Insights from Personal Photos

Youhee Choi, Jackie Kim, Yuntao Wang, Kexin Huang, Jeff Gonda, Kevin Feng

All from your face?

Concept: Insights from Personal Photos

Introduced by Gerry Pesavento, Sr. Director Yahoo! Inc.

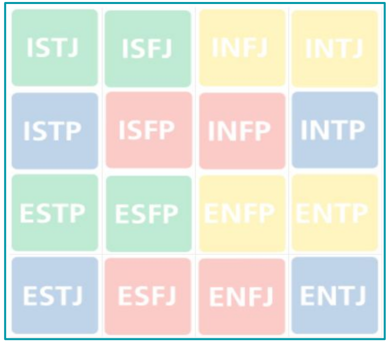
From a users photos, one can compute an accurate contextual advertising profile including hobbies, events, age, ethnicity, gender, work/home address, and current product ownership. Currently advertising profiles are done through web clicks and purchase intent; a more accurate profile is possible through photo analysis. This project can be done using photo repositories (Flickr, Facebook, etc) and Tensorflow and AI APIs (Google, Microsoft, etc).

Can we identify one's personality type from a profile picture?

Social Network Profile

Face Recognition

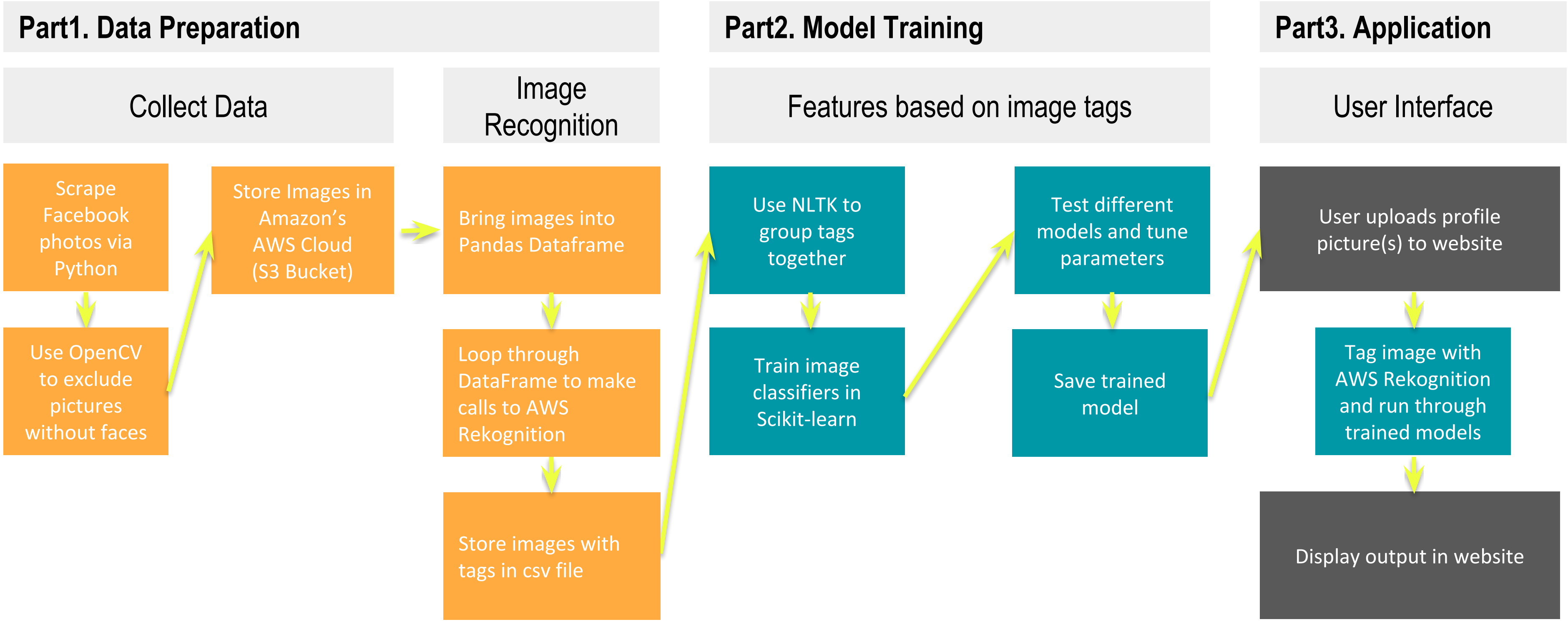
Myers-Briggs Types



Approach

	Steps	Tools
Collect Data	• Scrape 1,000 profile pictures from members of each of the 16 Myers-Briggs personality groups (ENTJ, INFP, etc.) on Facebook	- Facebook API - Python Web-scraping - Amazon AWS S3 storage
Tag Photos	• Write Python script to call Amazon's image tagging API 16,000 times to tag all photos and store them in csv files	- Amazon Rekognition API - Python Scripting - OpenCV
Train Models	• Group similar tags together using NLTK • Build, tune, and test predictive models with scikit-learn (SKL), using grouped tags as features	- NLTK Wordnet - SKL Decision Trees - SKL SVM - SKL Logistic Regression - TensorFlow & Keras
Apply Insights	• Analyze correlations between tags and personality traits • Communicate findings	- Matplotlib - Google Slides

Technical Architecture



[1] Collect the Data

```
import requests
import pandas as pd
import numpy as np
import json
import time

