Xinyuan Chen HW3 (main)

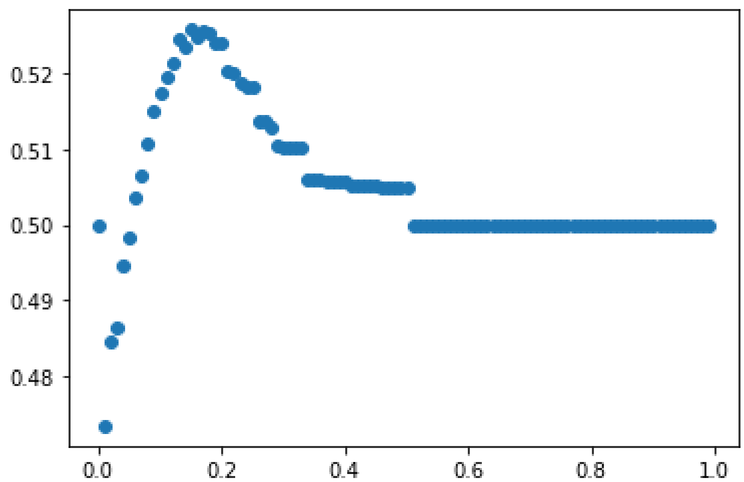
Approach

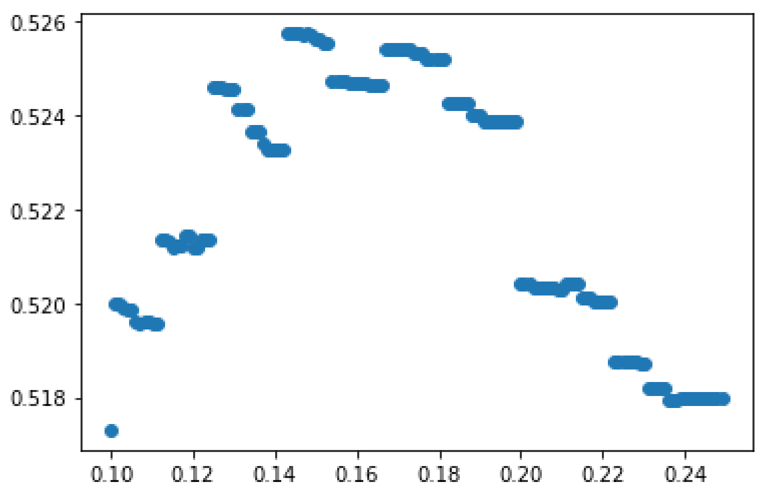
Find a weight for Jaccard similarity (see baseline.ipynb) and recipe popularity percentile (see baselines.ipy), and calculate the weighed average.

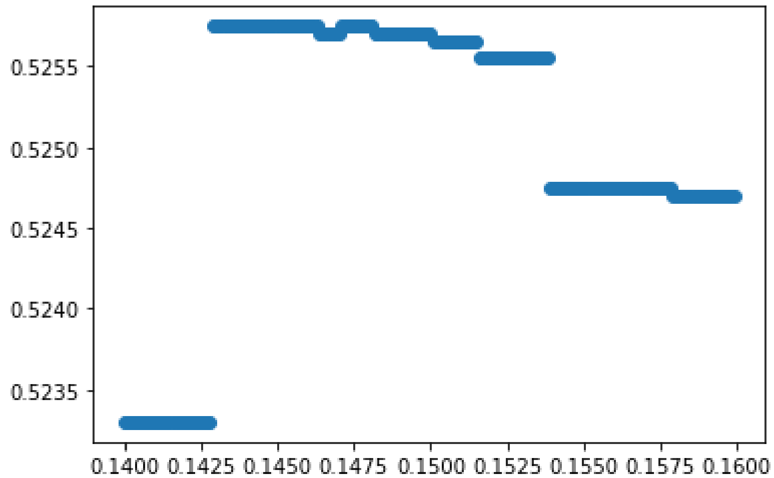
If the weighed average >= a value we set (here we use 0.5), the user_id and recipe_id combination is predicted to be actually have happened.

In [ ]:
```
# make ipynb files importable
from jupyter_utils_tddschn import notebook_importer

# import the baseline.ipynb file
import baseline
```

```
importing Jupyter notebook from baseline.ipynb
```







In [ ]:
```
from baseline import *
```

In [ ]:
```
d.shape
X_train.shape
recipes_train = X_train[:, :-1]
recipes_train = recipes_train.copy()
recipes_train.resize((X_train.shape[0],))
recipes_train.shape
```

Out[ ]:
```
(490000,)
```

In [ ]:
```
from collections import Counter, defaultdict
import numpy as np
recipe_counts = Counter(recipes_train)
```

