CIND 820 Project - What makes a Board Game Highly Rated?

BY: Francis Semenuk - 500185004

Supervisor: Ceni Babaoglu

```
import pandas as pd
from patsy import dmatrices
import numpy as np
#import os
import matplotlib.pyplot as plt
import statsmodels.discrete.discrete_model as sm
import copy as cp
#import io
import graphviz
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from·sklearn·import·tree,·linear_model,·svm,·datasets,·metrics
from·sklearn.model_selection·import·train_test_split,GridSearchCV,·KFold,·StratifiedKFold,·cr
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
from sklearn.decomposition import PCA
from IPython.display import Image
from typing import Tuple
import pydotplus
#import sys
import seaborn as sns
#sys.path
#sys.path.append("/usr/lib/jvm/java-11-openjdk-amd64/bin/")
#os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64/"


df = pd.read_csv("/content/drive/MyDrive/BoardgameswithExpansions.csv")
## view the first few rows of the data
df.head()
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: Dtyp
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	row_names	game.id	game.type	details.description	details.im
0	1	1	boardgame	Die Macher is a game about seven sequential po...	//cf.geek images.com/images/pic159509
1	2	2	boardgame	Dragonmaster is a trick-taking card game based	//cf.geek images.com/images/pic184174

```
df.describe()
```

	row_names	game.id	details.maxplayers	details.maxplaytime	detai
count	76688.000000	76688.000000	76685.000000	76685.000000	766
mean	42479.366628	81220.433432	5.647467	49.104153	
std	25722.198312	68265.573055	57.767585	351.301987	
min	1.000000	1.000000	0.000000	0.000000	
25%	20242.750000	22008.750000	2.000000	5.000000	
50%	41129.500000	55239.500000	4.000000	30.000000	
75%	64029.250000	141893.500000	6.000000	60.000000	
max	90400.000000	226264.000000	11299.000000	60120.000000	

8 rows × 54 columns

```
df.info()
```

```
 26  stats.averageweight                        76688 non-null  float64
 27  stats.bayesaverage                         76688 non-null  float64
 28  stats.family.abstracts.bayesaverage        867 non-null    float64
 29  stats.family.abstracts.pos                 858 non-null    float64
 30  stats.family.cgs.bayesaverage              266 non-null    float64
 31  stats.family.cgs.pos                       264 non-null    float64
 32  stats.family.childrensgames.bayesaverage   671 non-null    float64
 33  stats.family.childrensgames.pos            670 non-null    float64
 34  stats.family.familygames.bayesaverage      1573 non-null   float64
 35  stats.family.familygames.pos               1558 non-null   float64
 36  stats.family.partygames.bayesaverage       441 non-null    float64
 37  stats.family.partygames.pos                436 non-null    float64
 38  stats.family.strategygames.bayesaverage    1651 non-null   float64
 39  stats.family.strategygames.pos             1631 non-null   float64
 40  stats.family.thematic.bayesaverage         879 non-null    float64
 41  stats.family.thematic.pos                  870 non-null    float64
 42  stats.family.wargames.bayesaverage         2562 non-null   float64
```

```
 43  stats.family.wargames.pos                 2530 non-null   float64
 44  stats.median                              76688 non-null  int64
 45  stats.numcomments                         76688 non-null  int64
 46  stats.numweights                          76688 non-null  int64
 47  stats.owned                               76688 non-null  int64
 48  stats.stddev                              76688 non-null  float64
 49  stats.subtype.boardgame.bayesaverage      13867 non-null  float64
 50  stats.subtype.boardgame.pos               13693 non-null  float64
 51  stats.trading                             76688 non-null  int64
 52  stats.usersrated                          76688 non-null  int64
 53  stats.wanting                             76688 non-null  int64
 54  stats.wishing                             76688 non-null  int64
 55  polls.language_dependence                 17840 non-null  object
 56  polls.suggested_numplayers.1              15148 non-null  object
 57  polls.suggested_numplayers.10             734 non-null    object
 58  polls.suggested_numplayers.2              17712 non-null  object
 59  polls.suggested_numplayers.3              12781 non-null  object
 60  polls.suggested_numplayers.4              13058 non-null  object
 61  polls.suggested_numplayers.5              7128 non-null   object
 62  polls.suggested_numplayers.6              5046 non-null   object
 63  polls.suggested_numplayers.7              1918 non-null   object
 64  polls.suggested_numplayers.8              1690 non-null   object
 65  polls.suggested_numplayers.9              776 non-null    object
 66  polls.suggested_numplayers.Over           13899 non-null  object
 67  polls.suggested_playerage                 15949 non-null  float64
 68  attributes.t.links.concat.2....           65 non-null     object
 69  stats.family.amiga.bayesaverage           1 non-null      float64
 70  stats.family.amiga.pos                    1 non-null      float64
 71  stats.family.arcade.bayesaverage          1 non-null      float64
 72  stats.family.arcade.pos                   1 non-null      float64
 73  stats.family.atarist.bayesaverage         1 non-null      float64
 74  stats.family.atarist.pos                  1 non-null      float64
 75  stats.family.commodore64.bayesaverage     1 non-null      float64
 76  stats.family.commodore64.pos              1 non-null      float64
 77  stats.subtype.rpgitem.bayesaverage        2 non-null      float64
 78  stats.subtype.rpgitem.pos                 2 non-null      float64
 79  stats.subtype.videogame.bayesaverage      1 non-null      float64
 80  stats.subtype.videogame.pos               1 non-null      float64
 81  expansions                                76688 non-null  int64
dtypes: float64(42), int64(12), object(28)
memory usage: 48.0+ MB
```

