<u>**NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD**</u>

Data Mining(Assignment 01)

<u>**Assignment: 01**</u>

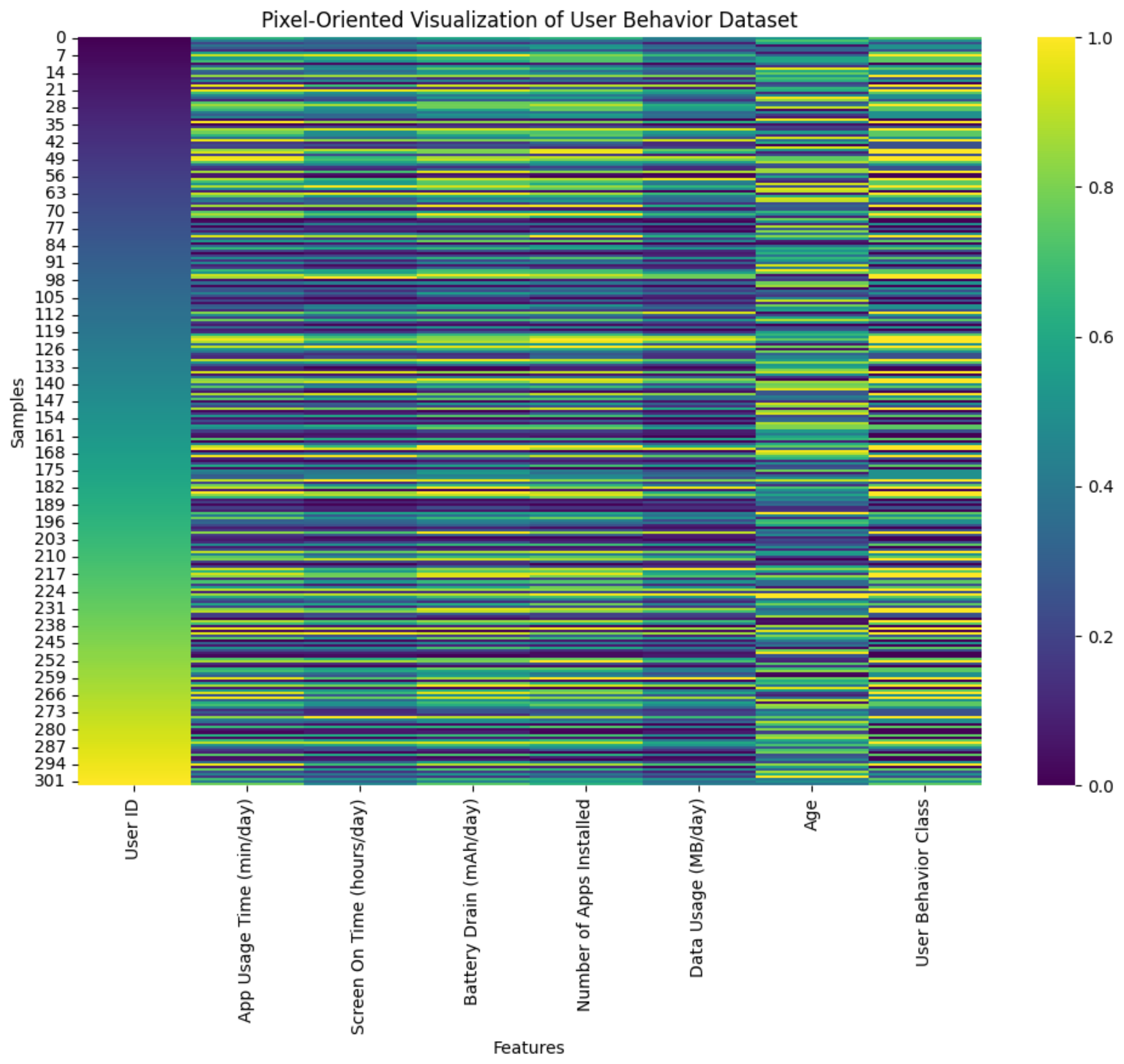
Submitted to
Dr moiz ullah Ghori

Submitted By
Adeel Naeem
(BSAI-146)

Submission Date: octuber 6th,2024

Pixel-Oriented Visualization:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

