Model Selection

Bayesian Optimization for model selection to maximize accuracy.

In [1]:

```
from src.classifier.ThreadedBayesianSearcher impo
from src.classifier.BayesianTabularModelSearch im
from src.classifier.CustomTabularModel import Cus
from src.recommender.BayesianRecommender import Ba
from bayes_opt import BayesianOptimization
from bayes_opt import UtilityFunction
import numpy as np

import matplotlib.pyplot as plt
from matplotlib import gridspec
%matplotlib inline
```

In [2]:

```
bayesian_optimizer = BayesianSearcher(20)
model = CustomTabularModel(0.5, False, 1000, {'la
bayesian_optimizer.run_optimization(model, {'laye
bayesian_optimizer.run_optimization(model, {'laye
bayesian_optimizer.run_optimization(model, {'laye
```

```
Epoch [illegible]: early stopping
```

```
25.00% [5/20
00:01<00:04]
```

epoch	train_loss	valid_loss	accuracy	time
0	0.765960	0.698588	0.179669	00:00
1	0.733799	0.691646	0.820331	00:00
2	0.715128	0.680893	0.820331	00:00
3	0.697245	0.664149	0.820331	00:00
4	0.672947	0.642763	0.820331	00:00

```
100.00% [14/14
00:00<00:00]
```

```
Epoch 5: early stopping
```

In [3]:
```
def maximization_function(**params: dict):
    """
    The function whose value we want to maximize.

    :param params: The parameters to set to the mo
    :return: The value of the metric used by the r
             Expected to be the validation acc.
    """
    params['early_stopping'] = False
    model.reset_params(params)
    return model.train(epochs=20, k=1)
```

In [4]:
```
x = [{'layer1': i} for i in range(2, 400, 100)]
y = [maximization_function(**layer_size) for layer
```