```
dfLess = df.copy()
percent_missing = dfLess.isnull().sum()* 100 / len(dfLess)
missing_value_df = pd.DataFrame({'column_name': dfLess.columns,
                                 'percent_missing': percent_missing})
missing_value_df
```

	column_name	percent_mis
row_names	row_names	0.0
game.id	game.id	0.0
game.type	game.type	0.0
details.description	details.description	0.0
details.image	details.image	7.6
...	...	
stats.subtype.rpgitem.bayesaverage	stats.subtype.rpgitem.bayesaverage	99.9
stats.subtype.rpgitem.pos	stats.subtype.rpgitem.pos	99.9

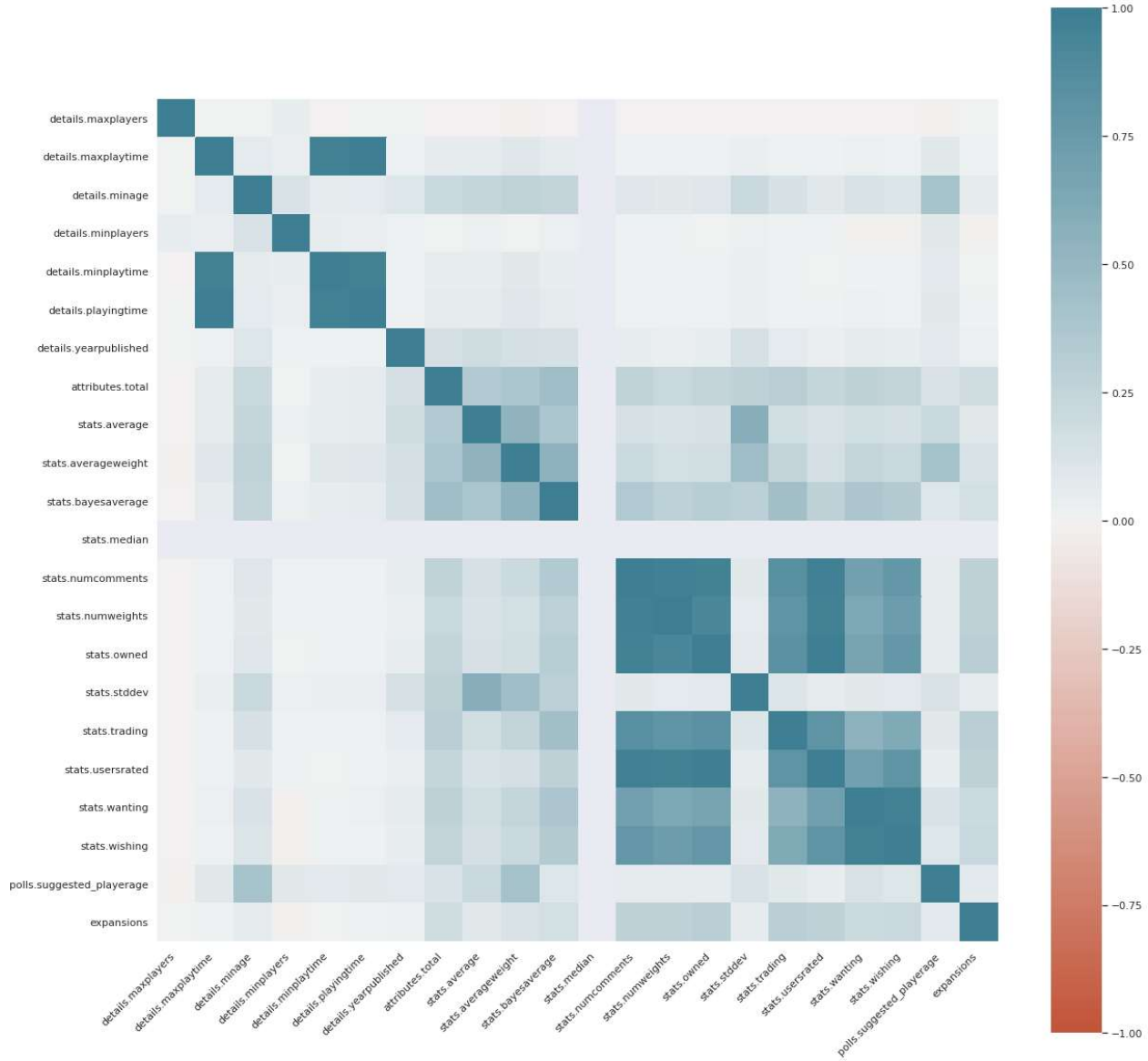
```
#removing columns with very high amount of null values

