In [1]:

```
import numpy as np
import pandas as pd
pd.options.display.max_columns=100
from sklearn.model_selection import train_test_split, cross_val_score, cross
import sklearn.metrics
from sklearn.preprocessing import StandardScaler as SSc
from sklearn.tree import DecisionTreeRegressor as DTR
from sklearn.ensemble import RandomForestRegressor as RFR
from sklearn.neighbors import KNeighborsRegressor as KNR
import matplotlib.pyplot as plt
%matplotlib inline

#set width of window to preference
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
display(HTML("<style>.output.output_scroll{ height:100% !important; }</style
```

In [2]:

```
data = pd.read_csv("Data-Prepped.csv",index_col=0)
data = data.astype(np.float32)
data.head()
```

Out[2]:

	Bronze	Silver	Gold	Platinum	Diamond	Master	GrandMaster	LeagueIndex	Age	HoursPerWe
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	5.0	27.0	1
1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	5.0	23.0	1
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	4.0	30.0	1
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	3.0	19.0	2
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	3.0	32.0	1

1 - Trying to predict the player's APM (Actions Per Minute)

Split data into X,Y where Y is the length of the game, then normalize the inputs.

In [3]:

```
X = data.iloc[:,data.columns != 'APM']
Y = data.iloc[:,data.columns == 'APM']
#transform input data (normalize scaling)
ssc = SSc()
Xft = ssc.fit_transform(X)
X = pd.DataFrame(Xft)
print("Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes: ")
Xtr,Xtst,Ytr,Ytst = train_test_split(X,Y.values.ravel(),test_size=0.2,random
print(Xtr.shape,Xtst.shape,Ytr.shape,Ytst.shape)
```

```
Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes:
(2670, 25) (668, 25) (2670,) (668,)
```

In [4]:

```
print("Classifier scores:\n"+"-"*18+"\n")
for i in range(3,8,2):
    tree = DTR(max_depth=i+1)
    tree.fit(Xtr,Ytr)
    scr = cross_val_score(tree,Xtst,Ytst, cv=5)
    print("Tree of depth "+str(i)+"\nscore avg:"+str(sum(scr)/5)+"\nscore =

for i in range(1,28,3):
    dpth = 7
    forest = RFR(n_estimators=i,max_depth=dpth)
    forest.fit(Xtr,Ytr)
    scr = cross_val_score(forest,Xtst,Ytst, cv=5)
    print("\nRandom Forest trees = "+str(i)+" depth = "+str(dpth)+" \nscore

print("\n"+"-"*64)

for i in range(1,10):
    knn = KNR(n_neighbors=i)
    knn.fit(Xtr,Ytr)
    scr = cross_val_score(knn,Xtst,Ytst, cv=5)
    print("\nK-Nearest Neighbors "+str(i)+"-neighbors\nscore avg:"+str(sum(s
          str(scr))
```

```
Classifier scores:
------------------

Tree of depth 3
score avg:0.8032197759729188
score = [0.8271212  0.83885572 0.69060868 0.78965485 0.86985842]

----------------------------------------------------------------
