CS57300: Assignment 2
Name: Tunazzina Islam, PUID: 0031294421

1. Preprocessing:

   python preprocess.py dating-full.csv dating.csv

   Outputs of preprocessing.py

   Quotes removed from 8316 cells.
   Standardized 5707 cells to lower case.
   Value assigned for male in column gender: 1 .
   Value assigned for European/Caucasian-American in column race: 2 .
   Value assigned for Latino/Hispanic American in column race_o: 3 .
   Value assigned for law in column field: 121 .
   Mean of attractive_important: 0.22
   Mean of sincere_important: 0.17
   Mean of intelligence_important: 0.2
   Mean of funny_important: 0.17
   Mean of ambition_important: 0.11
   Mean of shared_interests_important: 0.12
   Mean of pref_o_attractive: 0.22
   Mean of pref_o_sincere: 0.17
   Mean of pref_o_intelligence: 0.2
   Mean of pref_o_funny: 0.17
   Mean of pref_o_ambitious: 0.11
   Mean of pref_o_shared_interests: 0.12

2. Visualizing interesting trends in data:

(i) python 2_1.py dating.csv

Output of 2_1.py

Mean values for six attributes in female dataset: 0.1796307098937738 0.18339475722085813 0.2091357042235831 0.17278176768808653 0.13008502285557544 0.12497203811812302

Mean values for six attributes in male dataset: 0.26117173247332454 0.1658662081727771 0.1984411430404305 0.17613518306570297 0.0872587409410902 0.11112699230667464

Barplot:

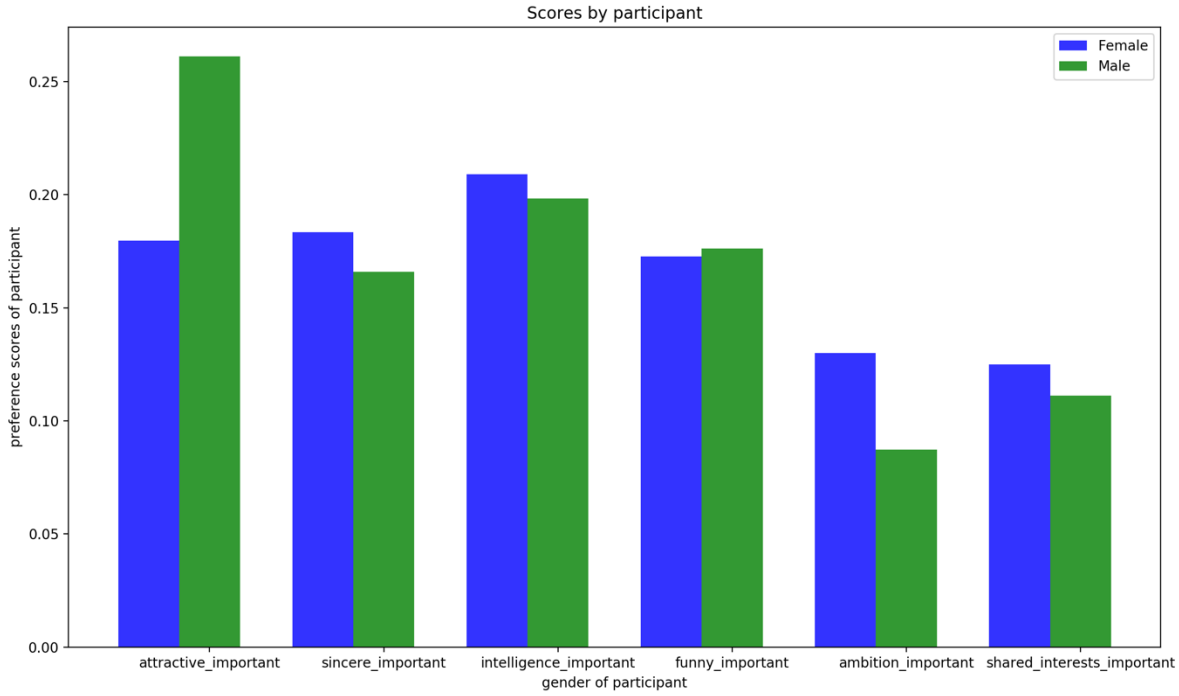

Figure1. Scores by participant

This barplot (Fig. 1) represents how females and males values the six attributes i.e. attractiveness, sincerity, intelligence, funny, ambitious, and shared interests in their romantic partners differently. The importance of to be funny is preferred almost equally by both females and males.