In [ ]:
```
recipe_counts.most_common(3)
```

Out[ ]:
```
[(32445558, 4671), (95482435, 3418), (54496210, 2847)]
```

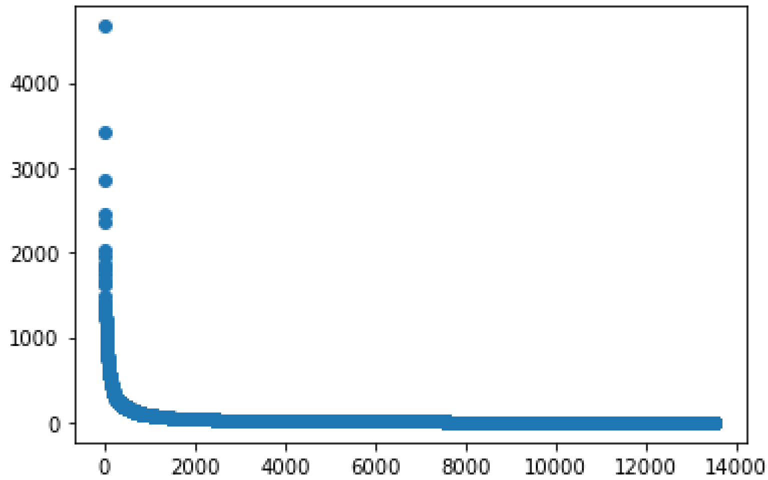
In [ ]:
```
most_common_recipes = recipe_counts.most_common()
```

In [ ]:
```
most_common_recipes_id_normalized = [(i + 1, most_common_recipes[i][1]) for i i
```

In [ ]:
```
len(most_common_recipes_id_normalized)
most_common_recipes_id_normalized[:3], most_common_recipes_id_normalized[-3:]
```

Out[ ]:
```
([(1, 4671), (2, 3418), (3, 2847)], [(13514, 1), (13515, 1), (13516, 1)])
```

In [ ]:
```
plt.scatter(*zip(*most_common_recipes_id_normalized, strict=True))
plt.show()
```



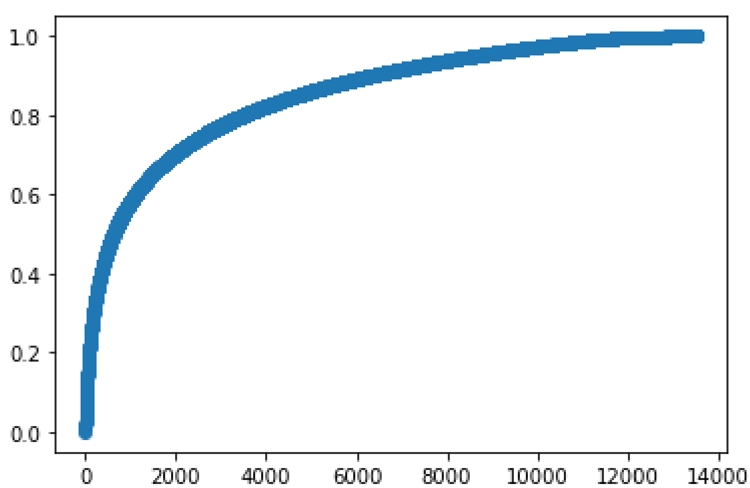
In [ ]:
```
recipe_quantiles: dict[int, float] = {}
cum_count = 0
for recipe, count in most_common_recipes:
        recipe_quantiles[recipe] = cum_count / len(recipes_train)
        cum_count += count
```

In [ ]:
```
import itertools
dict(itertools.islice(recipe_quantiles.items(), 5))
```

Out[ ]:
```
{32445558: 0.0,
 95482435: 0.00953265306122449,
 54496210: 0.016508163265306124,
 43615275: 0.022318367346938775,
 59129763: 0.027351020408163264}
```

In [ ]:
```
recipe_quantiles_values = list(recipe_quantiles.values())
recipe_quantiles_id_normalized: list[int, float] = [(i + 1, recipe_quantiles_va
```

In [ ]:
```
plt.scatter(*zip(*recipe_quantiles_id_normalized, strict=True))
plt.show()
```



In [ ]:
```
from collections import defaultdict
jaccard_max_sims_dict: dict[tuple[int, int], float] = defaultdict(float)
for u, r in X_test_and_random:
        jaccard_max_sims_dict[(u, r)] = jaccard_sim_max(u, r)
```

In [ ]:
```
def get_prediction_score(weight1: float, weight2: float, user_id: int, recipe_i
        # sim_max = jaccard_sim_max(user_id, recipe_id)
        sim_max = jaccard_max_sims_dict[(user_id, recipe_id)]
        return weight1 * sim_max + weight2 * (1 - recipe_quantiles.get(recipe_i

def get_prediction(weight1: float, weight2: float, user_id: int, recipe_id: int
        return get_prediction_score(weight1, weight2, user_id, recipe_id) >= 0.
```

In [ ]:
```
# get the number of elements in np.arange(start, stop, step)
def numpy_arange_element_count(start: float, stop: float, step: float) -> int:
        mod = (stop - start) / step
        return int(mod) + int(mod != int(mod))
```