Tree of depth 5
score avg:0.845846191947276
score = [0.88849921 0.8856103  0.74909063 0.83008348 0.87594734]

----------------------------------------------------------------
Tree of depth 7
score avg:0.8696179557268688
score = [0.8737981  0.89376504 0.86539501 0.83220082 0.88293081]

----------------------------------------------------------------

Random Forest trees = 1 depth = 7
score avg: 0.8478416594793909
scores: [0.83496781 0.8392719  0.85509484 0.85377152 0.85610223]

Random Forest trees = 4 depth = 7
score avg: 0.9143970869915241
scores: [0.92281844 0.90360367 0.92070465 0.90471198 0.9201467 ]

Random Forest trees = 7 depth = 7
score avg: 0.9193531761594309
scores: [0.93155436 0.92729023 0.88264547 0.92572652 0.9295493 ]

Random Forest trees = 10 depth = 7
score avg: 0.9240256353990427
scores: [0.93817753 0.93067714 0.89429252 0.93548159 0.9214994 ]
```

```
Random Forest trees = 13 depth = 7
score avg: 0.9217937557059777
scores: [0.93433215 0.93401824 0.9019924  0.92232971 0.91629628]

Random Forest trees = 16 depth = 7
score avg: 0.9257613477843986
scores: [0.93492841 0.94244993 0.9039587  0.92778045 0.91968925]

Random Forest trees = 19 depth = 7
score avg: 0.9286730721115417
scores: [0.93727028 0.94030842 0.91005841 0.92586479 0.92986347]

Random Forest trees = 22 depth = 7
score avg: 0.9312856557053116
scores: [0.93657706 0.94391803 0.91792289 0.93190683 0.92610346]

Random Forest trees = 25 depth = 7
score avg: 0.9297893563415878
scores: [0.93953064 0.9424512  0.9057713  0.92473428 0.93645936]

----------------------------------------------------------------

K-Nearest Neighbors 1-neighbors
score avg:0.6567899023671828
score = [0.71022497 0.64850297 0.64872142 0.56369455 0.7128056 ]

K-Nearest Neighbors 2-neighbors
score avg:0.7317652289384065
score = [0.73767447 0.7115865  0.76003885 0.69521691 0.75430942]

K-Nearest Neighbors 3-neighbors
score avg:0.7515982037258839
score = [0.75454142 0.73871477 0.77797304 0.72133181 0.76542998]

K-Nearest Neighbors 4-neighbors
score avg:0.7502555506003818
score = [0.74036176 0.73786996 0.78074976 0.72811242 0.76418385]

K-Nearest Neighbors 5-neighbors
score avg:0.7588877042305268
score = [0.751122   0.74559278 0.77733696 0.74497376 0.77541301]

K-Nearest Neighbors 6-neighbors
score avg:0.7554305770305947
score = [0.75837805 0.74490975 0.75560472 0.76142531 0.75683505]

K-Nearest Neighbors 7-neighbors
score avg:0.7501583728502097
score = [0.75785751 0.73121346 0.75348483 0.76157765 0.74665842]

K-Nearest Neighbors 8-neighbors
score avg:0.7498361810705756
score = [0.74728066 0.73483882 0.75520183 0.77412768 0.73773192]

K-Nearest Neighbors 9-neighbors
```

```
score avg:0.7492486658812062
score = [0.74692561 0.73230282 0.75005114 0.77822466 0.7387391 ]
```

