Project

June 8, 2020

0.1 Imports

```
In [74]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import graphviz
         from sklearn.svm import SVC
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.model_selection import GridSearchCV
         from sklearn.preprocessing import StandardScaler
         from sklearn.pipeline import make_pipeline
         from sklearn.ensemble import RandomForestClassifier
         from sklearn import tree
         from sklearn import preprocessing
         from sklearn.tree import DecisionTreeClassifier
```

0.2 Path

```
In [2]: pwd

Out[2]: '/Users/ramosem/Documents/SyracuseUniversity/3rd_Quarter/IST707/Project'

In [3]: path = '/Users/ramosem/Documents/SyracuseUniversity/3rd_Quarter/IST707/Project/'
```

0.3 Read Data

```
In [4]: df = pd.read_csv(path + 'kaggle_Interests_group.csv')

In [5]: df.head()

Out[5]:   group  grand_tot_interests  interest1  interest2  interest3  interest4  \
        0     C                   17        NaN        NaN        NaN        NaN
        1     C                   43        1.0        NaN        NaN        NaN
        2     C                   27        NaN        NaN        NaN        NaN
        3     C                   34        NaN        NaN        NaN        NaN
        4     C                   36        NaN        NaN        NaN        NaN

           interest5  interest6  interest7  interest8  ...  interest208  interest209  \
0         NaN        NaN        NaN        NaN  ...            NaN           NaN
1         1.0        NaN        NaN        NaN  ...            NaN           NaN
2         NaN        NaN        NaN        NaN  ...            NaN           NaN
3         NaN        NaN        NaN        NaN  ...            NaN           NaN
4         1.0        NaN        NaN        NaN  ...            NaN           NaN

   interest210  interest211  interest212  interest213  interest214  \
0          NaN          NaN          NaN          NaN          NaN
1          1.0          NaN          NaN          NaN          NaN
2          1.0          NaN          NaN          NaN          NaN
3          NaN          1.0          NaN          NaN          NaN
4          1.0          NaN          NaN          NaN          NaN

   interest215  interest216  interest217
0          NaN          NaN          NaN
1          1.0          1.0          NaN
2          1.0          1.0          NaN
3          1.0          1.0          NaN
4          1.0          1.0          NaN

[5 rows x 219 columns]
```

```
In [6]: len(df)
```

```
Out[6]: 6340
```

1 Preprocessing

```
In [7]: cat = df['group'].value_counts().reset_index(drop=False)
```

```
In [8]: cat.columns= ['group', 'count']
```

```
In [9]: cat['GroupPerc'] = cat['count']/cat['count'].sum()
```

```
In [10]: cat
```

```
Out[10]:   group  count  GroupPerc
         0     I   1809   0.285331
         1     P   1731   0.273028
         2     C   1725   0.272082
         3     R   1075   0.169558
```

```
In [11]: df.describe()
```

```
Out[11]:        grand_tot_interests  interest1  interest2  interest3  interest4  \
         count          6340.000000      993.0        1.0       35.0       25.0
         mean             37.312303        1.0        1.0        1.0        1.0
         std              15.729872        0.0        NaN        0.0        0.0
         min               1.000000        1.0        1.0        1.0        1.0
```

```
25%              28.000000        1.0        1.0        1.0        1.0
50%              39.000000        1.0        1.0        1.0        1.0
75%              48.000000        1.0        1.0        1.0        1.0
max             104.000000        1.0        1.0        1.0        1.0

        interest5    interest6  interest7  interest8   interest9  ...  \
count  798.000000  3394.000000        1.0       93.0  333.000000  ...
mean     1.001253     1.000589        1.0        1.0    1.003003  ...
std      0.035400     0.024271        NaN        0.0    0.054800  ...
min      1.000000     1.000000        1.0        1.0    1.000000  ...
25%      1.000000     1.000000        1.0        1.0    1.000000  ...
50%      1.000000     1.000000        1.0        1.0    1.000000  ...
75%      1.000000     1.000000        1.0        1.0    1.000000  ...
max      2.000000     2.000000        1.0        1.0    2.000000  ...

       interest208  interest209  interest210  interest211  interest212  \
count   118.000000        110.0  5037.000000  2474.000000   877.000000
mean      1.008475          1.0     1.000596     1.000808     1.002281
std       0.092057          0.0     0.024400     0.028427     0.047727
min       1.000000          1.0     1.000000     1.000000     1.000000
25%       1.000000          1.0     1.000000     1.000000     1.000000
50%       1.000000          1.0     1.000000     1.000000     1.000000
75%       1.000000          1.0     1.000000     1.000000     1.000000
max       2.000000          1.0     2.000000     2.000000     2.000000

       interest213  interest214  interest215  interest216  interest217
count          2.0         72.0  4943.000000  4058.000000        147.0
mean           1.0          1.0     1.000202     1.000246          1.0
std            0.0          0.0     0.014223     0.015698          0.0
min            1.0          1.0     1.000000     1.000000          1.0
25%            1.0          1.0     1.000000     1.000000          1.0
50%            1.0          1.0     1.000000     1.000000          1.0
75%            1.0          1.0     1.000000     1.000000          1.0
max            1.0          1.0     2.000000     2.000000          1.0

[8 rows x 218 columns]
```

