

# XOKind - Machine Learning/Data Science Intern Interview

## Yelp rating predictions

### Traditional Machine learning Vs Graph Machine Learning

#### Traditional Machine Learning - Multilayer Perceptron Neural Networks

```
In [1]: #Importing necessary Libraries

import warnings

warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from keras.utils import np_utils
```

Using TensorFlow backend.

In [9]: *#Function to generate confusion matrix images from confusion matrix array*

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
    plt.figure()
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
    title_for_figure = "results/" + title + ".png"
    plt.savefig(title_for_figure)
```

In [ ]:

In [3]: *# path to data files*

```
business_json_path = 'dataset/business.json'
review_json_path = 'dataset/review.json'
user_json_path = 'dataset/user.json'
```

In [4]: *#read business file and extract restaurant data*

```
size = 500000

business = pd.read_json(business_json_path, lines=True,
                        dtype={'business_id':str,'name':str,
                              'address':str,'city':str,
                              'latitude':float,'longitude':float,
                              'state':str,'postal_code':str,
                              'stars':float,'review_count':int,
                              'is_open':int,
                              'attributes':object,'categories':object,
                              'hours':object},
                        chunksize=size)

business_drop_columns = ['name', 'address', 'city', 'state', 'postal_code',
                          'latitude', 'longitude', 'attributes', 'hours']
chunk_list_business = []

for chunk_business in business:
    # Drop columns that aren't needed
    chunk_business = chunk_business.drop(business_drop_columns, axis=1)

    # Renaming column name to avoid conflicts
    chunk_business.rename(columns={'stars': 'business_stars', 'review_count':
    'business_review_count',
                                'review_stars': 'business_review_stars'
    }, inplace=True)

    chunk_business = chunk_business[chunk_business['categories'].str.contains(
    'Restaurants', case=True,na=False)]

    chunk_list_business.append(chunk_business)

df_restaurants = pd.concat(chunk_list_business, ignore_index=True, join='outer', axis=0)
```

```
In [6]: df_restaurants.head()
```

```
Out[6]:
```

	business_id	business_stars	business_review_count	is_open	categories
0	pQeaRpvuhoEqudo3uymHIQ	4.5	5	1	Ethnic Food, Food Trucks, Specialty Food, Impo...
1	CsLQLiRoafpJPJSkNX2h5Q	3.0	5	0	Food, Restaurants, Grocery, Middle Eastern
2	eBEfgOPG7pvFhb2wcG9l7w	4.5	4	1	Restaurants, Cheesesteaks, Pouteries
3	lu7vtrp_bE9PnxWfA8g4Pg	4.5	7	1	Japanese, Fast Food, Food Court, Restaurants
4	9sRGfSVEfLhN_km60YruTA	3.0	3	1	Persian/Iranian, Turkish, Middle Eastern, Rest...

```
In [5]: #Delete non-essential data to save memory
```

```
del chunk_business  
del chunk_list_business
```

```

In [7]: # =====
=
# Reviews data
# =====
=
size = 500000

review = pd.read_json(review_json_path, lines=True,
                      dtype={'review_id':str,'user_id':str,
                            'business_id':str,'stars':int,
                            'date':str,'text':str,'useful':int,
                            'funny':int,'cool':int},
                      chunksize=size)

chunk_list = []
for chunk_review in review:

    # Drop columns that aren't needed
    chunk_review = chunk_review.drop(['text', 'date', 'review_id','useful','funny','cool'], axis=1)

    # Renaming column name to avoid conflicts
    chunk_review = chunk_review.rename(columns={'stars': 'review_stars'})

    # Inner merge with edited business file so only reviews related to the restaurants remain
    chunk_merged = pd.merge(df_restaurants, chunk_review, on='business_id', how='inner')

    # Show feedback on progress
    print(f"{chunk_merged.shape[0]} out of {size:,} related reviews")

    chunk_list.append(chunk_merged)

# After trimming down the review file, concatenate all relevant data back to one dataframe
df_restaurant_reviews = pd.concat(chunk_list, ignore_index=True, join='outer', axis=0)

df_restaurant_reviews.head()

```

330699 out of 500,000 related reviews  
303722 out of 500,000 related reviews  
311627 out of 500,000 related reviews  
319846 out of 500,000 related reviews  
296928 out of 500,000 related reviews  
331148 out of 500,000 related reviews  
312615 out of 500,000 related reviews  
312472 out of 500,000 related reviews  
323678 out of 500,000 related reviews  
299857 out of 500,000 related reviews  
315579 out of 500,000 related reviews  
325414 out of 500,000 related reviews  
307972 out of 500,000 related reviews  
330227 out of 500,000 related reviews  
311351 out of 500,000 related reviews  
311776 out of 500,000 related reviews  
11081 out of 500,000 related reviews

Out[7]:

	business_id	business_stars	business_review_count	is_open	categories	
0	pQeaRpvuhoEqudo3uymHIQ	4.5	5	1	Ethnic Food, Food Trucks, Specialty Food, Impo...	eSc
1	pQeaRpvuhoEqudo3uymHIQ	4.5	5	1	Ethnic Food, Food Trucks, Specialty Food, Impo...	5So:
2	pQeaRpvuhoEqudo3uymHIQ	4.5	5	1	Ethnic Food, Food Trucks, Specialty Food, Impo...	Oh
3	CsLQLiRoafpJPJSkNX2h5Q	3.0	5	0	Food, Restaurants, Grocery, Middle Eastern	wh
4	CsLQLiRoafpJPJSkNX2h5Q	3.0	5	0	Food, Restaurants, Grocery, Middle Eastern	TC

In [8]: *#Delete non-essential data to save memory*