Random Forests can get up to ~94% accuracy with 16-23 trees of depth 7-8

K-Nearest Neighbors optimizes at ~76% accuracy with 6-7 neighbors

I think this is somewhat easy to predict because a player will almost always being doing approximately the same number of things within a certain timeframe in a game even if those things vary based on how the game progresses.

---

2 - Trying to predict a player's league

Split data into X,Y where Y is the kills, deaths and assists, then normalize the inputs.

In [5]:
```
X = data.iloc[:,8:]
Y = data.iloc[:,7]
#transform input data (normalize scaling)
ssc = SSc()
Xft = ssc.fit_transform(X)
X = pd.DataFrame(Xft)
print("Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes: ")
Xtr,Xtst,Ytr,Ytst = train_test_split(X,Y,test_size=0.2,random_state=2021)
print(Xtr.shape,Xtst.shape,Ytr.shape,Ytst.shape)
```

```
Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes:
(2670, 18) (668, 18) (2670,) (668,)
```

In [11]:

```
print("Classifier scores:\n"+"-"*18+"\n")
for i in range(1,5):
    tree = DTR(max_depth=i+1)
    tree.fit(Xtr,Ytr)
    scr = cross_val_score(tree,Xtst,Ytst, cv=5)
    print("Tree of depth "+str(i)+"\nscore avg:"+str(sum(scr)/5)+"\nscore =

for i in range(1,28,3):
    dpth = 12
    forest = RFR(n_estimators=i,max_depth=dpth)
    forest.fit(Xtr,Ytr)
    scr = cross_val_score(forest,Xtst,Ytst, cv=5)
    print("\nRandom Forest trees = "+str(i)+" depth = "+str(dpth)+" \nscore

print("\n"+"-"*64)

for i in range(1,10):
    knn = KNR(n_neighbors=i)
    knn.fit(Xtr,Ytr)
    scr = cross_val_score(knn,Xtst,Ytst, cv=5)
    print("\nK-Nearest Neighbors "+str(i)+"-neighbors\nscore avg:"+str(sum(s
          str(scr))
```

```
Classifier scores:
------------------

Tree of depth 1
score avg:0.3403440334409956
score = [0.38913359 0.36448855 0.34194207 0.24783451 0.35832145]

----------------------------------------------------------------
Tree of depth 2
score avg:0.3576686385080065
score = [0.43001907 0.33181201 0.33993957 0.30824462 0.37832793]

----------------------------------------------------------------
Tree of depth 3
score avg:0.35534726368757885
score = [0.42045359 0.33843993 0.38518068 0.27673255 0.35592957]

----------------------------------------------------------------
Tree of depth 4
score avg:0.3437868693835167
score = [0.45785779 0.36198845 0.35051617 0.23465786 0.31391407]

----------------------------------------------------------------

Random Forest trees = 1 depth = 12
score avg: 0.16651361836639175
scores: [0.19109599 0.38560725 0.03772168 0.06057304 0.15757012]

Random Forest trees = 4 depth = 12
score avg: 0.4299119387716333
scores: [0.46009776 0.40339719 0.4225009  0.39959224 0.46397161]

Random Forest trees = 7 depth = 12
score avg: 0.47414492121139273
```

```
scores: [0.54401832 0.48157266 0.44580631 0.40166658 0.49766074]

Random Forest trees = 10 depth = 12
score avg: 0.4682302568297917
scores: [0.50873854 0.47764976 0.40907191 0.42672866 0.51896242]

Random Forest trees = 13 depth = 12
score avg: 0.48673703880318425
scores: [0.55207908 0.5033478  0.41467391 0.4489497  0.5146347 ]

Random Forest trees = 16 depth = 12
score avg: 0.48770802867130625
scores: [0.57171225 0.50088887 0.43372915 0.42382127 0.5083886 ]

Random Forest trees = 19 depth = 12
score avg: 0.5125709723931277
scores: [0.58235608 0.50315627 0.48518814 0.46464476 0.52750961]

Random Forest trees = 22 depth = 12
score avg: 0.527082017479236
scores: [0.59369482 0.53164308 0.46624569 0.51947506 0.52435144]

Random Forest trees = 25 depth = 12
score avg: 0.5061117249618455
scores: [0.56030407 0.54918495 0.47218451 0.43340516 0.51547995]

----------------------------------------------------------------

K-Nearest Neighbors 1-neighbors
score avg:0.10295583239477739
score = [0.07348811 0.09218017 0.19117773 0.02511415 0.132819  ]

K-Nearest Neighbors 2-neighbors
score avg:0.3200218692541493
score = [0.35337191 0.29660059 0.27550991 0.31378424 0.36084269]

K-Nearest Neighbors 3-neighbors
score avg:0.3849283563213092
score = [0.4309387  0.34792927 0.3683605  0.34785642 0.4295569 ]

K-Nearest Neighbors 4-neighbors
score avg:0.4469098057902462
score = [0.49452021 0.4153093  0.4077581  0.42116153 0.49579989]

K-Nearest Neighbors 5-neighbors
score avg:0.46910458896327506
score = [0.51576886 0.44191614 0.46809393 0.44447489 0.47526913]

K-Nearest Neighbors 6-neighbors
score avg:0.47194486551093523
score = [0.49751385 0.45595194 0.47760902 0.43431317 0.49433636]

K-Nearest Neighbors 7-neighbors
score avg:0.4819129880013323
score = [0.5077249  0.47250793 0.48422813 0.43436075 0.51074323]

K-Nearest Neighbors 8-neighbors
```

```
score avg:0.4869628347995569
score = [0.52101065 0.47084084 0.47406188 0.44675727 0.52214353]

K-Nearest Neighbors 9-neighbors
score avg:0.4840630024593172
score = [0.52451131 0.45479938 0.46845173 0.44591716 0.52663543]
```

It seems very difficult to predict someone's league based on the data I have. I think that this is partially due to the skill levels not being vastly different between leagues and different players may be good at some skills and poor at others, i.e. platinum level in worker production but only silver level in army management

---

3 - Trying to predict a player's Action Latency (How long between focusing on a location and giving a command in that location)