Columns

```
In [12]: yParm = ['group']
         yParmStr = 'group'
         xParm = df.columns[2:]
```

1.0.1 Get rid of the ones with a small and big amounts of positive responses

```
In [13]: goodCols = []
         for col in xParm:
             if (df[col].count() < 10):
```

```
         #         print(col)
                 continue
             elif (df[col].count() > 5990):
                 print(col)
                 continue
             else:
                 goodCols.append(col)
```

```
interest162
interest183
```

```
In [14]: len(goodCols)
```

```
Out[14]: 175
```

```
In [15]: len(xParm) - len(goodCols)
```

```
Out[15]: 42
```

```
In [16]: xParm = goodCols
```

1.0.2 Change twos to ones

```
In [17]: for col in xParm:
             df[col] = [x if pd.isnull(x) else 1 for x in df[col].values]
```

1.0.3 Get rid off of well distributed interests among all categories.

```
In [18]: catCount = df[xParm + yParm].groupby(yParmStr).count().transpose()
```

```
In [19]: catCount['Total'] = catCount[['C', 'I', 'P', 'R']].sum(axis=1)
```

```
In [20]: for col in ['C', 'I', 'P', 'R']:
             catCount[col+'_perc'] = catCount[col]*100/cat.loc[cat['group']==col]['count'].valu
```

```
In [21]: catCount['TotalPerc'] = catCount[['C_perc', 'I_perc', 'P_perc', 'R_perc']].sum(axis=1
```

```
In [22]: for col in ['C', 'I', 'P', 'R']:
             catCount[col+'_norm'] = catCount[col+'_perc']*100/catCount['TotalPerc']
```

```
In [23]: badRow = []
         goodRow = []
         for idx, row in catCount.iterrows():
             if (row['C_norm'] > 30) | (row['C_norm'] < 20):
                 goodRow.append(idx)
                 continue
             elif (row['I_norm'] > 30) | (row['I_norm'] < 20):
                 goodRow.append(idx)
                 continue
```

```
    elif (row['P_norm'] > 30) | (row['P_norm'] < 20):
        goodRow.append(idx)
        continue
    elif (row['R_norm'] > 30) | (row['R_norm'] < 20):
        goodRow.append(idx)
        continue
    else:
        badRow.append(idx)
```

```
In [24]: len(badRow)
```

```
Out[24]: 51
```

```
In [25]: catCount.loc[catCount.index.isin(badRow)].head()
```

```
Out[25]: group            C     I     P    R  Total     C_perc     I_perc     P_perc  \
         interest6      859  1032   888  615   3394  49.797101  57.048093  51.299827
         interest12    1211  1235  1248  825   4519  70.202899  68.269762  72.097054
         interest14      86   111   100   68    365   4.985507   6.135987   5.777008
         interest15     455   545   559  333   1892  26.376812  30.127142  32.293472
         interest16    1218  1219  1251  823   4511  70.608696  67.385296  72.270364

         group           R_perc  TotalPerc     C_norm     I_norm     P_norm     R_norm
         interest6    57.209302 215.354323  23.123335  26.490340  23.821127  26.565198
         interest12   76.744186 287.313901  24.434216  23.761385  25.093479  26.710920
         interest14    6.325581  23.224083  21.466971  26.420792  24.875073  27.237163
         interest15   30.976744 119.774170  22.022120  25.153288  26.961967  25.862625
         interest16   76.558140 286.822495  24.617559  23.493728  25.196895  26.691818
```

```
In [26]: xParm = goodRow
```

```
In [27]: len(xParm)
```

```
Out[27]: 124
```