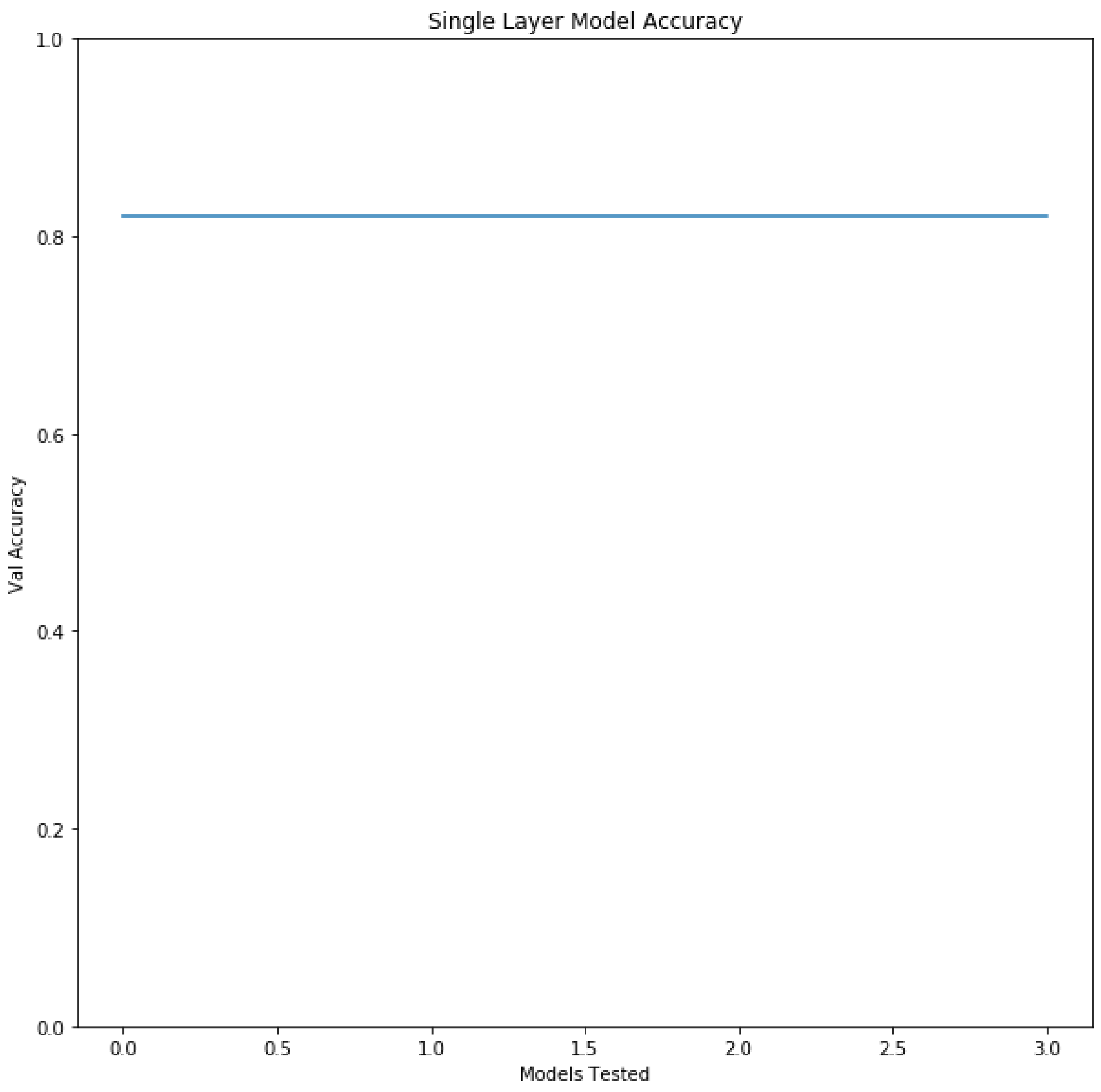
```
50.00% [10/20
00:02<00:02]
```

epoch	train_loss	valid_loss	accuracy	time
0	0.719603	0.751872	0.179669	00:00
1	0.745910	0.747806	0.179669	00:00
2	0.730650	0.738172	0.184397	00:00
3	0.718957	0.723776	0.200946	00:00
4	0.704679	0.705083	0.248227	00:00
5	0.696500	0.682889	0.486998	00:00
6	0.692060	0.656228	0.817967	00:00
7	0.677321	0.627614	0.817967	00:00
8	0.656529	0.597951	0.817967	00:00
9	0.639671	0.568748	0.820331	00:00

```
100.00% [14/14
00:00<00:00]
```

In [5]:
```
x_i = list(range(0, len(x)))

plt.figure(figsize=(10,10))
plt.title("Single Layer Model Accuracy")
plt.ylim((0, 1))
plt.xlabel("Models Tested")
plt.ylabel("Val Accuracy")
plt.plot(x_i, y)
```

```
[<matplotlib.lines.Line2D at 0x11b9804e0>]
```



In [6]:
```
ef posterior(optimizer, x_obs, y_obs, grid):
    optimizer._gp.fit(x_obs, y_obs)

    mu, sigma = optimizer._gp.predict(grid, return_s
    return mu, sigma



ef plot_gp(optimizer, x, y):
    fig = plt.figure(figsize=(10, 10))
    steps = len(optimizer.space)
    fig.suptitle(
        'Gaussian Process and Utility Function After
        fontdict={'size': 30}
    )

    gs = gridspec.GridSpec(2, 1, height_ratios=[3, 1
    axis = plt.subplot(gs[0])
    acq = plt.subplot(gs[1])

    x_obs = np.array([[res["params"]["layer1"]] for
    y_obs = np.array([res["target"] for res in optim

    mu, sigma = posterior(optimizer, x_obs, y_obs, x
    axis.plot(x, y, linewidth=3, label='Target')
    axis.plot(x_obs.flatten(), y_obs, 'D', markersiz
    axis.plot(x, mu, '--', color='k', label='Predict

    axis.fill(np.concatenate([x, x[::-1]]),
              np.concatenate([mu - 1.9600 * sigma, (
              alpha=.6, fc='c', ec='None', label='95
    axis.set_xlim((-2, 10))
    axis.set_ylim((None, None))
    axis.set_ylabel('f( layer1 )', fontdict={'size':
    axis.set_xlabel('layer1', fontdict={'size': 20})

    utility_function = UtilityFunction(kind="ucb", k
    utility = utility_function.utility(x, optimizer.
    acq.plot(x, utility, label='Utility Function', c
    acq.plot(x[np.argmax(utility)], np.max(utility),
             label=u'Next Best Guess', markerfacecol
    acq.set_xlim((-2, 10))
    acq.set_ylim((0, np.max(utility) + 0.5))
    acq.set_ylabel('Utility', fontdict={'size': 20})
    acq.set_xlabel('layer1', fontdict={'size': 20})

    axis.legend(loc=2, bbox_to_anchor=(1.01, 1), bor
    acq.legend(loc=2, bbox_to_anchor=(1.01, 1), bord
In [7]: esian_optimizer.optimizer.maximize(init_points=0, 
    t_gp(bayesian_optimizer.optimizer, np.array(x_i).re
```

```
00:01<00:03]
```