dfLess.drop(df.iloc[:,28:44], axis=1, inplace=True)
dfLess.drop(df.iloc[:,68:81], axis=1, inplace=True)
dfLess.drop(df.columns[[0,1,2,3,4,10,12,16,18,20,21,49,50]], axis=1, inplace=True)
dfLess.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76688 entries, 0 to 76687
Data columns (total 40 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   details.maxplayers              76685 non-null  float64
 1   details.maxplaytime             76685 non-null  float64
 2   details.minage                  76685 non-null  float64
 3   details.minplayers              76685 non-null  float64
 4   details.minplaytime             76685 non-null  float64
 5   details.playingtime             76685 non-null  float64
 6   details.yearpublished           76685 non-null  float64
 7   attributes.boardgameartist      27520 non-null  object
 8   attributes.boardgamecategory    75297 non-null  object
 9   attributes.boardgamedesigner    67101 non-null  object
 10  attributes.boardgamefamily      39776 non-null  object
 11  attributes.boardgamemechanic    62971 non-null  object
 12  attributes.boardgamepublisher   76593 non-null  object
 13  attributes.total                76688 non-null  int64
 14  stats.average                   76688 non-null  float64
 15  stats.averageweight             76688 non-null  float64
 16  stats.bayesaverage              76688 non-null  float64
 17  stats.median                    76688 non-null  int64
 18  stats.numcomments               76688 non-null  int64
 19  stats.numweights                76688 non-null  int64
 20  stats.owned                     76688 non-null  int64
 21  stats.stddev                    76688 non-null  float64
 22  stats.trading                   76688 non-null  int64
 23  stats.usersrated                76688 non-null  int64
 24  stats.wanting                   76688 non-null  int64
 25  stats.wishing                   76688 non-null  int64
```

```
 26  polls.language_dependence        17840 non-null  object
 27  polls.suggested_numplayers.1     15148 non-null  object
 28  polls.suggested_numplayers.10    734 non-null    object
 29  polls.suggested_numplayers.2     17712 non-null  object
 30  polls.suggested_numplayers.3     12781 non-null  object
 31  polls.suggested_numplayers.4     13058 non-null  object
 32  polls.suggested_numplayers.5     7128 non-null   object
 33  polls.suggested_numplayers.6     5046 non-null   object
 34  polls.suggested_numplayers.7     1918 non-null   object
 35  polls.suggested_numplayers.8     1690 non-null   object
 36  polls.suggested_numplayers.9     776 non-null    object
 37  polls.suggested_numplayers.Over  13899 non-null  object
 38  polls.suggested_playerage        15949 non-null  float64
 39  expansions                       76688 non-null  int64
dtypes: float64(12), int64(10), object(18)
memory usage: 23.4+ MB
```

```
corr = dfLess.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20,220,n=200),
    square=True
)
sns.set(rc = {'figure.figsize':(20,20)})
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
)
```

```
[Text(0.5, 0, 'details.maxplayers'),
 Text(1.5, 0, 'details.maxplaytime'),
 Text(2.5, 0, 'details.minage'),
 Text(3.5, 0, 'details.minplayers'),
 Text(4.5, 0, 'details.minplaytime'),
 Text(5.5, 0, 'details.playingtime'),
 Text(6.5, 0, 'details.yearpublished'),
 Text(7.5, 0, 'attributes.total'),
 Text(8.5, 0, 'stats.average'),
 Text(9.5, 0, 'stats.averageweight'),
 Text(10.5, 0, 'stats.bayesaverage'),
 Text(11.5, 0, 'stats.median'),
 Text(12.5, 0, 'stats.numcomments'),
 Text(13.5, 0, 'stats.numweights'),
 Text(14.5, 0, 'stats.owned'),
 Text(15.5, 0, 'stats.stddev'),
 Text(16.5, 0, 'stats.trading'),
 Text(17.5, 0, 'stats.usersrated'),
 Text(18.5, 0, 'stats.wanting'),
 Text(19.5, 0, 'stats.wishing'),
 Text(20.5, 0, 'polls.suggested_playerage'),
 Text(21.5, 0, 'expansions')]
```



corr

	details.maxplayers	details.maxplaytime	details.minage
details.maxplayers	1.000000	0.000201	0.000134

Several of the columns are found to be highly correlated (>0.8) and will be removed from the model since the feature will be well represented by the remaining feature.

```
#remove highly correlated columns
dfLess.drop(['details.maxplaytime', 'stats.numweights', 'stats.owned', 'stats.wishing', 'stat
dfLess.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76688 entries, 0 to 76687
Data columns (total 34 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   details.maxplayers                76685 non-null  float64
 1   details.minage                    76685 non-null  float64
 2   details.minplayers                76685 non-null  float64
 3   details.playingtime               76685 non-null  float64
 4   details.yearpublished             76685 non-null  float64
 5   attributes.boardgameartist        27520 non-null  object
 6   attributes.boardgamecategory      75297 non-null  object
 7   attributes.boardgamedesigner      67101 non-null  object
 8   attributes.boardgamefamily        39776 non-null  object
 9   attributes.boardgamemechanic      62971 non-null  object
 10  attributes.boardgamepublisher     76593 non-null  object
 11  attributes.total                  76688 non-null  int64
 12  stats.average                     76688 non-null  float64
 13  stats.averageweight               76688 non-null  float64
 14  stats.bayesaverage                76688 non-null  float64
 15  stats.median                      76688 non-null  int64
 16  stats.stddev                      76688 non-null  float64
 17  stats.trading                     76688 non-null  int64
 18  stats.usersrated                  76688 non-null  int64
 19  stats.wanting                     76688 non-null  int64
 20  polls.language_dependence         17840 non-null  object
 21  polls.suggested_numplayers.1      15148 non-null  object
 22  polls.suggested_numplayers.10     734 non-null    object
 23  polls.suggested_numplayers.2      17712 non-null  object
 24  polls.suggested_numplayers.3      12781 non-null  object
 25  polls.suggested_numplayers.4      13058 non-null  object
 26  polls.suggested_numplayers.5      7128 non-null   object
 27  polls.suggested_numplayers.6      5046 non-null   object
 28  polls.suggested_numplayers.7      1918 non-null   object
 29  polls.suggested_numplayers.8      1690 non-null   object
 30  polls.suggested_numplayers.9      776 non-null    object
 31  polls.suggested_numplayers.Over   13899 non-null  object
 32  polls.suggested_playerage         15949 non-null  float64
 33  expansions                        76688 non-null  int64
dtypes: float64(10), int64(6), object(18)
memory usage: 19.9+ MB
```