Males favor attractiveness more among 6 attributes in their romantic partners.

Females prefer intelligence more than other 5 attributes in their romantic partners.

(ii) python 2_2.py dating.csv

Output of 2_2.py

Number of distinct values for attractive_partner attribute: 17
Number of distinct values for sincere_partner attribute: 13
Number of distinct values for intelligence_parter attribute: 17
Number of distinct values for funny_partner attribute: 16
Number of distinct values for ambition_partner attribute: 14
Number of distinct values for shared_interests_partner attribute: 15
success_rate_attractive partner: 0.858267716535
success_rate_sincere : 0.544368600683
success_rate_intelligence: 0.523297491039
success_rate_funny: 0.732087227414
success_rate_ambition: 0.525059665871
success_rate_shared_interests: 0.790960451977

Scatter Plot Attractive Partner:

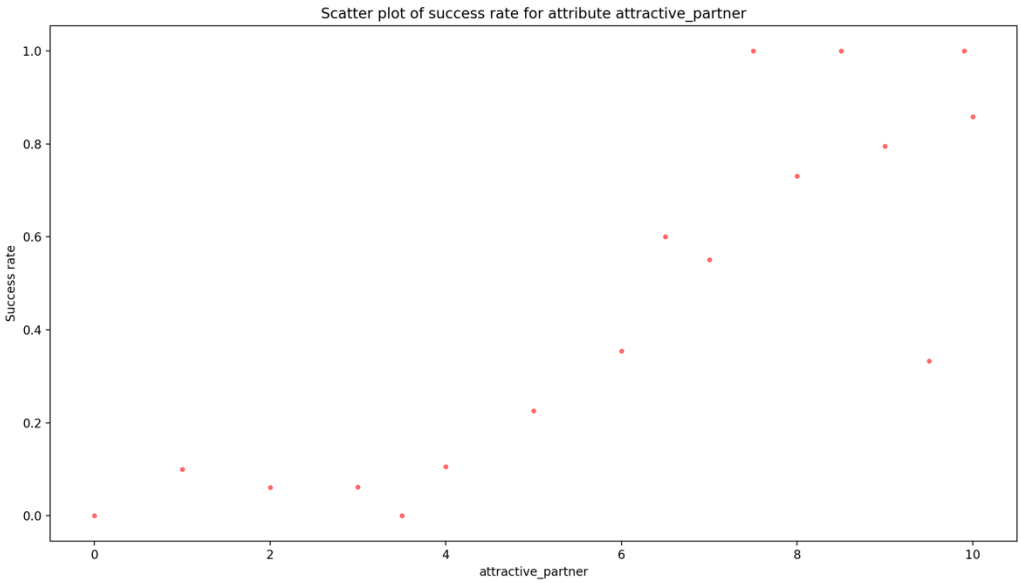

Figure2: Scatter plot of success rate for attribute attractive_partner

From the scatter plot of success rate for attribute attractive_partner (Fig. 2), we can observe that success rate increases for the values > 4. For values 9.9, 8.5 and 7.5, success rate of attractive partner is 100%. For values 0 and 3.5 the success rate is 0%.

Scatter Plot Sincere Partner:

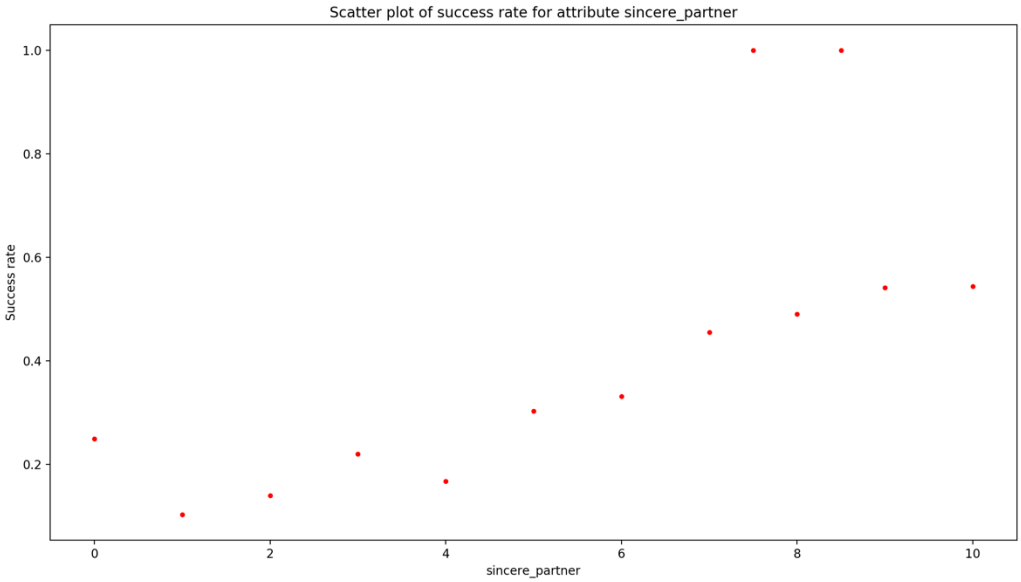

Figure 3: Scatter plot of success rate for attribute sincere_partner

Sincere partners having values 7.5 and 8.5 have 100% success rate (Fig. 3).

Scatter Plot Intelligent Partner:

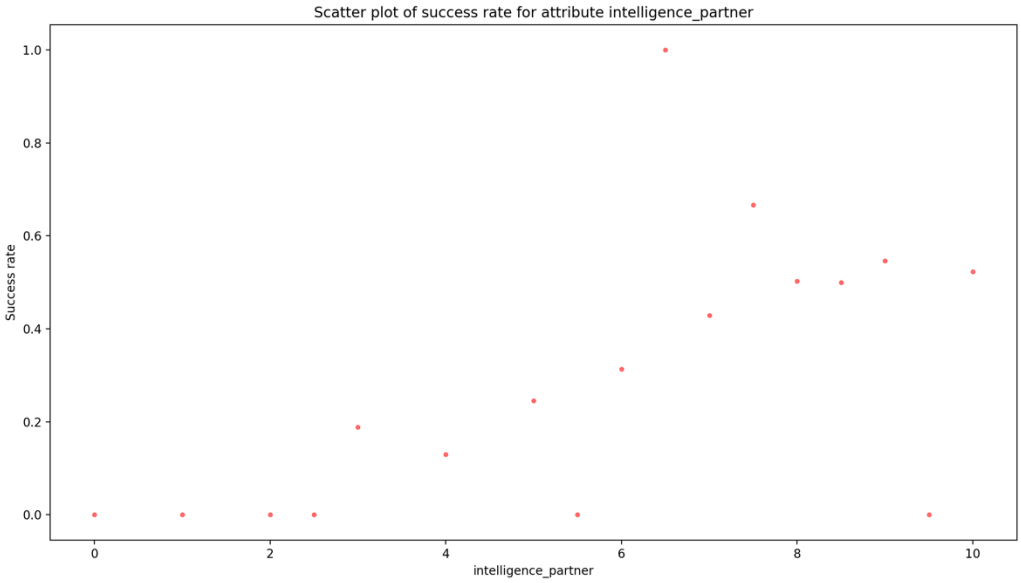

Figure 4: Scatter plot of success rate for attribute intelligence_partner

Fig. 4 is interesting because we can observe 0% success rate for six different values including 0, 1, 2, 2.5, 5.5, and 9.5.

Scatter Plot Funny Partner:

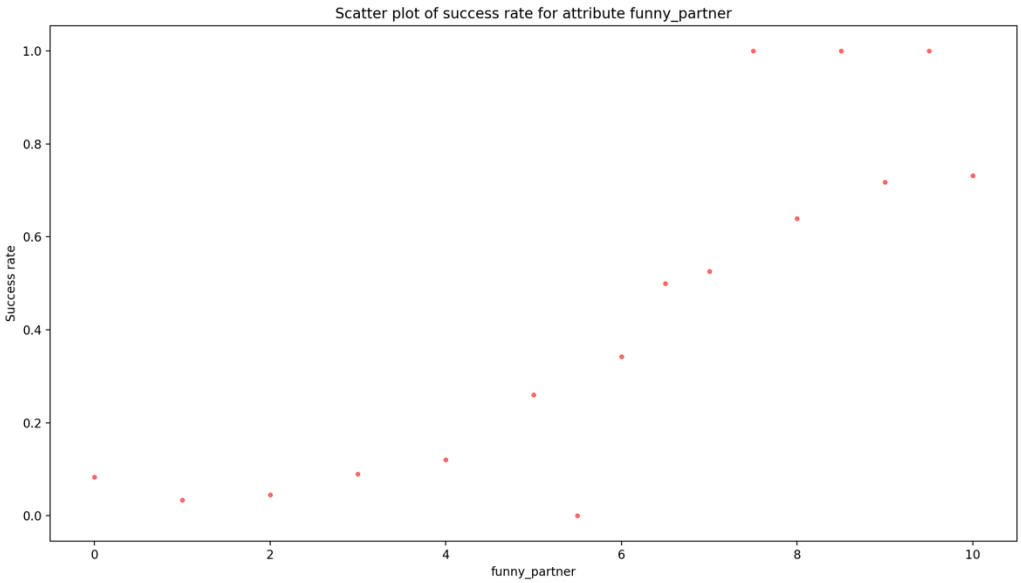

Figure 5: Scatter plot of success rate for attribute funny_partner

In case of funny partner, success rate increases for value from 1-10 except for value 5.5.

Scatter Plot Ambitious Partner:

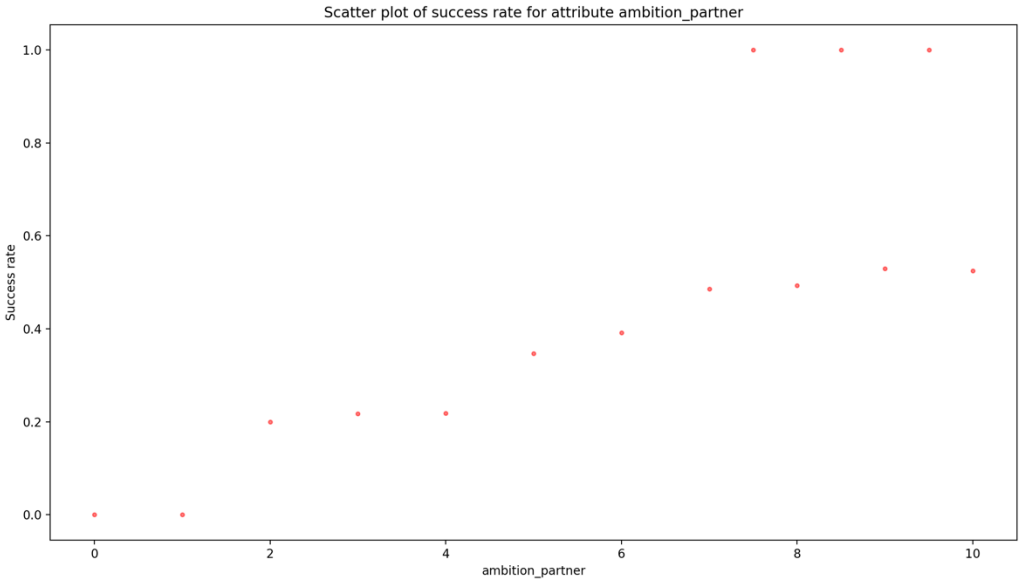

Figure 6: Scatter plot of success rate for attribute ambition_partner

From Fig. 6 we can say that for values 0 and 1 the success rate is 0%. Success rate increases for value > 1. Success rate becomes steady state for value 2-4. For values 9.5, 7.5, 8.5 the success rate of ambitious partner is 100%.

