

Data Intake Report

Name: Cloud and API deployment

Report date: 7th July 2024

Internship Batch: LISUM34

Version:1.0

Data intake by: Biswadip Bhattacharyya

Data intake reviewer:<intern who reviewed the report>

Data storage location: <location URL eg: github, cloud>

Tabular data details:

Total number of observations	1000
Total number of files	1
Total number of features	7
Base format of the file	.csv
Size of the data	20 kb

1.Dataset:

The Dataset provides a comprehensive view into the dynamics of online matchmaking interactions. It captures essential variables that influence the likelihood of successful matches across different genders.

Variables:

- **Gender:** 0 (Male), 1 (Female)
- **PurchasedVIP:** 0 (No), 1 (Yes)
- **Income:** Annual income in USD
- **Children:** Number of children
- **Age:** Age of the user
- **Attractiveness:** Subjective rating of attractiveness (1-10)
- **Matches:** Number of matches obtained based on criteria

Target Variable:

- **Matches:** Number of matches received, indicative of success rate in online dating

2. Investigation of data:

```
import pandas as pd
```

```
df=pd.read_csv("C:/ads/DataGlacier/week4/Online_Dating_Behavior_Dataset.csv")  
df
```

✓ 1.0s

	Gender	PurchasedVIP	Income	Children	Age	Attractiveness	Matches
0	0	1	51777	3	47	5	70
1	1	0	36646	0	42	7	130
2	0	0	53801	1	25	5	0
3	0	0	56105	0	35	8	0
4	0	0	55597	1	36	6	0
...
995	0	0	36799	0	28	1	0
996	0	1	43882	2	46	9	70
997	1	0	49629	2	49	6	120
998	1	0	45706	1	22	8	140
999	0	0	43075	0	43	3	0

1000 rows × 7 columns

```
df.info()
```

```
[2]
```

```
✓ 0.0s
```

```
..
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1000 entries, 0 to 999
```

```
Data columns (total 7 columns):
```

#	Column	Non-Null Count	Dtype
0	Gender	1000 non-null	int64
1	PurchasedVIP	1000 non-null	int64
2	Income	1000 non-null	int64
3	Children	1000 non-null	int64
4	Age	1000 non-null	int64
5	Attractiveness	1000 non-null	int64
6	Matches	1000 non-null	int64

```
dtypes: int64(7)
```

```
memory usage: 54.8 KB
```

```
df.isna().sum()
```

```
[3]
```

```
✓ 0.0s
```

```
..
```

```
Gender      0
```

```
PurchasedVIP  0
```

```
Income      0
```

```
Children    0
```

```
Age         0
```

```
Attractiveness  0
```

```
Matches     0
```

```
dtype: int64
```

```
duplicates=df.duplicated()  
duplicates
```

✓ 0.0s

```
0      False  
1      False  
2      False  
3      False  
4      False
```

...