# Facebook Graph API access token. needed to use API
access_token='EAAZAQLburfFIBACUkZBKfu4FF26naC1clijmZBaETGZBjjVTmrNAkNEfd3sPsaUnuvF0OcZCBRzXXyWoxTS2JJR7RfB
nHxzVcwzcywUULKQFRiYI9sUJ1m0kzZATqAIHQWmM1s9ZAyOL8BZCIXZAyfed6nl85B6DRGDhd8Ac0gjxLogZDZD'
```

```
personality_types_str = ['INTP', 'ESFJ']
personality_types = {}

columns = ['name','user_id', 'profile_picture','personality_type','faces']
# users dataframe containing personality type and photos
users = pd.DataFrame(columns = columns)

# mapping to facebook group id. link to group is facebook.com/groups/[group_id]
group_id = {'INTP': '506879209400576', 'ESFJ': '2215581049'}

# map personality types in personality_type dictionary to .json data from facebook api using group id (per
 group) and access token (static)
# '/members?limit=' determines how many photos to extract
for idx, i in enumerate(personality_types_str):
    personality_types[i] = requests.get('https://graph.facebook.com/v2.5/' + group_id[i] + \
                                        '/members?limit=30&access_token=' + access_token).json()
```

```
# extract data from json files to populate users dataframe
# '/picture?height=' is adjustable to change image size
for j in personality_types_str:
    for i in personality_types[j]['data']:
        temp = pd.DataFrame({'name': i['name'], 'user_id': i['id'], 'profile_picture': 'https://graph.face
book.com/' + \
                             i['id'] + '/picture?height=900&width=900','personality_type': ['INTP']}, inde
x = None)
        users = pd.concat([users,temp], axis = 0)

users.reset_index(drop = True, inplace = True)
print('Users Dataframe sample:\n',users.head(),'\n\nNumber of users:\n', users.shape[0])
```

Level 1.

Personality type Facebook group

Level 2.

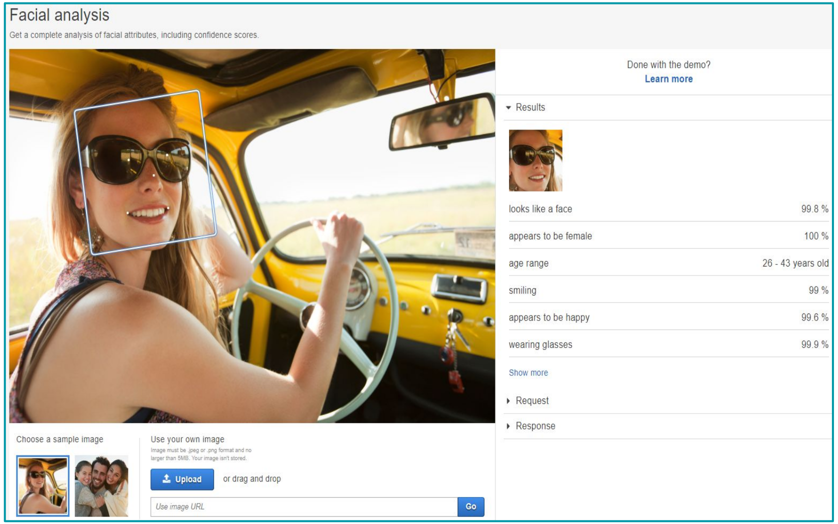
Extract profile photos for each personality types

Level 3.