epoch	train_loss	valid_loss	accuracy	time
0	0.652627	0.679543	0.791962	00:00
1	0.659102	0.675948	0.791962	00:00
2	0.675877	0.666772	0.813239	00:00
3	0.684686	0.649839	0.820331	00:00
4	0.676672	0.630394	0.820331	00:00
5	0.668888	0.608128	0.820331	00:00

```
100.00% [14/14
00:00<00:00]

Epoch 6: early stopping

35.00% [7/20
00:01<00:02]
```

Recommendation

Via Baysian Optimization, find the params that need to change to reduce the likelihood someone experience depression.

In [18]:

```
from src.recommender.JSONParamReader import JSONPa
import pprint
import warnings
from pandas.io.json import json
import pandas as pd
from bayes_opt import BayesianOptimization, Events
from fastai.basic_data import DataBunch, Tensor
import numpy as np
from src.classifier.CustomTabularModel import Cust
from src.data.DataCsvInterface import DataCsvInte
from src.recommender.JSONParamReader import JSONPa

warnings.simplefilter(action='ignore', category=Fu

bayesian_optimizer = BayesianRecommender(40, 3, 3
# Init the model with the best params
model = CustomTabularModel(0.5, False, 1000, {'lay
best_params = JSONParamReader('classifier/logs').
model.reset_params(best_params)
pp = pprint.PrettyPrinter(indent=4)
print("        Using Model Architecture")
pp.pprint( best_params)
```

```
        Using Model Architecture
{'dropout': 0.4170220047, 'layer1': 288.409472883
4}
```

In [9]:

```
data = model.input_data.train_ds[40][0]
data_parsed = []
for element in data.data:
    if type(element) is Tensor:
        data_parsed += list(element.numpy())
    else:
        data_parsed += element



data_init = {key: data_parsed[i] for i, key in enu
cr = bayesian_optimizer.get_ranges(data.names, moc
column_range = {key: cr[key] for key in cr if key
model.train(90)
```

```
5.56% [5/90
00:01<00:23]
```

epoch	train_loss	valid_loss	accuracy	time
0	0.724703	0.694174	0.417021	00:00
1	0.701253	0.675544	0.846808	00:00
2	0.687708	0.654574	0.846808	00:00
3	0.677194	0.629963	0.846808	00:00
4	0.661005	0.595601	0.846808	00:00

```
100.00% [8/8
00:00<00:00]


Epoch 5: early stopping


  0.8468084931373596
```