In [12]:

```
X = data.iloc[:,data.columns != 'ActionLatency']
Y = data.iloc[:,data.columns == 'ActionLatency']
#transform input data (normalize scaling)
ssc = SSc()
Xft = ssc.fit_transform(X)
X = pd.DataFrame(Xft)
print("Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes: ")
Xtr,Xtst,Ytr,Ytst = train_test_split(X,Y.values.ravel(),test_size=0.2,random
print(Xtr.shape,Xtst.shape,Ytr.shape,Ytst.shape)
```

```
Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes:
(2670, 25) (668, 25) (2670,) (668,)
```

In [21]:

```
print("Classifier scores:\n"+"-"*18+"\n")
for i in range(1,6):
    tree = DTR(max_depth=i+1)
    tree.fit(Xtr,Ytr)
    scr = cross_val_score(tree,Xtst,Ytst, cv=5)
    print("Tree of depth "+str(i)+"\nscore avg:"+str(sum(scr)/5)+"\nscore =

for i in range(1,28,3):
    dpth = 6
    forest = RFR(n_estimators=i,max_depth=dpth)
    forest.fit(Xtr,Ytr)
    scr = cross_val_score(forest,Xtst,Ytst, cv=5)
    print("\nRandom Forest trees = "+str(i)+" depth = "+str(dpth)+" \nscore

print("\n"+"-"*64)

for i in range(1,10):
    knn = KNR(n_neighbors=i)
    knn.fit(Xtr,Ytr)
    scr = cross_val_score(knn,Xtst,Ytst, cv=5)
    print("\nK-Nearest Neighbors "+str(i)+"-neighbors\nscore avg:"+str(sum(s
          str(scr))
```

```
Classifier scores:
------------------

Tree of depth 1
score avg:0.6368445728023168
score = [0.67453979 0.69461853 0.6032955  0.53266655 0.6791025 ]

----------------------------------------------------------------
Tree of depth 2
score avg:0.7292429401021139
score = [0.71571997 0.77642274 0.76269725 0.65691945 0.7344553 ]

----------------------------------------------------------------
Tree of depth 3
score avg:0.7338384142991337
score = [0.71933252 0.78908394 0.77211738 0.6944908  0.69416744]

----------------------------------------------------------------
Tree of depth 4
score avg:0.7621940043128811
score = [0.76226648 0.78999556 0.79599369 0.73398329 0.72873101]

----------------------------------------------------------------
Tree of depth 5
score avg:0.71589816741241
score = [0.72061511 0.78254823 0.75697259 0.65591432 0.66344059]

----------------------------------------------------------------

Random Forest trees = 1 depth = 6
score avg: 0.6954008106423517
scores: [0.73602529 0.69138366 0.66629902 0.77570485 0.60759123]

Random Forest trees = 4 depth = 6
```

```
score avg: 0.7968041830675918
scores: [0.8390759  0.86124611 0.78216867 0.7239608  0.77756942]

Random Forest trees = 7 depth = 6
score avg: 0.8108869939299705
scores: [0.8568268  0.85139396 0.79100138 0.74684657 0.80836625]

Random Forest trees = 10 depth = 6
score avg: 0.822396081571539
scores: [0.88012572 0.82218555 0.8144945  0.77964819 0.81552645]

Random Forest trees = 13 depth = 6
score avg: 0.8308613306326844
scores: [0.86538593 0.85308082 0.82893158 0.79594302 0.81096531]

Random Forest trees = 16 depth = 6
score avg: 0.8315495166943754
scores: [0.88690377 0.85654462 0.81642564 0.79599531 0.80187824]

Random Forest trees = 19 depth = 6
score avg: 0.8316884107478358
scores: [0.8828878  0.85768836 0.8190716  0.78530291 0.81349138]

Random Forest trees = 22 depth = 6
score avg: 0.8353398390240121
scores: [0.89403732 0.85107986 0.80278528 0.80950568 0.81929106]

Random Forest trees = 25 depth = 6
score avg: 0.8373155139367032
scores: [0.88648812 0.85072581 0.82459565 0.8102271  0.81454088]

----------------------------------------------------------------

K-Nearest Neighbors 1-neighbors
score avg:0.4967991346021403
score = [0.57375707 0.56133706 0.52090344 0.44574187 0.38225625]

K-Nearest Neighbors 2-neighbors
score avg:0.6002036374004344
score = [0.71175563 0.66321011 0.54621368 0.58903421 0.49080455]

K-Nearest Neighbors 3-neighbors
score avg:0.6256613410033605
score = [0.71251288 0.66218515 0.57393715 0.60759285 0.57207867]

K-Nearest Neighbors 4-neighbors
score avg:0.6327625523381311
score = [0.70587075 0.63960528 0.6093669  0.60105896 0.60791087]

K-Nearest Neighbors 5-neighbors
score avg:0.6442528138352206
score = [0.71631854 0.64257956 0.61793102 0.63054578 0.61388917]

K-Nearest Neighbors 6-neighbors
score avg:0.6382444904641797
score = [0.71949757 0.62786982 0.59256097 0.63820137 0.61309273]
```

```
K-Nearest Neighbors 7-neighbors
score avg:0.6486630083076939
score = [0.7238587  0.63864377 0.6280587  0.64216898 0.61058489]

K-Nearest Neighbors 8-neighbors
score avg:0.6445864392614078
score = [0.71430014 0.63324547 0.62762667 0.63882143 0.60893848]

K-Nearest Neighbors 9-neighbors
score avg:0.643854407181036
score = [0.70474343 0.6459684  0.62335641 0.64235545 0.60284834]
```