1.0.4 Get rid of Anomalous Individuals

```
In [28]: df['grand_tot_interests'] = df[xParm].sum(axis=1)
```

Zero Interests

```
In [29]: df = df.loc[df['grand_tot_interests']!=0].reset_index(drop=True)
```

```
In [30]: len(df)
```

```
Out[30]: 5788
```

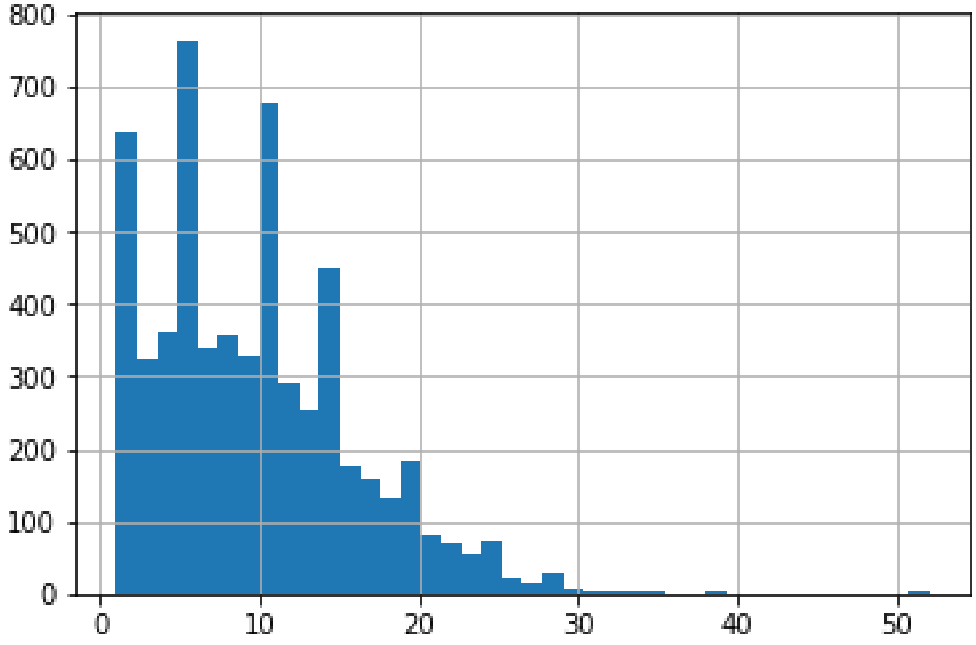
Anomaly in distribution

```
In [31]: df['grand_tot_interests'].describe()
```

```
Out[31]: count    5788.000000
         mean        9.720974
         std         6.236600
         min         1.000000
         25%         5.000000
         50%         9.000000
         75%        14.000000
         max        52.000000
         Name: grand_tot_interests, dtype: float64
```