```
# investigate columns with many groups

li = [5,6,7,8,9,10,20,21,22,23,24,25,26,27,28,29,30,31,32]

for i in li:
  l =dfLess.iloc[:,i].unique()

  print(i, dfLess.columns[i], len(l))
```

```
    5 attributes.boardgameartist 12528
    6 attributes.boardgamecategory 14383
    7 attributes.boardgamedesigner 21215
    8 attributes.boardgamefamily 10233
    9 attributes.boardgamemechanic 9011
    10 attributes.boardgamepublisher 23387
    20 polls.language_dependence 6
    21 polls.suggested_numplayers.1 4
    22 polls.suggested_numplayers.10 4
    23 polls.suggested_numplayers.2 4
    24 polls.suggested_numplayers.3 4
    25 polls.suggested_numplayers.4 4
    26 polls.suggested_numplayers.5 4
    27 polls.suggested_numplayers.6 4
    28 polls.suggested_numplayers.7 4
    29 polls.suggested_numplayers.8 4
    30 polls.suggested_numplayers.9 4
    31 polls.suggested_numplayers.Over 4
    32 polls.suggested_playerage 13
```

Columns with many groupings (greater than 13 groups in this case) will be dropped as the information gained from these columns will be limited.

```
dfLess.drop(dfLess.columns[[5,6, 7, 8, 9, 10, 15]], axis=1, inplace=True)
dfLess.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 76688 entries, 0 to 76687
    Data columns (total 27 columns):
     #   Column                          Non-Null Count  Dtype
    ---  ------                          --------------  -----
     0   details.maxplayers              76685 non-null  float64
     1   details.minage                  76685 non-null  float64
     2   details.minplayers              76685 non-null  float64
     3   details.playingtime             76685 non-null  float64
     4   details.yearpublished           76685 non-null  float64
     5   attributes.total                76688 non-null  int64
     6   stats.average                   76688 non-null  float64
     7   stats.averageweight             76688 non-null  float64
     8   stats.bayesaverage              76688 non-null  float64
     9   stats.stddev                    76688 non-null  float64
     10  stats.trading                   76688 non-null  int64
     11  stats.usersrated                76688 non-null  int64
```

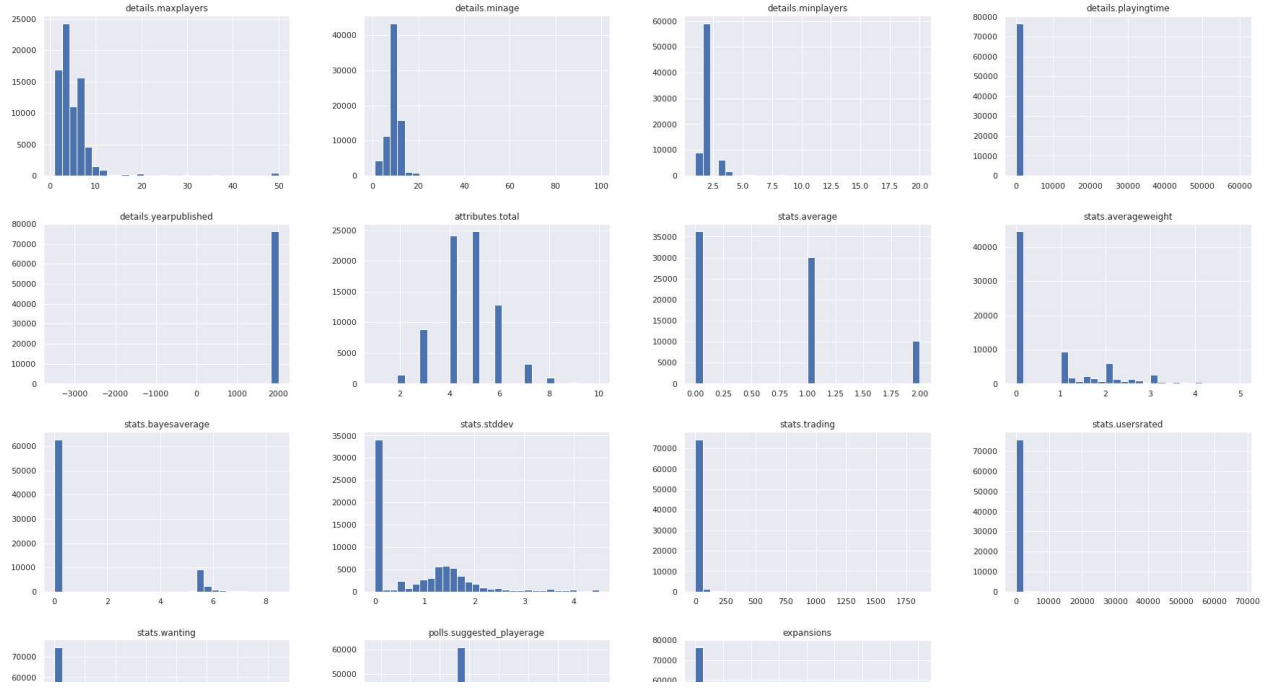
```
 12  stats.wanting                    76688 non-null  int64
 13  polls.language_dependence        17840 non-null  object
 14  polls.suggested_numplayers.1     15148 non-null  object
 15  polls.suggested_numplayers.10    734 non-null    object
 16  polls.suggested_numplayers.2     17712 non-null  object
 17  polls.suggested_numplayers.3     12781 non-null  object
 18  polls.suggested_numplayers.4     13058 non-null  object
 19  polls.suggested_numplayers.5     7128 non-null   object
 20  polls.suggested_numplayers.6     5046 non-null   object
 21  polls.suggested_numplayers.7     1918 non-null   object
 22  polls.suggested_numplayers.8     1690 non-null   object
 23  polls.suggested_numplayers.9     776 non-null    object
 24  polls.suggested_numplayers.Over  13899 non-null  object
 25  polls.suggested_playerage        15949 non-null  float64
 26  expansions                       76688 non-null  int64
dtypes: float64(10), int64(5), object(12)
memory usage: 15.8+ MB
```