Random Forests can get up to ~82% accuracy with 10-13 trees of depth 4-6

K-Nearest Neighbors optimizes at ~65% accuracy with 7 neighbors

This seems fairly easy to predict which is likely because players will be more or less consistent in their reaction speed (especially since these are ranked games most people won't play if they're overly fatigued so they will be more consistent).

---

4 - Trying to predict how many PACs a player goes through in a game

In [24]:
```
X = data.iloc[:,data.columns != 'NumberOfPACs']
Y = data.iloc[:,data.columns == 'NumberOfPACs']
#transform input data (normalize scaling)
ssc = SSc()
Xft = ssc.fit_transform(X)
X = pd.DataFrame(Xft)
print("Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes: ")
Xtr,Xtst,Ytr,Ytst = train_test_split(X,Y.values.ravel(),test_size=0.2,random
print(Xtr.shape,Xtst.shape,Ytr.shape,Ytst.shape)
```

```
Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes:
(2670, 25) (668, 25) (2670,) (668,)
```

In [39]:

```
print("Classifier scores:\n"+"-"*18+"\n")
for i in range(1,6):
    tree = DTR(max_depth=i+1)
    tree.fit(Xtr,Ytr)
    scr = cross_val_score(tree,Xtst,Ytst, cv=5)
    print("Tree of depth "+str(i)+"\nscore avg:"+str(sum(scr)/5)+"\nscore =

for i in range(5,29,2):
    dpth = 12
    forest = RFR(n_estimators=i,max_depth=dpth)
    forest.fit(Xtr,Ytr)
    scr = cross_val_score(forest,Xtst,Ytst, cv=5)
    print("\nRandom Forest trees = "+str(i)+" depth = "+str(dpth)+" \nscore

print("\n"+"-"*64)

for i in range(1,13):
    knn = KNR(n_neighbors=i)
    knn.fit(Xtr,Ytr)
    scr = cross_val_score(knn,Xtst,Ytst, cv=5)
    print("\nK-Nearest Neighbors "+str(i)+"-neighbors\nscore avg:"+str(sum(s
          str(scr))
```

```
Classifier scores:
------------------

Tree of depth 1
score avg:0.6711838824362658
score = [0.65221136 0.73829029 0.66072585 0.64460341 0.6600885 ]

----------------------------------------------------------------
Tree of depth 2
score avg:0.7524418025822117
score = [0.79386859 0.79296622 0.69178476 0.71300382 0.77058562]

----------------------------------------------------------------
Tree of depth 3
score avg:0.7853211685955301
score = [0.80870162 0.79723699 0.76438274 0.75593614 0.80034836]

----------------------------------------------------------------
Tree of depth 4
score avg:0.7954759079649211
score = [0.81214763 0.7889671  0.80237457 0.76950567 0.80438458]

----------------------------------------------------------------
Tree of depth 5
score avg:0.795451799869187
score = [0.82894022 0.76949658 0.78655605 0.78639268 0.80587347]

----------------------------------------------------------------

Random Forest trees = 1 depth = 12
score avg: 0.7233187484025182
scores: [0.80647551 0.7089456  0.67467661 0.70045749 0.72603853]
```

```
Random Forest trees = 3 depth = 12
score avg: 0.8511951143410219
scores: [0.86301682 0.8499355  0.85156137 0.8073744  0.88408747]

Random Forest trees = 5 depth = 12
score avg: 0.8608491746190865
scores: [0.90061903 0.84725096 0.83909636 0.83373748 0.88354206]

Random Forest trees = 7 depth = 12
score avg: 0.8661870398093934
scores: [0.88010034 0.88015369 0.85421815 0.83711146 0.87935156]

Random Forest trees = 9 depth = 12
score avg: 0.8884269161117594