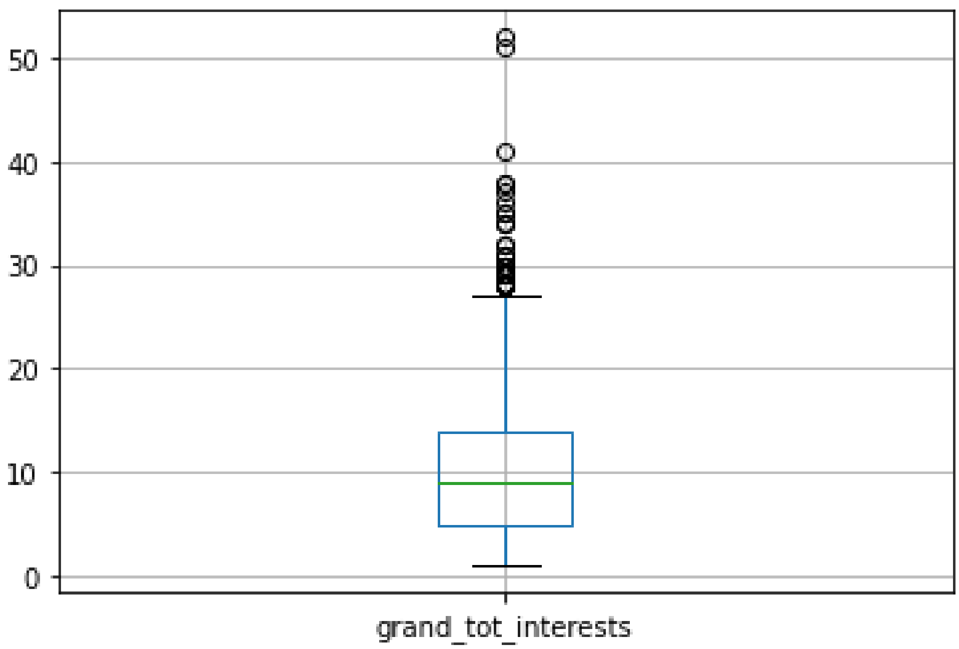
```
In [32]: df['grand_tot_interests'].hist(bins=40)
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x102865358>
```



```
In [33]: df[['grand_tot_interests']].boxplot()
```

```
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1166189b0>
```