```
#column cleaning - moving items that don't make sense

dfLess.loc[dfLess['details.minplayers'] >20, 'details.minplayers'] =20
dfLess.loc[dfLess['details.maxplayers'] >50, 'details.maxplayers'] =50


#set rating categories
dfLess.loc[dfLess['stats.average'] <5, 'stats.average'] = 0
dfLess.loc[dfLess['stats.average'] >=7, 'stats.average'] = 2
dfLess.loc[(dfLess['stats.average'] >=5) & (dfLess['stats.average']<7), 'stats.average'] = 1




#changing 0s for columns that do not make sense to allow for imputation
NAcols = ['details.yearpublished', 'details.playingtime','details.minage', 'details.maxplayer
for i in NAcols:
  dfLess[i] = dfLess[i].replace(0, np.nan)



NaNcols = ['polls.suggested_numplayers.10','polls.suggested_numplayers.1', 'polls.suggested_r

for x in NaNcols:
  dfLess[x] = dfLess[x].fillna('NotRecommended')

dfLess['polls.suggested_playerage'] = dfLess['polls.suggested_playerage'].fillna(dfLess['poll
dfLess['polls.language_dependence'] = dfLess['polls.language_dependence'].fillna('Not Rated')


for col in dfLess.columns:
  if dfLess[col].isna().sum()!=0:
    print(col,dfLess[col].isna().sum())
```

```
 details.maxplayers 5272
```

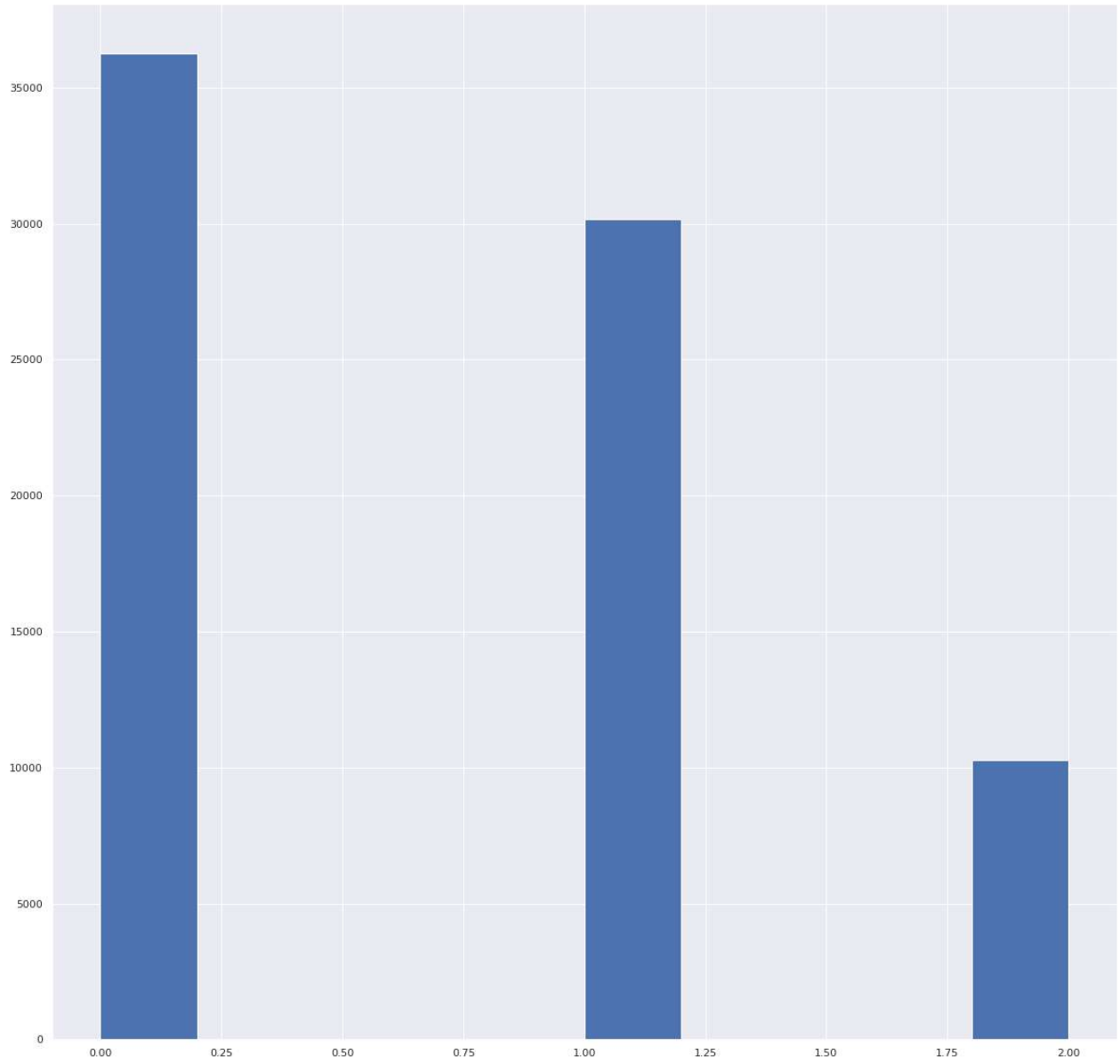
```
     details.minage 19826
     details.minplayers 1783
     details.playingtime 18311
     details.yearpublished 7900


