# p2

August 7, 2018

## 1 DMG2 Assignment Problem 2

*Purity, Entropy, Information Gain*

In [17]:
```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('white')
```

In [18]:
```
DATA_DIR = '/home/jishnu/Documents/ISB/Term3/dmg2/assignments/hw_assignment1/dmg2/data
train = pd.read_csv(os.path.join(DATA_DIR,'train.csv'),usecols=['V{0}'.format(i) for
test = pd.read_csv(os.path.join(DATA_DIR,'test.csv'),usecols=['V{0}'.format(i) for i

train.columns
```

Out[18]:
```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11',
       'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21',
       'V22', 'V23'],
      dtype='object')
```

In [19]:
```
train.head()
```

Out[19]:

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | ... | V14 | V15 | V16 | V17 | V18 | V19 | V20 | V21 | V22 | V23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | p | x | s | n | t | p | f | c | n | k | ... | s | w | w | p | w | o | p | k | s | u |
| 1 | e | x | s | y | t | a | f | c | b | k | ... | s | w | w | p | w | o | p | n | n | g |
| 2 | e | x | s | g | f | n | f | w | b | k | ... | s | w | w | p | w | o | e | n | a | g |
| 3 | p | x | y | w | t | p | f | c | n | p | ... | s | w | w | p | w | o | p | k | v | g |
| 4 | e | x | y | y | t | a | f | c | b | n | ... | s | w | w | p | w | o | p | k | s | m |

[5 rows x 23 columns]

In [20]:
```
#for col_no in range(1,24):
    #print('V{0}'.format(col_no))
#    train['V{0}'.format(col_no)] =
train_cat = train.astype('category')
train_cat.dtypes
```

```
Out[20]: V1     category
         V2     category
         V3     category
         V4     category
         V5     category
         V6     category
         V7     category
         V8     category
         V9     category
         V10    category
         V11    category
         V12    category
         V13    category
         V14    category
         V15    category
         V16    category
         V17    category
         V18    category
         V19    category
         V20    category
         V21    category
         V22    category
         V23    category
         dtype: object
```

```
In [21]: train_cat.describe().T
```

```
Out[21]:     count unique top  freq
         V1   4907      2   e  2535
         V2   4907      6   x  2198
         V3   4907      4   y  1998
         V4   4907     10   n  1372
         V5   4907      2   f  2862
         V6   4907      9   n  2148
         V7   4907      2   f  4772
         V8   4907      2   c  4135
         V9   4907      2   b  3419
         V10  4907     12   b  1040
         V11  4907      2   t  2783
         V12  4907      4   b  3820
         V13  4907      4   s  3098
         V14  4907      4   s  2966
         V15  4907      9   w  2651
         V16  4907      9   w  2595
         V17  4907      1   p  4907
         V18  4907      4   w  4781
         V19  4907      3   o  4506
         V20  4907      5   p  2411
```

```
V21  4907      9   w  1449
V22  4907      6   v  2440
V23  4907      7   d  1938
```

```
In [22]: pd.DataFrame(train_cat['V1'].value_counts()).reset_index()
```

```
Out[22]:   index    V1
0      e  2535
1      p  2372
```

```
In [25]: v2_grouped = train_cat.groupby(by='V2')

         v2_v1_df = pd.DataFrame(v2_grouped.V1.value_counts()).rename(columns={'V1':'count'}).r

         v2_v1_df
```

```
Out[25]:   V2 V1  count
0  b  e    242
1  b  p     26
2  c  p      3
3  f  e    989
4  f  p    936
5  k  p    361
6  k  e    131
7  s  e     21
8  x  e   1152
9  x  p   1046
```

```
In [1]: 242/(242+26)
```

```
Out[1]: 0.9029850746268657
```

```
In [68]: max_sum = 0
         for sub_class in v2_v1_df['V2'].unique():
             e_count = v2_v1_df.loc[(v2_v1_df['V2'] == sub_class) & (v2_v1_df['V1'] == 'e')]['
             p_count = v2_v1_df.loc[(v2_v1_df['V2'] == sub_class) & (v2_v1_df['V1'] == 'p')]['
             try:
                 e_count = int(e_count)
             except:
                 e_count = 0
             try:
                 p_count = int(p_count)
             except:
                 p_count = 0
             print(e_count,p_count)
             max_sum += np.max([e_count,p_count])
             #print(np.max([e_count,p_count]))
         print(max_sum/4907)
```

```
242 26
0 3
989 936
131 361
21 0
1152 1046
0.564092113308
```

```
In [63]: v2_grouped.count()
```

```
Out[63]:       V1    V3    V4    V5    V6    V7    V8    V9   V10   V11  ...    V14  \
         V2                                                              ...
         b    268   268   268   268   268   268   268   268   268   268  ...    268
         c      3     3     3     3     3     3     3     3     3     3  ...      3
         f   1925  1925  1925  1925  1925  1925  1925  1925  1925  1925  ...   1925
         k    492   492   492   492   492   492   492   492   492   492  ...    492
         s     21    21    21    21    21    21    21    21    21    21  ...     21
         x   2198  2198  2198  2198  2198  2198  2198  2198  2198  2198  ...   2198

              V15   V16   V17   V18   V19   V20   V21   V22   V23
         V2
         b    268   268   268   268   268   268   268   268   268
         c      3     3     3     3     3     3     3     3     3
         f   1925  1925  1925  1925  1925  1925  1925  1925  1925
         k    492   492   492   492   492   492   492   492   492
         s     21    21    21    21    21    21    21    21    21
         x   2198  2198  2198  2198  2198  2198  2198  2198  2198

         [6 rows x 22 columns]
```

```
In [69]: np.sum(pd.DataFrame(train_cat['V1'].value_counts()).reset_index()['V1'])
```

```
Out[69]: 4907
```

```
In [104]: purity_table = pd.DataFrame(columns=['feature','accuracy','gini_index','1-entropy'])
          record_count = np.sum(pd.DataFrame(train_cat['V1'].value_counts()).reset_index()['V1
          for col_no in range(2,24):
              feature = 'V{0}'.format(col_no)
              feature_grouped = train_cat.groupby(by=feature)
              feature_v1_df = pd.DataFrame(feature_grouped.V1.value_counts()).rename(columns={
              max_sum,gini_purity,entropy = 0,0,0
              for sub_class in feature_v1_df[feature].unique():
                  e_count = feature_v1_df.loc[(feature_v1_df[feature] == sub_class) & (feature_
                  p_count = feature_v1_df.loc[(feature_v1_df[feature] == sub_class) & (feature_
                  try:
                      e_count = int(e_count)
                  except:
                      e_count = 0
```