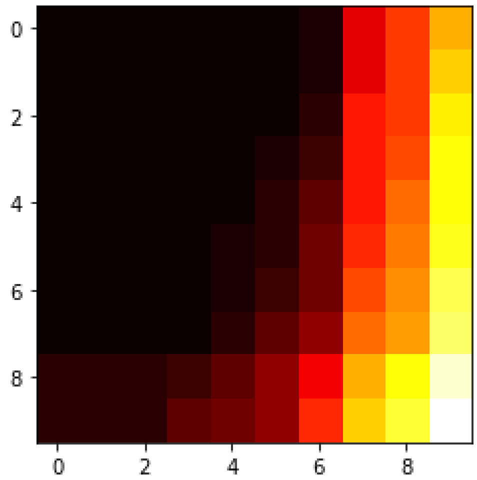
In [ ]:
```
import seaborn as sns

def get_accuracies(start, stop, step, start2, stop2, step2) -> list[tuple[float
        accuracies: list[tuple[float, float, float]] = []
        d1, d2 = numpy_arange_element_count(start, stop, step), numpy_arange_el
        for weight1 in np.arange(start, stop, step):
                for weight2 in np.arange(start2, stop2, step2):
                        predictions_binary = [get_prediction(weight1, weight2,
                        accuracy = sum(predictions_binary) / len(predictions_bi
                        accuracies.append((weight1, weight2, accuracy))
        arr = np.zeros((d1, d2))
        return accuracies

def plot_heatmap(accuracies: list[tuple[float, float, float]]) -> None:
        x, y, z = zip(*accuracies)
        heatmap, _, _ = np.histogram2d(x, y, weights=z)
        plt.clf()
        plt.imshow(heatmap, cmap='hot')
        plt.show()
```

In [ ]:
```
accu = get_accuracies(0, 1, 0.1, 0, 1, 0.1)
```

In [ ]:
```
accu[:3], accu[8:12]
accu[:5], accu[-5:]
```

Out[ ]:
```
([(0.0, 0.0, 0.0),
  (0.0, 0.1, 0.0),
  (0.0, 0.2, 0.0),
  (0.0, 0.30000000000000004, 0.0),
  (0.0, 0.4, 0.0)],
 [(0.9, 0.5, 0.0004),
  (0.9, 0.6000000000000001, 0.00085),
  (0.9, 0.7000000000000001, 0.00135),
  (0.9, 0.8, 0.0016),
  (0.9, 0.9, 0.002)])
```

In [ ]:
```
plot_heatmap(accu)
```

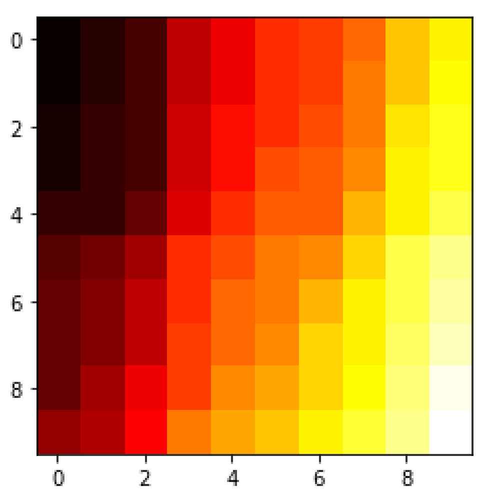


In [ ]:
```
accu2 = get_accuracies(0.5, 1, 0.05, 0.5, 1, 0.05)
```

In [ ]:
```
accu2[:5], accu2[-5:]
```

Out[ ]:
```
([(0.5, 0.5, 0.0001),
  (0.5, 0.55, 0.0002),
  (0.5, 0.6000000000000001, 0.0003),
  (0.5, 0.6500000000000001, 0.0007),
  (0.5, 0.7000000000000002, 0.00085)],
 [(0.9500000000000004, 0.7500000000000002, 0.00155),
  (0.9500000000000004, 0.8000000000000003, 0.0017),
  (0.9500000000000004, 0.8500000000000003, 0.00185),
  (0.9500000000000004, 0.9000000000000004, 0.00205),
  (0.9500000000000004, 0.9500000000000004, 0.0023)])
```

