Import of packages

```
import os
import sys
import glob
import pandas
import numpy
import matplotlib
import matplotlib.pyplot
import seaborn
import sklearn
import sklearn.preprocessing
import sklearn.model_selection
import sklearn.metrics
import sklearn.ensemble
import sklearn.linear_model
import sklearn.tree
import sklearn.neighbors
import sklearn.svm
import sklearn.cluster
import sklearn.decomposition
import sklearn.manifold
import sklearn.metrics
import sklearn.pipeline
import sklearn.model_selection
import sklearn.preprocessing
import sklearn.ensemble
import sklearn.linear_model
import sklearn.tree
import sklearn.neighbors
import sklearn.svm
import sklearn.cluster
import sklearn.decomposition
import sklearn.manifold
import sklearn.metrics
from flask import Flask
from flask import render_template
from flask import redirect
from flask import request
from flask import jsonify
from flask.ext.bootstrap
from flask.ext.bootstrap.nav
from flask.ext.bootstrap.nav.sphinxdoc
from flask.ext.bootstrap.utils
from flask.ext.bootstrap.utils.bootstrap
from flask.ext.bootstrap.utils.bootstrap.get_nav_menu_items
from flask.ext.bootstrap.utils.bootstrap.get_msg_categories
from flask.ext.bootstrap.utils.bootstrap.get_flashed_messages
from flask.ext.bootstrap.utils.bootstrap.wtf
from flask.ext.bootstrap.utils.bootstrap.wtf.wtform_fields
from flask.ext.bootstrap.utils.bootstrap.wtf.wtform_fields.groupedselectfield
from flask.ext.bootstrap.utils.bootstrap.wtf.wtform_fields.groupedselectfield.GroupedSelectField
```

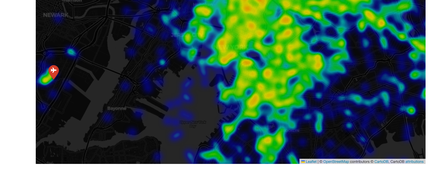
Part 1: Data Preprocessing

Yellow Taxi dataset

Downloading

```
# Download yellow taxi data, three months, parquet format
# ------------------------------------------------------------------------------
# Downloads and stores the yellow taxi data from the given base URL.
# ------------------------------------------------------------------------------
# Usage:
# ./download_yellow_taxi_data.sh <base_URL>
# ------------------------------------------------------------------------------
# Author: Yoav Wulf
# ------------------------------------------------------------------------------
# Last Edit: 2/2/2022
# ------------------------------------------------------------------------------
# Notes:
# - This is just wget with flags to download the yellow taxi data.
# - Stores data in ./data/yellow_taxi/.
# ------------------------------------------------------------------------------
base_URL="$1"
# Check if the first argument is empty.
if [ -z "$base_URL" ]; then
  echo -e "Usage: ./download_yellow_taxi_data.sh <base_URL>\n\nDownloads the yellow taxi data from the given base URL.\n\nNote: The base URL should be the URL that the data is stored under, without the trailing slash.\n\nExample: ./download_yellow_taxi_data.sh https://www.example.com/data/yellow_taxi\n\nWill download the data from https://www.example.com/data/yellow_taxi/yellow_tripdata_2021-01.parquet\n\nNote: This script will download the data to ./data/yellow_taxi/, if this directory does not exist, it will be created.\n\nNote: This script will download the data for the three months of 2021.\n\nNote: This script will not download the data if it already exists.\n\nNote: This script requires wget to be installed.\n\nNote: This script requires curl to be installed." exit 1
fi
# Set the base URL.
base_URL="${base_URL}/yellow_tripdata_"
# Set the directory to download the data to.
data_dir="./data/yellow_taxi/"
# Create the directory if it does not exist.
mkdir -p "$data_dir"
# Loop through the months.
for month in 01 02 03; do
  # Set the URL for the data.
  data_URL="${base_URL}2021-${month}.parquet"
  # Set the output file.
  output_file="${data_dir}yellow_tripdata_2021-${month}.parquet"
  # Download the data.
  wget -O "$output_file" "$data_URL"
  # Check if the download was successful.
  if [ $? -eq 0 ]; then
    echo -e "Downloaded ${data_URL} to ${output_file}\n\n\n
```

```
-- ORDER BY pickup_zone, t.hour;
-- ORDER BY pickup_zone, t.hour
-- ORDER BY pickup_zone, t.hour
-- ORDER BY pickup_zone, t.hour
```



Heatmap with animation

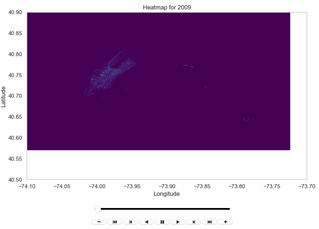
Please do not execute the commented codes below until all static plots have been generated. Once you execute the command matplotlib.use("Agg"), the plot will be generated in the backend and will not be displayed in the notebook.

