Practical-11

Aim:- To Implement big data visualization using data shader.

Libraries:-

1) **datashader**:
   - ➢ A high-performance visualization library that efficiently renders large datasets by transforming data into images. It is particularly useful for visualizing big data.
2) **pandas**:
   - ➢ A data manipulation library for Python. It is used to handle and preprocess the dataset, like converting date columns and ensuring latitude and longitude values are numeric.
3) **numpy**:
   - ➢ A library for numerical computations in Python. It is often used to handle arrays and matrices, although it isn't explicitly used in this code but supports underlying operations.
4) **matplotlib**:
   - ➢ A plotting library that allows static data visualizations. In this case, it is used to display the Datashader output as a static image.
5) **datashader.transfer_functions**:
   - ➢ Provides functions for transforming the aggregated data into visually appealing images. Here, it is used to apply shading (tf.shade) to the aggregated data.
6) **colorcet**:
   - ➢ A library for perceptually uniform colormaps. It is used to choose a color palette (cc.fire) for the interactive plot.
7) **holoviews**:
   - ➢ A library that provides high-level building blocks for building interactive visualizations. It works seamlessly with Datashader to create interactive plots.
8) **bokeh**:
   - ➢ A visualization library for creating interactive plots. It integrates with HoloViews to provide interactivity like zooming and hovering over the data.

Theory:-

- **Datashader** is used for efficient rendering of large datasets into images. It aggregates data points (e.g., latitude and longitude) and applies visual transformations to create meaningful plots.
- **Matplotlib** is used to display static plots for big data, generated by Datashader.
- **HoloViews** and **Bokeh** are used for creating interactive visualizations, which allow zooming and data exploration in real-time. HoloViews works with Datashader to render large datasets interactively.
- The **count()** operation in Datashader aggregates data points (e.g., Uber ride locations) based on their longitude and latitude, creating a plot where the density of points is visualized through color shading.