```
del chunk_review
del chunk_merged
del chunk_list
```

```

In [10]: # =====
#
# User data
# =====

size = 500000

user = pd.read_json(user_json_path, lines=True,
                    dtype={'user_id':str,'name':str,
                           'yelping_since':str,'review_count':int,
                           'friends':object,'useful':int,
                           'funny':int,'cool':int,'fans':int,
                           'elite':list, 'average_stars':float,'compliment_h
ot':int,
                           'compliment_more':int,'compliment_more':int,'comp
liment_profile':int,
                           'compliment_cute':int,'compliment_list':int,'comp
liment_note':int,
                           'compliment_plain':int,'compliment_cool':int,'com
pliment_funny':int,
                           'compliment_writer':int,'compliment_photos':int},
                    chunksize=size)

user_drop_columns = ['name', 'yelping_since', 'friends']

chunk_list_user = []

for chunk_user in user:
    # Drop columns that aren't needed
    chunk_user = chunk_user.drop(user_drop_columns, axis=1)

    # Renaming column name to avoid conflicts
    chunk_user.rename(columns={'review_count': 'user_review_count', 'average_s
tars': 'user_average_stars'})

    chunk_list_user.append(chunk_user)

# concatenate to one dataframe
df_user = pd.concat(chunk_list_user, ignore_index=True, join='outer', axis=0)

df_user.head()

```

Out[10]:

	user_id	review_count	useful	funny	cool	
0	ntlvpPzc8eglvk92iDIAw	553	628	225	227	
1	FOBRPIBHa3WPHFB5qYDIVg	564	790	316	400	2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022,2023,2024,2025,2026,2027,2028,2029,2030,2031,2032,2033,2034,2035,2036,2037,2038,2039,2040,2041,2042,2043,2044,2045,2046,2047,2048,2049,2050,2051,2052,2053,2054,2055,2056,2057,2058,2059,2060,2061,2062,2063,2064,2065,2066,2067,2068,2069,2070,2071,2072,2073,2074,2075,2076,2077,2078,2079,2080,2081,2082,2083,2084,2085,2086,2087,2088,2089,2090,2091,2092,2093,2094,2095,2096,2097,2098,2099,2100,2101,2102,2103,2104,2105,2106,2107,2108,2109,2110,2111,2112,2113,2114,2115,2116,2117,2118,2119,2120,2121,2122,2123,2124,2125,2126,2127,2128,2129,2130,2131,2132,2133,2134,2135,2136,2137,2138,2139,2140,2141,2142,2143,2144,2145,2146,2147,2148,2149,2150,2151,2152,2153,2154,2155,2156,2157,2158,2159,2160,2161,2162,2163,2164,2165,2166,2167,2168,2169,2170,2171,2172,2173,2174,2175,2176,2177,2178,2179,2180,2181,2182,2183,2184,2185,2186,2187,2188,2189,2190,2191,2192,2193,2194,2195,2196,2197,2198,2199,2200,2201,2202,2203,2204,2205,2206,2207,2208,2209,2210,2211,2212,2213,2214,2215,2216,2217,2218,2219,2220,2221,2222,2223,2224,2225,2226,2227,2228,2229,2230,2231,2232,2233,2234,2235,2236,2237,2238,2239,2240,2241,2242,2243,2244,2245,2246,2247,2248,2249,2250,2251,2252,2253,2254,2255,2256,2257,2258,2259,2260,2261,2262,2263,2264,2265,2266,2267,2268,2269,2270,2271,2272,2273,2274,2275,2276,2277,2278,2279,2280,2281,2282,2283,2284,2285,2286,2287,2288,2289,2290,2291,2292,2293,2294,2295,2296,2297,2298,2299,2300,2301,2302,2303,2304,2305,2306,2307,2308,2309,2310,2311,2312,2313,2314,2315,2316,2317,2318,2319,2320,2321,2322,2323,2324,2325,2326,2327,2328,2329,2330,2331,2332,2333,2334,2335,2336,2337,2338,2339,2340,2341,2342,2343,2344,2345,2346,2347,2348,2349,2350,2351,2352,2353,2354,2355,2356,2357,2358,2359,2360,2361,2362,2363,2364,2365,2366