In [78]:
```
# matplotlib.use("Agg")
# def visualize_heatmap(ax, year):
#     query = f"""
#     SELECT start_lat, start_lon
#     FROM (
#         SELECT start_lat, start_lon
#         FROM yellow_taxi_database
#         WHERE start_lat IS NOT NULL AND start_lon IS NOT NULL AND strftime('%Y', pickup_datetime) = '{year}'
#         UNION ALL
#         SELECT start_lat, start_lon
#         FROM uber_database
#         WHERE start_lat IS NOT NULL AND start_lon IS NOT NULL AND strftime('%Y', pickup_datetime) = '{year}'
#     )
#     """
#     df = pd.read_sql_query(query, conn)
#
#     # Center map on NYC
#     nyc_map = folium.Map(location=[40.7128, -74.0060], zoom_start=12, tiles='CartoDB dark_matter', attr='<a href="https://carto.com/attrib
#
#     # Create a heatmap
#     heatmap_data = [[row['start_lat'], row['start_lon']] for index, row in df.iterrows()]
#     plugins.HeatMap(heatmap_data, radius=12, max_zoom=20, blur=10).add_to(nyc_map)
#
#     # Add airport markers
#     airports = {
#         'JFK': [40.6413, -73.7781],
#         'LGA': [40.7769, -73.8740],
#         'EWR': [40.6895, -74.1745]
#     }
#
#     for airport, coords in airports.items():
#         folium.Marker(
#             location=coords,
#             popup=f"{airport} Airport",
#             icon=folium.Icon(icon='plane', prefix='fa', color='red')
#         ).add_to(nyc_map)
#
#     # Add fullscreen button
#     plugins.Fullscreen(position='topright', title='Expand me', title_cancel='Exit me', force_separate_button=True).add_to(nyc_map)
#
#     ax.clear()
#     ax.set_title(f"Heatmap for {year}")
#     ax.hist2d(df['start_lat'], df['start_lat'], bins=[300, 300], cmap='viridis',
#               range=[[df['start_lat'].min(), df['start_lat'].max()], [df['start_lat'].min(), df['start_lat'].max()]])
#     ax.set_xlim(-74.1, -73.7)
#     ax.set_ylim(40.5, 40.9)
#     ax.set_xlabel("Longitude")
#     ax.set_ylabel("Latitude")
#
#     return ax
```

Display animated heatmap

In [79]:
```
# conn = sqlite3.connect("project.db")

# fig, ax = plt.subplots(figsize=(10, 6))

# def update(year):
#     visualize_heatmap(ax, str(year))

# years = range(2009, 2016)
# ani = FuncAnimation(fig, update, frames=years, repeat=True)

# # Display the animation in the Jupyter notebook
# HTML(ani.to_jshtml())
```

Out[79]:



Display animated heatmap as GIF

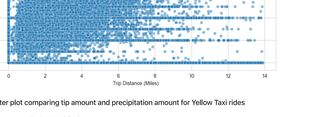
In [72]:
```
# # Save the animation as a GIF
# ani.save("heatmap_animation.gif", writer='pillow', fps=2)

# # Display the GIF in the Jupyter notebook
# Image(filename="heatmap_animation.gif")
```

Out[72]:
```
<IPython.core.display.Image object>
```

Scatter plot comparing tip amount and distance for Yellow Taxi ride

If you are using matplotlib.use("Agg") to implement an animated heatmap, please note that you will not be able to display the following plots if the previous matplotlib.use("Agg") command is executed.

In [63]:
```
def visualize_scatter_tip_distance():
    query = "SELECT Tip_Amt, Real_Distance FROM yellow_taxi_database;"
    df = pd.read_sql_query(query, conn)

    # Remove outliers
    df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]

    # Apply filters to remove outliers
    filtered_data = df[(df['Tip_Amt'] > 0) & (df['Real_Distance'] > 0)]


    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x='Real_Distance', y='Tip_Amt', alpha=0.5)
    plt.title('Tip Amount vs. Trip Distance')
    plt.xlabel('Trip Distance (Miles)')
    plt.ylabel('Tip Amount ($)')
    plt.show()

# Switch back to the default backend "TkAgg"
import matplotlib
visualize_scatter_tip_distance()
```



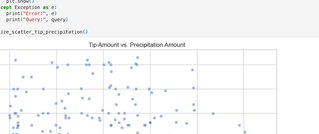
Scatter plot comparing tip amount and precipitation amount for Yellow Taxi rides

