high between 14:00 and 17:00 before declining continuously throughout the night.

These findings provide insights into the distinct patterns of peak hours and periods of high bicycle usage on both weekdays and weekends. Understanding these patterns can be valuable for urban planning, optimizing infrastructure, and implementing targeted interventions to support and encourage bicycle usage during specific hours of the day.

5 Conclusion

This project explored the patterns of bicycle usage Konstanz using two data sources. Analyzing weather conditions, temperature, rainfall, and temporal factors, we identified key insights. Bicycle usage was influenced by weather types, with light summer rain and warm weather leading to higher usage. Higher temperatures correlated with increased usage, except for a plateau between 25-30 degrees Celsius. Rainfall significantly decreased bicycle counts, while peak usage occurred in July, August, and September. Weekdays exhibited higher usage compared to weekends, with Wednesday, Tuesday, and Thursday showing the highest usage. Peak bicycle usage hours occurred during weekdays, with significant spikes during morning and evening commute hours. Weekends exhibited a more gradual increase in usage throughout the day.

In conclusion, this project shed light on the intricate patterns of bicycle usage, providing valuable insights for urban planners, transportation authorities, and policymakers. The findings emphasized the importance of weather conditions, temperature ranges, rainfall, and temporal factors in understanding and predicting bicycle usage. By harnessing this knowledge, cities can develop targeted strategies to promote cycling, enhance infrastructure, and improve the overall urban mobility experience. Ultimately, this endeavor contributes to creating more sustainable, efficient, and cyclist-friendly urban environments.