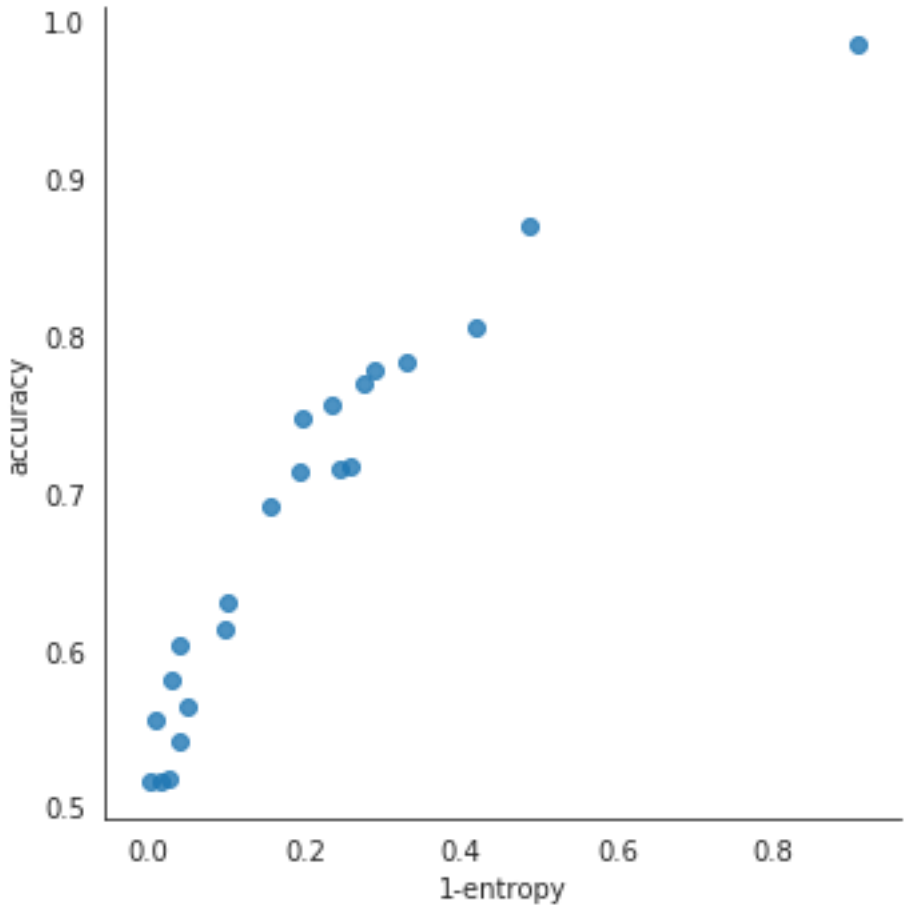
```
                try:
                    p_count = int(p_count)
                except:
                    p_count = 0
                max_sum += np.max([e_count,p_count])
                gini_purity += ((e_count/(e_count+p_count))**2 + (p_count/(e_count+p_count))
                e_prob = e_count / (e_count + p_count)
                p_prob = p_count / (e_count + p_count)
                if e_prob == 0.0:
                    entropy += ( p_prob * np.log2(1 / p_prob) ) * (e_count + p_count)
                elif p_prob == 0.0:
                    entropy += ( e_prob * np.log2(1 / e_prob) ) * (e_count + p_count)
                else:
                    entropy += ( e_prob * np.log2(1 / e_prob) + p_prob * np.log2(1 / p_prob)

            accuracy = np.round(max_sum / record_count, 4)
            gini_purity = np.round(gini_purity / record_count, 4)
            entropy = np.round(entropy / record_count, 4)
            purity_table = purity_table.append({'feature' : feature, 'accuracy' : accuracy,
        purity_table
```