scores: [0.90509469 0.89714673 0.88227538 0.85613881 0.90147897]

Random Forest trees = 11 depth = 12
score avg: 0.8816327807203601
scores: [0.89748125 0.88272939 0.87430599 0.85968137 0.8939659 ]

Random Forest trees = 13 depth = 12
score avg: 0.8813749064955434
scores: [0.90104295 0.88134592 0.88173668 0.8460195  0.89672948]

Random Forest trees = 15 depth = 12
score avg: 0.8834599101907656
scores: [0.89471964 0.88934624 0.88405741 0.8586297  0.89054655]

Random Forest trees = 17 depth = 12
score avg: 0.888668847949833
scores: [0.89899032 0.88673676 0.90308979 0.85777703 0.89675034]

Random Forest trees = 19 depth = 12
score avg: 0.8862816789742614
scores: [0.90625194 0.8774489  0.88439895 0.86465642 0.89865218]

Random Forest trees = 21 depth = 12
score avg: 0.8872633735882859
scores: [0.90387807 0.88511498 0.8836466  0.86472524 0.89895197]

Random Forest trees = 23 depth = 12
score avg: 0.8872386878249859
scores: [0.90779422 0.88309058 0.8874417  0.85563959 0.90222734]

Random Forest trees = 25 depth = 12
score avg: 0.8901885062917098
scores: [0.91194891 0.89599677 0.87896613 0.86338242 0.90064829]

Random Forest trees = 27 depth = 12
score avg: 0.8895945763886779
scores: [0.91258872 0.88119552 0.8782688  0.86920135 0.9067185 ]

Random Forest trees = 29 depth = 12
score avg: 0.8915580962633088
scores: [0.90151595 0.8876261  0.88790153 0.87161603 0.90913088]

Random Forest trees = 31 depth = 12
```

```
score avg: 0.8887661131720529
scores: [0.91500542 0.88683182 0.87939532 0.86002864 0.90256937]

Random Forest trees = 33 depth = 12
score avg: 0.8903643827336175
scores: [0.91019089 0.88910979 0.8871321  0.86620369 0.89918544]

----------------------------------------------------------------

K-Nearest Neighbors 1-neighbors
score avg:0.44494208915711086
score = [0.44979573 0.51368914 0.52685444 0.3209304  0.41344074]

K-Nearest Neighbors 2-neighbors
score avg:0.5302496646466325
score = [0.54363035 0.56971774 0.56213113 0.4302024  0.54556671]

K-Nearest Neighbors 3-neighbors
score avg:0.5812188765368039
score = [0.61669775 0.59762856 0.57723834 0.5119907  0.60253903]

K-Nearest Neighbors 4-neighbors
score avg:0.5988011004068212
score = [0.6260471  0.59965166 0.59902305 0.53125477 0.63802892]

K-Nearest Neighbors 5-neighbors
score avg:0.6194586489198692
score = [0.65059941 0.60786478 0.62635514 0.55012275 0.66235116]

K-Nearest Neighbors 6-neighbors
score avg:0.6371545853734133
score = [0.67333975 0.61724685 0.64807919 0.56629885 0.68080829]

K-Nearest Neighbors 7-neighbors
score avg:0.6337332348341291
score = [0.6604809  0.61087002 0.6307411  0.58923364 0.67734051]

K-Nearest Neighbors 8-neighbors
score avg:0.6311869521353295
score = [0.66178938 0.60628534 0.62416428 0.58318291 0.68051286]

K-Nearest Neighbors 9-neighbors
score avg:0.632885355382013
score = [0.67668746 0.60831042 0.61907927 0.57492296 0.68542667]

K-Nearest Neighbors 10-neighbors
score avg:0.6343371385873067
score = [0.68224372 0.59672552 0.61501991 0.59069253 0.687004  ]

K-Nearest Neighbors 11-neighbors
score avg:0.6368196828026085
score = [0.68112977 0.60771908 0.62019947 0.59000226 0.68504784]

K-Nearest Neighbors 12-neighbors
score avg:0.6317836859241253
score = [0.6813803  0.60809615 0.60372562 0.58645652 0.67925983]
```