Scatter Plot Shared_Interest Partner:

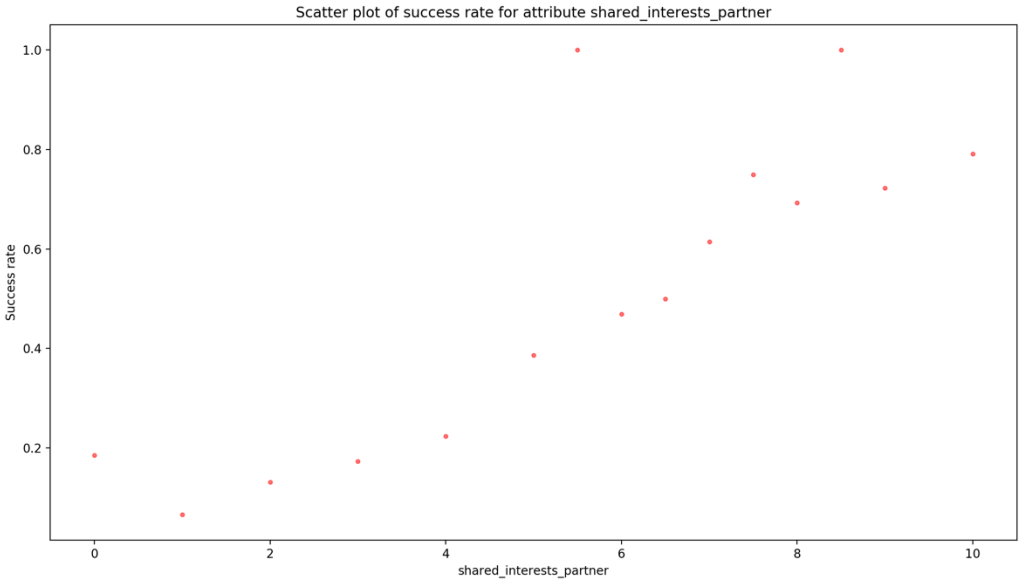

Figure 7: Scatter plot of success rate for attribute shared_interests_partner

Success rate increases sharply from values from 2 to 5.5 (Fig. 7). Overall shared interest partner has incremental success rate.

3. Convert continuous attributes to categorical attributes
   python discretize.py dating.csv dating-binned.csv
   Output of discretize.py

```
age :  [3032, 3390, 297, 20, 5]
age_o :  [2990, 3362, 371, 16, 5]
importance_same_race :  [2980, 1213, 977, 1013, 561]
importance_same_religion :  [3203, 1188, 1110, 742, 501]
pref_o_attractive :  [4333, 1987, 344, 51, 29]
pref_o_sincere :  [1416, 4378, 865, 79, 6]
pref_o_intelligence :  [666, 3935, 1873, 189, 81]
pref_o_funny :  [1255, 4361, 1048, 55, 25]
pref_o_ambitious :  [1963, 2352, 2365, 42, 22]
pref_o_shared_interests :  [1506, 2068, 1981, 1042, 147]
attractive_important :  [4323, 2017, 328, 57, 19]
sincere_important :  [546, 2954, 2782, 377, 85]
intelligence_important :  [630, 3976, 1861, 210, 67]
funny_important :  [1282, 4306, 1070, 58, 28]
ambition_important :  [1913, 2373, 2388, 49, 21]
shared_interests_important :  [1464, 2197, 1950, 1007, 126]
attractive :  [131, 726, 899, 4122, 866]
sincere :  [57, 228, 352, 2715, 3392]
intelligence :  [127, 409, 732, 3190, 2286]
funny :  [19, 74, 1000, 2338, 3313]
ambition :  [225, 697, 559, 2876, 2387]
attractive_partner :  [284, 948, 2418, 2390, 704]
sincere_partner :  [94, 353, 1627, 3282, 1388]
intelligence_parter :  [36, 193, 1509, 3509, 1497]
funny_partner :  [279, 733, 2296, 2600, 836]
ambition_partner :  [119, 473, 2258, 2804, 1090]
shared_interests_partner :  [701, 1269, 2536, 1774, 464]
sports :  [650, 961, 1369, 2077, 1687]
tvsports :  [2151, 1292, 1233, 1383, 685]
exercise :  [619, 952, 1775, 2115, 1283]
dining :  [39, 172, 1118, 2797, 2618]
museums :  [117, 732, 1417, 2737, 1741]
art :  [224, 946, 1557, 2500, 1517]
hiking :  [963, 1386, 1575, 1855, 965]
gaming :  [2565, 1522, 1435, 979, 243]
clubbing :  [912, 1068, 1668, 2193, 903]
reading :  [131, 398, 1071, 2317, 2827]
tv :  [1188, 1216, 1999, 1642, 699]
```