# Assuming you have downloaded the dataset from Kaggle and it's in the same directory
df = pd.read_csv(r'C:\Users\Adeel\Downloads\user_behavior_dataset.csv')
df.head(5)

numerical_columns = df.select_dtypes(include=[np.number]).columns
data = df[numerical_columns]
data = data.fillna(0)

scaler = MinMaxScaler()
data_normalized = scaler.fit_transform(data)

plt.figure(figsize=(12, 8))
sns.heatmap(data_normalized, cmap="viridis", cbar=True, xticklabels=numerical_columns)
plt.title('Pixel-Oriented Visualization of User Behavior Dataset')
plt.xlabel('Features')
plt.ylabel('Samples')
plt.show()
```



Chernoff Faces:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

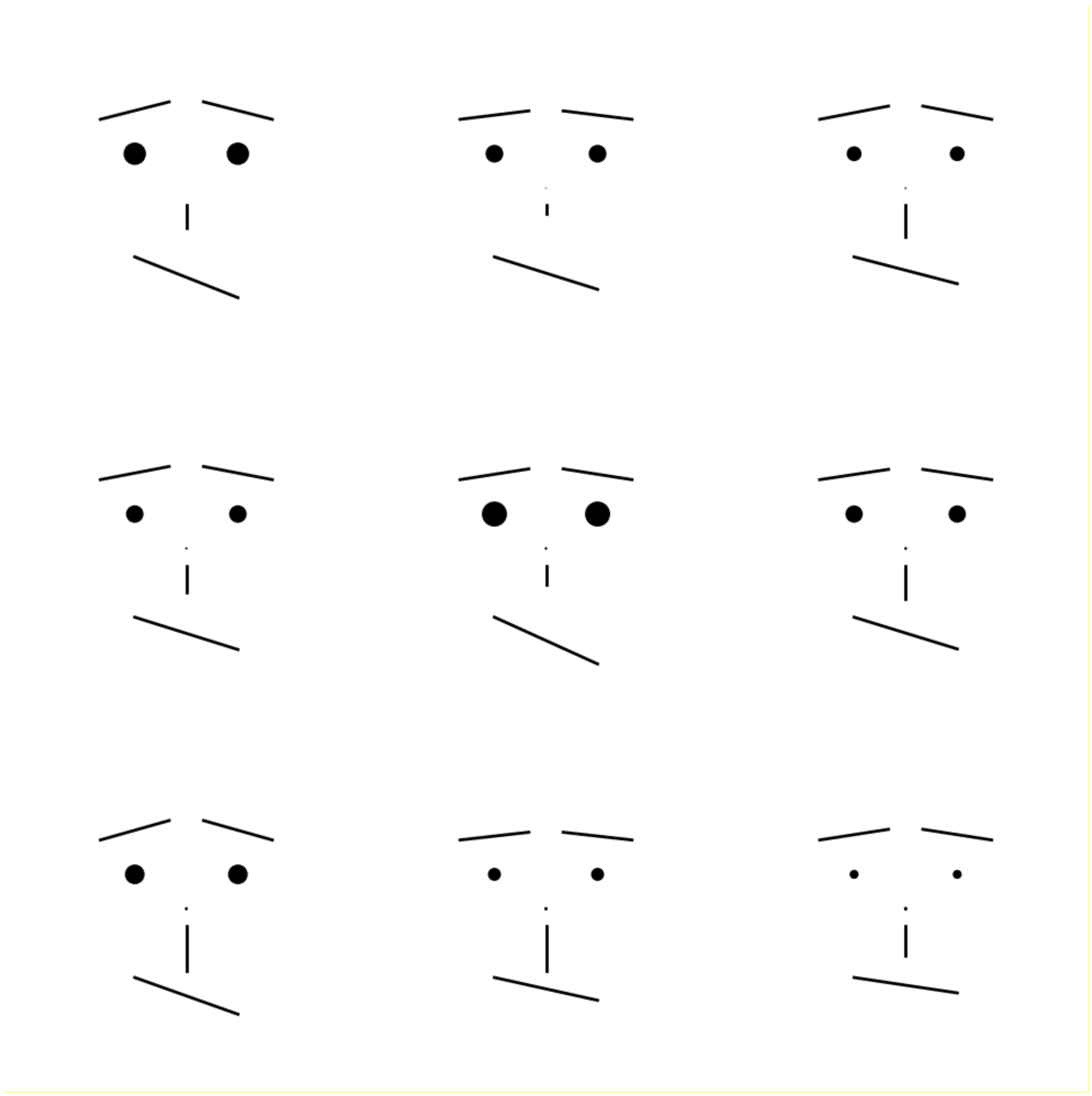
from matplotlib.patches import Ellipse

df = pd.read_csv(r'C:\Users\Adeel\Downloads\Stores.csv')

print(df.head())

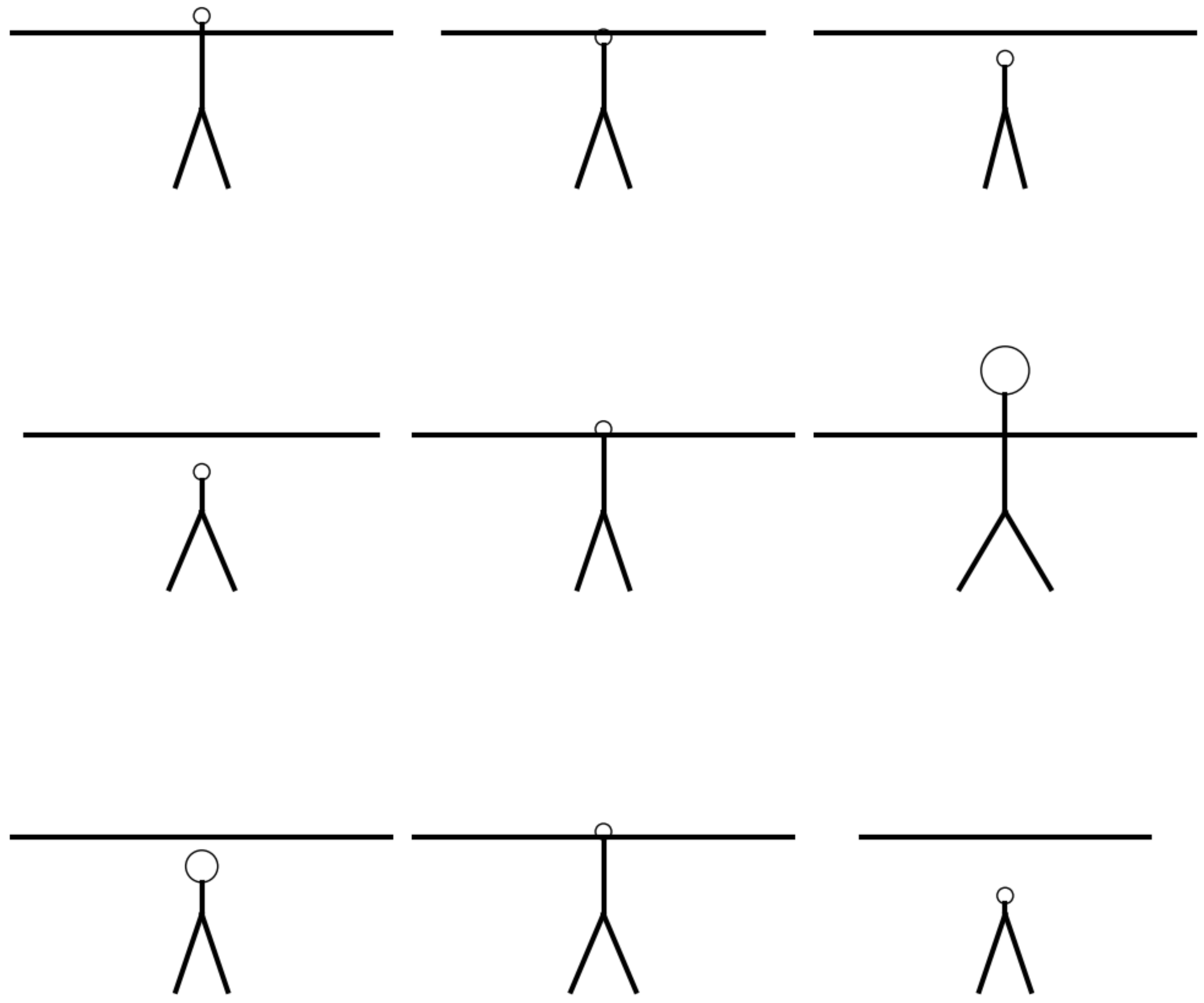
data = df[['Store ID ', 'Store_Area', 'Items_Available', 'Daily_Customer_Count', 'Store_Sales']].dropna()

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_normalized = scaler.fit_transform(data)
def draw_chernoff_face(ax, face_size, eye_size, mouth_curve, nose_length, eyebrow_angle):
    face = Ellipse((0.5, 0.5), width=0.6 * face_size, height=0.8 * face_size, facecolor='bisque',