Random Forests can get up to ~88-89% accuracy with 15-20 trees of depth 7-9

K-Nearest Neighbors optimizes at ~63% accuracy with 6 neighbors

---

5 - Trying to predict how many workers a player produces per a given amount of time

In [42]:
```
X = data.iloc[:,data.columns != 'WorkersMade']
Y = data.iloc[:,data.columns == 'WorkersMade']
#transform input data (normalize scaling)
ssc = SSc()
Xft = ssc.fit_transform(X)
X = pd.DataFrame(Xft)
print("Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes: ")
Xtr,Xtst,Ytr,Ytst = train_test_split(X,Y.values.ravel(),test_size=0.2,random
print(Xtr.shape,Xtst.shape,Ytr.shape,Ytst.shape)
```

```
Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes:
(2670, 25) (668, 25) (2670,) (668,)
```

In [47]:

```
print("Classifier scores:\n"+"-"*18+"\n")
for i in range(1,4):
    tree = DTR(max_depth=i+1)
    tree.fit(Xtr,Ytr)
    scr = cross_val_score(tree,Xtst,Ytst, cv=5)
    print("Tree of depth "+str(i)+"\nscore avg:"+str(sum(scr)/5)+"\nscore =

for i in range(1,29,3):
    dpth = 7
    forest = RFR(n_estimators=i,max_depth=dpth)
    forest.fit(Xtr,Ytr)
    scr = cross_val_score(forest,Xtst,Ytst, cv=5)
    print("\nRandom Forest trees = "+str(i)+" depth = "+str(dpth)+" \nscore

print("\n"+"-"*64)

for i in range(1,100,5):
    knn = KNR(n_neighbors=i)
    knn.fit(Xtr,Ytr)
    scr = cross_val_score(knn,Xtst,Ytst, cv=5)
    print("\nK-Nearest Neighbors "+str(i)+"-neighbors\nscore avg:"+str(sum(s
          str(scr))
```

```
Classifier scores:
------------------

Tree of depth 1
score avg:0.09921620818689791
score = [ 0.15621548  0.04931906  0.17505456  0.14254622 -0.02705429]

----------------------------------------------------------------
Tree of depth 2
score avg:0.03793248019195825
score = [ 0.15725063  0.00895247  0.16464879  0.14688575 -0.28807523]

----------------------------------------------------------------
Tree of depth 3
score avg:-0.0816491485962854
score = [ 0.07337114 -0.06509747  0.19142933  0.11847134 -0.72642009]

----------------------------------------------------------------

Random Forest trees = 1 depth = 7
score avg: -0.7414144706368789
scores: [-0.43475375 -1.42011903 -0.74392732 -0.17368379 -0.93458846]

Random Forest trees = 4 depth = 7
score avg: 0.08577150304789016
scores: [0.1715485  0.06118097 0.06990596 0.05953024 0.06669184]

Random Forest trees = 7 depth = 7
score avg: 0.07146769135154671
scores: [ 0.12213111  0.03732887  0.1954776   0.14339019 -0.14098931]

Random Forest trees = 10 depth = 7
score avg: 0.0982629499686811
scores: [ 0.16924934  0.02908447  0.17478122  0.12999787 -0.01179816]
```

```
Random Forest trees = 13 depth = 7
score avg: 0.11775964543271784
scores: [ 0.15729425  0.07824387  0.2705754   0.10673915 -0.02405445]

Random Forest trees = 16 depth = 7
score avg: 0.13047721413864638
scores: [ 0.16856197  0.17102691  0.25400847  0.14632945 -0.08754073]

Random Forest trees = 19 depth = 7
score avg: 0.14751612352814897
scores: [0.20922185 0.05859395 0.26878019 0.16896952 0.03201511]

Random Forest trees = 22 depth = 7
score avg: 0.15309172509825641
scores: [ 0.20649196  0.15201127  0.22486803  0.1917899  -0.00970254]

Random Forest trees = 25 depth = 7