theater : [288, 811, 1585, 2300, 1760]
movies : [144, 462, 530, 2783, 2825]
concerts : [222, 777, 1752, 2282, 1711]
music : [62, 196, 1106, 2583, 2797]
shopping : [1093, 1098, 1709, 1643, 1201]
yoga : [2285, 1392, 1369, 1056, 642]
interests_correlate : [75, 985, 2312, 2597, 775]
expected_happy_with_sd_people : [321, 1262, 3292, 1596, 273]
like : [273, 865, 2539, 2560, 507]

4. Training-Test Split
   python split.py dating-binned.csv trainingSet.csv testSet.csv

5. Implement a Naïve Bayes Classifier
   1) python 5_1.py trainingSet.csv testSet.csv
      Output of 5_1.py:

      Training Accuracy: 75.06%
      Testing Accuracy: 72.63%

   2) python 5_2.py dating.csv
      Output of 5_2.py:

```
Bin size:  2
Training Accuracy: 72.77%
Testing Accuracy: 70.33%
Bin size:  5
Training Accuracy: 75.06%
Testing Accuracy: 72.63%
Bin size:  10
Training Accuracy: 75.36%
Testing Accuracy: 72.55%
Bin size:  50
Training Accuracy: 75.08%
Testing Accuracy: 72.55%
Bin size:  100
Training Accuracy: 75.06%
Testing Accuracy: 72.33%
Bin size:  200
Training Accuracy: 75.06%
Testing Accuracy: 72.33%
```

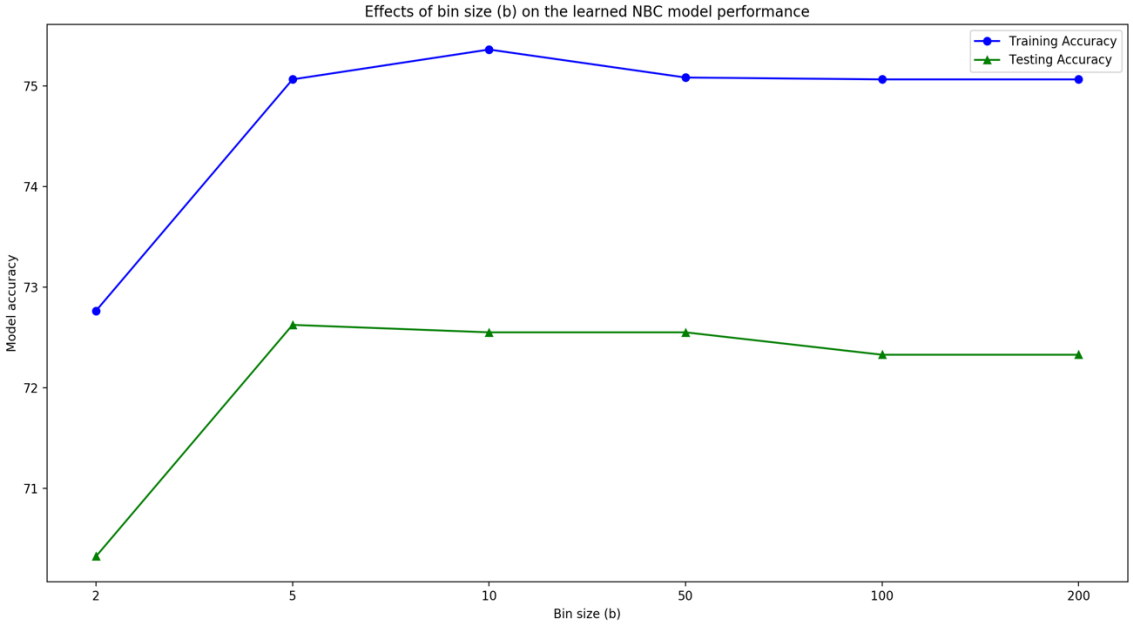

Figure 8: Effects of bin size (b) on the learned NBC model performance

Both training and testing accuracy increase when bin size is increased from 2 to 5. As we discretized the continuous attributes and most of attributes (47 among 54) have value from 0-10, so 5 equal sized bins are the better choice for binning. For bin size 10, 50, 100 and 200 the values of these 47 categorical attributes fell in one bin. So, the accuracy for both cases did not change that much. However, training accuracy is larger than testing accuracy regardless of bin size (Fig. 8).