edgecolor='black')
    ax.add_patch(face)
    ax.add_patch(plt.Circle((0.35, 0.6), 0.05 * eye_size, color='black'))
    ax.add_patch(plt.Circle((0.65, 0.6), 0.05 * eye_size, color='black'))
    mouth = plt.Line2D([0.35, 0.65], [0.3, 0.3 - 0.2 * mouth_curve], color='black', lw=2)
    ax.add_line(mouth)
    nose = plt.Line2D([0.5, 0.5], [0.45, 0.45 - 0.2 * nose_length], color='black', lw=2)
    ax.add_line(nose)
    left_eyebrow = plt.Line2D([0.25, 0.45], [0.7, 0.7 + 0.1 * eyebrow_angle], color='black', lw=2)
    right_eyebrow = plt.Line2D([0.55, 0.75], [0.7 + 0.1 * eyebrow_angle, 0.7], color='black',
lw=2)
    ax.add_line(left_eyebrow)
    ax.add_line(right_eyebrow)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.set_aspect('equal')
    ax.axis('off')
fig, axes = plt.subplots(3, 3, figsize=(10, 10))
axes = axes.flatten()
for i in range(9):
    ax = axes[i]

    face_size = data_normalized[i, 0]        # Sales
    eye_size = data_normalized[i, 1]         # Profit
    mouth_curve = data_normalized[i, 2]      # Discount
    nose_length = data_normalized[i, 3]      # Quantity
    eyebrow_angle = data_normalized[i, 4]    # Shipping Cost

    draw_chernoff_face(ax, face_size, eye_size, mouth_curve, nose_length, eyebrow_angle)

plt.tight_layout()
```