,2367,2368,2369,2370,2371,2372,2373,2374,2375,2376,2377,2378,2379,2380,2381,2382,2383,2384,2385,2386,2387,2388,2389,2390,2391,2392,2393,2394,2395,2396,2397,2398,2399,2400,2401,2402,2403,2404,2405,2406,2407,2408,2409,2410,2411,2412,2413,2414,2415,2416,2417,2418,2419,2420,2421,2422,2423,2424,2425,2426,2427,2428,2429,2430,2431,2432,2433,2434,2435,2436,2437,2438,2439,2440,2441,2442,2443,2444,2445,2446,2447,2448,2449,2450,2451,2452,2453,2454,2455,2456,2457,2458,2459,2460,2461,2462,2463,2464,2465,2466,2467,2468,2469,2470,2471,2472,2473,2474,2475,2476,2477,2478,2479,2480,2481,2482,2483,2484,2485,2486,2487,2488,2489,2490,2491,2492,2493,2494,2495,2496,2497,2498,2499,2500,2501,2502,2503,2504,2505,2506,2507,2508,2509,2510,2511,2512,2513,2514,2515,2516,2517,2518,2519,2520,2521,2522,2523,2524,2525,2526,2527,2528,2529,2530,2531,2532,2533,2534,2535,2536,2537,2538,2539,2540,2541,2542,2543,2544,2545,2546,2547,2548,2549,2550,2551,2552,2553,2554,2555,2556,2557,2558,2559,2560,2561,2562,2563,2564,2565,2566,2567,2568,2569,2570,2571,2572,2573,2574,2575,2576,2577,2578,2579,2580,2581,2582,2583,2584,2585,2586,2587,2588,2589,2590,2591,2592,2593,2594,2595,2596,2597,2598,2599,2600,2601,2602,2603,2604,2605,2606,2607,2608,2609,2610,2611,2612,2613,2614,2615,2616,2617,2618,2619,2620,2621,2622,2623,2624,2625,2626,2627,2628,2629,2630,2631,2632,2633,2634,2635,2636,2637,2638,2639,2640,2641,2642,2643,2644,2645,2646,2647,2648,2649,2650,2651,2652,2653,2654,2655,2656,2657,2658,2659,2660,2661,2662,2663,2664,2665,2666,2667,2668,2669,2670,2671,2672,2673,2674,2675,2676,2677,2678,2679,2680,2681,2682,2683,2684,2685,2686,2687,2688,2689,2690,2691,2692,2693,2694,2695,2696,2697,2698,2699,2700,2701,2702,2703,2704,2705,2706,2707,2708,2709,2710,2711,2712,2713,2714,2715,2716,2717,2718,2719,2720,2721,2722,2723,2724,2725,2726,2727,2728,2729,2730,2731,2732,2733,2734,2735,2736,2737,2738,2739,2740,2741,2742,2743,2744,2745,2746,2747,2748,2749,2750,2751,2752,2753,2754,2755,2756,2757,2758,2759,2760,2761,2762,2763,2764,2765,2766,2767,2768,2769,2770,2771,2772,2773,2774,2775,2776,2777,2778,2779,2780,2781,2782,2783,2784,2785,2786,2787,2788,2789,2790,2791,2792,2793,2794,2795,2796,2797,2798,2799,2800,2801,2802,2803,2804,2805,2806,2807,2808,2809,2810,2811,2812,2813,2814,2815,2816,2817,2818,2819,2820,2821,2822,2823,2824,2825,2826,2827,2828,2829,2830,2831,2832,2833,2834,2835,2836,2837,2838,2839,2840,2841,2842,2843,2844,2845,2846,2847,2848,2849,2850,2851,2852,2853,2854,2855,2856,2857,2858,2859,2860,2861,2862,2863,2864,2865,2866,2867,2868,2869,2870,2871,2872,2873,2874,2875,2876,2877,2878,2879,2880,2881,2882,2883,2884,2885,2886,2887,2888,2889,2890,2891,2892,2893,2894,2895,2896,2897,2898,2899,2900,2901,2902,2903,2904,2905,2906,2907,2908,2909,2910,2911,2912,2913,2914,2915,2916,2917,2918,2919,2920,2921,2922,2923,2924,2925,2926,2927,2928,2929,2930,2931,2932,2933,2934,2935,2936,2937,2938,2939,2940,2941,2942,2943,2944,2945,2946,2947,2948,2949,2950,2951,2952,2953,2954,2955,2956,2957,2958,2959,2960,2961,2962,2963,2964,2965,2966,2967,2968,2969,2970,2971,2972,2973,2974,2975,2976,2977,2978,2979,2980,2981,2982,2983,2984,2985,2986,2987,2988,2989,2990,2991,2992,2993,2994,2995,2996,2997,2998,2999,3000,3001,3002,3003,3004,3005,3006,3007,3008,3009,3010,3011,3012,3013,3014,3015,3016,3017,3018,3019,3020,3021,3022,3023,3024,3025,3026,3027,3028,3029,3030,3031,3032,3033