3) python 5_3.py trainingSet.csv testSet.csv
   Output of 5_3.py:

   f: 0.01
   Training Accuracy: 86.79%
   Testing Accuracy: 69.36%
   f: 0.1
   Training Accuracy: 77.18%
   Testing Accuracy: 71.96%
   f: 0.2
   Training Accuracy: 74.21%
   Testing Accuracy: 72.18%
   f: 0.5
   Training Accuracy: 76.16%
   Testing Accuracy: 72.11%
   f: 0.6
   Training Accuracy: 75.65%

Testing Accuracy: 72.33%
f: 0.75
Training Accuracy: 74.76%
Testing Accuracy: 72.11%
f: 0.9
Training Accuracy: 74.91%
Testing Accuracy: 72.4%
f: 1
Training Accuracy: 75.06%
Testing Accuracy: 72.63%

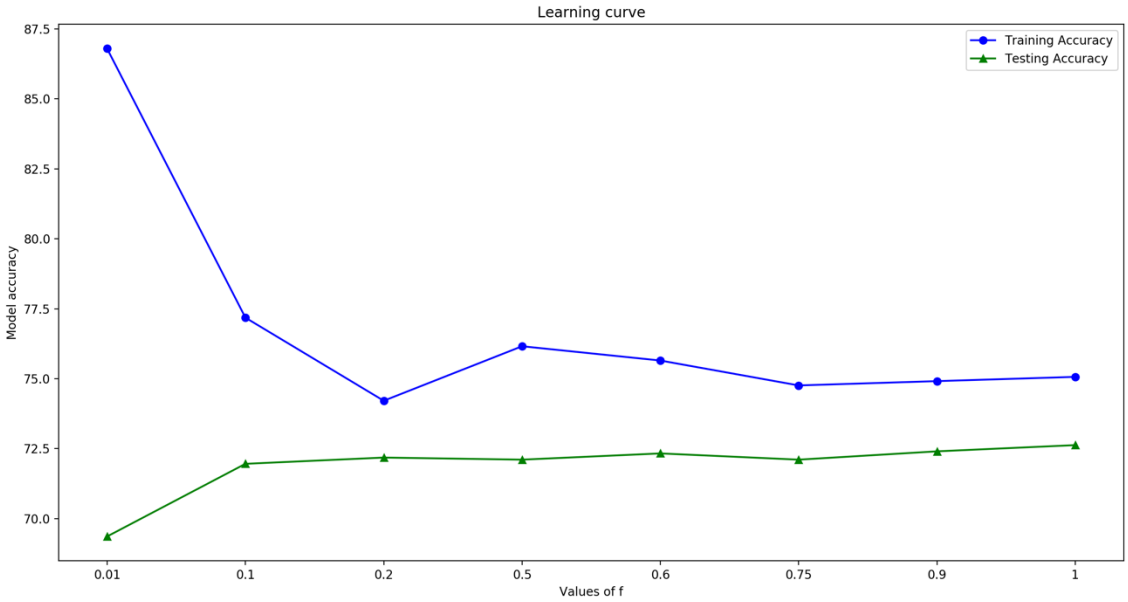

Figure 9: Effects of randomly sample a fraction of training dataset on accuracy

In the plot randomly sample a fraction of training dataset vs accuracy (Fig. 9), training accuracy is higher (86.79%) when there is approximately 1% of training data in the sample training set. We can see the lowest testing values (69.36%) for f = 0.01.
In the Fig. 9 we can see the testing accuracy remains almost steady for f = {0.1, 0.2, 0.5, 0.6, 0.75, 0.9, 1}.
For f =1, meaning we have all training data in our sample training data. Here, training data is 80% (trainingSet.csv file) and testing data is 20% (testSet.csv file). Testing accuracy is the highest (72.63%) among all other testing accuracy in this case. We got the both training and testing accuracy same as question 5. 1) (5_1.py).
However, training accuracy is larger than testing accuracy regardless of values of f.