In [ ]:
```
plot_heatmap(accu2)
```



In [ ]:
```
accu3 = get_accuracies(1, 11, 0.5, 1, 11, 0.5)
```

In [ ]:
```
# accu3[:5], accu3[-5:]
accu3
```

Out[ ]:
```
[(1.0, 1.0, 0.0177),
 (1.0, 1.5, 0.01915),
 (1.0, 2.0, 0.02115),
 (1.0, 2.5, 0.0223),
 (1.0, 3.0, 0.0236),
 (1.0, 3.5, 0.0243),
 (1.0, 4.0, 0.02475),
 (1.0, 4.5, 0.02535),
 (1.0, 5.0, 0.02595),
 (1.0, 5.5, 0.02635),
 (1.0, 6.0, 0.0265),
 (1.0, 6.5, 0.0267),
 (1.0, 7.0, 0.02685),
 (1.0, 7.5, 0.02685),
 (1.0, 8.0, 0.02705),
 (1.0, 8.5, 0.02725),
 (1.0, 9.0, 0.0274),
 (1.0, 9.5, 0.02745),
 (1.0, 10.0, 0.0276),
 (1.0, 10.5, 0.02775),
 (1.5, 1.0, 0.0462),
 (1.5, 1.5, 0.04765),
 (1.5, 2.0, 0.04995),
 (1.5, 2.5, 0.05085),
 (1.5, 3.0, 0.0519),
 (1.5, 3.5, 0.0525),
 (1.5, 4.0, 0.05315),
 (1.5, 4.5, 0.0537),
 (1.5, 5.0, 0.05425),
 (1.5, 5.5, 0.0545),
 (1.5, 6.0, 0.0547),
 (1.5, 6.5, 0.0548),
 (1.5, 7.0, 0.05495),
 (1.5, 7.5, 0.05505),
 (1.5, 8.0, 0.0552),
 (1.5, 8.5, 0.05535),
 (1.5, 9.0, 0.05555),
 (1.5, 9.5, 0.0557),
 (1.5, 10.0, 0.0559),
 (1.5, 10.5, 0.056),
 (2.0, 1.0, 0.08485),
 (2.0, 1.5, 0.08635),
 (2.0, 2.0, 0.0884),
 (2.0, 2.5, 0.0893),
 (2.0, 3.0, 0.0903),
 (2.0, 3.5, 0.09075),
 (2.0, 4.0, 0.09125),
 (2.0, 4.5, 0.0919),
 (2.0, 5.0, 0.09235),
 (2.0, 5.5, 0.0927),
 (2.0, 6.0, 0.0928),
 (2.0, 6.5, 0.09285),
 (2.0, 7.0, 0.093),
 (2.0, 7.5, 0.09315),
 (2.0, 8.0, 0.09335),
 (2.0, 8.5, 0.0936),
 (2.0, 9.0, 0.09375),
 (2.0, 9.5, 0.0939),
 (2.0, 10.0, 0.094),
 (2.0, 10.5, 0.0942),
```

```
(2.5, 1.0, 0.1252),
(2.5, 1.5, 0.12695),
(2.5, 2.0, 0.1289),
(2.5, 2.5, 0.12965),
(2.5, 3.0, 0.13045),
(2.5, 3.5, 0.13105),
(2.5, 4.0, 0.13175),
(2.5, 4.5, 0.1322),
(2.5, 5.0, 0.1325),
(2.5, 5.5, 0.13285),
(2.5, 6.0, 0.1329),
(2.5, 6.5, 0.13295),
(2.5, 7.0, 0.13305),
(2.5, 7.5, 0.1333),
(2.5, 8.0, 0.1335),
(2.5, 8.5, 0.1336),
(2.5, 9.0, 0.13375),
(2.5, 9.5, 0.134),
(2.5, 10.0, 0.1342),
(2.5, 10.5, 0.1343),
(3.0, 1.0, 0.15795),
(3.0, 1.5, 0.15955),
(3.0, 2.0, 0.1614),
(3.0, 2.5, 0.16195),
(3.0, 3.0, 0.1629),
(3.0, 3.5, 0.1635),
(3.0, 4.0, 0.1641),
(3.0, 4.5, 0.16445),
(3.0, 5.0, 0.1649),
(3.0, 5.5, 0.1652),
(3.0, 6.0, 0.1653),
(3.0, 6.5, 0.16545),
(3.0, 7.0, 0.16565),
(3.0, 7.5, 0.16565),
(3.0, 8.0, 0.16575),
(3.0, 8.5, 0.16605),
(3.0, 9.0, 0.16625),
(3.0, 9.5, 0.1663),
(3.0, 10.0, 0.1665),
(3.0, 10.5, 0.1666),
(3.5, 1.0, 0.18975),
(3.5, 1.5, 0.19135),
(3.5, 2.0, 0.193),
(3.5, 2.5, 0.19355),
(3.5, 3.0, 0.1943),
(3.5, 3.5, 0.1951),
(3.5, 4.0, 0.19545),
(3.5, 4.5, 0.1958),
(3.5, 5.0, 0.1962),
(3.5, 5.5, 0.1965),
(3.5, 6.0, 0.1966),
(3.5, 6.5, 0.19675),
(3.5, 7.0, 0.19695),
(3.5, 7.5, 0.1971),
(3.5, 8.0, 0.1972),