Use Photo recognition API for face & personality relationship analysis

[2] Tagging Images Using Amazon Rekognition

Tagging 1 photo at a time



16,000 photos = 1,000 per type

```python
import boto3
import pandas as pd
from botocore.exceptions import ClientError

s3 = boto3.resource('s3')
s3bucket = s3.Bucket('dataxteamprojectfacebookphotos')
bucket='dataxteamprojectfacebookphotos'
client=boto3.client('rekognition')


def callRekognition(MBTItype):
    MBTI = list(s3bucket.objects.filter(Prefix='Facebook/'+str(MBTItype)))

    MBTIlist = list()

    for i in MBTI:
        MBTIlist.append(i.key)

    MBTIphotos = pd.DataFrame(MBTIlist,columns=['fileName'])

    imageNames = []
    Labels = []
    faceDetails = []

    for i in range(0,1000):
        try:
            response = client.detect_labels(Image={'S3Object':{'Bucket':bucket,'Name':MBTIphotos.fileName[i]}}\
                                            ,MinConfidence=50,MaxLabels=50)
            response1 = client.detect_faces(Image={'S3Object':{'Bucket':bucket,'Name':MBTIphotos.fileName[i]}}\
                                            ,Attributes=['ALL'])
            Labels.append(response['Labels'])
            faceDetails.append(response1['FaceDetails'])
            imageNames.append(MBTIlist[i])
            print(str(i) + " photos complete")
        except (ClientError,ValueError):
            continue

    MBTItags = pd.DataFrame(imageNames,columns=['fileName'])
    MBTItags['Labels'] = Labels
    MBTItags['FaceDetails'] = faceDetails
    MBTItags.to_csv(str(MBTItype)+'-1000.csv')
```

[3] Data Preprocessing – Feature Extraction

One-hot encoding tags

- Over 1,700 total features > preliminary models severely **overfitting**
- Need to reduce tags by grouping features

	Human	People	Person	Apparel	Clothing	Maillot	Female	Dress	Bra	Lingerie	...	Ribs	Jaguar	Toucan	Christmas Stocking	Stocking	Steak	I-E	S-N	T-F	J-P
0	1	1	1	1	1	0	0	0	0	0	...	0	0	0	0	0	0	E	N	F	J
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	E	N	F	J
2	1	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	E	N	F	J
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	E	N	F	J
4	1	1	1	0	0	0	1	0	0	0	...	0	0	0	0	0	0	E	N	F	J

5 rows × 1765 columns

[4] Data Preprocessing - Feature Grouping

```python
tag_list = list(df.columns.values)
tag_list.remove('I-E')

# use wordnet to get similarity scores between words in similarity_df

list1 = tag_list
list2 = tag_list

similarity_df = pd.DataFrame(index = list1, columns = list1)

# find word similary score between each word in tag_list

for word1 in list1:
    for word2 in list2:
        syns1 = wordnet.synsets(word1)
        syns2 = wordnet.synsets(word2)
        if len(syns1) == 0:
            d = None
        elif len(syns2) == 0:
            d = None
        else:
            d = syns1[0].wup_similarity(syns2[0])
            similarity_df.loc[word1, word2] = d
```

```python
# create df of what other tag each tag is most similar to (None is no similar words, i.e. word not in dictionary)

most_similar_df = pd.DataFrame(index = tag_list, columns = ['most_similar_word'])

for tag in list1:
    most_similar_percent = similarity_df[tag].sort_values(ascending = False)
    most_similar_word = most_similar_percent.index.values[1]

    if pd.isnull(most_similar_percent[0]):
        most_similar_df.loc[tag] = None
    else:
        most_similar_df.loc[tag] = most_similar_word
```