for col in dfLess.columns:
  if dfLess[col].isna().sum()!=0:
    dfLess[col] = dfLess[col].fillna(dfLess[col].mean())
    print(col,dfLess[col].isna().sum())

     details.maxplayers 0
     details.minage 0
     details.minplayers 0
     details.playingtime 0
     details.yearpublished 0


dfLess.hist(bins=30, figsize=(30,20))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fac910c9f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac90448690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac90476c10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac9042f910>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fac903e3e10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac903a9350>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac9035f8d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac90314d10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fac90314d50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac902d7390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac90247bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac90207110>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fac901bf610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac901f4b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac901abfd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fac9016d550>]],
      dtype=object)
```




```
dfLess['stats.average'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fac99fef2d0>
```



The above graph displays an asymmetrical distribution of the data - where highly rated games account for less than a third of each of the other two groups by comparison.

```
dfLess['stats.average'] = dfLess['stats.average'].astype(str)
dfLess.loc[dfLess['stats.average'] =='0.0', 'stats.average'] = 'low'
dfLess.loc[dfLess['stats.average'] =='2.0', 'stats.average'] = 'high'
dfLess.loc[dfLess['stats.average'] =='1.0', 'stats.average'] = 'med'

#one-hot encoding of the remaining categorical variables
ohe = pd.get_dummies(dfLess, columns = ['polls.language_dependence', 'polls.suggested_numplay
                                        'polls.suggested_numplayers.10', 'polls.suggested_num
                                         'polls.suggested_numplayers.5', 'polls.suggested_num
                                         'polls.suggested_numplayers.9', 'polls.suggested_num
```

The data will be explored via classification modeling using a stratified classificaiton approach, and component reduction through principal component analysis as well as a regular 10-fold model.

```
#Data preparation for decision tree modelling
X = ohe.drop('stats.average', axis=1)
y = ohe[['stats.average']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=100)


kFolds = StratifiedKFold(n_splits=10, shuffle=True, random_state=100)
cnt =1
for train, test in kFolds.split(X,y):
  print(f'Fold:{cnt} successfully split')
  cnt+=1
```

```
Fold:1 successfully split
Fold:2 successfully split
Fold:3 successfully split
Fold:4 successfully split
Fold:5 successfully split
Fold:6 successfully split
Fold:7 successfully split
Fold:8 successfully split
Fold:9 successfully split
Fold:10 successfully split
```

Checking max depth tuning to see what the optimal tree size would look like to create a template.

```
#depth tuning
max_depth = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

for val in max_depth:
    score = cross_val_score(tree.DecisionTreeClassifier(max_depth= val, random_state= 100), X
    print(f'Average score({val}): {"{:.3f}".format(score.mean())}')
```

```
Average score(1): 0.693
Average score(2): 0.693
```

```
Average score(3): 0.701
Average score(4): 0.708
Average score(5): 0.711
Average score(6): 0.727
Average score(7): 0.735
Average score(8): 0.743
Average score(9): 0.746
Average score(10): 0.752
Average score(11): 0.756
Average score(12): 0.756
Average score(13): 0.757
Average score(14): 0.757
Average score(15): 0.756
```

```
cvs = cross_val_score(tree.DecisionTreeClassifier(criterion="gini", max_depth=12, random_stat
print(f'Fold scores: {cvs}')
print(f'Average score: {cvs.mean()}')
```

```
Fold scores: [0.76450645 0.75746512 0.75733472 0.74755509 0.75433564 0.76046421
 0.75498761 0.7659408  0.74908712 0.75117371]
Average score: 0.7562850470833773
```

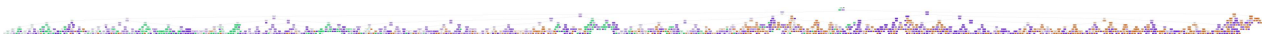
Using a stratified 10-fold decision tree classificaiton model - this yielded an overall score of ~0.756. The folding of the model ensures the variance of model is reduced from the different iteration samples used.