(3.5, 8.5, 0.1973),
(3.5, 9.0, 0.19745),
(3.5, 9.5, 0.19775),
(3.5, 10.0, 0.198),
(3.5, 10.5, 0.1981),
```

```
(4.0, 1.0, 0.21735),
(4.0, 1.5, 0.2185),
(4.0, 2.0, 0.2201),
(4.0, 2.5, 0.2208),
(4.0, 3.0, 0.2216),
(4.0, 3.5, 0.22225),
(4.0, 4.0, 0.2225),
(4.0, 4.5, 0.22305),
(4.0, 5.0, 0.2234),
(4.0, 5.5, 0.2236),
(4.0, 6.0, 0.2237),
(4.0, 6.5, 0.22395),
(4.0, 7.0, 0.2241),
(4.0, 7.5, 0.22415),
(4.0, 8.0, 0.2244),
(4.0, 8.5, 0.2245),
(4.0, 9.0, 0.2248),
(4.0, 9.5, 0.2249),
(4.0, 10.0, 0.22505),
(4.0, 10.5, 0.22525),
(4.5, 1.0, 0.24125),
(4.5, 1.5, 0.24275),
(4.5, 2.0, 0.2441),
(4.5, 2.5, 0.24465),
(4.5, 3.0, 0.24555),
(4.5, 3.5, 0.24615),
(4.5, 4.0, 0.2465),
(4.5, 4.5, 0.2469),
(4.5, 5.0, 0.2473),
(4.5, 5.5, 0.2476),
(4.5, 6.0, 0.2477),
(4.5, 6.5, 0.24795),
(4.5, 7.0, 0.248),
(4.5, 7.5, 0.24815),
(4.5, 8.0, 0.24845),
(4.5, 8.5, 0.24865),
(4.5, 9.0, 0.2488),
(4.5, 9.5, 0.24905),
(4.5, 10.0, 0.24915),
(4.5, 10.5, 0.2493),
(5.0, 1.0, 0.26275),
(5.0, 1.5, 0.2641),
(5.0, 2.0, 0.2653),
(5.0, 2.5, 0.26595),
(5.0, 3.0, 0.2671),
(5.0, 3.5, 0.2675),
(5.0, 4.0, 0.26785),
(5.0, 4.5, 0.2683),
(5.0, 5.0, 0.26875),
(5.0, 5.5, 0.2691),
(5.0, 6.0, 0.26915),
(5.0, 6.5, 0.2694),
(5.0, 7.0, 0.2695),
(5.0, 7.5, 0.2696),
(5.0, 8.0, 0.2698),
(5.0, 8.5, 0.26995),
(5.0, 9.0, 0.27015),
(5.0, 9.5, 0.27025),
(5.0, 10.0, 0.2704),
(5.0, 10.5, 0.27045),
```

```
(5.5, 1.0, 0.28195),
(5.5, 1.5, 0.28335),
(5.5, 2.0, 0.2845),
(5.5, 2.5, 0.28525),
(5.5, 3.0, 0.2861),
(5.5, 3.5, 0.2864),
(5.5, 4.0, 0.28685),
(5.5, 4.5, 0.28715),
(5.5, 5.0, 0.28755),
(5.5, 5.5, 0.28785),
(5.5, 6.0, 0.28795),
(5.5, 6.5, 0.288),
(5.5, 7.0, 0.28815),
(5.5, 7.5, 0.2882),
(5.5, 8.0, 0.28845),
(5.5, 8.5, 0.28865),
(5.5, 9.0, 0.2888),
(5.5, 9.5, 0.289),
(5.5, 10.0, 0.28915),
(5.5, 10.5, 0.28925),
(6.0, 1.0, 0.30105),
(6.0, 1.5, 0.3023),
(6.0, 2.0, 0.30335),
(6.0, 2.5, 0.30415),
(6.0, 3.0, 0.3049),
(6.0, 3.5, 0.30535),
(6.0, 4.0, 0.3056),
(6.0, 4.5, 0.3061),
(6.0, 5.0, 0.30645),
(6.0, 5.5, 0.3067),
(6.0, 6.0, 0.3068),
(6.0, 6.5, 0.30705),
(6.0, 7.0, 0.30715),
(6.0, 7.5, 0.30725),
(6.0, 8.0, 0.3074),
(6.0, 8.5, 0.3075),
(6.0, 9.0, 0.30765),
(6.0, 9.5, 0.3078),
(6.0, 10.0, 0.3079),
(6.0, 10.5, 0.308),
(6.5, 1.0, 0.31825),
(6.5, 1.5, 0.31965),
(6.5, 2.0, 0.321),
(6.5, 2.5, 0.3214),
(6.5, 3.0, 0.3223),
(6.5, 3.5, 0.3229),
(6.5, 4.0, 0.3232),
(6.5, 4.5, 0.3236),
(6.5, 5.0, 0.3239),
(6.5, 5.5, 0.3241),
(6.5, 6.0, 0.32425),
(6.5, 6.5, 0.32435),
(6.5, 7.0, 0.3244),
(6.5, 7.5, 0.32445),
(6.5, 8.0, 0.32455),
(6.5, 8.5, 0.3246),
(6.5, 9.0, 0.32485),
(6.5, 9.5, 0.32505),
(6.5, 10.0, 0.3251),
(6.5, 10.5, 0.32515),
```