NLTK Wordnet

Group features (tags) by similarity score

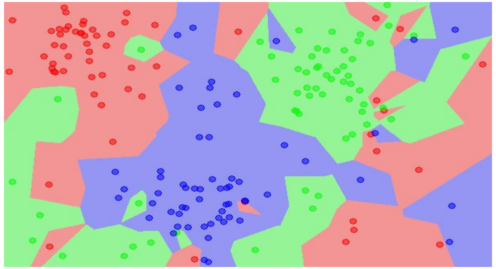
	most_similar_word
Dress	Sari
Bra	Underwear
Lingerie	Underwear
Underwear	Lingerie
Art	Mosaic

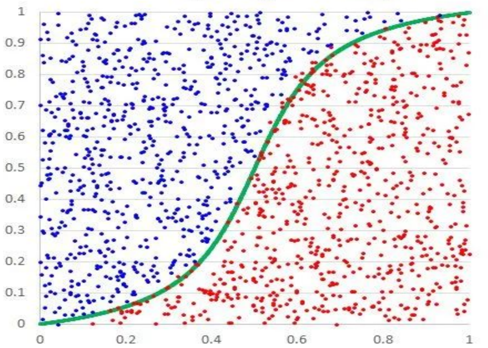
[5] Testing Models

K-Nearest Neighbors

```
model_fit(KNeighborsClassifier(n_neighbors = 10), X_train, X_test, y_train, y_test)
```

```
Train accuracy:  51.50656
Test accuracy:  49.74811
```





Logistic Regression

```
model_fit(LogisticRegression(penalty = 'l2', C = 10), X_train, X_test, y_train, y_test)
```

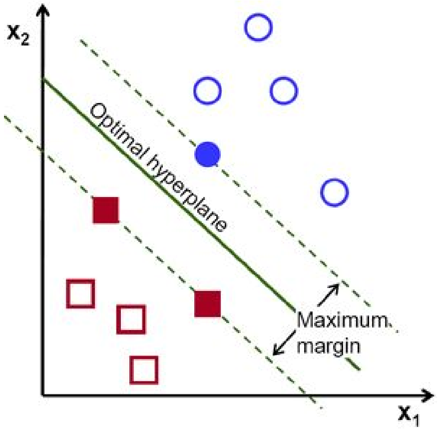
```
Train accuracy:  54.35171
Test accuracy:  55.24769
```

[5] Testing Models Cont…

SVM

```
model_fit(SVC(), X_train, X_test, y_train, y_test)
```

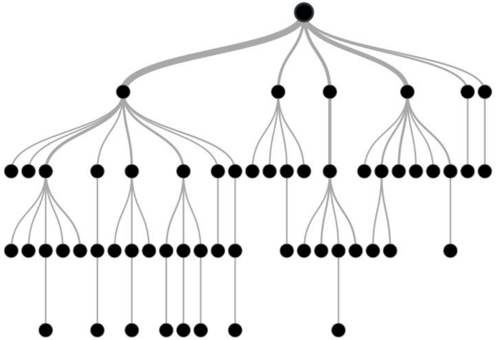
```
Train accuracy:  53.96325
Test accuracy:  55.62552
```



Decision Trees

```
model_fit(RandomForestClassifier(bootstrap = False, criterion = 'gini', max_depth = None, max_features = 1, \
                                 min_samples_leaf = 15, min_samples_split = 2, n_estimators = 300), \
          X_train, X_test, y_train, y_test)
```

```
Train accuracy:  54.30971
Test accuracy:  55.58354
```



Random Forest
Adaboost
XGBoost
ExtraTrees
...

[5] Testing Models Cont...