Out[104]:

| | feature | accuracy | gini_index | 1-entropy |
|---|---|---|---|---|
| 0 | V2 | 0.5641 | 0.5318 | 0.0518 |
| 1 | V3 | 0.5816 | 0.5199 | 0.0293 |
| 2 | V4 | 0.6028 | 0.5268 | 0.0402 |
| 3 | V5 | 0.7473 | 0.6292 | 0.1979 |
| 4 | V6 | 0.9851 | 0.9713 | 0.9063 |
| 5 | V7 | 0.5166 | 0.5087 | 0.0144 |
| 6 | V8 | 0.6138 | 0.5597 | 0.0992 |
| 7 | V9 | 0.7559 | 0.6477 | 0.2347 |
| 8 | V10 | 0.8046 | 0.7334 | 0.4204 |
| 9 | V11 | 0.5561 | 0.5065 | 0.0094 |
| 10 | V12 | 0.6303 | 0.5603 | 0.1000 |
| 11 | V13 | 0.7771 | 0.6751 | 0.2877 |
| 12 | V14 | 0.7687 | 0.6674 | 0.2749 |
| 13 | V15 | 0.7171 | 0.6404 | 0.2598 |
| 14 | V16 | 0.7149 | 0.6347 | 0.2460 |
| 15 | V17 | 0.5166 | 0.5006 | 0.0008 |
| 16 | V18 | 0.5176 | 0.5129 | 0.0257 |
| 17 | V19 | 0.5417 | 0.5249 | 0.0415 |
| 18 | V20 | 0.7824 | 0.6886 | 0.3295 |
| 19 | V21 | 0.8686 | 0.7853 | 0.4867 |
| 20 | V22 | 0.7131 | 0.6129 | 0.1930 |
| 21 | V23 | 0.6913 | 0.5976 | 0.1578 |

### 1.1 Plotting Accuracy vs 1-Entropy

```
In [106]: sns.lmplot(x='1-entropy',y='accuracy',data=purity_table,fit_reg=False)
          plt.show();
```




It is observed that as the entropy decreases, the accuracy increases as the purity increases.

```
In [107]: purity_table.loc[purity_table['accuracy'] == np.max(purity_table['accuracy'])]

Out[107]:   feature  accuracy  gini_index  1-entropy
          4      V6    0.9851      0.9713     0.9063

In [108]: purity_table.loc[purity_table['1-entropy'] == np.max(purity_table['1-entropy'])]

Out[108]:   feature  accuracy  gini_index  1-entropy
          4      V6    0.9851      0.9713     0.9063

In [110]: purity_table.loc[purity_table['gini_index'] == np.max(purity_table['gini_index'])]

Out[110]:   feature  accuracy  gini_index  1-entropy
          4      V6    0.9851      0.9713     0.9063
```

### 1.2 Google Form Answers

- Feature with highest accuracy : V6
- Accuracy of feature with highest accuracy : 0.9851
- Feature with lowest entropy : V6
- Lowest Entropy : 0.0937
- Feature with highest Gini Index : V6
- Highest Gini Index across all features : 0.9713