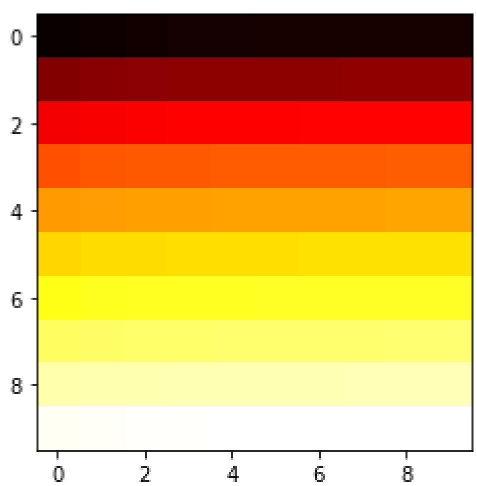
```
(7.0, 1.0, 0.33535),
(7.0, 1.5, 0.3368),
(7.0, 2.0, 0.33805),
(7.0, 2.5, 0.3386),
(7.0, 3.0, 0.3394),
(7.0, 3.5, 0.33985),
(7.0, 4.0, 0.34005),
(7.0, 4.5, 0.34045),
(7.0, 5.0, 0.34075),
(7.0, 5.5, 0.34095),
(7.0, 6.0, 0.3411),
(7.0, 6.5, 0.34115),
(7.0, 7.0, 0.3412),
(7.0, 7.5, 0.3413),
(7.0, 8.0, 0.34145),
(7.0, 8.5, 0.34155),
(7.0, 9.0, 0.3417),
(7.0, 9.5, 0.3418),
(7.0, 10.0, 0.34195),
(7.0, 10.5, 0.3421),
(7.5, 1.0, 0.35105),
(7.5, 1.5, 0.3524),
(7.5, 2.0, 0.3537),
(7.5, 2.5, 0.3543),
(7.5, 3.0, 0.35505),
(7.5, 3.5, 0.3554),
(7.5, 4.0, 0.35565),
(7.5, 4.5, 0.356),
(7.5, 5.0, 0.3563),
(7.5, 5.5, 0.35645),
(7.5, 6.0, 0.35655),
(7.5, 6.5, 0.3567),
(7.5, 7.0, 0.3568),
(7.5, 7.5, 0.35685),
(7.5, 8.0, 0.35695),
(7.5, 8.5, 0.3571),
(7.5, 9.0, 0.3573),
(7.5, 9.5, 0.35755),
(7.5, 10.0, 0.35765),
(7.5, 10.5, 0.35785),
(8.0, 1.0, 0.3657),
(8.0, 1.5, 0.36695),
(8.0, 2.0, 0.36805),
(8.0, 2.5, 0.36855),
(8.0, 3.0, 0.36925),
(8.0, 3.5, 0.36965),
(8.0, 4.0, 0.36985),
(8.0, 4.5, 0.37015),
(8.0, 5.0, 0.37045),
(8.0, 5.5, 0.37075),
(8.0, 6.0, 0.3708),
(8.0, 6.5, 0.37095),
(8.0, 7.0, 0.3711),
(8.0, 7.5, 0.37115),
(8.0, 8.0, 0.3714),
(8.0, 8.5, 0.3716),
(8.0, 9.0, 0.37175),
(8.0, 9.5, 0.37185),
(8.0, 10.0, 0.372),
(8.0, 10.5, 0.3721),
```

```
(8.5, 1.0, 0.3799),
(8.5, 1.5, 0.38105),
(8.5, 2.0, 0.3824),
(8.5, 2.5, 0.3827),
(8.5, 3.0, 0.38335),
(8.5, 3.5, 0.3838),
(8.5, 4.0, 0.3841),
(8.5, 4.5, 0.3844),
(8.5, 5.0, 0.3848),
(8.5, 5.5, 0.385),
(8.5, 6.0, 0.3851),
(8.5, 6.5, 0.38525),
(8.5, 7.0, 0.38535),
(8.5, 7.5, 0.3855),
(8.5, 8.0, 0.38575),
(8.5, 8.5, 0.3858),
(8.5, 9.0, 0.3861),
(8.5, 9.5, 0.3863),
(8.5, 10.0, 0.38645),
(8.5, 10.5, 0.38655),
(9.0, 1.0, 0.395),
(9.0, 1.5, 0.3961),
(9.0, 2.0, 0.39725),
(9.0, 2.5, 0.39765),
(9.0, 3.0, 0.3984),
(9.0, 3.5, 0.3988),
(9.0, 4.0, 0.39905),
(9.0, 4.5, 0.3994),
(9.0, 5.0, 0.3997),
(9.0, 5.5, 0.3999),
(9.0, 6.0, 0.39995),
(9.0, 6.5, 0.40005),
(9.0, 7.0, 0.4002),
(9.0, 7.5, 0.40025),
(9.0, 8.0, 0.40035),
(9.0, 8.5, 0.4005),
(9.0, 9.0, 0.40065),
(9.0, 9.5, 0.40075),
(9.0, 10.0, 0.40085),
(9.0, 10.5, 0.40115),
(9.5, 1.0, 0.40905),
(9.5, 1.5, 0.4102),
(9.5, 2.0, 0.4111),
(9.5, 2.5, 0.41175),
(9.5, 3.0, 0.4123),
(9.5, 3.5, 0.4126),
(9.5, 4.0, 0.41285),
(9.5, 4.5, 0.41305),
(9.5, 5.0, 0.41335),
(9.5, 5.5, 0.4137),
(9.5, 6.0, 0.41385),
(9.5, 6.5, 0.4139),
(9.5, 7.0, 0.4141),
(9.5, 7.5, 0.41425),
(9.5, 8.0, 0.41435),
(9.5, 8.5, 0.4144),
(9.5, 9.0, 0.4147),
(9.5, 9.5, 0.4149),
(9.5, 10.0, 0.415),
(9.5, 10.5, 0.4151),
```

```
 (10.0, 1.0, 0.42365),
 (10.0, 1.5, 0.42475),
 (10.0, 2.0, 0.4257),
 (10.0, 2.5, 0.4262),
 (10.0, 3.0, 0.4267),
 (10.0, 3.5, 0.4271),
 (10.0, 4.0, 0.42735),
 (10.0, 4.5, 0.4276),
 (10.0, 5.0, 0.42795),
 (10.0, 5.5, 0.42815),
 (10.0, 6.0, 0.4282),
 (10.0, 6.5, 0.42825),
 (10.0, 7.0, 0.42845),
 (10.0, 7.5, 0.4285),
 (10.0, 8.0, 0.42885),
 (10.0, 8.5, 0.42895),
 (10.0, 9.0, 0.4292),
 (10.0, 9.5, 0.4294),
 (10.0, 10.0, 0.4295),
 (10.0, 10.5, 0.42965),
 (10.5, 1.0, 0.43915),
 (10.5, 1.5, 0.4401),
 (10.5, 2.0, 0.4411),
 (10.5, 2.5, 0.44155),
 (10.5, 3.0, 0.44205),
 (10.5, 3.5, 0.44235),
 (10.5, 4.0, 0.4427),
 (10.5, 4.5, 0.44305),
 (10.5, 5.0, 0.44345),
 (10.5, 5.5, 0.4437),
 (10.5, 6.0, 0.44385),
 (10.5, 6.5, 0.444),
 (10.5, 7.0, 0.4441),
 (10.5, 7.5, 0.4442),
 (10.5, 8.0, 0.44455),
 (10.5, 8.5, 0.4447),
 (10.5, 9.0, 0.4449),
 (10.5, 9.5, 0.44495),
 (10.5, 10.0, 0.44505),
 (10.5, 10.5, 0.4452)]
```