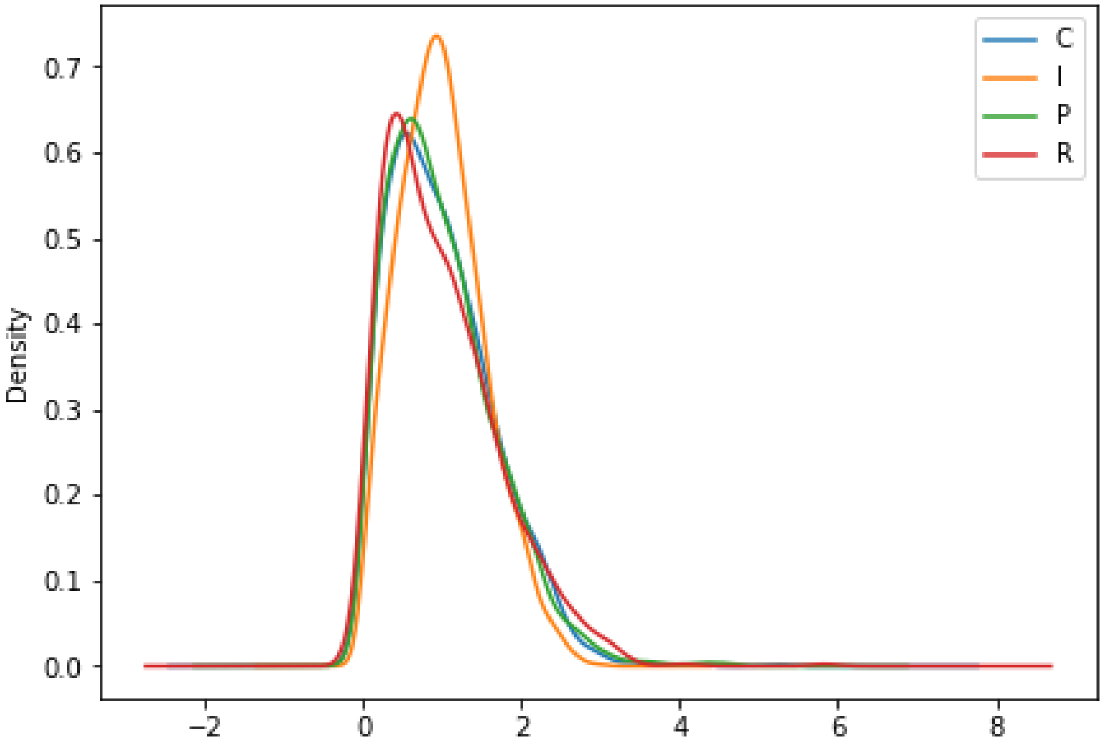


```
In [34]: df['interest9'].value_counts()

Out[34]: 1.0    333
         Name: interest9, dtype: int64

In [35]: df[xParm].describe()

Out[35]:        interest1  interest3  interest4  interest5  interest8  interest9  \
         count      993.0       35.0       25.0      798.0       93.0      333.0
         mean         1.0        1.0        1.0        1.0        1.0        1.0
         std          0.0        0.0        0.0        0.0        0.0        0.0
         min          1.0        1.0        1.0        1.0        1.0        1.0
         25%          1.0        1.0        1.0        1.0        1.0        1.0
         50%          1.0        1.0        1.0        1.0        1.0        1.0
         75%          1.0        1.0        1.0        1.0        1.0        1.0
         max          1.0        1.0        1.0        1.0        1.0        1.0

                interest11  interest13  interest19  interest20  ...  interest201  \
         count       175.0        18.0       141.0       165.0  ...       1671.0
         mean          1.0         1.0         1.0         1.0  ...          1.0
         std           0.0         0.0         0.0         0.0  ...          0.0
         min           1.0         1.0         1.0         1.0  ...          1.0
         25%           1.0         1.0         1.0         1.0  ...          1.0
         50%           1.0         1.0         1.0         1.0  ...          1.0
         75%           1.0         1.0         1.0         1.0  ...          1.0
         max           1.0         1.0         1.0         1.0  ...          1.0
```