,3034,3035,3036,3037,3038,3039,3040,3041,3042,3043,3044,3045,3046,3047,3048,3049,3050,3051,3052,3053,3054,3055,3056,3057,3058,3059,3060,3061,3062,3063,3064,3065,3066,3067,3068,3069,3070,3071,3072,3073,3074,3075,3076,3077,3078,3079,3080,3081,3082,3083,3084,3085,3086,3087,3088,3089,3090,3091,3092,3093,3094,3095,3096,3097,3098,3099,3100,3101,3102,3103,3104,3105,3106,3107,3108,3109,3110,3111,3112,3113,3114,3115,3116,3117,3118,3119,3120,3121,3122,3123,3124,3125,3126,3127,3128,3129,3130,3131,3132,3133,3134,3135,3136,3137,3138,3139,3140,3141,3142,3143,3144,3145,3146,3147,3148,3149,3150,3151,3152,3153,3154,3155,3156,3157,3158,3159,3160,3161,3162,3163,3164,3165,3166,3167,3168,3169,3170,3171,3172,3173,3174,3175,3176,3177,3178,3179,3180,3181,3182,3183,3184,3185,3186,3187,3188,3189,3190,3191,3192,3193,3194,3195,3196,3197,3198,3199,3200,3201,3202,3203,3204,3205,3206,3207,3208,3209,3210,3211,3212,3213,3214,3215,3216,3217,3218,3219,3220,3221,3222,3223,3224,3225,3226,3227,3228,3229,3230,3231,3232,3233,3234,3235,3236,3237,3238,3239,3240,3241,3242,3243,3244,3245,3246,3247,3248,3249,3250,3251,3252,3253,3254,3255,3256,3257,3258,3259,3260,3261,3262,3263,3264,3265,3266,3267,3268,3269,3270,3271,3272,3273,3274,3275,3276,3277,3278,3279,3280,3281,3282,3283,3284,3285,3286,3287,3288,3289,3290,3291,3292,3293,3294,3295,3296,3297,3298,3299,3300,3301,3302,3303,3304,3305,3306,3307,3308,3309,3310,3311,3312,3313,3314,3315,3316,3317,3318,3319,3320,3321,3322,3323,3324,3325,3326,3327,3328,3329,3330,3331,3332,3333,3334,3335,3336,3337,3338,3339,3340,3341,3342,3343,3344,3345,3346,3347,3348,3349,3350,3351,3352,3353,3354,3355,3356,3357,3358,3359,3360,3361,3362,3363,3364,3365,3366,3367,3368,3369,3370,3371,3372,3373,3374,3375,3376,3377,3378,3379,3380,3381,3382,3383,3384,3385,3386,3387,3388,3389,3390,3391,3392,3393,3394,3395,3396,3397,3398,3399,3400,3401,3402,3403,3404,3405,3406,3407,3408,3409,3410,3411,3412,3413,3414,3415,3416,3417,3418,3419,3420,3421,3422,3423,3424,3425,3426,3427,3428,3429,3430,3431,3432,3433,3434,3435,3436,3437,3438,3439,3440,3441,3442,3443,3444,3445,3446,3447,3448,3449,3450,3451,3452,3453,3454,3455,3456,3457,3458,3459,3460,3461,3462,3463,3464,3465,3466,3467,3468,3469,3470,3471,3472,3473,3474,3475,3476,3477,3478,3479,3480,3481,3482,3483,3484,3485,3486,3487,3488,3489,3490,3491,3492,3493,3494,3495,3496,3497,3498,3499,3500,3501,3502,3503,3504,3505,3506,3507,3508,3509,3510,3511,3512,3513,3514,3515,3516,3517,3518,3519,3520,3521,3522,3523,3524,3525,3526,3527,3528,3529,3530,3531,3532,3533,3534,3535,3536,3537,3538,3539,3540,3541,3542,3543,3544,3545,3546,3547,3548,3549,3550,3551,3552,3553,3554,3555,3556,3557,3558,3559,3560,3561,3562,3563,3564,3565,3566,3567,3568,3569,3570,3571,3572,3573,3574,3575,3576,3577,3578,3579,3580,3581,3582,3583,3584,3585,3586,3587,3588,3589,3590,3591,3592,3593,3594,3595,3596,3597,3598,3599,3600,3601,3602,3603,3604,3605,3606,3607,3608,3609,3610,3611,3612,3613,3614,3615,3616,3617,3618,3619,3620,3621,3622,3623,3624,3625,3626,3627,3628,3629,3630,3631,3632,3633,3634,3635,3636,3637,3638,3639,3640,3641,3642,3643,3644,3645,3646,3647,3648,3649,3650,3651,3652,3653,3654,3655,3656,3657,3658,3659,3660,3661,3662,3663,3664,3665,3666,3667,3668,3669,3670,3671,3672,3673,3674,3675,3676,3677,3678,3679,3680,3681,3682,3683,3684,3685,3686,3687,3688,3689,3690,3691,3692,3693,3694,3695,3696,3697,3698,3699,3700