In [ ]:

```
plot_heatmap(accu3)
```



In the weight ranges used by `accu3`, weight1 dominates.

In [ ]:
```
accu4 = get_accuracies(11, 21, 0.5, 15, 16, 1)
accu4
```

Out[ ]:
```
[(11.0, 15, 0.4598),
 (11.5, 15, 0.4725),
 (12.0, 15, 0.48775),
 (12.5, 15, 0.5004),
 (13.0, 15, 0.513),
 (13.5, 15, 0.52475),
 (14.0, 15, 0.5362),
 (14.5, 15, 0.54705),
 (15.0, 15, 0.55455),
 (15.5, 15, 0.5641),
 (16.0, 15, 0.5726),
 (16.5, 15, 0.5813),
 (17.0, 15, 0.5894),
 (17.5, 15, 0.59615),
 (18.0, 15, 0.6026),
 (18.5, 15, 0.60945),
 (19.0, 15, 0.61465),
 (19.5, 15, 0.6197),
 (20.0, 15, 0.6243),
 (20.5, 15, 0.6285)]
```

In [ ]:
```
accu5 = get_accuracies(21, 41, 1, 15, 16, 1)
accu5
```

Out[ ]:
```
[(21, 15, 0.633),
 (22, 15, 0.64095),
 (23, 15, 0.64825),
 (24, 15, 0.65405),
 (25, 15, 0.6597),
 (26, 15, 0.66435),
 (27, 15, 0.66885),
 (28, 15, 0.67275),
 (29, 15, 0.6766),
 (30, 15, 0.6794),
 (31, 15, 0.6824),
 (32, 15, 0.6851),
 (33, 15, 0.68785),
 (34, 15, 0.69075),
 (35, 15, 0.69325),
 (36, 15, 0.69515),
 (37, 15, 0.69745),
 (38, 15, 0.69985),
 (39, 15, 0.70185),
 (40, 15, 0.70395)]
```

In [ ]:
```
accu6 = get_accuracies(41, 61, 0.5, 15, 16, 1)
accu6
```

Out[ ]:
```
[(41.0, 15, 0.7052),
 (41.5, 15, 0.7057),
 (42.0, 15, 0.70665),
 (42.5, 15, 0.7072),
 (43.0, 15, 0.70795),
 (43.5, 15, 0.7086),
 (44.0, 15, 0.70945),
 (44.5, 15, 0.7101),
 (45.0, 15, 0.71075),
 (45.5, 15, 0.7115),
 (46.0, 15, 0.71205),
 (46.5, 15, 0.7126),
 (47.0, 15, 0.7132),
 (47.5, 15, 0.7135),
 (48.0, 15, 0.71375),
 (48.5, 15, 0.7143),
 (49.0, 15, 0.71435),
 (49.5, 15, 0.71495),
 (50.0, 15, 0.71535),
 (50.5, 15, 0.7159),
 (51.0, 15, 0.7162),
 (51.5, 15, 0.7163),
 (52.0, 15, 0.71675),
 (52.5, 15, 0.71715),
 (53.0, 15, 0.71755),
 (53.5, 15, 0.71755),
 (54.0, 15, 0.71815),
 (54.5, 15, 0.71855),
 (55.0, 15, 0.7189),
 (55.5, 15, 0.7194),
 (56.0, 15, 0.71965),
 (56.5, 15, 0.71995),
 (57.0, 15, 0.7202),
 (57.5, 15, 0.72065),
 (58.0, 15, 0.7211),
 (58.5, 15, 0.72135),
 (59.0, 15, 0.72195),
 (59.5, 15, 0.7225),
 (60.0, 15, 0.7227),
 (60.5, 15, 0.7229)]
```