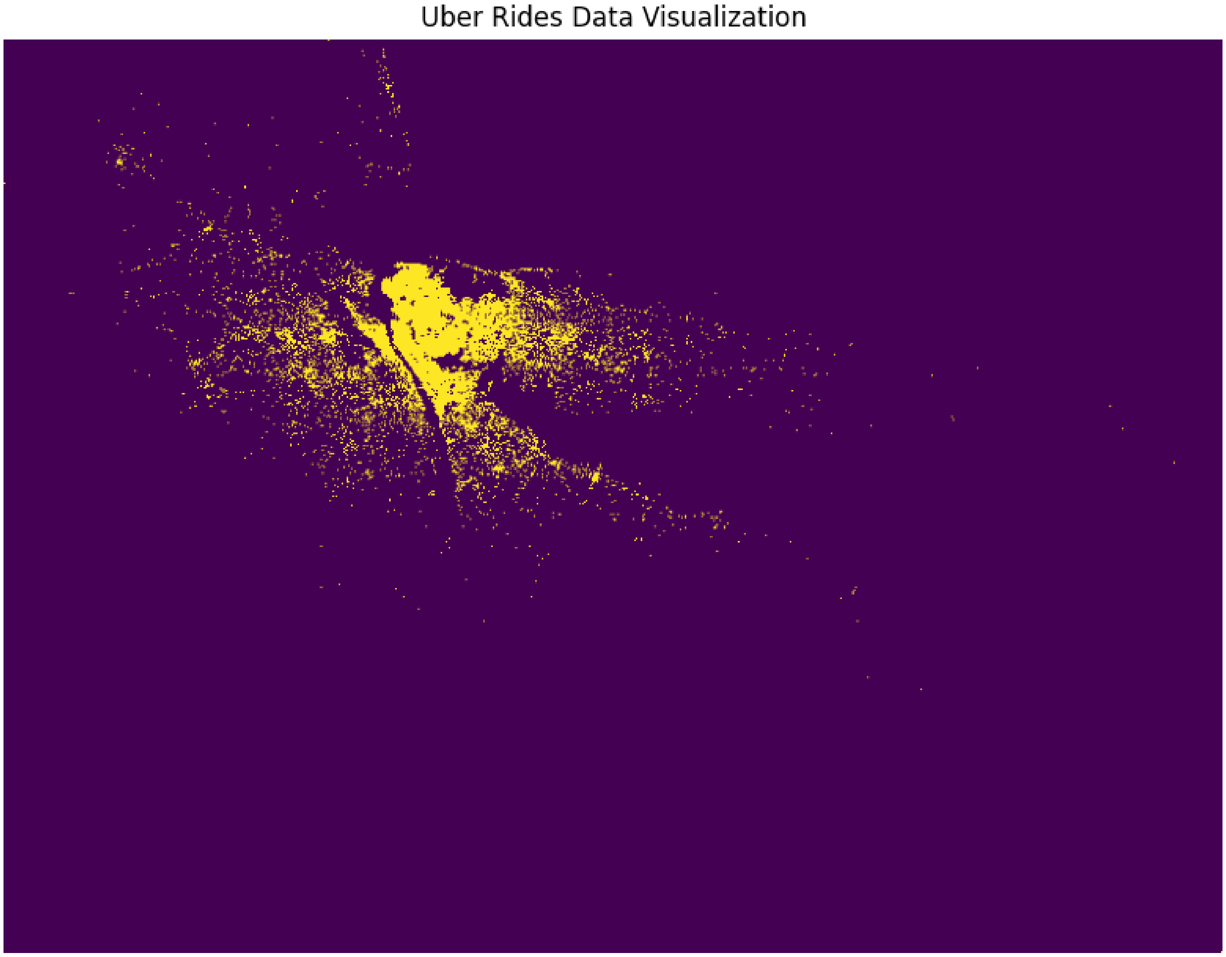
Code:-

```
 # Install necessary libraries for Google Colab

!pip install -U datashader

!pip install -U colorcet

!pip install -U holoviews

!pip install -U bokeh

!pip install -U pandas

# Import necessary libraries

import datashader as ds

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from datashader import transfer_functions as tf
from datashader.colors import viridis
import colorcet as cc
import holoviews as hv
from holoviews.operation.datashader import datashade
import bokeh.io

# Enable the Bokeh extension for interactive plots
hv.extension('bokeh')
bokeh.io.output_notebook()  # Enable Bokeh output in Jupyter notebook
# Step 1: Load the dataset (using a URL path)
url = 'https://raw.githubusercontent.com/plotly/datasets/master/uber-rides-data1.csv'
df = pd.read_csv(url)
# Step 2: Convert the 'Date/Time' column to datetime objects
df['Date/Time'] = pd.to_datetime(df['Date/Time'])
# Step 3: Extract latitude and longitude
df['lat'] = df['Lat'].astype(float)  # Ensure 'Lat' is numeric
df['lon'] = df['Lon'].astype(float)  # Ensure 'Lon' is numeric
# Step 4: Create a Datashader Canvas
cvs = ds.Canvas(plot_width=800, plot_height=600)
# Step 5: Aggregate the data points
agg = cvs.points(df, 'lon', 'lat', ds.count())
# Step 6: Apply shading for visualization
```

```
img = tf.shade(agg, cmap=viridis, how='eq_hist')
# Step 7: Display the visualization using Matplotlib (Static Image)
plt.figure(figsize=(10, 10))
plt.imshow(img.data)   # Access the image data using img.data instead of
img.to_array()
plt.axis('off')  # Hide the axis for better presentation
plt.title('Uber Rides Data Visualization')  # Add a title to the image
plt.show()
# Step 8: Display using HoloViews and Bokeh (Interactive Plot)
# Create a HoloViews dataset with 'lon' and 'lat' dimensions
hv_ds = hv.Points(df, ['lon', 'lat'])
# Create a datashaded plot with color map from colorcet
shaded_plot = datashade(hv_ds, cmap=cc.fire, width=800, height=600)
# Add interactivity with hover tool
shaded_plot = shaded_plot.opts(
    tools=['hover'],  # Enable hover tool
    width=800,
    height=600,
    title="Uber Rides Data",
    bgcolor='white',  # Set the background color to white
    toolbar='above'   # Place the toolbar above the plot
)
# Display the interactive plot
```

```
hv.render(shaded_plot)  # Use hv.render to display in a separate HTML file
```

Output:-



Prcatical-12

Aim:- To implement sentiment analysis using text blob.

Libraries:-

1) **text blob**:

   - **Text Blob** is a simple NLP library that provides easy-to-use tools for processing text, including performing sentiment analysis. It returns two important values: **polarity** (a measure of sentiment) and **subjectivity** (a measure of personal opinion).