,3701,3702,3703,3704,3705,3706,3707,3708,3709,3710,3711,3712,3713,3714,3715,3716,3717,3718,3719,3720,3721,3722,3723,3724,3725,3726,3727,3728,3729,3730,3731,3732,3733,3734,3735,3736,3737,3738,3739,3740,3741,3742,3743,3744,3745,3746,3747,3748,3749,3750,3751,3752,3753,3754,3755,3756,3757,3758,3759,3760,3761,3762,3763,3764,3765,3766,3767,3768,3769,3770,3771,3772,3773,3774,3775,3776,3777,3778,3779,3780,3781,3782,3783,3784,3785,3786,3787,3788,3789,3790,3791,3792,3793,3794,3795,3796,3797,3798,3799,3800,3801,3802,3803,3804,3805,3806,3807,3808,3809,3810,3811,3812,3813,3814,3815,3816,3817,3818,3819,3820,3821,3822,3823,3824,3825,3826,3827,3828,3829,3830,3831,3832,3833,3834,3835,3836,3837,3838,3839,3840,3841,3842,3843,3844,3845,3846,3847,3848,3849,3850,3851,3852,3853,3854,3855,3856,3857,3858,3859,3860,3861,3862,3863,3864,3865,3866,3867,3868,3869,3870,3871,3872,3873,3874,3875,3876,3877,3878,3879,3880,3881,3882,3883,3884,3885,3886,3887,3888,3889,3890,3891,3892,3893,3894,3895,3896,3897,3898,3899,3900,3901,3902,3903,3904,3905,3906,3907,3908,3909,3910,3911,3912,3913,3914,3915,3916,3917,3918,3919,3920,3921,3922,3923,3924,3925,3926,3927,3928,3929,3930,3931,3932,3933,3934,3935,3936,3937,3938,3939,3940,3941,3942,3943,3944,3945,3946,3947,3948,3949,3950,3951,3952,3953,3954,3955,3956,3957,3958,3959,3960,3961,3962,3963,3964,3965,3966,3967,3968,3969,3970,3971,3972,3973,3974,3975,3976,3977,3978,3979,3980,3981,3982,3983,3984,3985,3986,3987,3988,3989,3990,3991,3992,3993,3994,3995,3996,3997,3998,3999,4000,4001,4002,4003,4004,4005,4006,4007,4008,4009,4010,4011,4012,4013,4014,4015,4016,4017,4018,4019,4020,4021,4022,4023,4024,4025,4026,4027,4028,4029,4030,4031,4032,4033,4034,4035,4036,4037,4038,4039,4040,4041,4042,4043,4044,4045,4046,4047,4048,4049,4050,4051,4052,4053,4054,4055,4056,4057,4058,4059,4060,4061,4062,4063,4064,4065,4066,4067,4068,4069,4070,4071,4072,4073,4074,4075,4076,4077,4078,4079,4080,4081,4082,4083,4084,4085,4086,4087,4088,4089,4090,4091,4092,4093,4094,4095,4096,4097,4098,4099,4100,4101,4102,4103,4104,4105,4106,4107,4108,4109,4110,4111,4112,4113,4114,4115,4116,4117,4118,4119,4120,4121,4122,4123,4124,4125,4126,4127,4128,4129,4130,4131,4132,4133,4134,4135,4136,4137,4138,4139,4140,4141,4142,4143,4144,4145,4146,4147,4148,4149,4150,4151,4152,4153,4154,4155,4156,4157,4158,4159,4160,4161,4162,4163,4164,4165,4166,4167,4168,4169,4170,4171,4172,4173,4174,4175,4176,4177,4178,4179,4180,4181,4182,4183,4184,4185,4186,4187,4188,4189,4190,4191,4192,4193,4194,4195,4196,4197,4198,4199,4200,4201,4202,4203,4204,4205,4206,4207,4208,4209,4210,4211,4212,4213,4214,4215,4216,4217,4218,4219,4220,4221,4222,4223,4224,4225,4226,4227,4228,4229,4230,4231,4232,4233,4234,4235,4236,4237,4238,4239,4240,4241,4242,4243,4244,4245,4246,4247,4248,4249,4250,4251,4252,4253,4254,4255,4256,4257,4258,4259,4260,4261,4262,4263,4264,4265,4266,4267,4268,4269,4270,4271,4272,4273,4274,4275,4276,4277,4278,4279,4280,4281,4282,4283,4284,4285,4286,4287,4288,4289,4290,4291,4292,4293,4294,4295,4296,4297,4298,4299,4300,4301,4302,4303,4304,4305,4306,4307,4308,4309,4310,4311,4312,4313,4314,4315,4316,4317,4318,4319,4320,4321,4322,4323,4324,4325,4326,4327,4328,4329,4330,4331,4332,4333,4334,4335,4336,4337,4338,4339,4340,4341,4342,4343,4344,4345,4346,4347,4348,4349,4350,4351,4352,4353,4354,4355,4356

In [11]: *#Delete non-essential data to save memory*