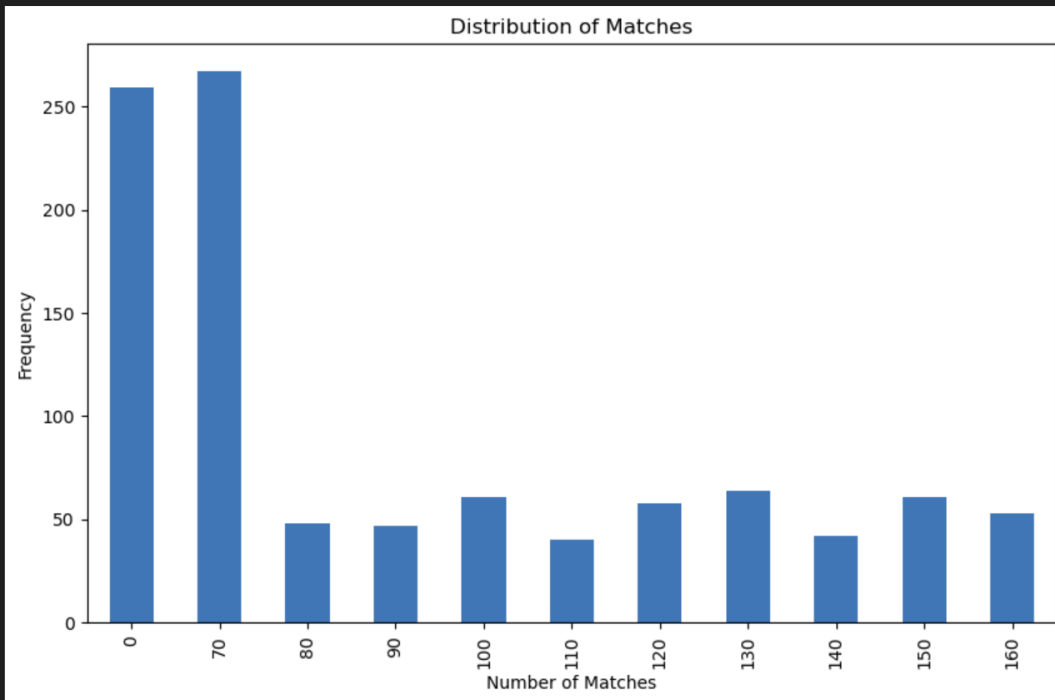
```
995     False  
996     False  
997     False  
998     False  
999     False
```

```
Length: 1000, dtype: bool
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
df['Matches'].value_counts().sort_index().plot(kind='bar')
plt.title('Distribution of Matches')
plt.xlabel('Number of Matches')
plt.ylabel('Frequency')
plt.show()
```

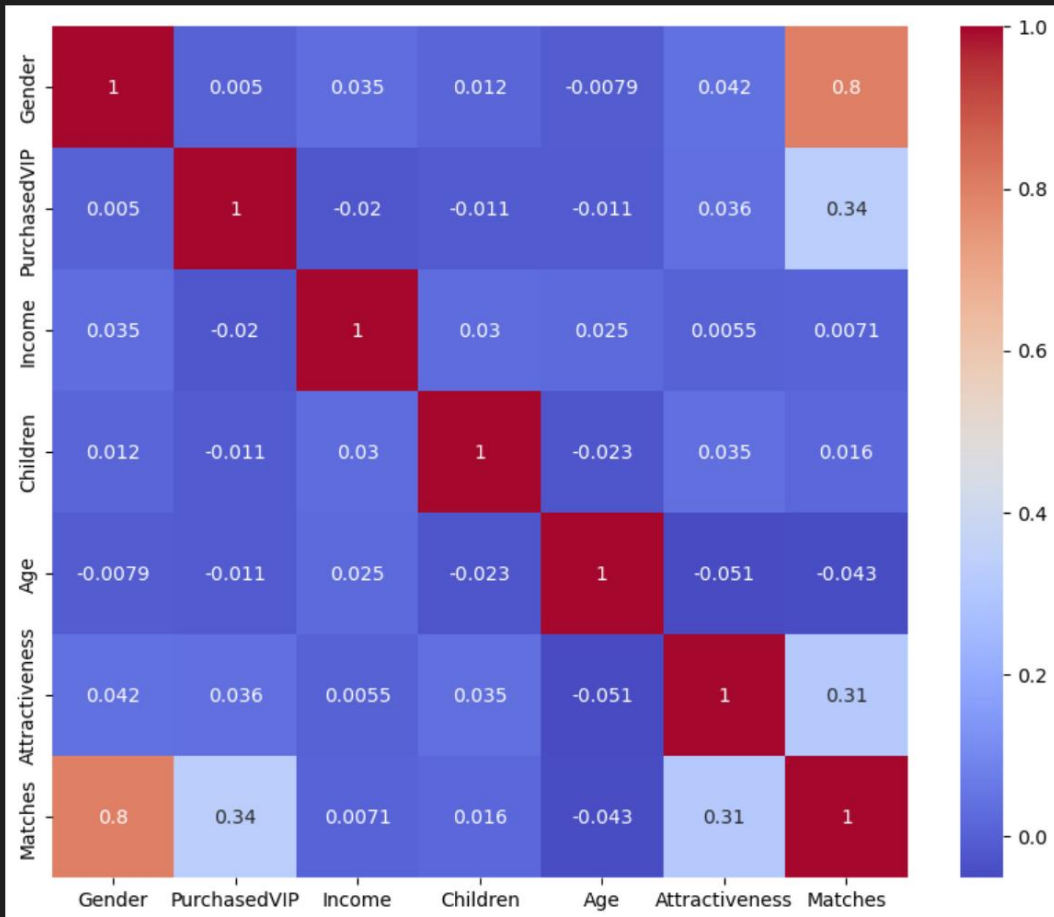
✓ 0.9s



```
import seaborn as sns
corr_matrix = df.corr()

# Plot heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

✓ 1.3s