```
#Decision tree view determined by tuning
clf = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=12)
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=12, random_state=100)
```

```
features = ohe.columns
features = features.drop('stats.average')
clsN = y_train.iloc[::,-1].unique()
clsN = clsN[::-1]
graphDF = export_graphviz(clf,feature_names = features, class_names = clsN ,rounded = True,
# Draw graph
graph= pydotplus.graph_from_dot_data(graphDF)
graph.write_png("myTree.png")
# Show graph
Image(graph.create_png())
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.340408 to fit

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.340408 to fit
```



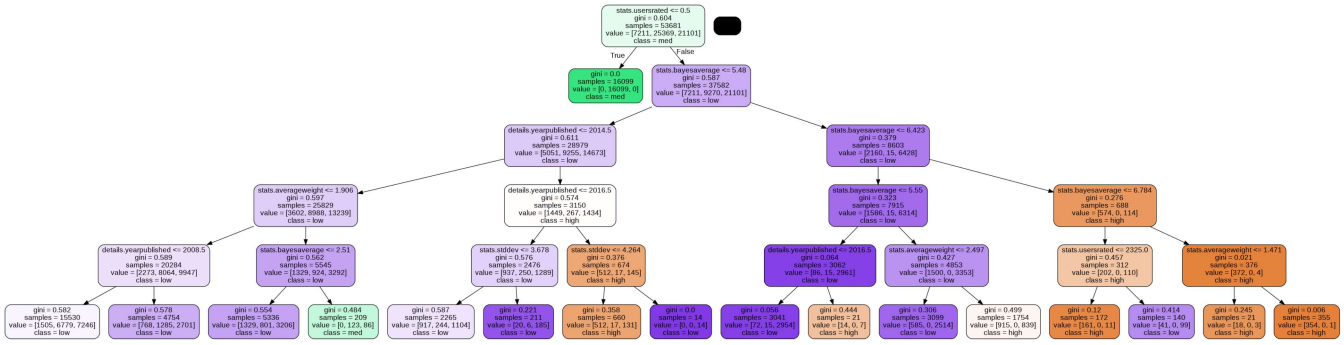
Double-click (or enter) to edit

```
py_pred = clf.predict(X_test)
clf.score(X_test,y_test)
```

```
0.7523797105228843
```

For a decision tree model taken using a single train-test (70/30) at the optimal tree depth of 12 - the results yielded a Model score of ~0.75, meaning that it was able to correctly classify a data row 3 out of 4 times! A simplified tree is displayed below.

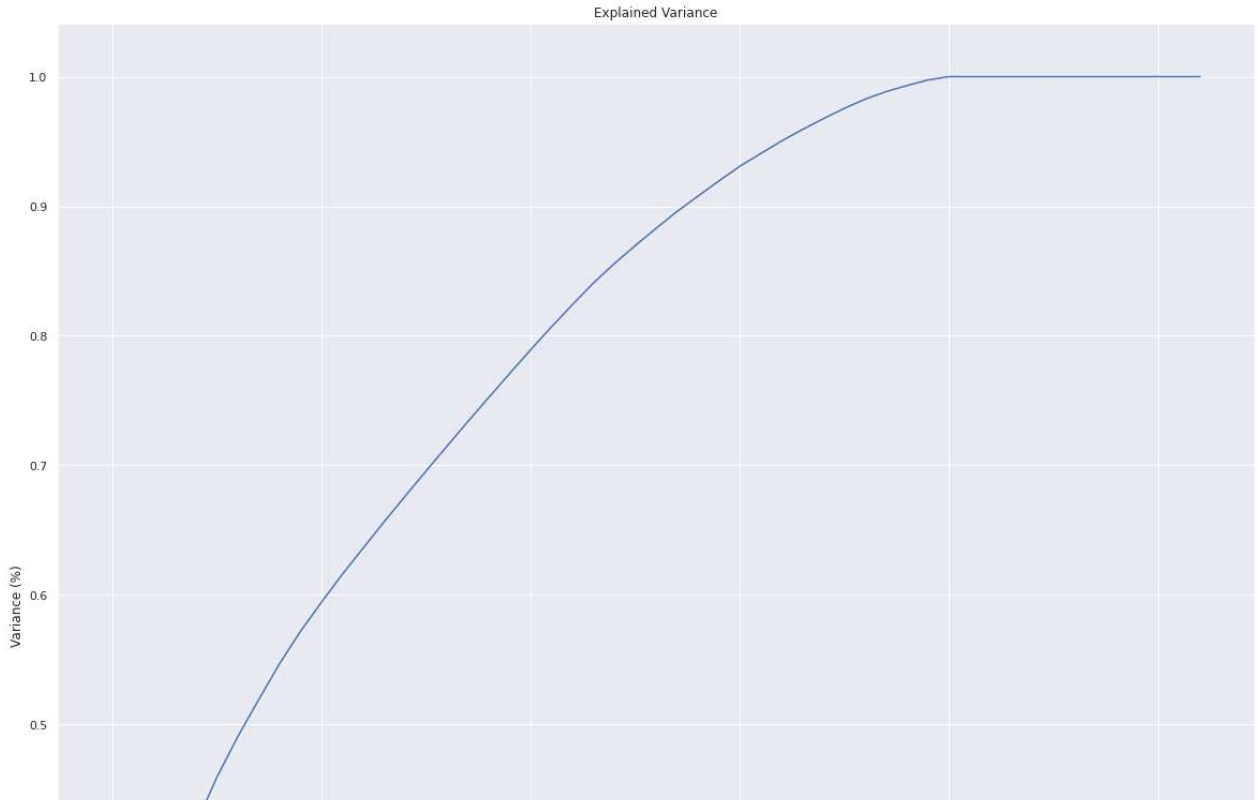
```
clf = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=5)#), min_samples_
clf.fit(X_train, y_train)
graphDF = export_graphviz(clf,feature_names = features, class_names = clsN ,rounded = True,
# Draw graph
graph = pydotplus.graph_from_dot_data(graphDF)
graph.write_png("myTree.png")
# Show graph
Image(graph.create_png())
```



Next I'll explore to see if using Principle component Analysis for feature reduction, if this will yield better results.