```
del chunk_user
del chunk_list_user
```

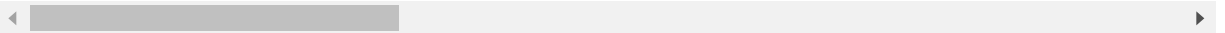
In [12]: *# Merge users and restaurant reviews data --> this dataframe will contain information about user, restaurant and review*

```
merged_df = df_user.merge(df_restaurant_reviews, how='inner', left_on=["user_id"], right_on=["user_id"])
merged_df.head()
```

Out[12]:

	user_id	review_count	useful	funny	cool	elite	fans	average_stars	compliments
0	ntlvfPzc8eglqvK92iDIAw	553	628	225	227		14	3.57	
1	ntlvfPzc8eglqvK92iDIAw	553	628	225	227		14	3.57	
2	ntlvfPzc8eglqvK92iDIAw	553	628	225	227		14	3.57	
3	ntlvfPzc8eglqvK92iDIAw	553	628	225	227		14	3.57	
4	ntlvfPzc8eglqvK92iDIAw	553	628	225	227		14	3.57	

5 rows × 25 columns



In [13]: *#Delete non-essential data to save memory*

```
del df_user
del df_restaurant_reviews
```



```

In [14]: # Create a new feature mean compliment score for each users

merged_drop_columns = ['business_id', 'user_id', 'elite']

merged_df.drop(merged_drop_columns, axis = 1, inplace = True)

compliment_columns = ['compliment_cool', 'compliment_cute', 'compliment_funny',
,
                        'compliment_hot', 'compliment_list', 'complimen
t_more',
                        'compliment_note', 'compliment_photos', 'compli
ment_plain',
                        'compliment_profile', 'compliment_writer']

merged_df['mean_compliment_score'] = merged_df.loc[:, compliment_columns].mea
n(axis=1)

merged_df.drop(compliment_columns, axis = 1, inplace = True)

merged_df.head()

```

```

Out[14]:

```

	review_count	useful	funny	cool	fans	average_stars	business_stars	business_review_coun
0	553	628	225	227	14	3.57	3.5	72
1	553	628	225	227	14	3.57	3.5	65
2	553	628	225	227	14	3.57	4.0	413
3	553	628	225	227	14	3.57	3.5	72
4	553	628	225	227	14	3.57	4.5	19

```
In [16]: # Expand by restaurant category to investigate restaurant categories and their
         overall count in data
```

```
df_yelp_expand_by_category = merged_df.assign(categories = merged_df.categories
s
        .str.split(', ').explode('categories'))
df_yelp_category_count = df_yelp_expand_by_category.categories.value_counts()
df_yelp_expand_by_category.head()
```

Out[16]:

	review_count	useful	funny	cool	fans	average_stars	business_stars	business_review_coun
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721

```
In [18]: df_yelp_category_count.head(11)
```

```
Out[18]: Restaurants      5055992
          Food              1394078
          Nightlife         1267512
          Bars              1227097
          American (Traditional) 902047
          American (New)      881417
          Breakfast & Brunch  838307
          Sandwiches         586848
          Mexican            499055
          Burgers            494426
          Pizza              479792
          Name: categories, dtype: int64
```

```
In [19]: # Selecting top 10 restaurants based on count
top_10_restaurants = list(df_yelp_category_count.index.values)[1:11] #first element is Restaurant, so index 1 to 11

df_yelp_top10 = df_yelp_expand_by_category.loc[df_yelp_expand_by_category['categories'].isin(top_10_restaurants)]

df_yelp_top10.head()
```

```
Out[19]:
```

	review_count	useful	funny	cool	fans	average_stars	business_stars	business_review_count
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721
2	553	628	225	227	14	3.57	4.0	413
2	553	628	225	227	14	3.57	4.0	413

```
In [25]: # Create One Hot Encoding for categories column

df_yelp_top10_ohe = pd.get_dummies(data = df_yelp_top10, prefix = 'is',
                                   columns = ['categories'],
                                   drop_first= True, sparse = True)
df_yelp_top10_ohe.head()
```

```
Out[25]:
```

	review_count	useful	funny	cool	fans	average_stars	business_stars	business_review_count
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721
0	553	628	225	227	14	3.57	3.5	721
2	553	628	225	227	14	3.57	4.0	413
2	553	628	225	227	14	3.57	4.0	413

```
In [21]: #Delete non-essential data to save memory

del df_yelp_expand_by_category
del merged_df
```

```
In [27]: #Standardize columns - (x- mean(x))/std(x) --> important for Gradient Descent
         based algorithms such as MLPNN

cols_to_norm = ['review_count', 'useful', 'funny', 'cool', 'fans', 'average_stars',
               'business_stars',
               'business_review_count', 'mean_compliment_score']

df_yelp_top10_ohe[cols_to_norm] = df_yelp_top10_ohe[cols_to_norm].apply(lambda
x: (x - np.mean(x)) / (np.std(x)))

df_yelp_top10_ohe.head()
```

Out[27]:

	review_count	useful	funny	cool	fans	average_stars	business_stars	business_review_count	mean_compliment_score
0	1.236976	0.074949	0.010345	-0.028027	0.021685	-0.258375	-0.409617	0.000000	0.000000
0	1.236976	0.074949	0.010345	-0.028027	0.021685	-0.258375	-0.409617	0.000000	0.000000
0	1.236976	0.074949	0.010345	-0.028027	0.021685	-0.258375	-0.409617	0.000000	0.000000
2	1.236976	0.074949	0.010345	-0.028027	0.021685	-0.258375	0.411236	0.000000	0.000000
2	1.236976	0.074949	0.010345	-0.028027	0.021685	-0.258375	0.411236	0.000000	0.000000