In [10]:

```
bayesian_optimizer.run_optimization(model, data_i
```

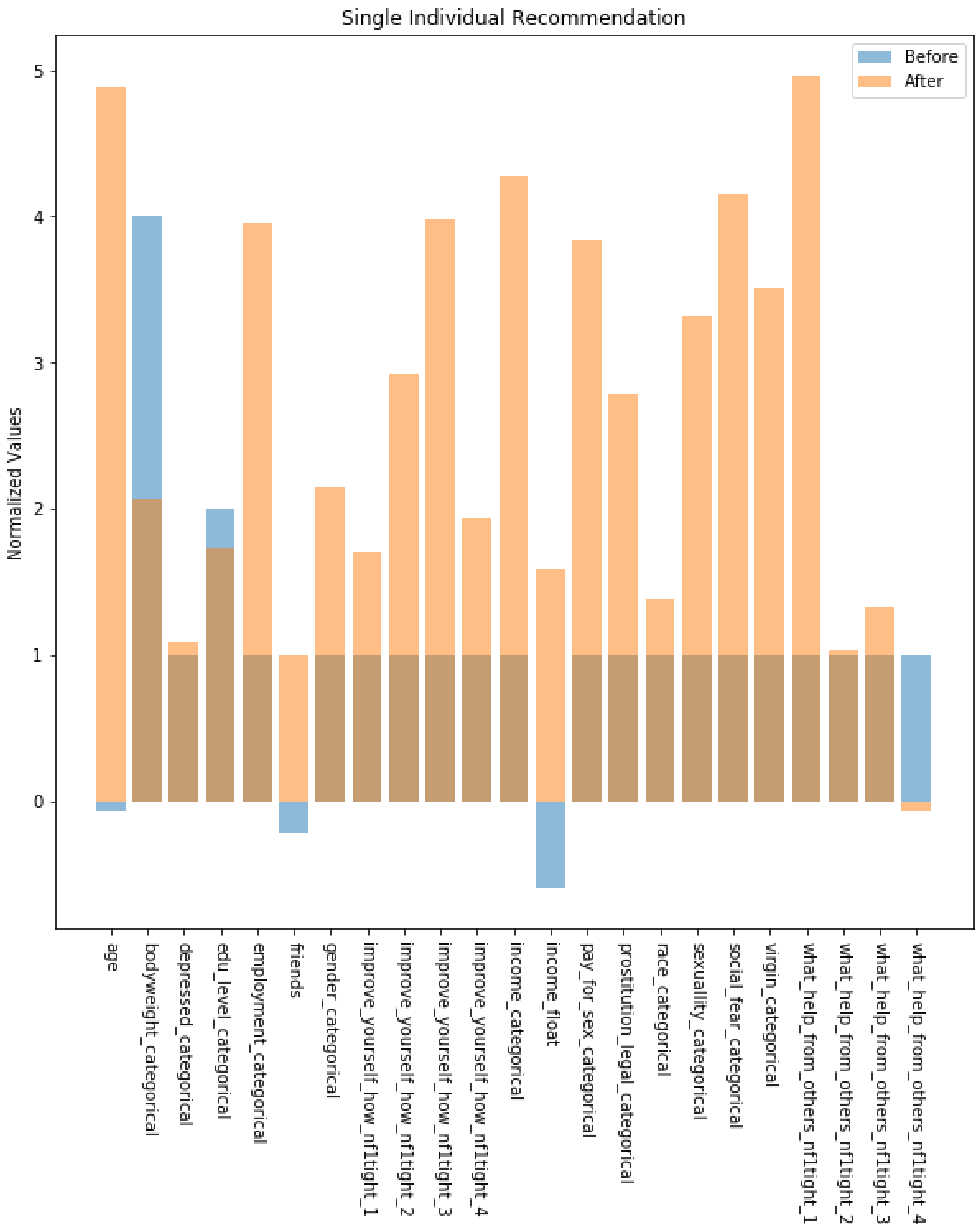
```
Value to maximize: 0.5386492013931274
Value to maximize: 0.5354037284851074
Value to maximize: 0.5352365970611572
Value to maximize: 0.5357813835144043
Value to maximize: 0.5376389026641846
Value to maximize: 0.5368748903274536
Value to maximize: 0.5376715064048767
Keeping results: {'target': 0.5376715064048767, 'p
arams': {'bodyweight_categorical': 4.8783209416137
39, 'depressed_categorical': 2.072849677402941, 'e
du_level_categorical': 1.0860365316185034, 'employ
ment_categorical': 1.7265832253091842, 'friends':
3.962085077884159, 'gender_categorical': 1.0, 'imp
rove_yourself_how_nf1tight_1': 2.1449551022643814,
'improve_yourself_how_nf1tight_2': 1.7099886999834
206, 'improve_yourself_how_nf1tight_3': 2.92042465
95239005, 'improve_yourself_how_nf1tight_4': 3.978
675136750419, 'income_categorical': 1.937975231937
7288, 'income_float': 4.273223183728743, 'pay_for_
sex_categorical': 1.580509044918251, 'prostitution
_legal_categorical': 3.8389922468172273, 'race_cat
egorical': 2.787271231476358, 'sexuallity_categori
cal': 1.3794389192775334, 'social_fear_categorica
l': 3.3186926830768413, 'virgin_categorical': 4.14
8099264094315, 'what_help_from_others_nf1tight_1':
3.5108531513462173, 'what_help_from_others_nf1tigh
t_2': 4.966456126592732, 'what_help_from_others_nf
1tight_3': 1.028714301411156, 'what_help_from_othe
rs_nf1tight_4': 1.3198236320579462}}
```