```
#Attempt feature reduction via PCA
tansformX = StandardScaler().fit_transform(X)
```

```
pca=PCA()
x_train,x_test,y_train,y_test = train_test_split(tansformX,y,test_size = 0.30, random_state=
principalComponents = pca.fit_transform(tansformX)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance (%)') #for each component
plt.title('Explained Variance')
plt.show()
```



The graph of the Variance shows about 95% of the variance of the data can be explained by about 31 of the columns.

```
principal = PCA(n_components=31)
new_data = principal.fit_transform(tansformX)
j = []
for i in range(1,32):
  s = str('PC-'+str(i))
  j.append(s)
pca = pd.DataFrame(new_data,columns=j)
pca
```

	PC-1	PC-2	PC-3	PC-4	PC-5	PC-6	PC-7	
0	6.004505	-3.929580	0.946890	8.438855	-1.808738	1.760019	0.456501	-0.
1	3.641615	-3.187910	-0.515126	1.696523	0.753273	0.393796	0.336526	2.
2	7.246894	-7.292955	2.556275	12.453320	2.300025	-0.909594	2.830145	-4.
3	4.236470	-4.232743	-0.034574	-0.184406	2.601574	-0.491229	0.601720	2.
4	11.859029	-7.480321	-0.599125	20.331732	-2.226362	0.662437	2.612939	-5

```
x_train,x_test,y_train,y_test = train_test_split(pca, y, test_size = 0.30, random_state= 100)
clf = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=12)
clf.fit(x_train,y_train)
clf.score(x_test,y_test)
```

```
0.6677967575085844
```

Exploring with PCA resulted in an overall reduction in performance of the model, dropping to ~0.668. In this case PCA would not be worthwile using for feature reduction, possible due in part because of many of the columns being generated from earlier splits using the one hot encoding.

```
#k fold decision tree confusion matrix initialization
def cross_val_predict(model, kfold : KFold, X : np.array, y : np.array) -> Tuple[np.array, np

    model_ = cp.deepcopy(model)

    no_classes = len(np.unique(y))

    actual_classes = np.empty([0], dtype=int)
    predicted_classes = np.empty([0], dtype=int)
    predicted_proba = np.empty([0, no_classes])

    for train_ndx, test_ndx in kfold.split(X):

        train_X, train_y, test_X, test_y = X[train_ndx], y[train_ndx], X[test_ndx], y[test_nd

        actual_classes = np.append(actual_classes, test_y)

        model_.fit(train_X, train_y) #cross val score here?
        predicted_classes = np.append(predicted_classes, model_.predict(test_X))

        try:
            predicted_proba = np.append(predicted_proba, model_.predict_proba(test_X), axis=0
        except:
            predicted_proba = np.append(predicted_proba, np.zeros((len(test_X), no_classes),

    return actual_classes, predicted_classes, predicted_proba
```

```
#Plotting k-fold decision tree results on confusion matrix
def plot_confusion_matrix(actual_classes : np.array, predicted_classes : np.array, sorted_lab

    matrix = confusion_matrix(actual_classes, predicted_classes, labels=sorted_labels)

    plt.figure(figsize=(12.8,6))
    sns.heatmap(matrix, annot=True, xticklabels=sorted_labels, yticklabels=sorted_labels, cma
    plt.xlabel('Predicted'); plt.ylabel('Actual'); plt.title('KFold Decision Classifier Confu

    plt.show()
    return matrix



kfold = KFold(n_splits=10, random_state=100, shuffle=True)
actual_classes, predicted_classes, _ = cross_val_predict(DecisionTreeClassifier(max_depth=12)
cfm = plot_confusion_matrix(actual_classes, predicted_classes, ['high', 'med', 'low'])
```


```
acc = (cfm[0,0]+cfm[1,1]+cfm[2,2])/cfm.sum()*100

recHigh = cfm[0,0]/(cfm[0,0:3].sum())*100
recMed = cfm[1,1]/cfm[1,0:3].sum()*100
recLow = cfm[2,2]/(cfm[2,0:3].sum())*100
presHigh = cfm[0,0]/(cfm[0:3,0].sum())*100
presMed = cfm[1,1]/(cfm[0:3,1].sum())*100
presLow = cfm[2,2]/(cfm[0:3,2].sum())*100
fScoreH = 2*recHigh*presHigh/(recHigh+presHigh)
fScoreM = 2*recMed*presMed/(recMed+presMed)
```