CNN

- 2,000 photos per personality type

```
Train on 200 samples, validate on 100 samples
Epoch 1/30
200/200 [==============================] - 1s 3ms/step - loss: 3.6301 - acc: 0.5550 - val_loss: 0.8199 - val_acc: 0.5300
Epoch 30/30
200/200 [==============================] - 0s 2ms/step - loss: 0.0479 - acc: 0.9800 - val_loss: 1.8618 - val_acc: 0.5200
Saved model to disk
Done!
```

```
Train on 23679 samples, validate on 6020 samples
Epoch 1/30
23679/23679 [==============================] - 12s 486us/step - loss: 0.7361 - acc: 0.5000 - val_loss: 0.6932 - val_a
cc: 0.4998
```

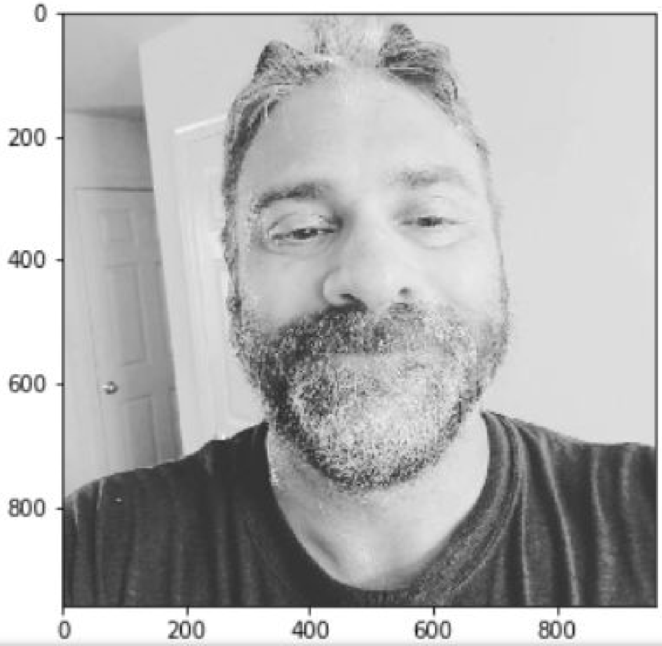
```
Epoch 30/30
23679/23679 [==============================] - 12s 487us/step - loss: 0.6551 - acc: 0.5860 - val_loss: 0.8028 - val_a
cc: 0.5083
```

[5] Testing Models Cont...