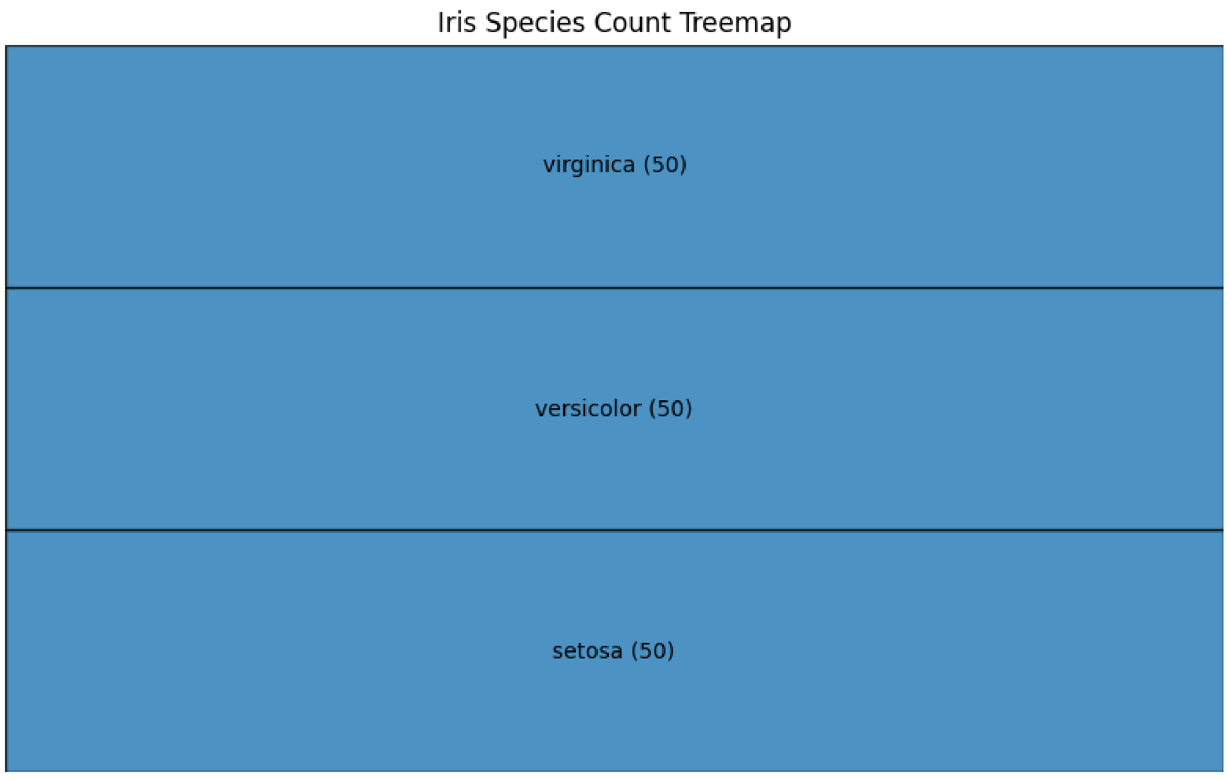
Stick Figures:

```
import pandas as pd

import matplotlib.pyplot as plt

import numpy as np


df = pd.read_csv(r'C:\Users\Adeel\Downloads\iris.csv')

print(df.head())


features = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]


from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

normalized_features = scaler.fit_transform(features)


def draw_stick_figure(ax, body_height, arm_length, leg_length, head_size):

    # Draw body (vertical line)

    ax.plot([0.5, 0.5], [0.3, 0.3 + body_height], color='black', lw=3)  # body


    # Draw head (circle on top of body)

    head = plt.Circle((0.5, 0.3 + body_height + head_size / 2), radius=head_size / 2, color='black', 
fill=False)

    ax.add_patch(head)

    # Draw arms (horizontal line from the middle of the body)


    ax.plot([0.5 - arm_length, 0.5 + arm_length], [0.5, 0.5], color='black', lw=3)  # arms


    # Draw legs (two diagonal lines starting from bottom of body)

    ax.plot([0.5, 0.5 - leg_length], [0.3, 0.1], color='black', lw=3)  # left leg

    ax.plot([0.5, 0.5 + leg_length], [0.3, 0.1], color='black', lw=3)  # right leg
    # Set plot limits and turn off axis
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.set_aspect('equal')
    ax.axis('off')
fig, axes = plt.subplots(3, 3, figsize=(10, 10))
axes = axes.flatten()

# Visualizing the first 9 flowers in the dataset
for i in range(9):
    ax = axes[i]

    # Map the Iris flower features to stick figure attributes
    body_height = normalized_features[i, 0]  # Sepal length
    arm_length = normalized_features[i, 1]   # Sepal width
    leg_length = normalized_features[i, 2]   # Petal length
    head_size = normalized_features[i, 3]    # Petal width

    # Draw the stick figure using the mapped features
    draw_stick_figure(ax, body_height, arm_length, leg_length, head_size)
plt.tight_layout()
plt.show()
```