In [11]:

```
after_data = bayesian_optimizer.results[0]['params
data_init = {key: data_parsed[i] for i, key in enu

# Sort both dictionaries by keys
data_init = {i:data_init[i] for i in sorted(data_i
after_data = {i:after_data[i] for i in sorted([_ f
              if i != 1}

# Exclude changing your sexuality lol
# del data_init['sexuallity_categorical']
# del after_data['sexuallity_categorical']
# del data_init['pay_for_sex_categorical']
# del after_data['pay_for_sex_categorical']
# del data_init['virgin_categorical']
# del after_data['virgin_categorical']



# Before Data
feature_data = [data_init[_] for _ in data_init]
feature_pos = list(range(len(feature_data)))
objects = [_ for _ in data_init]
# Fill in any missing data for running predictions
for key in data_init:
    if key not in after_data:
        print("    "+ str(key))
        after_data[key] = data_init[key]

after_feature_data = [after_data[_] for _ in after
after_feature_pos = list(range(len(after_feature_d
after_objects = [_ for _ in after_data]
```

```
font = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 22}



plt.figure(figsize=(10, 10))
b1 = plt.bar(feature_pos, feature_data, align='cen
b2 = plt.bar(after_feature_pos, after_feature_data
plt.xticks(feature_pos, objects, rotation=-90)
plt.ylabel('Normalized Values')
plt.title('Single Individual Recommendation')
plt.legend((b1[0], b2[0]), ('Before', 'After'))
plt.show()
```

```
age
```


Single Individual Recommendation
Before
After
Normalized Values
5
4
3
2
1
0
age
bodyweight_categorical
depressed_categorical
edu_level_categorical
employment_categorical
friends
gender_categorical
improve_yourself_how_nf1tight_1
improve_yourself_how_nf1tight_2
improve_yourself_how_nf1tight_3
improve_yourself_how_nf1tight_4
income_categorical
income_float
pay_for_sex_categorical
prostitution_legal_categorical
race_categorical
sexuallity_categorical
social_fear_categorical
virgin_categorical
what_help_from_others_nf1tight_1
what_help_from_others_nf1tight_2
what_help_from_others_nf1tight_3
what_help_from_others_nf1tight_4

In [12]:

```
data_init
```

```
{'age': -0.0751691,
 'bodyweight_categorical': 4,
 'depressed_categorical': 1,
 'edu_level_categorical': 2,
 'employment_categorical': 1,
 'friends': -0.21313006,
 'gender_categorical': 1,
 'improve_yourself_how_nf1tight_1': 1,
 'improve_yourself_how_nf1tight_2': 1,
 'improve_yourself_how_nf1tight_3': 1,
 'improve_yourself_how_nf1tight_4': 1,
 'income_categorical': 1,
 'income_float': -0.59635216,
 'pay_for_sex_categorical': 1,
 'prostitution_legal_categorical': 1,
 'race_categorical': 1,
 'sexuallity_categorical': 1,
 'social_fear_categorical': 1,
 'virgin_categorical': 1,
 'what_help_from_others_nf1tight_1': 1,
 'what_help_from_others_nf1tight_2': 1,
 'what_help_from_others_nf1tight_3': 1,
 'what_help_from_others_nf1tight_4': 1}
```

In [13]:

```
changes = np.abs(np.subtract(feature_data, after_
directions = np.sign(np.subtract(feature_data, aft
```

In [14]:

```
n = 3
max_categories = np.argsort(changes)[-n:][::-1]
```

In [15]:
```
model = CustomTabularModel(0.5, False, 1000, {'la
```

In [16]:
```
model.predict(after_data)
```

```
(Category 1, tensor(0), tensor([0.5356, 0.464
4]))
```

In [17]:
```
after_data
```

```
{'bodyweight_categorical': 4.878320941613739,
 'depressed_categorical': 2.072849677402941,
 'edu_level_categorical': 1.0860365316185034,
 'employment_categorical': 1.7265832253091842,
 'friends': 3.962085077884159,
 'gender_categorical': 1.0,
 'improve_yourself_how_nf1tight_1': 2.1449551022
643814,
 'improve_yourself_how_nf1tight_2': 1.7099886999
834206,
 'improve_yourself_how_nf1tight_3': 2.9204246595
239005,
 'improve_yourself_how_nf1tight_4': 3.9786751367
50419,
 'income_categorical': 1.9379752319377288,
 'income_float': 4.273223183728743,
 'pay_for_sex_categorical': 1.580509044918251,
 'prostitution_legal_categorical': 3.83899224681
72273,
 'race_categorical': 2.787271231476358,
 'sexuallity_categorical': 1.3794389192775334,
 'social_fear_categorical': 3.318692683076841З,
 'virgin_categorical': 4.148099264094315,
 'what_help_from_others_nf1tight_1': 3.510853151
3462173,
 'what_help_from_others_nf1tight_2': 4.966456126
592732,
 'what_help_from_others_nf1tight_3': 1.028714301
411156,
 'what_help_from_others_nf1tight_4': 1.319823632
0579462,
 'age': -0.0751691}
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```