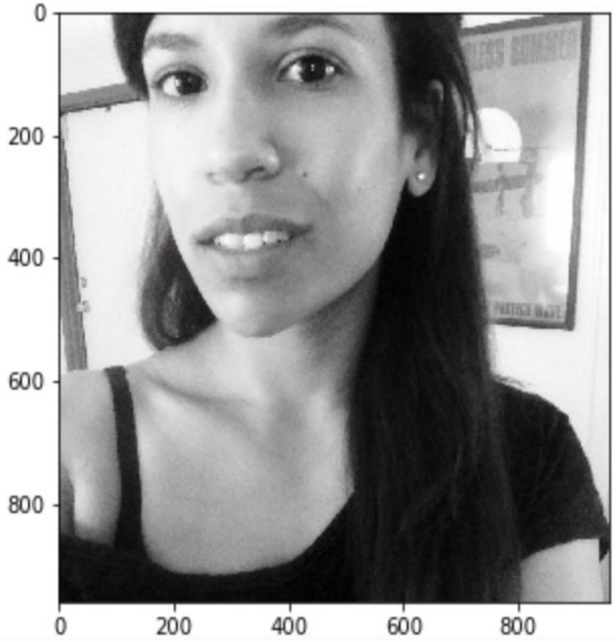
CNN

- 2,000 photos per personality type

I think this is a I with 96.4937% probability



I think this is a E type person with 73.67375% probability

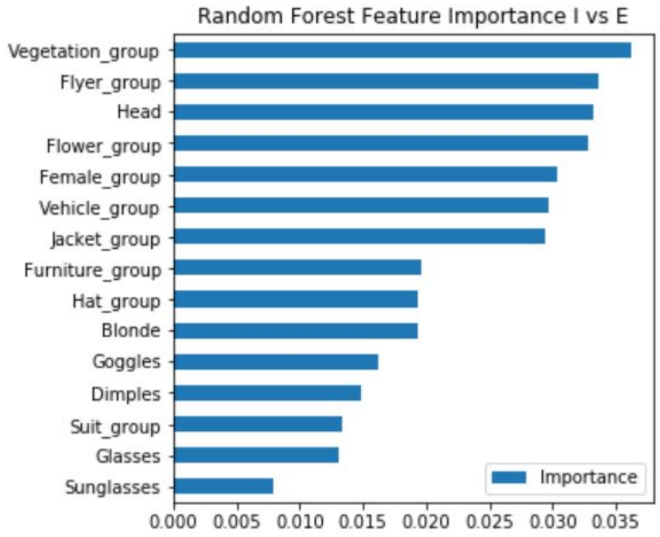


[6] Results

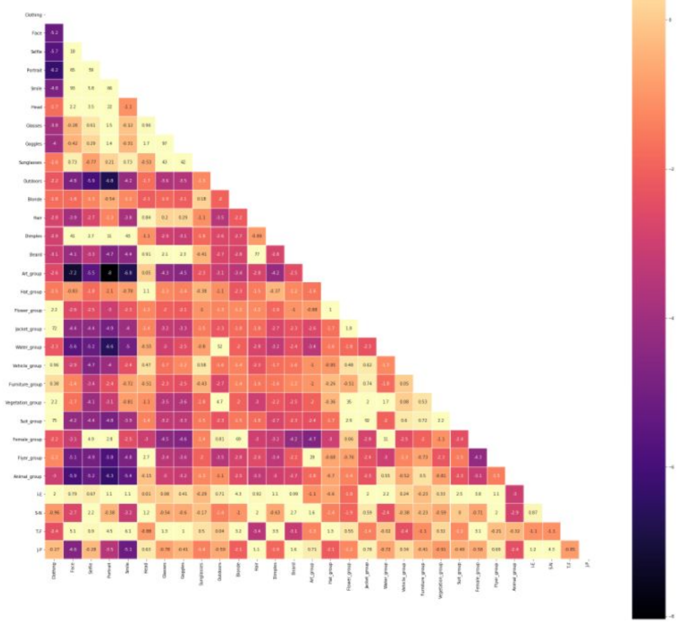
> 55% Test Accuracy

- Predicting all 4 personality type combinations (I/E, S/N, T/F, J/P) : **Hypertuned Random Forest**

Test accuracy: 55.3736356003



Tag Correlation Matrix



[6] Results

Introverted (I) vs Extroverted (E)

1. Female (E) - 3.8%
2. Animals (I) - 3%
3. Sunglasses (I) - 2.9%
4. Suit (E) - 2.5%
5. Flowers (I) - 1.8%

Sensing (S) vs Intuition (N)

1. Smile (S) - 3.2%
2. Animal (S) - 2.9%
3. Beard (N) - 2.7%
4. Selfie (N) - 2.2%
5. Art (N) - 1.6%

Thinking (T) vs Feeling (F)

1. Smile (F) - 6.1%
2. Face (F) - 5.1%
3. Beard (T) - 3.1%
4. Female (F) - 3.1%
5. Vehicle (T) - 2.4%

Judging (J) vs Perceiving (P)

1. Smile (J) - 5.1%
2. Animal (J) - 2.4%
3. Hat (J) - 2.1%
4. Beard (P) - 1.6%
5. Sunglasses (J) - 1.4%

Learning Path

Photos

How can we filter out noisy data from social media APIs?

Database

How do we best store and transfer a large amount of image data?

Analysis

How can we counteract matrix sparsity for the best possible analysis?