```
import numpy as np

# Calculate quartiles
Q1 = np.percentile(df['Income'], 25)
Q3 = np.percentile(df['Income'], 75)

# Calculate IQR
IQR = Q3 - Q1

# Define limits
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

# Remove outliers
filtered_data = df[(df['Income'] >= lower_limit) & (df['Income'] <= upper_limit)]

print("Original data:", df)
print("Filtered data (without outliers):", filtered_data)
Q1
```

✓ 0.0s

Original data:		Gender	PurchasedVIP	Income	Children	Age	Attractiveness	Matches
0	0	1	51777	3	47	5	70	
1	1	0	36646	0	42	7	130	
2	0	0	53801	1	25	5	0	
3	0	0	56105	0	35	8	0	
4	0	0	55597	1	36	6	0	
..	
995	0	0	36799	0	28	1	0	
996	0	1	43882	2	46	9	70	
997	1	0	49629	2	49	6	120	
998	1	0	45706	1	22	8	140	
999	0	0	43075	0	43	3	0	

[1000 rows x 7 columns]

Filtered data (without outliers):		Gender	PurchasedVIP	Income	Children	Age	Attractiveness	Matches
0	0	1	51777	3	47	5	70	
1	1	0	36646	0	42	7	130	
2	0	0	53801	1	25	5	0	
3	0	0	56105	0	35	8	0	

3. Model building:

```
x=filtered_data.drop(columns=['Matches'])
y=filtered_data['Matches']
```

✓ 0.0s

+ Code

+ Markdo

x

✓ 0.0s

	Gender	PurchasedVIP	Income	Children	Age	Attractiveness
0	0	1	51777	3	47	5
1	1	0	36646	0	42	7
2	0	0	53801	1	25	5
3	0	0	56105	0	35	8
4	0	0	55597	1	36	6
...
995	0	0	36799	0	28	1
996	0	1	43882	2	46	9
997	1	0	49629	2	49	6
998	1	0	45706	1	22	8
999	0	0	43075	0	43	3

992 rows × 6 columns

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Check the shapes of the resulting datasets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

✓ 0.3s

```
X_train shape: (793, 6)
X_test shape: (199, 6)
y_train shape: (793,)
y_test shape: (199,)
```



```

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Fit the scaler on the training data and transform the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data (note: we only transform the test data, not fit)
X_test_scaled = scaler.transform(X_test)

# Check the scaled training data
print("Scaled X_train:\n", X_train_scaled)
print("Scaled X_test:\n", X_test_scaled)

```

✓ 0.0s

```

Scaled X_train:
[[-1.01653005 -0.97385982 -0.4262749  1.01920173  1.46738229  0.17512899]
 [ 0.98373875  1.02684183 -0.64275935  1.01920173 -1.71224557  1.22457954]
 [ 0.98373875 -0.97385982 -0.42688586 -1.0013657  -0.72546451  0.17512899]
 ...
 [ 0.98373875 -0.97385982  0.86377286  0.00891802 -0.83510685  0.87476269]
 [ 0.98373875 -0.97385982  0.14405879  0.00891802 -0.06761047  1.22457954]
 [ 0.98373875  1.02684183 -0.63838078  1.01920173  0.59024357 -1.22413841]]