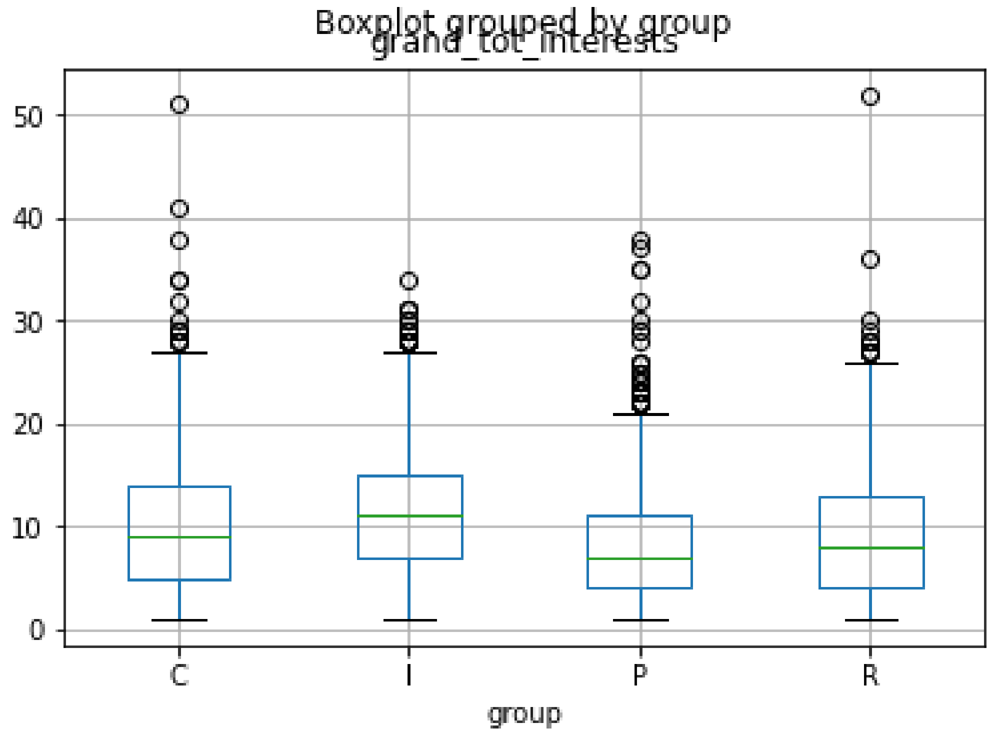
```
       interest203  interest204  interest205  interest208  interest209  \
count         26.0        117.0        134.0        118.0        110.0
mean           1.0          1.0          1.0          1.0          1.0
std            0.0          0.0          0.0          0.0          0.0
min            1.0          1.0          1.0          1.0          1.0
25%            1.0          1.0          1.0          1.0          1.0
50%            1.0          1.0          1.0          1.0          1.0
75%            1.0          1.0          1.0          1.0          1.0
max            1.0          1.0          1.0          1.0          1.0

       interest211  interest212  interest214  interest217
count       2474.0        877.0         72.0        147.0
mean           1.0          1.0          1.0          1.0
std            0.0          0.0          0.0          0.0
min            1.0          1.0          1.0          1.0
25%            1.0          1.0          1.0          1.0
50%            1.0          1.0          1.0          1.0
75%            1.0          1.0          1.0          1.0
max            1.0          1.0          1.0          1.0

[8 rows x 124 columns]
```

```
In [36]: df['Zscore'] = df.groupby('group')['grand_tot_interests'].apply(lambda x: x.div(x.mea

         plt.figure(figsize=(7,5))
         df.groupby('group').Zscore.plot.kde()
         plt.legend()
```

```
Out[36]: <matplotlib.legend.Legend at 0x116724a58>
```



```
In [37]: df['grand_tot_interests']

Out[37]: 0       15.0
         1        9.0
         2        7.0
         3       18.0
         4        4.0
                 ...
         5783    14.0
         5784     9.0
         5785    12.0
         5786    13.0
         5787    26.0
         Name: grand_tot_interests, Length: 5788, dtype: float64

In [38]: df.boxplot(column='grand_tot_interests',by='group')

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x11680dd30>
```



```
In [39]: df = df.loc[df['grand_tot_interests']<28].reset_index(drop=True)
```

```
In [40]: len(df)
```

```
Out[40]: 5737
```

2 Analysis

```
In [45]: for col in xParm:
             df[col] = [0 if pd.isnull(x) else 1 for x in df[col].values]
```

2.1 Decision Tree

```
In [46]: parameters = {'n_estimators':[10, 30, 50, 70], 'criterion':['entropy'],
                       'max_depth':[10, 20, 25]}
```

```
In [47]: rf = GridSearchCV(RandomForestClassifier(), parameters, n_jobs=5, cv=5, verbose=10)
```

```
In [48]: rf = rf.fit(df[xParm].values, df[yParm].astype(str).values)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```