```
In [28]: #Create training and testing sets for model training

x_train, x_test, y_train, y_test = train_test_split(df_yelp_top10_ohe.drop(['review_stars'], axis = 1).values,
                                                    df_yelp_top10_ohe[['review_stars']].values, test_size=0.2,
                                                    random_state=0)
```

In [29]: #converting from object type to float

```
x_train = x_train.astype(float)
x_test = x_test.astype(float)
```

In [30]: #Label encoding and One hot Encoding for target variable

```
encoder = LabelEncoder()
encoder.fit(y_train)
encoded_y_train = encoder.transform(y_train)

y_train_ohe = np_utils.to_categorical(encoded_y_train)
```

In [31]: *# To calculate class weights = to address imbalanced class size*

```
(unique, counts) = np.unique(y_train, return_counts=True)
counts = counts/sum(counts)

inv_counts = 1/counts

class_weights = {}

for i in range(len(unique)):
    class_weights[i] = inv_counts[i]
```

In [32]: class\_weights

Out[32]: {0: 8.186850818570097,  
1: 10.986759413267187,  
2: 7.994886929429132,  
3: 4.088424089304806,  
4: 2.3971543056885567}

In [33]: tf.keras.backend.clear\_session()

*# Configure a simple MLPNN model with many of the default parameters.*

```
model = tf.keras.models.Sequential([tf.keras.layers.Dense(40, input_dim=x_train.shape[1], activation=tf.nn.relu),
                                     tf.keras.layers.Dense(20, activation=tf.nn.relu),
                                     tf.keras.layers.Dense(5, activation=tf.nn.softmax)])
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\ops\init\_ops.py:1251: calling VarianceScaling.\_\_init\_\_ (from tensorflow.python.ops.init\_ops) with dtype is deprecated and will be removed in a future version.

Instructions for updating:

Call initializer instance with the dtype argument instead of passing it to the constructor

In [34]: *#Model training and saving history of training and validation accuracies*

```
history = model.fit(  
    x_train,  
    y_train_ohe,  
    batch_size=1000,  
    epochs=10,  
    verbose=1,  
    validation_split=0.1,  
    class_weight=class_weights)
```

Train on 6170816 samples, validate on 685647 samples

Epoch 1/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.6  
582 - acc: 0.4521 - val\_loss: 6.5993 - val\_acc: 0.4611

Epoch 2/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
903 - acc: 0.4577 - val\_loss: 6.5793 - val\_acc: 0.4573

Epoch 3/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
767 - acc: 0.4593 - val\_loss: 6.5686 - val\_acc: 0.4645

Epoch 4/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
681 - acc: 0.4606 - val\_loss: 6.5614 - val\_acc: 0.4632

Epoch 5/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
590 - acc: 0.4618 - val\_loss: 6.5515 - val\_acc: 0.4683

Epoch 6/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
509 - acc: 0.4628 - val\_loss: 6.5472 - val\_acc: 0.4595

Epoch 7/10

6170816/6170816 [=====] - 17s 3us/sample - loss: 6.5  
453 - acc: 0.4630 - val\_loss: 6.5459 - val\_acc: 0.4676

Epoch 8/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
376 - acc: 0.4635 - val\_loss: 6.5304 - val\_acc: 0.4700

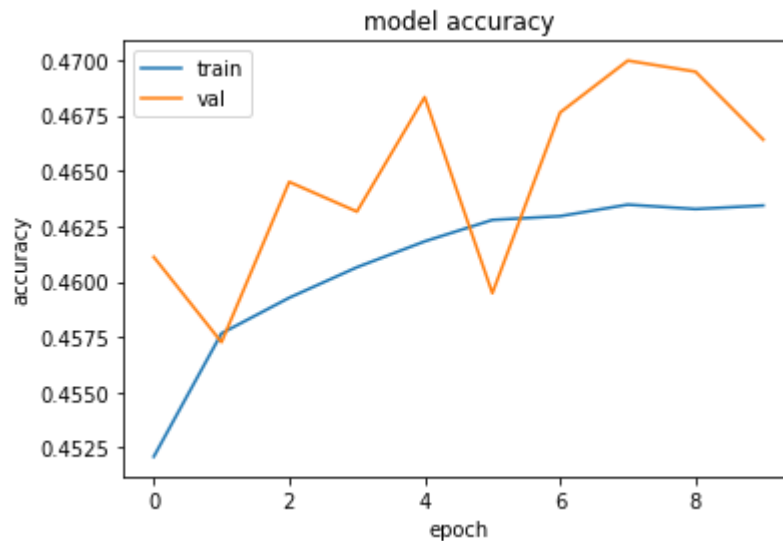
Epoch 9/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
282 - acc: 0.4633 - val\_loss: 6.5242 - val\_acc: 0.4695