Scaled X_test:
[[-1.01653005  1.02684183 -0.83439799  0.00891802  0.69988591 -0.52450471]
 [-1.01653005  1.02684183  0.36471816 -1.0013657  0.48060123 -1.57395526]
 [ 0.98373875 -0.97385982  0.834243  2.02948545 -1.49296089 -0.87432156]
 ...
 [-1.01653005 -0.97385982 -1.16818728  1.01920173  0.15167421  0.17512899]
 [ 0.98373875  1.02684183 -0.63267847 -1.0013657  1.02881293 -0.17468786]
 [-1.01653005 -0.97385982 -1.33131432  1.01920173  0.26131655 -0.17468786]]

```

```

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import joblib
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model on the scaled training data
rf_regressor.fit(X_train_scaled, y_train)

# Make predictions on the test set
y_pred = rf_regressor.predict(X_test_scaled)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", mse**0.5)

joblib.dump(rf_regressor, 'random_forest_model.pkl')
joblib.dump(scaler, 'scaler.pkl')

```

14] ✓ 0.4s

```

Mean Squared Error: 0.0
Root Mean Squared Error: 0.0

```

```
['scaler.pkl']
```

4. app.py

```

add.py > predict
1  from flask import Flask, request, render_template
2  import joblib
3  import numpy as np
4
5  app = Flask(__name__)
6
7  # Load the trained model and scaler
8  model = joblib.load('random_forest_model.pkl')
9  scaler = joblib.load('scaler.pkl')
10
11 @app.route('/')
12 def index():
13     return render_template('index.html')
14
15 @app.route('/predict', methods=['POST'])
16 def predict():
17     gender = int(request.form['gender'])
18     vip = int(request.form['vip'])
19     income = float(request.form['income'])
20     children = int(request.form['children'])
21     age = int(request.form['age'])
22     attractiveness = int(request.form['attractiveness'])
23
24     # Create the input array for prediction
25     input_data = np.array([[gender, vip, income, children, age, attractiveness]])
26
27     # Scale the input data
28     input_data_scaled = scaler.transform(input_data)
29
30     # Perform the prediction
31     predicted_matches = model.predict(input_data_scaled)[0]
32
33     return f'<h1>Predicted Matches: {predicted_matches}</h1>'
34
35 if __name__ == '__main__':
36     app.run(debug=True)
37

```

Index.html

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Predict Online Dating Matches</title>
7      <link rel="stylesheet" href="{ url_for('static', filename='styles.css') }" >
8  </head>
9  <body>
10     <div class="container">
11         <h1>Predict Online Dating Matches</h1>
12         <form action="/predict" method="post">
13             <label for="gender">Gender:</label>
14             <select id="gender" name="gender">
15                 <option value="0">Male</option>
16                 <option value="1">Female</option>
17             </select>
18
19             <label for="vip">Purchased VIP:</label>
20             <select id="vip" name="vip">
21                 <option value="0">No</option>
22                 <option value="1">Yes</option>
23             </select>
24
25             <label for="income">Income (USD):</label>
26             <input type="number" id="income" name="income" required>
27
28             <label for="children">Number of Children:</label>
29             <input type="number" id="children" name="children" required>
30
31             <label for="age">Age:</label>
32             <input type="number" id="age" name="age" required>
33
34             <label for="attractiveness">Attractiveness (1-10):</label>
35             <input type="number" id="attractiveness" name="attractiveness" min="1" max="10" required>
36
37             <button type="submit">Predict Matches</button>
38         </form>
39     </div>
40 </body>
41 </html>
42