```
[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done   3 tasks      | elapsed:    0.2s
[Parallel(n_jobs=5)]: Done   8 tasks      | elapsed:    0.6s
[Parallel(n_jobs=5)]: Done  15 tasks      | elapsed:    1.2s
[Parallel(n_jobs=5)]: Done  22 tasks      | elapsed:    2.2s
[Parallel(n_jobs=5)]: Done  31 tasks      | elapsed:    3.8s
[Parallel(n_jobs=5)]: Done  40 tasks      | elapsed:    5.0s
[Parallel(n_jobs=5)]: Done  51 tasks      | elapsed:    6.4s
[Parallel(n_jobs=5)]: Done  58 out of  60 | elapsed:    7.7s remaining:    0.3s
[Parallel(n_jobs=5)]: Done  60 out of  60 | elapsed:    7.7s finished
/Users/ramosem/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_search.py:739: Da
  self.best_estimator_.fit(X, y, **fit_params)
```

2.1.1 Check Out the Best Model

```
In [49]: rf_model = rf.best_estimator_
         print (rf.best_score_, rf.best_params_)
```

```
0.5847999173726948 {'criterion': 'entropy', 'max_depth': 25, 'n_estimators': 70}
```

```
In [50]: rf_model
```

```
Out[50]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='entropy', max_depth=25, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=70,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)
```

2.1.2 Prediction

```
In [51]: df['predict_rf'] = rf_model.predict(df[xParm].values)
```

```
In [52]: df[['predict_rf', 'group']].head()
```

```
Out[52]:   predict_rf group
         0          C     C
         1          C     C
         2          C     C
         3          C     C
         4          C     C
```

```
In [53]: df['correct_rf'] = df['predict_rf'] == df['group']
```

```
In [54]: df[['group', 'correct_rf']].groupby(['group', 'correct_rf']).size().unstack(fill_valu
Out[54]: correct_rf group  False  True
         0              C    156  1364
         1              I     97  1588
         2              P     45  1531
         3              R    144   812
```

```
In [55]: df['correct_rf'].value_counts()
```

```
Out[55]: True     5295
         False     442
         Name: correct_rf, dtype: int64
```

```
In [56]: print("Total Correctly Predicted ", len(df.loc[df['correct_rf']])/len(df))
```

```
Total Correctly Predicted  0.9229562489105805
```

2.1.3 Find Splits

```
In [57]: colImp = pd.DataFrame(zip(xParm, rf_model.feature_importances_), columns=['Interest',
```

```
In [116]: xParm = colImp.sort_values('Importance', ascending=False)['Interest'].values[0:30]
```

3 Decision Tree

3.1 Train

```
In [117]: parameters = {'criterion':['entropy'],
                        'max_depth':[5,10,15,20], 'max_features':[5,10,15,20, 30], 'max_leaf_no
```

```
In [118]: dt = GridSearchCV(DecisionTreeClassifier(), parameters, n_jobs=5, cv=5, verbose=10)
```

```
In [119]: dt = dt.fit(df[xParm].values, df[yParm].values)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits


[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Batch computation too fast (0.0414s.) Setting batch_size=2.
[Parallel(n_jobs=5)]: Done   3 tasks      | elapsed:    0.1s
[Parallel(n_jobs=5)]: Done   8 tasks      | elapsed:    0.1s
[Parallel(n_jobs=5)]: Batch computation too fast (0.0584s.) Setting batch_size=4.
[Parallel(n_jobs=5)]: Done  20 tasks      | elapsed:    0.1s
[Parallel(n_jobs=5)]: Batch computation too fast (0.1440s.) Setting batch_size=8.
[Parallel(n_jobs=5)]: Done  38 tasks      | elapsed:    0.3s
[Parallel(n_jobs=5)]: Done  78 tasks      | elapsed:    0.6s
[Parallel(n_jobs=5)]: Done 150 tasks      | elapsed:    0.9s
[Parallel(n_jobs=5)]: Done 238 tasks      | elapsed:    1.6s
[Parallel(n_jobs=5)]: Done 263 tasks      | elapsed:    1.8s
[Parallel(n_jobs=5)]: Done 290 tasks      | elapsed:    1.9s
[Parallel(n_jobs=5)]: Done 300 out of 300 | elapsed:    2.0s finished
```

3.2 Check out Best Model