Epoch 10/10

6170816/6170816 [=====] - 16s 3us/sample - loss: 6.5  
214 - acc: 0.4634 - val\_loss: 6.5143 - val\_acc: 0.4664

```
In [35]: # Plot training curves
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



```
In [36]: #Preparing targets in the test data for performance comparison + Predicting ratings for test data

y_test_encoded = encoder.transform(y_test)
y_test_ohe = np_utils.to_categorical(y_test_encoded)

y_pred = model.predict_classes(x_test)
y_pred += 1 #to match the labels as model outputs values 0-4 instead of 1-5 which label encoder uses

y_pred_encoded = encoder.transform(y_pred)
y_pred_ohe = np_utils.to_categorical(y_pred_encoded)
```

```
In [37]: #To print classification report with metrics such as accuracy, precision, recall and f1-score
```

```
target_names = ['1', '2', '3', '4', '5']
```

```
print(classification_report(y_test, y_pred, target_names=target_names))
```

	precision	recall	f1-score	support
1	0.50	0.56	0.53	208544
2	0.18	0.31	0.23	155708
3	0.26	0.31	0.28	214697
4	0.40	0.35	0.37	419794
5	0.71	0.59	0.64	715373
accuracy			0.47	1714116
macro avg	0.41	0.42	0.41	1714116
weighted avg	0.50	0.47	0.48	1714116

```
In [38]: #Print confusion matrix
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
```

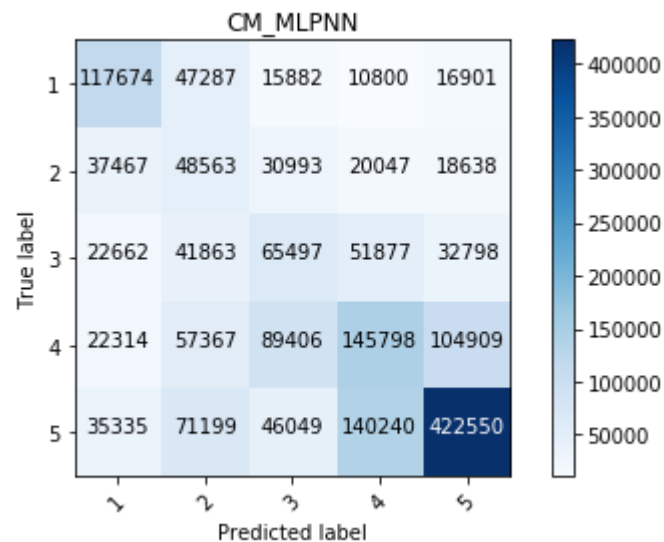
```
[[117674  47287  15882  10800  16901]
 [ 37467  48563  30993  20047  18638]
 [ 22662  41863  65497  51877  32798]
 [ 22314  57367  89406 145798 104909]
 [ 35335  71199  46049 140240 422550]]
```



```
In [39]: plot_confusion_matrix(cm, target_names, normalize = False, title = 'CM_MLPNN')
```

Confusion matrix, without normalization

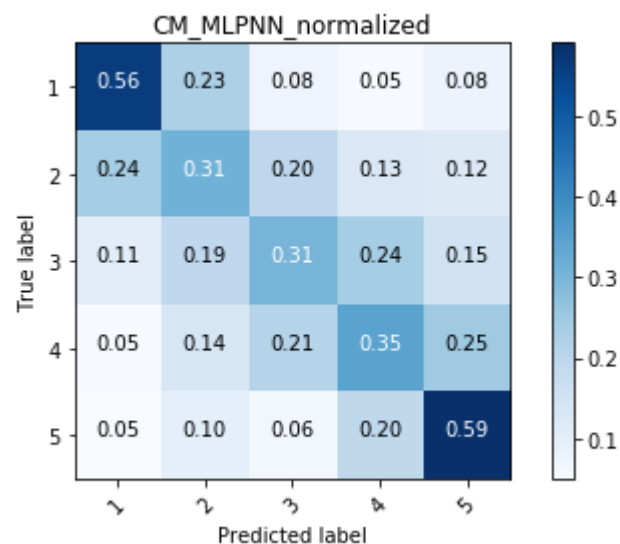
```
[[117674 47287 15882 10800 16901]
 [ 37467 48563 30993 20047 18638]
 [ 22662 41863 65497 51877 32798]
 [ 22314 57367 89406 145798 104909]
 [ 35335 71199 46049 140240 422550]]
```



```
In [40]: plot_confusion_matrix(cm, target_names, normalize = True, title = 'CM_MLPNN_normalized')
```

Normalized confusion matrix

```
[[0.56426462 0.22674831 0.07615659 0.05178763 0.08104285]
 [0.24062347 0.31188507 0.19904565 0.1287474 0.11969841]
 [0.10555341 0.19498642 0.30506714 0.2416289 0.15276413]
 [0.05315464 0.13665512 0.21297589 0.34730844 0.24990591]
 [0.04939381 0.0995271 0.06437062 0.19603759 0.59067088]]
```



**END**