```

CSS

atic > # styles.css > button

```
1  /* styles.css */
2  body {
3      font-family: Arial, sans-serif;
4      background: linear-gradient(to right, #6a11cb, #2575fc); /* Gradient background */
5      margin: 0;
6      padding: 0;
7      display: flex;
8      justify-content: center;
9      align-items: center;
10     height: 100vh;
11 }
12
13 .container {
14     background-color: rgba(255, 255, 255, 0.9); /* Semi-transparent white */
15     padding: 20px 40px;
16     border-radius: 10px;
17     box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
18     text-align: center;
19 }
20
21 h1 {
22     color: #333;
23     margin-bottom: 20px;
24 }
25
26 form {
27     display: flex;
28     flex-direction: column;
29     align-items: center;
30 }
31
32 label {
33     margin-bottom: 10px;
34     font-weight: bold;
35     color: #333;
36 }
37
38 input, select, button {
39     margin-bottom: 20px;
40     padding: 10px;
41     font-size: 16px;
42     width: 100%;
43     max-width: 300px;
44     box-sizing: border-box;
45 }
46
```

```

button {
  background-color: #5cb85c;
  color: #fff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

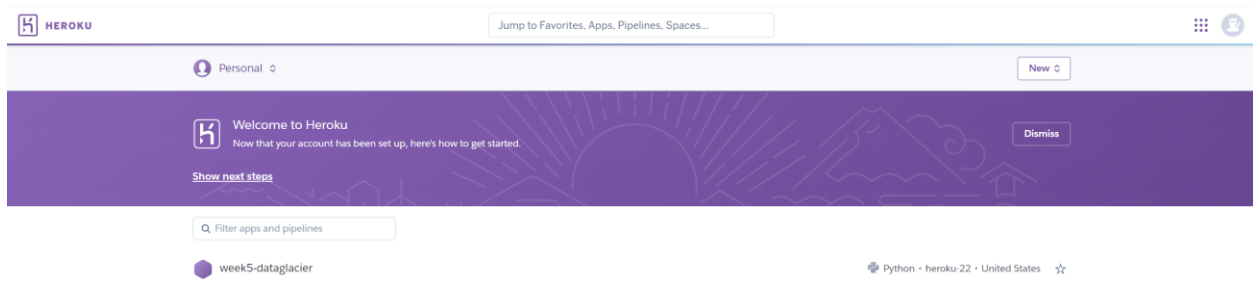
button:hover {
  background-color: #4cae4c;
}

```

5. Deployment of web app on Heroku:

Step1:-

Create an account



Step2:- Choose github repo

The screenshot shows the Heroku web interface for a repository named 'biswadip2291'. At the top, there's a header with the Heroku logo, the repository name, and a 'Public' badge. Below this, there are navigation links for 'main', '1 Branch', and '0 Tags'. A search bar labeled 'Go to file' is present, along with 'Add file' and 'Code' buttons. The main content area displays a list of files and folders:

File/Folder	Action	Time
dataset	Add files via upload	yesterday
static	Add files via upload	yesterday
templates	Add files via upload	yesterday
Deploying the model on flask.docx	Add files via upload	yesterday
Deploying the model on flask.pdf	Add files via upload	yesterday
Procfile	Update Procfile	20 hours ago
Untitled-1.ipynb	Add files via upload	yesterday
app.py	Add files via upload	yesterday
random_forest_model.pkl	Add files via upload	yesterday
requirements.txt	Update requirements.txt	1 hour ago
scaler.pkl	Add files via upload	yesterday

This is my Github Repo

Step 3:- Connect Heroku to github account

The screenshot shows the 'Deployment method' section in the Heroku dashboard. It lists three options: 'Heroku Git' (Use Heroku CLI), 'GitHub' (Connected), and 'Container Registry' (Use Heroku CLI). The 'GitHub' option is highlighted with a green checkmark. Below this, a message states 'App connected to GitHub' and 'Code diffs, manual and auto deploys are available for this app.' A box shows the connection details: 'Connected to biswadip2291/heroko by biswadip2291' with a 'Disconnect...' button. A link to 'activity feed' is also provided.

Step4:-

Deploy the application

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

Choose a branch to deploy

main

Receive code from GitHub

Build main 89dc1047

```
-----> Installing requirements with pip
-----> Discovering process types
Profile declares types -> web
-----> Compressing...
Done: 161.5M
-----> Launching...
Released v28
https://week5-dataglacier-8ffdd1be04fd.herokuapp.com/ deployed to Heroku
```

☒ Autoscroll with output

[View build log](#)

Release phase

Deploy to Heroku

Step 5:- Run the application

week5-dataglacier-8ffdd1be04fd.herokuapp.com

Predict Online Dating Matches

Gender:

Purchased VIP:

Income (USD):

Number of Children:

Age:

Attractiveness (1-10):

← → ↻ 🔍 week5-dataglacier-8ffdd1be04fd.herokuapp.com/predict

Predicted Matches: 70.0