```
In [120]: dt_model = dt.best_estimator_
          print (dt.best_score_, dt.best_params_)
```

```
0.5600478752327691 {'criterion': 'entropy', 'max_depth': 20, 'max_features': 15, 'max_leaf_node
```

```
In [121]: dt_model
```

```
Out[121]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                                 max_depth=20, max_features=15, max_leaf_nodes=10,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, presort='deprecated',
                                 random_state=None, splitter='best')
```

3.3 Predict

```
In [122]: df['predict_dt'] = dt_model.predict(df[xParm].values)
```

```
In [123]: df[['predict_dt', 'group']].head()
```

```
Out[123]:   predict_dt group
          0          C     C
          1          I     C
          2          C     C
          3          C     C
          4          C     C
```

```
In [124]: df['correct_dt'] = df['predict_dt'] == df['group']
```

```
In [125]: df[['group', 'correct_dt']].groupby(['group', 'correct_dt']).size().unstack(fill_valu
```

```
Out[125]: correct_dt group  False  True
          0              C    690   830
          1              I    561  1124
          2              P    388  1188
          3              R    849   107
```

```
In [126]: df['correct_dt'].value_counts()
```

```
Out[126]: True     3249
          False    2488
          Name: correct_dt, dtype: int64
```

```
In [127]: print("Total Correctly Predicted ", len(df.loc[df['correct_dt']])/len(df))
```

```
Total Correctly Predicted  0.5663238626459822
```

3.4 Plot Desicion Tree

```
In [128]: dot_data = tree.export_graphviz(dt_model, out_file=None)
          graph = graphviz.Source(dot_data)
          graph.render("Personality Decision Tree")

Out[128]: 'Personality Decision Tree.pdf'
```

```
In [129]: dot_data = tree.export_graphviz(dt_model, out_file=None,
                                          feature_names=xParm,
                                          class_names=df['group'].unique(),
                                          filled=True, rounded=True,
                                          special_characters=True)
          graph = graphviz.Source(dot_data)
```

```
In [153]: # plt.figure(figsize=(5,5))
          # graph
```

4 Group By Group in Remaining Parameters

```
In [150]: df[np.append(xParm, yParm)].groupby(yParmStr).sum().transpose().reset_index(drop=Fals

Out[150]: group        index    C     I     P    R
          0      interest201  300  1131    94  112
          1      interest200  892  1455   432  497
          2       interest48  174   935   646  100
          3      interest196  550    48    42   42
          4      interest135  560   122   137   66
          5      interest211  471   792   835  330
          6       interest79  548   780   493  332
          7       interest63  778  1143   773  541
          8      interest112  631   803   571  368
          9      interest164  951  1448  1191  583
```

```
In [151]: df[np.append(xParm, yParm)].groupby(yParmStr).sum().transpose().reset_index(drop=Fals

Out[151]: group        index    C     I    P    R
          10      interest62  738  1030  626  444
          11     interest186   62   225  257   41
          12     interest147  278   394  287  181
          13      interest44  293   481  353  195
          14      interest65  336   383  268  237
          15      interest88  185   100  116  195
          16     interest187  216   224  205  169
          17       interest1  269   324  197  179
          18     interest149  428   578  360  245
          19     interest181  190   348  184  104
```

```
In [152]: df[np.append(xParm, yParm)].groupby(yParmStr).sum().transpose().reset_index(drop=Fals
Out[152]: group          index    C    I    P    R
          20     interest142  424  574  386  258
          21     interest144  347  523  342  216
          22     interest212  181  301  238  116
          23      interest43  217  321  229  144
          24      interest82  286  161  160   92
          25     interest153  133   91   62  128
          26      interest41  194  182  118  130
          27     interest146  139   93  117   84
          28     interest118  127  199  184  100
          29       interest5  296  213  183   76
```