Tree Maps

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv(r'C:\Users\Adeel\Downloads\iris.csv')

df_grouped = df['species'].value_counts().reset_index()
df_grouped.columns = ['Species', 'Count']  # Rename columns for clarity

#  Normalize counts for the tree map
total_count = df_grouped['Count'].sum()
df_grouped['Count'] = df_grouped['Count'] / total_count  # Normalize

# Create Tree Map
plt.figure(figsize=(10, 6))

# Initialize position variables
x0 = 0
y0 = 0
width = 1
height = 1

# Draw rectangles for each species
for _, row in df_grouped.iterrows():
    # Calculate height based on normalized count
    sub_height = row['Count'] * height
    plt.gca().add_patch(plt.Rectangle((x0, y0), width, sub_height, alpha=0.8, edgecolor='black'))

    # Add label inside the rectangle
    plt.text(x0 + width / 2, y0 + sub_height / 2,
             f"{row['Species']} ({row['Count'] * total_count:.0f})",
             ha='center', va='center', fontsize=10)

    y0 += sub_height  # Move y position for the next species

#  Customize the plot
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.title('Iris Species Count Treemap')
plt.axis('off')  # Turn off the axis
plt.show()
```


Iris Species Count Treemap
virginica (50)
versicolor (50)
setosa (50)