2) **Vader Sentiment**:

   - **VADER** (Valence Aware Dictionary and sentiment Reasoned) is a rule-based sentiment analysis tool, specifically designed for analyzing social media text. It provides more fine-grained sentiment analysis, including **positive**, **negative**, **neutral**, and **compound** sentiment scores.

Theory:-

Text Blob is a simple library used for basic sentiment analysis. It evaluates text based on two key aspects:

1. **Polarity**: Measures sentiment from **-1** (negative) to **+1** (positive).
2. **Subjectivity**: Measures whether the text expresses personal opinions (**0** = objective, **1** = subjective).

Text Blob analyzes the overall emotional tone of the text, determining if it’s positive or negative and if it’s based on facts or opinions.

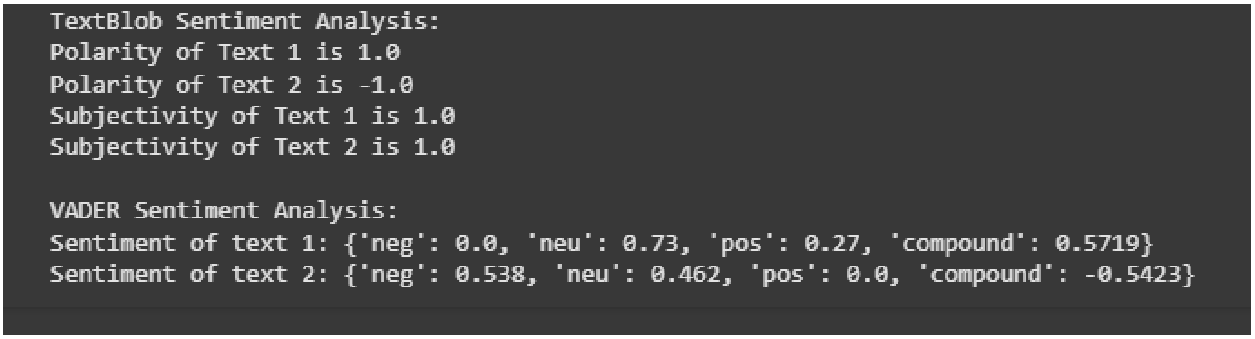
Text Blob provides a simple and effective way to determine the sentiment and subjectivity of text, making it useful for analyzing general sentiment in various types of content.

Code:-

```
# Install necessary libraries for Google Collab
!pip install -U textblob
!pip install -U vaderSentiment
# ------------------------------
# TextBlob Sentiment Analysis
# ------------------------------
from textblob import TextBlob
# Sample texts
text_1 = "The movie was so awesome."
text_2 = "The food here tastes terrible."
# Determining the Polarity using TextBlob
p_1 = TextBlob(text_1).sentiment.polarity
p_2 = TextBlob(text_2).sentiment.polarity
# Determining the Subjectivity using TextBlob
s_1 = TextBlob(text_1).sentiment.subjectivity
s_2 = TextBlob(text_2).sentiment.subjectivity
# Printing TextBlob Results
print("TextBlob Sentiment Analysis:")
print("Polarity of Text 1 is", p_1)
print("Polarity of Text 2 is", p_2)
print("Subjectivity of Text 1 is", s_1)
print("Subjectivity of Text 2 is", s_2)
# ------------------------------
# VADER Sentiment Analysis
# ------------------------------
```

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Initialize VADER sentiment analyzer

sentiment = SentimentIntensityAnalyzer()

# Sample texts for VADER analysis

text_1 = "The book was a perfect balance between writing style and plot."

text_2 = "The pizza tastes bad."

# Get sentiment scores for text_1 and text_2

sent_1 = sentiment.polarity_scores(text_1)

sent_2 = sentiment.polarity_scores(text_2)

# Printing VADER Results

print("\nVADER Sentiment Analysis:")

print("Sentiment of text 1:", sent_1)

print("Sentiment of text 2:", sent_2)
```

Output:-



```
TextBlob Sentiment Analysis:
Polarity of Text 1 is 1.0
Polarity of Text 2 is -1.0
Subjectivity of Text 1 is 1.0
Subjectivity of Text 2 is 1.0

VADER Sentiment Analysis:
Sentiment of text 1: {'neg': 0.0, 'neu': 0.73, 'pos': 0.27, 'compound': 0.5719}
Sentiment of text 2: {'neg': 0.538, 'neu': 0.462, 'pos': 0.0, 'compound': -0.5423}
```