In [ ]:
```
accu7 = get_accuracies(11, 101, 2, 15, 16, 1)
w1_to_accu = [(x, y) for x, _, y in accu7]
w1_to_accu
```

Out[ ]:
```
[(11, 0.4598),
 (13, 0.513),
 (15, 0.55455),
 (17, 0.5894),
 (19, 0.61465),
 (21, 0.633),
 (23, 0.64825),
 (25, 0.6597),
 (27, 0.66885),
 (29, 0.6766),
 (31, 0.6824),
 (33, 0.68785),
 (35, 0.69325),
 (37, 0.69745),
 (39, 0.70185),
 (41, 0.7052),
 (43, 0.70795),
 (45, 0.71075),
 (47, 0.7132),
 (49, 0.71435),
 (51, 0.7162),
 (53, 0.71755),
 (55, 0.7189),
 (57, 0.7202),
 (59, 0.72195),
 (61, 0.7235),
 (63, 0.72495),
 (65, 0.72625),
 (67, 0.72785),
 (69, 0.7287),
 (71, 0.7295),
 (73, 0.7305),
 (75, 0.7311),
 (77, 0.7321),
 (79, 0.7333),
 (81, 0.73385),
 (83, 0.7347),
 (85, 0.73545),
 (87, 0.73655),
 (89, 0.73725),
 (91, 0.73805),
 (93, 0.73875),
 (95, 0.73905),
 (97, 0.73965),
 (99, 0.74)]
```

In [ ]:
```
accu8 = get_accuracies(101, 201, 10, 15, 16, 1)
w1_to_accu = [(x, y) for x, _, y in accu8]
w1_to_accu
```

Out[ ]:
```
[(101, 0.7405),
 (111, 0.74235),
 (121, 0.74335),
 (131, 0.74415),
 (141, 0.7448),
 (151, 0.74595),
 (161, 0.74615),
 (171, 0.7479),
 (181, 0.7488),
 (191, 0.74945)]
```

In [ ]:
```
accu9 = get_accuracies(201, 2701, 100, 15, 16, 1)
w1_to_accu = [(x, y) for x, _, y in accu9]
w1_to_accu
```

Out[ ]:
```
[(201, 0.74975),
 (301, 0.7559),
 (401, 0.75715),
 (501, 0.75715),
 (601, 0.75715),
 (701, 0.75715),
 (801, 0.75715),
 (901, 0.75715),
 (1001, 0.75715),
 (1101, 0.75715),
 (1201, 0.75715),
 (1301, 0.75715),
 (1401, 0.75715),
 (1501, 0.75715),
 (1601, 0.75715),
 (1701, 0.75715),
 (1801, 0.75715),
 (1901, 0.75715),
 (2001, 0.75715),
 (2101, 0.75715),
 (2201, 0.75715),
 (2301, 0.75715),
 (2401, 0.75715),
 (2501, 0.75715),
 (2601, 0.75715)]
```

w1 doesn't seem to matter when it reaches 400.

In [ ]:
```
accu9 = get_accuracies(400, 401, 1, 10, 200, 10)
w2_to_accu = [(x, y) for _, x, y in accu9]
w2_to_accu
```

Out[ ]:
```
[(10, 0.75695),
 (20, 0.7573),
 (30, 0.7576),
 (40, 0.75775),
 (50, 0.7581),
 (60, 0.7583),
 (70, 0.75835),
 (80, 0.7585),
 (90, 0.7585),
 (100, 0.7586),
 (110, 0.7587),
 (120, 0.75885),
 (130, 0.7589),
 (140, 0.7589),
 (150, 0.759),
 (160, 0.75905),
 (170, 0.75915),
 (180, 0.75935),
 (190, 0.75945)]
```

In [ ]:
```
accu10 = get_accuracies(400, 401, 1, 200, 3000, 100)
w2_to_accu = [(x, y) for _, x, y in accu10]
w2_to_accu
```

Out[ ]:

```
[(200, 0.75945),
 (300, 0.76005),
 (400, 0.7605),
 (500, 0.76075),
 (600, 0.76085),
 (700, 0.76095),
 (800, 0.7611),
 (900, 0.7612),
 (1000, 0.7612),
 (1100, 0.7612),
 (1200, 0.7612),
 (1300, 0.7613),
 (1400, 0.76135),
 (1500, 0.76135),
 (1600, 0.7614),
 (1700, 0.76145),
 (1800, 0.76145),
 (1900, 0.7615),
 (2000, 0.7615),
 (2100, 0.7615),
 (2200, 0.7615),
 (2300, 0.7615),
 (2400, 0.7615),
 (2500, 0.7615),
 (2600, 0.7616),
 (2700, 0.7616),
 (2800, 0.7616),
 (2900, 0.7616)]
```

Conclusion

The accuracy increases as both weights increase, and stays 0.76 when weight1 set to >=400 and weight2 set to >= 2600.