Tip amount vs HOURLY Precipitation

In [64]:
```
def visualize_scatter_tip_hourly_precipitation():
    # Join the taxi_trips and hourly_weather_data tables
    query = """
        SELECT Tip_Amt, HourlyPrecipitation
        FROM yellow_taxi_database as taxi
        JOIN hourly_weather_database as weather
        ON DATE(taxi.Pickup_Datetime) = weather.Date AND strftime('%H', taxi.Pickup_Datetime) = weather.Hour
    """

    # Read the data into a Pandas DataFrame
    df = pd.read_sql_query(query, conn)

    # Apply filters to remove outliers
    df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
    filtered_data = df[(df['Tip_Amt'] > 0) & (df['Tip_Amt'] <= 30) & (df['HourlyPrecipitation'] > 0) & (df['HourlyPrecipitation'] <= 10)]

    # Create a scatter plot using Seaborn
    sns.set(style='whitegrid')
    scatter_plot = sns.scatterplot(x=filtered_data['HourlyPrecipitation'], y=filtered_data['Tip_Amt'], data=filtered_data, s=20, alpha=0.5,

    # Add a regression line
    sns.regplot(x='HourlyPrecipitation', y='Tip_Amt', data=filtered_data, scatter=False, line_kws={"color": "red"})

    # Customize plot appearance
    scatter_plot.set_xlabel('Hourly Precipitation (inches)', fontsize=12)
    scatter_plot.set_ylabel('Tip Amount (USD)', fontsize=12)
    scatter_plot.set_title('Tip Amount vs. Hourly Precipitation for Yellow Taxi Rides', fontsize=14)

    # Remove the top and right spines for a cleaner look
    sns.despine()

    # Show the plot
    plt.show()
    plt.show()

visualize_scatter_tip_hourly_precipitation()
```

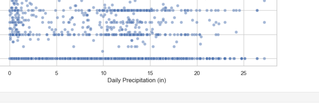
Tip amount vs DAILY Precipitation

In [68]:
```
def visualize_scatter_tip_daily_precipitation():
    # Join the taxi_trips and hourly_weather_data tables
    query = """
        SELECT Tip_Amt, DailyPrecipitation
        FROM yellow_taxi_database as taxi
        JOIN daily_weather_database as weather
        ON DATE(taxi.Pickup_Datetime) = weather.Date AND strftime('%H', taxi.Pickup_Datetime) = weather.Hour
    """

    # Read the data into a Pandas DataFrame
    df = pd.read_sql_query(query, conn)

    # Apply filters to remove outliers
    df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
    filtered_data = df[(df['Tip_Amt'] > 0) & (df['Tip_Amt'] <= 30) & (df['DailyPrecipitation'] > 0) & (df['DailyPrecipitation'] <= 10)]

    # Create a scatter plot using Seaborn
    sns.set(style='whitegrid')
    scatter_plot = sns.scatterplot(x=filtered_data['DailyPrecipitation'], y=filtered_data['Tip_Amt'], data=filtered_data, s=20, alpha=0.5, e

    # Add a regression line
    sns.regplot(x='DailyPrecipitation', y='Tip_Amt', data=filtered_data, scatter=False, line_kws={"color": "red"})

    # Customize plot appearance
    scatter_plot.set_xlabel('Daily Precipitation (inches)', fontsize=12)
    scatter_plot.set_ylabel('Tip Amount (USD)', fontsize=12)
    scatter_plot.set_title('Tip Amount vs. Hourly Precipitation for Yellow Taxi Rides', fontsize=14)

    # Remove the top and right spines for a cleaner look
    sns.despine()

    # Show the plot
    plt.show()
    plt.show()

visualize_scatter_tip_daily_precipitation()
```



Tip amount vs HOURLY Precipitation - IMPLEMENTATION 2

In [69]:
```
def visualize_scatter_tip_precipitation():
    query = """
    SELECT Tip_Amt, DailyPrecipitation
    FROM yellow_taxi_database
    JOIN daily_weather_database ON Pickup_Datetime = Record_Time;
    """
    try:
        df = pd.read_sql_query(query, conn)
        df['Tip_Amt'] = pd.to_numeric(df['Tip_Amt'], errors='coerce')
        df['DailyPrecipitation'] = pd.to_numeric(df['DailyPrecipitation'], errors='coerce')

        # Remove outliers using z-score normalization
        df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]

        # Create a scatter plot
        plt.figure(figsize=(10, 6))
        sns.scatterplot(data=df, x='DailyPrecipitation', y='Tip_Amt', alpha=0.5)

        plt.title('Tip Amount vs. Precipitation Amount')
        plt.xlabel('Daily Precipitation (in)')
        plt.ylabel('Tip Amount ($)')
        plt.show()
    except Exception as e:
        print("Error:", e)
        print("Query:", query)

visualize_scatter_tip_precipitation()
```



In [ ]: