analysis

October 10, 2023

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
import numpy as np
```

[3]:
```
files = ["asian", "black", "indian", "latino_hispanic", "maori",␣
 ↪"middle_eastern", "white", "pasifika"]

attributes = ["Gender", "Ethnicity", "Mood", "Age"]

def get_range(age_range):
    if age_range == "more than 70":
        return range(70, 120)
    bounds = str(age_range).split("-")

    if len(bounds) == 1:
        lower = (int(int(bounds[0]) / 10)) * 10
        return range(lower, lower + 10)

    return range(int(bounds[0]), int(bounds[1]) + 1)

def get_str_range(age):
    if int(age) >= 70:
        return "more than 70"
    r = get_range(age)
    return f"{r[0]}-{r[9]}"

def get_accuracies(df):
    c_results = {}
    for attr in attributes:
        gb = df.groupby(f"{attr} accurate")[f"{attr} accurate"].count()
        c_results[attr] = [(gb.index[i], gb[i]) for i in range(len(gb.index))]

    return c_results
```

[4]:
```
dfs = {}

per_category_results = {}
```

```
for f in files:
    path = f"{f}_stats.csv"
    df = pd.read_csv(path, index_col = "Unnamed: 0")
    df["True Ethnicity"] = df["True Ethnicity"].str.replace("_", " ").str.
 ↪lower()
    df["Classified Mood"] = df["Classified Mood"].str.replace("surprise",␣
 ↪"surprised")
    df["Classified Mood"] = df["Classified Mood"].str.replace("disgust",␣
 ↪"disgusted")
    if f == "pasifika" or f == "maori":
        df = df.sample(n = 100)
        df["True Gender"] = df["True Gender"].str.replace("Female", "Woman")
        df["True Gender"] = df["True Gender"].str.replace("Male", "Man")
        df["True Ethnicity"] = df["True Ethnicity"].str.replace("european",␣
 ↪"maori")
        df["True Age"] = df["True Age"].apply(get_str_range)

    df.insert(3, "Gender accurate", df["True Gender"] == df["Classified␣
 ↪Gender"])
    df.insert(7, "Ethnicity accurate", df["True Ethnicity"] == df["Classified␣
 ↪Ethnicity"])
    df.insert(11, "Mood accurate", df["True Mood"] == df["Classified Mood"])
    df.insert(15, "Age accurate", df.apply(lambda row: row["Classified Age"] in␣
 ↪get_range(row["True Age"]), axis = 1))
    accuracy = get_accuracies(df)

    per_category_results[f] = accuracy

    dfs[f] = df
```

[5]:
```
# All categories
all_df = pd.concat(dfs.values())

all_acc = get_accuracies(all_df)
```

[6]:
```
# Matrix generation

def confusion_matrix(group, attr, options, normalise = True):
    row = {}
    correct_option = list(group[f"True {attr}"])[0]
    for o in options:
        correct = correct_option == o
        coeff = (100 * (1 / len(group)) * (1 if correct else -1)) if normalise␣
 ↪else 1
        if attr == "Age":
            correct = correct_option == get_str_range(o)
```

```
            coeff *= -correct if correct and normalise else 1
            num_correct = group[["True Age", "Classified Age"]]\
                                                .apply(lambda row: (row["Classified␣
 ↪Age"] in get_range(row["True Age"])) & (get_str_range(row["Classified Age"])␣
 ↪== get_str_range(o)), axis = 1)\
                                                .sum()

            total_classified = coeff * group[["True Age", "Classified Age"]]\
                                                        .apply(lambda row:␣
 ↪int(row["Classified Age"] in get_range(o)), axis = 1)\
                                                        .sum()

            row[get_str_range(o)] = (num_correct, total_classified)

        else:
            num_correct = sum((group[f"Classified {attr}"] == group[f"True␣
 ↪{attr}"]) & (group[f"Classified {attr}"] == o))
            total_classified = coeff * sum(group[f"Classified {attr}"] == o)
            row[o] = (num_correct, total_classified)


    return pd.Series(row)

norm_matrices = {}
for attr in attributes:
    options = set(all_df[f"Classified {attr}"])
    matrix = all_df.groupby(f"True {attr}").apply(confusion_matrix, attr =␣
 ↪attr, options = options, normalise = True)
    new_order = list(filter(lambda i: i in matrix.columns, matrix.index))
    new_order = new_order + list(filter(lambda c: c not in new_order, matrix.
 ↪columns))
    matrix = matrix[list(new_order)]
    norm_matrices[attr] = matrix.applymap(lambda cell: cell[1])

c_matrices = {}
for att1 in attributes:
    c_matrices[att1] = {}
    for att2 in attributes:
        options = set(all_df[f"Classified {att2}"])
        matrix = all_df.groupby(f"True {att1}").apply(confusion_matrix, attr =␣
 ↪att2, options = options, normalise = False)
        c_matrices[att1][att2] = matrix
```

[13]:
```
# Precision/recall calculations

pr_results = {}
for attr, matrix in norm_matrices.items():
```

```
    prs = []
    for c in matrix.columns:
        tp = matrix[c].loc[[c]][0]
        fp = sum(matrix[c].abs()) - tp
        fn = matrix.loc[[c]].abs().sum(axis = 1)[0] - tp
        tn = all_df.groupby(f"True {attr}").size().sum() - all_df.
 ↪groupby(f"True {attr}").size()[c]
        precision = tp / (tp + fp)
        recall = tp / (tp + fn)
        specificity = tn / (tn + fp)

        prs.append((c, precision, recall, specificity))
    pr_results[attr] = prs
```

1 Visualisation

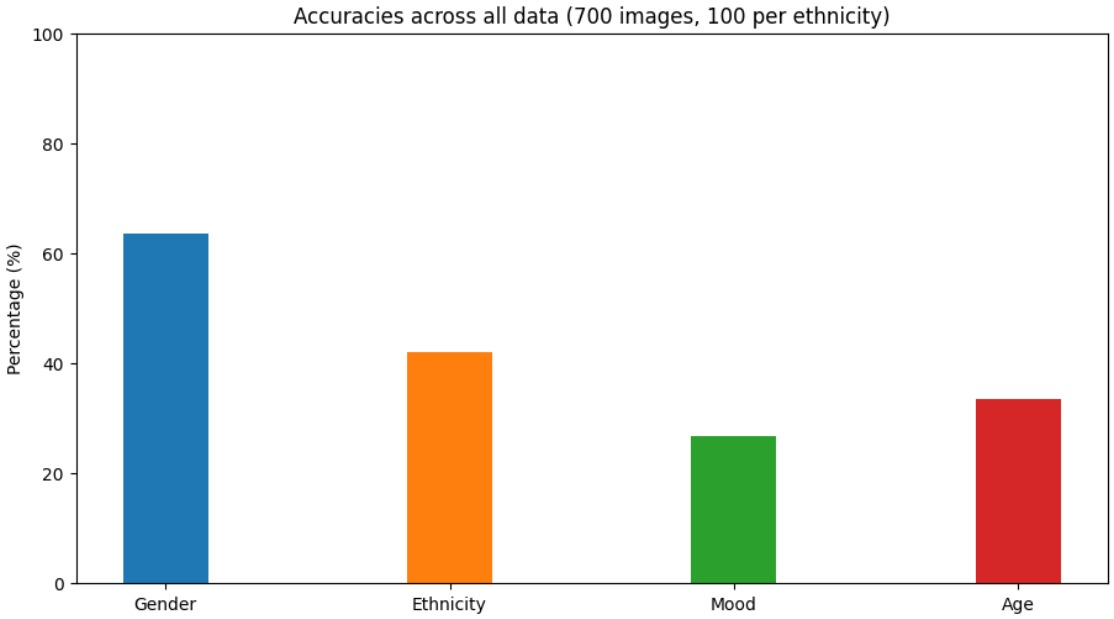
1.1 Accuracies

1.1.1 Total accuracy

[8]:
```
ethnicities = files

fig, ax = plt.subplots(layout = "constrained")
fig.set_size_inches(9, 5)
width = 0.3
x = np.arange(len(attributes))

false_heights = [all_acc[attr][0][1] / len(ethnicities) for attr in attributes]
true_heights = [100 - h for h in false_heights]
print(false_heights)

ax.bar(x, true_heights, width, color = ["tab:blue", "tab:orange", "tab:green",␣
 ↪"tab:red"])

ax.set_ylabel('Percentage (%)')
ax.set_title('Accuracies across all data (700 images, 100 per ethnicity)')
ax.set_xticks(x, attributes)
ax.set_ylim(0, 100)
```

```
[36.5, 58.0, 73.25, 66.5]
```

[8]:
```
(0.0, 100.0)
```

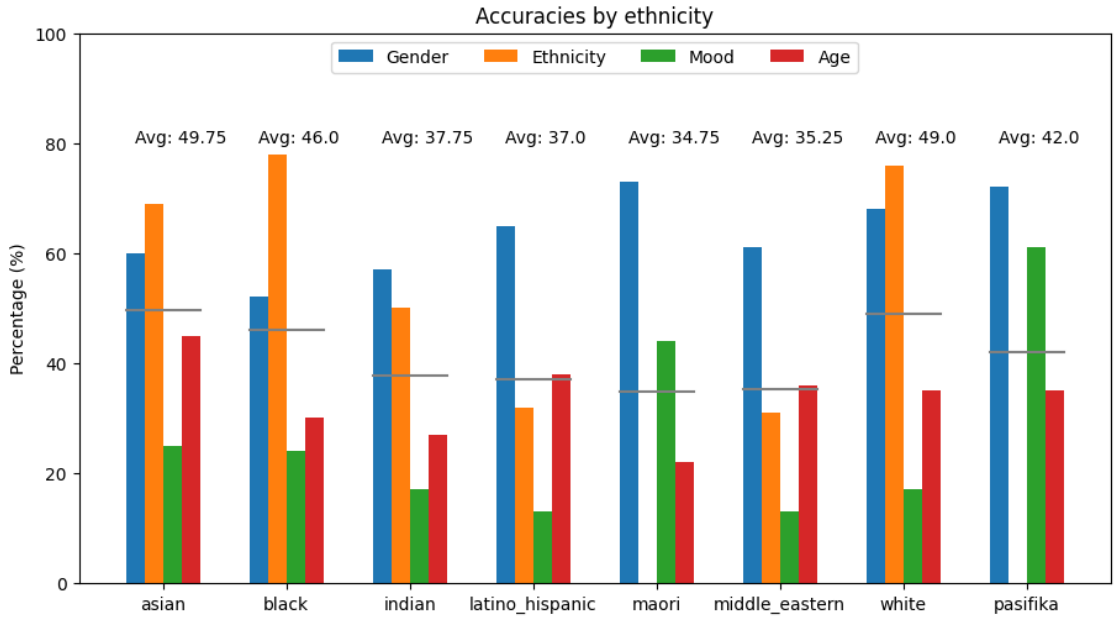


1.1.2 Accuracy by ethnicity

[9]:
```
fig, ax = plt.subplots(layout = "constrained")
fig.set_size_inches(9, 5)
multiplier = 0
width = 0.15
x = np.arange(len(ethnicities))


for attr in attributes:
    false_heights = [per_category_results[d][attr][0][1] for d in␣
 ↪per_category_results]
    true_heights = [100 - x for x in false_heights]

    offset = width * multiplier
    ax.bar(x + offset, true_heights, width, label = attr)

    multiplier += 1

for eth in range(len(ethnicities)):
    avg = sum(100 - per_category_results[ethnicities[eth]][x][0][1] for x in␣
 ↪per_category_results[ethnicities[eth]]) / len(attributes)
    ax.text(x[eth], 80, f"Avg: {avg}")
    ax.plot([x[eth] - width / 2, x[eth] + width * 3.5], [avg, avg], color =␣
 ↪"gray")
```

```
ax.set_ylabel('Percentage (%)')
ax.set_title('Accuracies by ethnicity')
ax.set_xticks(x + 1.5*width, ethnicities)
ax.legend(loc='upper center', ncols = 4)
ax.set_ylim(0, 100)
```

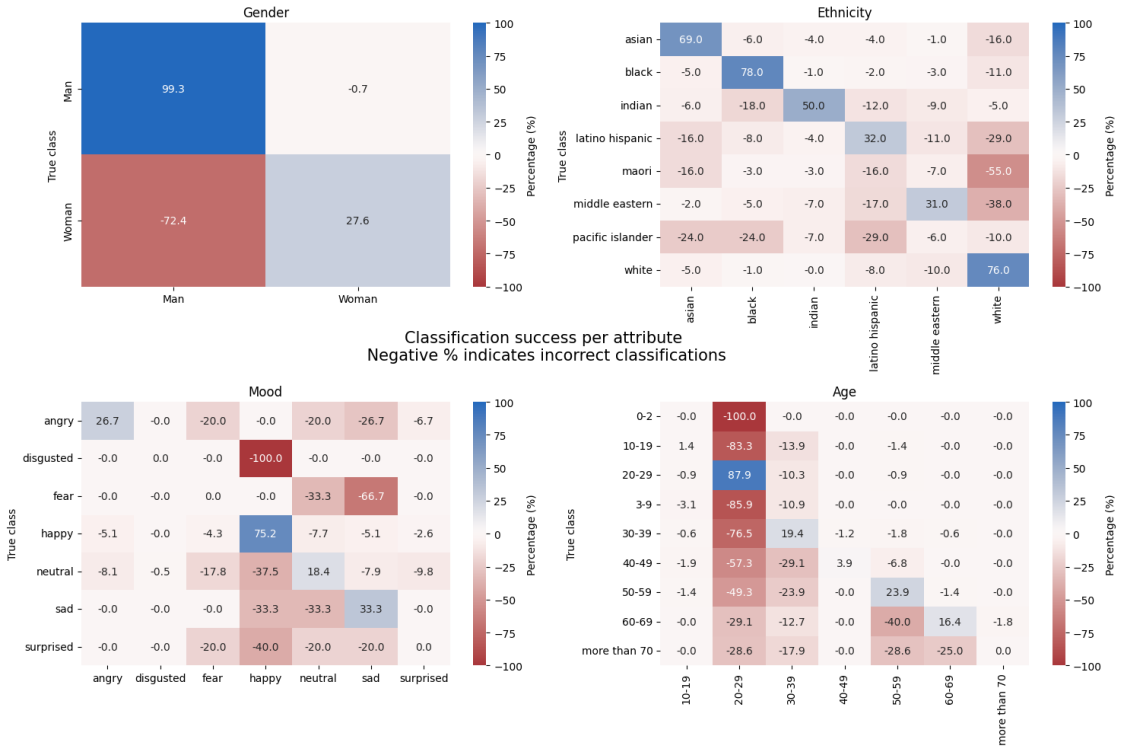
[9]: (0.0, 100.0)



1.1.3 Accuracy matrices

[10]:
```
fig, axes = plt.subplots(2, 2, figsize = (15, 10))

for i, (key, data) in enumerate(norm_matrices.items()):
    row, col = divmod(i, 2)
    sn.heatmap(data, fmt = '.1f', annot = True, cmap = "vlag_r", ax = axes[row,␣
 ↪col],
               vmin = -100, vmax = 100,
               cbar_kws = {"label": "Percentage (%)"})
    axes[row, col].set_title(key)
    axes[row, col].set_ylabel("True class")

plt.tight_layout()
plt.figtext(0.48, 0.52, "Classification success per attribute\n Negative %␣
 ↪indicates incorrect classifications", ha = "center", fontsize = 15)
```

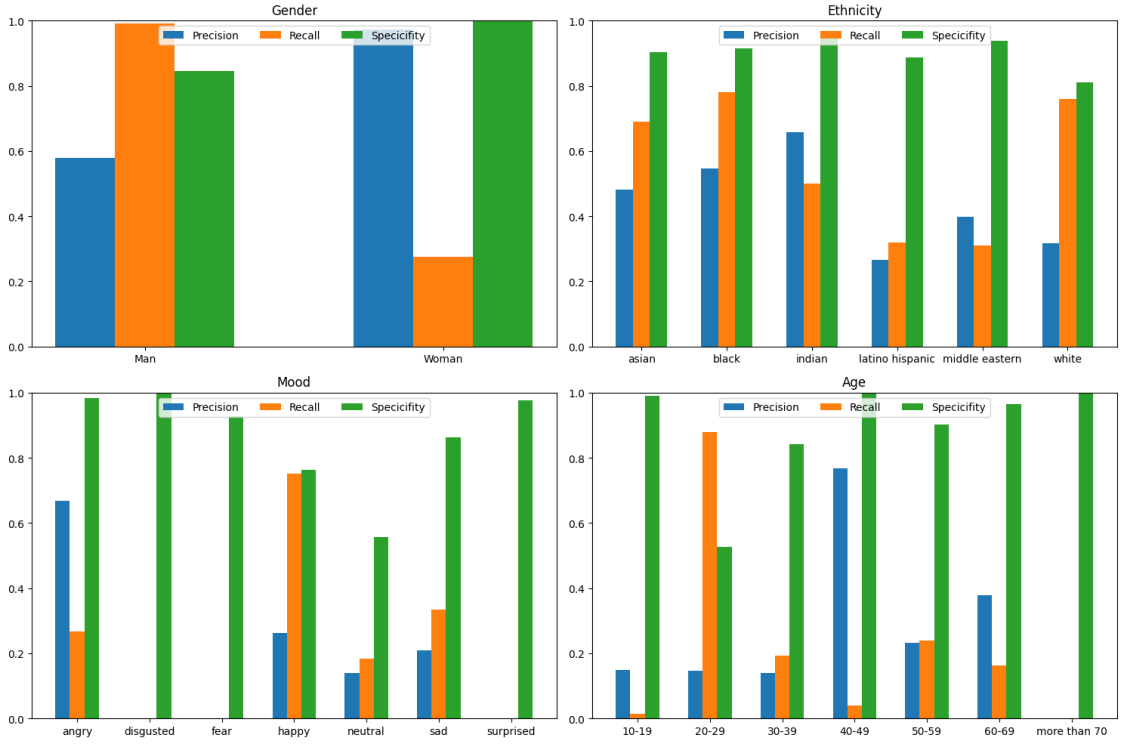
[10]: Text(0.48, 0.52, 'Classification success per attribute\n Negative % indicates incorrect classifications')



1.2 Precision, Recall, Sensitivity and Specicifity

[14]:
```
fig, axes = plt.subplots(2, 2, figsize = (15, 10))
width = 0.2
for i, (key, data) in enumerate(pr_results.items()):
    row, col = divmod(i, 2)
    multiplier = 0
    x = np.arange(len(data))

    for i in range(3):
        offset = width * multiplier
        axes[row, col].bar(x + offset, [t[i + 1] for t in data], width, label =␣
↪["Precision", "Recall", "Specicifity"][i])

        multiplier += 1

    axes[row, col].set_title(key)
    axes[row,col].set_xticks(x + width, [t[0] for t in data])
    axes[row, col].set_ylim(0, 1)
    axes[row, col].legend(loc = 'upper center', ncols = 3)
```

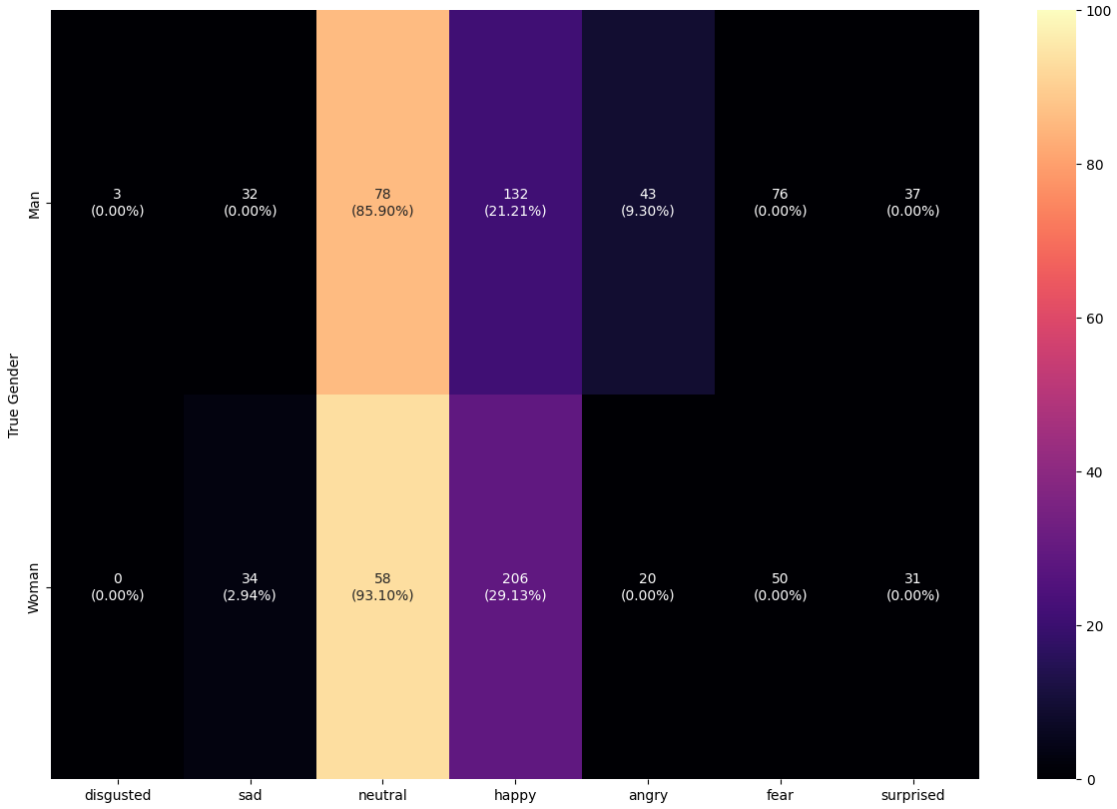
```
plt.tight_layout()
```

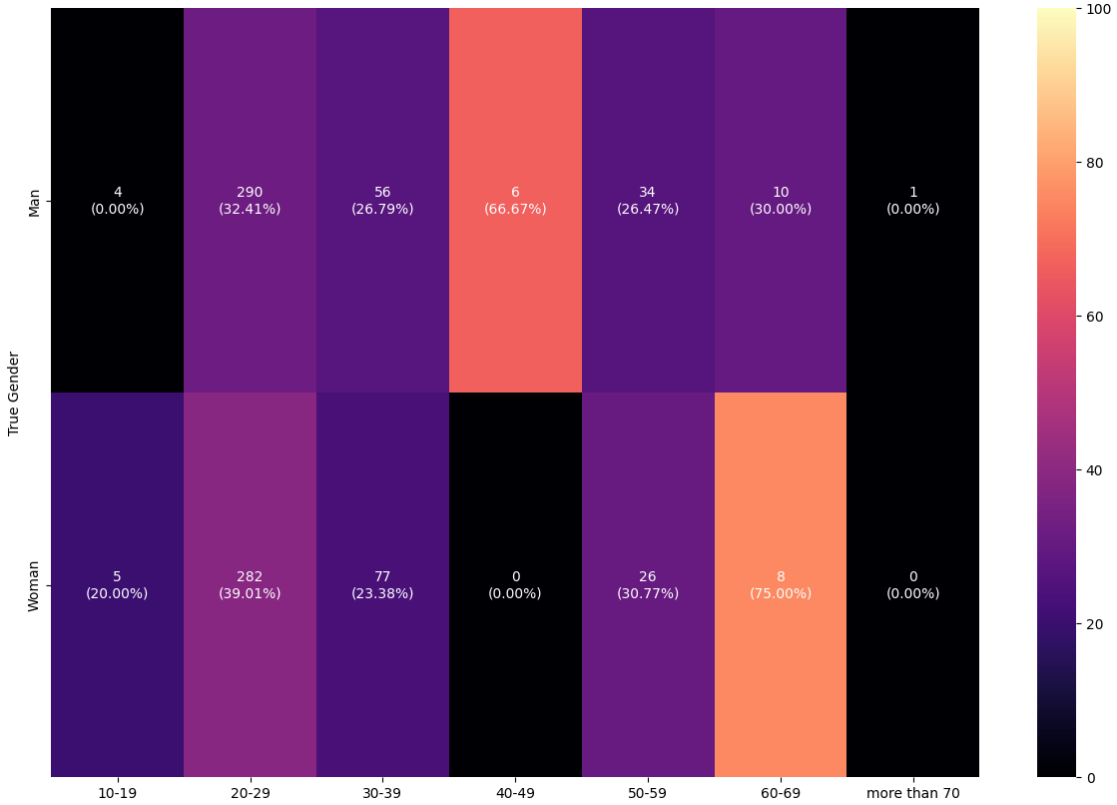


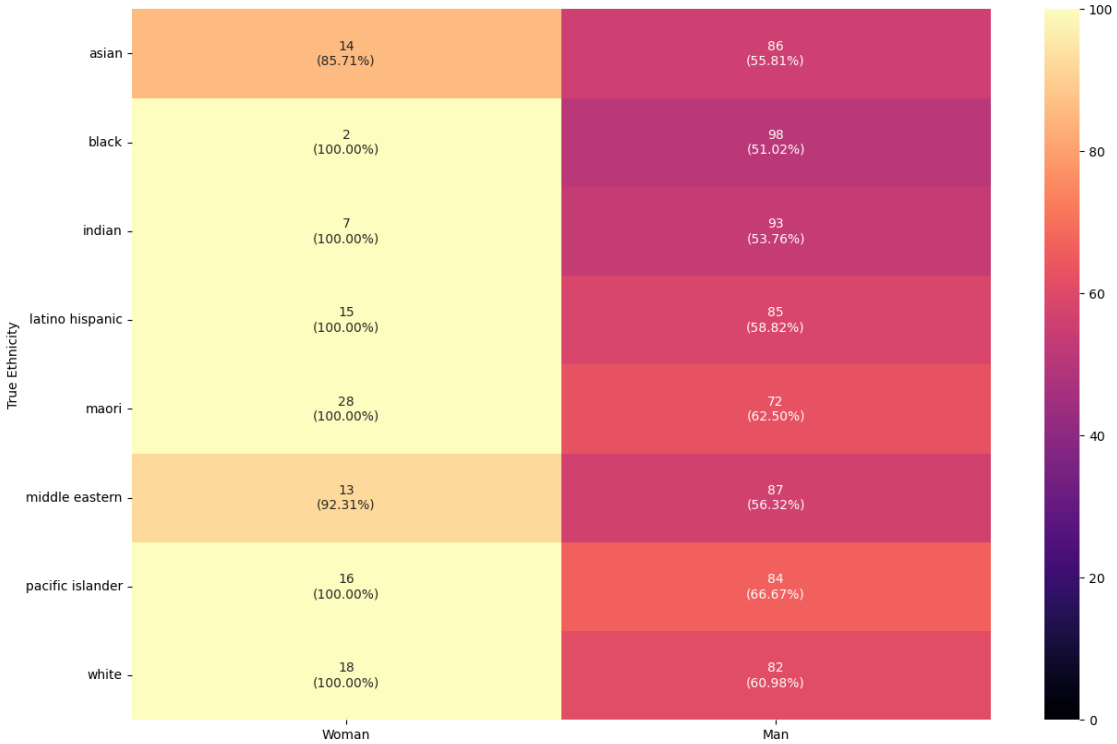
1.3 Bias heatmaps

[12]:
```
count = 0
for (att1, inner) in c_matrices.items():
    for (att2, matrix) in inner.items():
        if att1 == att2:
            continue
        figure = plt.figure(figsize = (15, 10))
        annot_matrix = matrix.applymap(lambda t: str(t[1]) + f"\n({(100 * t[0]/
 ↪t[1]) if t[1] != 0 else 0:.2f}%)")
        sn.heatmap(matrix.applymap(lambda t: (100 * t[0] / t[1]) if t[1] != 0␣
 ↪else 0),
                   vmin = 0,
                   vmax = 100,
                   cmap = "magma",
                   fmt = 's',
                   annot = annot_matrix)
        plt.show()

plt.tight_layout()
```


True Gender
Man
Woman
31
(64.52%)
53
(30.19%)
59
(38.98%)
67
(49.25%)
103
(34.95%)
88
(45.45%)
45
(66.67%)
67
(23.88%)
19
(42.11%)
76
(47.37%)
137
(29.20%)
55
(69.09%)
indian
latino hispanic
middle eastern
asian
white
black
100
80
60
40
20
0


True Gender
Man
Woman
3
(0.00%)
32
(0.00%)
78
(85.90%)
132
(21.21%)
43
(9.30%)
76
(0.00%)
37
(0.00%)
0
(0.00%)
34
(2.94%)
58
(93.10%)
206
(29.13%)
20
(0.00%)
50
(0.00%)
31
(0.00%)
disgusted
sad
neutral
happy
angry
fear
surprised
100
80
60
40
20
0

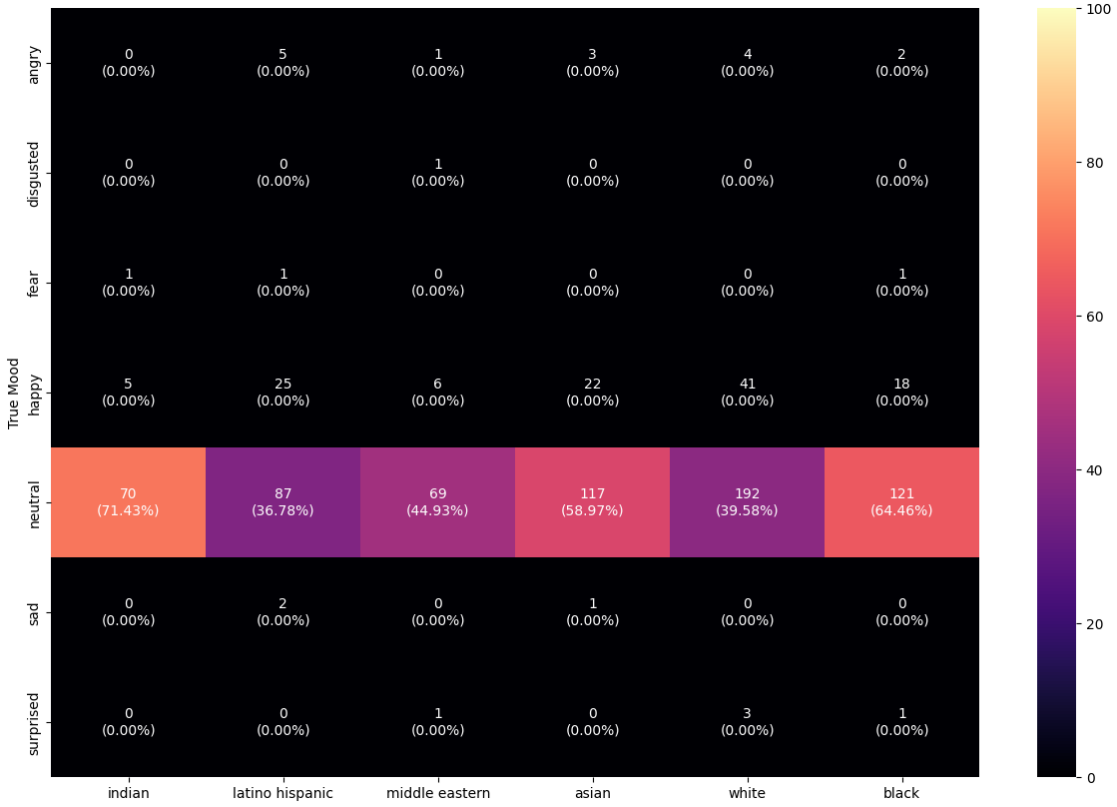

True Gender
Man
Woman
4
(0.00%)
290
(32.41%)
56
(26.79%)
6
(66.67%)
34
(26.47%)
10
(30.00%)
1
(0.00%)
5
(20.00%)
282
(39.01%)
77
(23.38%)
0
(0.00%)
26
(30.77%)
8
(75.00%)
0
(0.00%)
10-19
20-29
30-39
40-49
50-59
60-69
more than 70
100
80
60
40
20
0


True Ethnicity
asian
black
indian
latino hispanic
maori
middle eastern
pacific islander
white
14
(85.71%)
86
(55.81%)
2
(100.00%)
98
(51.02%)
7
(100.00%)
93
(53.76%)
15
(100.00%)
85
(58.82%)
28
(100.00%)
72
(62.50%)
13
(92.31%)
87
(56.32%)
16
(100.00%)
84
(66.67%)
18
(100.00%)
82
(60.98%)
Woman
Man
100
80
60
40
20
0


True Ethnicity
asian
black
indian
latino hispanic
maori
middle eastern
pacific islander
white
disgusted
sad
neutral
happy
angry
fear
surprised
0 (0.00%)
5 (0.00%)
25 (100.00%)
43 (0.00%)
7 (0.00%)
14 (0.00%)
6 (0.00%)
1 (0.00%)
8 (0.00%)
24 (100.00%)
29 (0.00%)
14 (0.00%)
13 (0.00%)
11 (0.00%)
0 (0.00%)
7 (0.00%)
17 (100.00%)
41 (0.00%)
7 (0.00%)
16 (0.00%)
12 (0.00%)
0 (0.00%)
10 (0.00%)
13 (100.00%)
42 (0.00%)
3 (0.00%)
15 (0.00%)
17 (0.00%)
0 (0.00%)
10 (0.00%)
13 (30.77%)
52 (75.00%)
7 (14.29%)
15 (0.00%)
3 (0.00%)
1 (0.00%)
7 (0.00%)
13 (100.00%)
35 (0.00%)
6 (0.00%)
26 (0.00%)
12 (0.00%)
0 (0.00%)
10 (10.00%)
14 (57.14%)
52 (94.23%)
10 (30.00%)
11 (0.00%)
3 (0.00%)
1 (0.00%)
9 (0.00%)
17 (100.00%)
44 (0.00%)
9 (0.00%)
16 (0.00%)
4 (0.00%)
100
80
60
40
20
0


True Ethnicity
asian
black
indian
latino hispanic
maori
middle eastern
pacific islander
white
10-19
20-29
30-39
40-49
50-59
60-69
more than 70
1 (0.00%)
85 (45.88%)
12 (41.67%)
1 (100.00%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
1 (100.00%)
78 (30.77%)
20 (25.00%)
0 (0.00%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
72 (27.78%)
24 (25.00%)
2 (50.00%)
2 (0.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
80 (36.25%)
15 (40.00%)
1 (100.00%)
2 (50.00%)
1 (100.00%)
0 (0.00%)
3 (0.00%)
38 (5.26%)
14 (14.29%)
1 (100.00%)
33 (36.36%)
9 (55.56%)
1 (0.00%)
0 (0.00%)
82 (37.80%)
16 (31.25%)
0 (0.00%)
0 (0.00%)
2 (0.00%)
0 (0.00%)
2 (0.00%)
54 (48.15%)
17 (11.76%)
0 (0.00%)
21 (19.05%)
6 (50.00%)
0 (0.00%)
1 (0.00%)
83 (39.76%)
15 (13.33%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
100
80
60
40
20
0

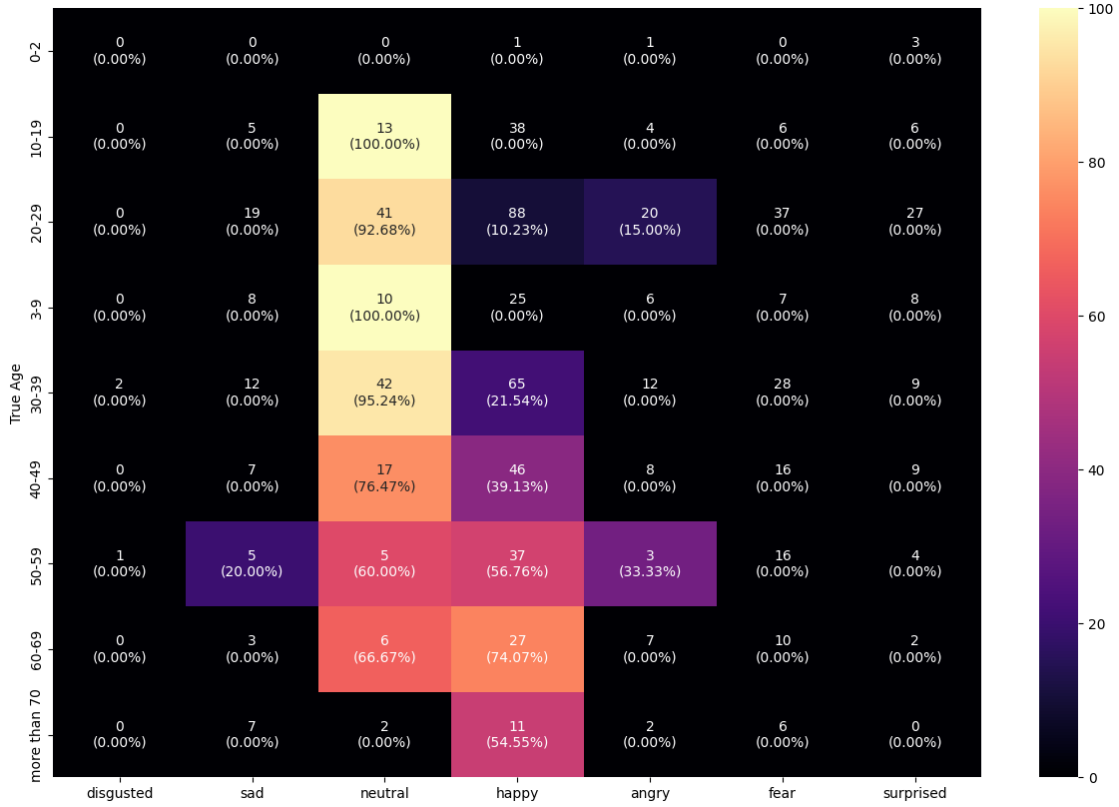

True Mood
angry
disgusted
fear
happy
neutral
sad
surprised
0
(0.00%)
15
(80.00%)
0
(0.00%)
1
(100.00%)
0
(0.00%)
3
(100.00%)
36
(100.00%)
81
(51.85%)
76
(96.05%)
580
(57.93%)
0
(0.00%)
3
(33.33%)
1
(100.00%)
4
(75.00%)
Woman
Man
100
80
60
40
20
0


True Mood
angry
disgusted
fear
happy
neutral
sad
surprised
0 (0.00%)
5 (0.00%)
1 (0.00%)
3 (0.00%)
4 (0.00%)
2 (0.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
5 (0.00%)
25 (0.00%)
6 (0.00%)
22 (0.00%)
41 (0.00%)
18 (0.00%)
70 (71.43%)
87 (36.78%)
69 (44.93%)
117 (58.97%)
192 (39.58%)
121 (64.46%)
0 (0.00%)
2 (0.00%)
0 (0.00%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
0 (0.00%)
3 (0.00%)
1 (0.00%)
indian
latino hispanic
middle eastern
asian
white
black
100
80
60
40
20
0


True Mood
angry
disgusted
fear
happy
neutral
sad
surprised
0 (0.00%)
11 (63.64%)
0 (0.00%)
0 (0.00%)
3 (33.33%)
1 (100.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
2 (100.00%)
1 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
0 (0.00%)
4 (0.00%)
55 (18.18%)
23 (13.04%)
1 (100.00%)
26 (30.77%)
7 (57.14%)
0 (0.00%)
4 (25.00%)
499 (36.67%)
109 (27.52%)
5 (60.00%)
29 (24.14%)
9 (33.33%)
1 (0.00%)
0 (0.00%)
2 (0.00%)
0 (0.00%)
0 (0.00%)
1 (100.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
2 (100.00%)
0 (0.00%)
0 (0.00%)
1 (0.00%)
1 (100.00%)
0 (0.00%)
10-19
20-29
30-39
40-49
50-59
60-69
more than 70
100
80
60
40
20
0


True Age
0-2
10-19
20-29
3-9
30-39
40-49
50-59
60-69
more than 70
0
(0.00%)
5
(80.00%)
6
(100.00%)
66
(48.48%)
36
(94.44%)
196
(53.06%)
4
(75.00%)
60
(60.00%)
27
(100.00%)
143
(66.43%)
18
(100.00%)
85
(58.82%)
9
(100.00%)
62
(53.23%)
10
(100.00%)
45
(64.44%)
3
(100.00%)
25
(60.00%)
Woman
Man
100
80
60
40
20
0


True Age
0-2
10-19
20-29
3-9
30-39
40-49
50-59
60-69
more than 70
indian
latino hispanic
middle eastern
asian
white
black
0 (0.00%)
2 (0.00%)
0 (0.00%)
0 (0.00%)
1 (100.00%)
2 (100.00%)
5 (100.00%)
11 (9.09%)
4 (25.00%)
10 (70.00%)
22 (45.45%)
20 (30.00%)
23 (65.22%)
36 (41.67%)
26 (50.00%)
41 (70.73%)
67 (41.79%)
39 (56.41%)
8 (62.50%)
14 (35.71%)
1 (0.00%)
21 (57.14%)
15 (26.67%)
5 (100.00%)
23 (78.26%)
20 (25.00%)
20 (45.00%)
29 (51.72%)
43 (46.51%)
35 (62.86%)
6 (66.67%)
10 (30.00%)
16 (37.50%)
11 (36.36%)
41 (17.07%)
19 (52.63%)
5 (40.00%)
13 (15.38%)
6 (16.67%)
13 (7.69%)
20 (15.00%)
14 (64.29%)
2 (50.00%)
12 (8.33%)
2 (0.00%)
13 (7.69%)
20 (15.00%)
6 (16.67%)
4 (0.00%)
2 (0.00%)
3 (33.33%)
5 (0.00%)
11 (0.00%)
3 (33.33%)
100
80
60
40
20
0



<Figure size 640x480 with 0 Axes>