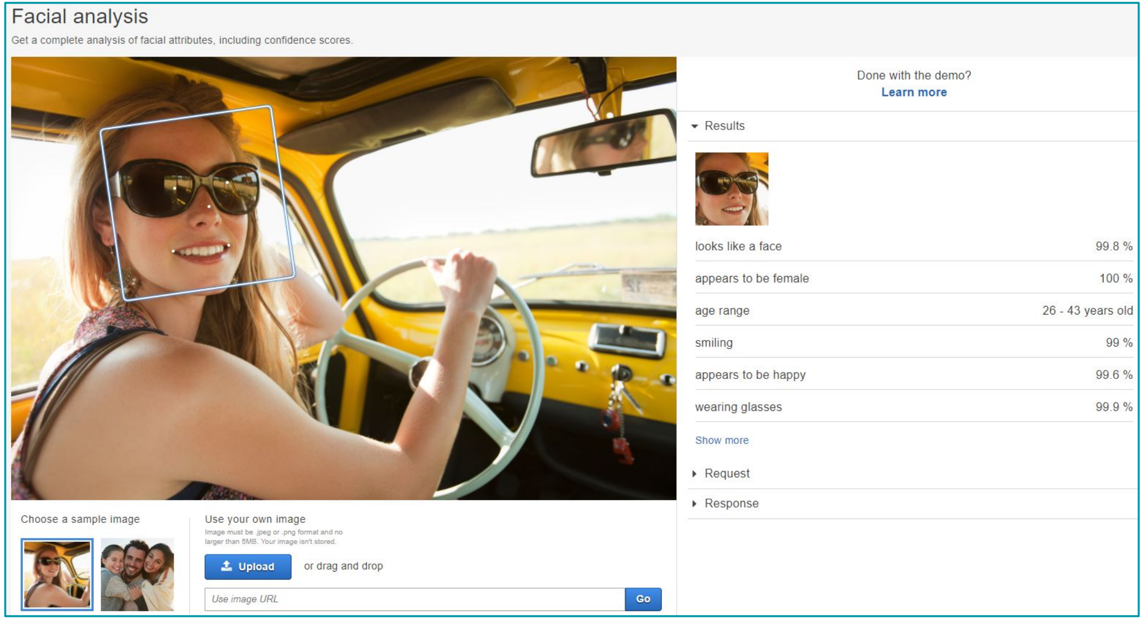
Application

How can our analysis be valuable?

Business opportunities
Marketing
User research
...

Intended User Interface

Existing image recognition service + ***Add personality tags for output***



Potential Uses

Targeted Marketing

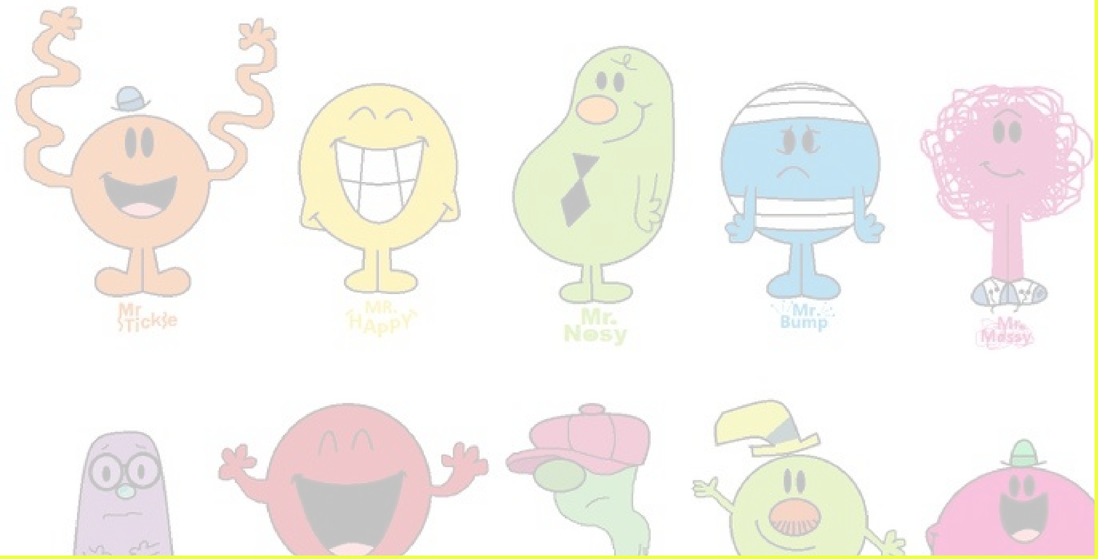
Marketers can create **personalized campaigns** based on a more concrete membership database

User Research

UI/UX and Product Designers can understand **user intent** and justify **user behavior**

Business Strategy

Project managers can more accurately **predict trends of target demographics**



Thank You

Special thanks to our mentors:

- Gerry Pesavento, Sr. Director Yahoo!
- Peter Cnudde, VP Yahoo!

https://github.com/jeff-gonda/data-x-team-project