score avg: 0.15023010929063746
scores: [0.1877657  0.11934215 0.24285998 0.12472995 0.07645276]

Random Forest trees = 28 depth = 7
score avg: 0.16603272658181517
scores: [0.20594698 0.17580056 0.24119975 0.18341532 0.02380102]

----------------------------------------------------------------

K-Nearest Neighbors 1-neighbors
score avg:-0.7312257724623157
score = [-0.62321509 -1.19330809 -0.43973199 -0.04945895 -1.35041475]

K-Nearest Neighbors 6-neighbors
score avg:0.03548319357503642
score = [ 0.08933592  0.07371106  0.06828884  0.14581637 -0.19973623]

K-Nearest Neighbors 11-neighbors
score avg:0.11560208752138815
score = [0.12870498 0.09134815 0.15270893 0.20348898 0.00175939]

K-Nearest Neighbors 16-neighbors
score avg:0.14332669723453076
score = [0.14846681 0.13908103 0.1965843  0.2153767  0.01712464]

K-Nearest Neighbors 21-neighbors
score avg:0.14343235900161871
score = [0.14674044 0.09937145 0.21303736 0.24634581 0.01166674]

K-Nearest Neighbors 26-neighbors
score avg:0.1419363473186543
score = [0.17568419 0.08775363 0.18530153 0.23327969 0.02766269]

K-Nearest Neighbors 31-neighbors
score avg:0.14073132627082235
score = [0.1651927  0.09864468 0.17957141 0.22424565 0.0360022 ]

K-Nearest Neighbors 36-neighbors
score avg:0.14968484169386184
```

```
score = [0.17586316 0.10736532 0.1882896  0.21784601 0.05906012]

K-Nearest Neighbors 41-neighbors
score avg:0.14053792375062693
score = [0.15564205 0.09977602 0.17768223 0.1986497  0.07093963]

K-Nearest Neighbors 46-neighbors
score avg:0.1469913750149638
score = [0.15848569 0.12252366 0.17369551 0.19544226 0.08480976]

K-Nearest Neighbors 51-neighbors
score avg:0.14898120250387417
score = [0.15656798 0.1228122  0.16910834 0.18954036 0.10687713]

K-Nearest Neighbors 56-neighbors
score avg:0.1482489008002776
score = [0.15179278 0.12101764 0.17444961 0.1905085  0.10347597]

K-Nearest Neighbors 61-neighbors
score avg:0.1476347372260527
score = [0.15648702 0.1099777  0.17628918 0.18636256 0.10905722]

K-Nearest Neighbors 66-neighbors
score avg:0.14910255609556575
score = [0.15160124 0.1265937  0.17633484 0.18411452 0.10686849]

K-Nearest Neighbors 71-neighbors
score avg:0.14867215203363282
score = [0.15787009 0.12952669 0.16509348 0.1831236  0.10774691]

K-Nearest Neighbors 76-neighbors
score avg:0.15337556643535716
score = [0.16242727 0.14108171 0.16430283 0.18583737 0.11322866]

K-Nearest Neighbors 81-neighbors
score avg:0.1546312773650498
score = [0.16153244 0.14864355 0.16174689 0.18748208 0.11375143]

K-Nearest Neighbors 86-neighbors
score avg:0.1591109815195545
score = [0.16113959 0.15905242 0.16595043 0.18442945 0.12498302]

K-Nearest Neighbors 91-neighbors
score avg:0.15549335258459768
score = [0.15858422 0.15033624 0.16543752 0.18192002 0.12118876]

K-Nearest Neighbors 96-neighbors
score avg:0.15373633369077455
score = [0.16131383 0.14569034 0.1637368  0.17720823 0.12073247]
```

It seems almost impossible to accurately predict this metric. I think this may be at least partially due to the nature of the game being that if players are playing aggressively then much fewer workers will be made while if players are playing defensive they will be making a lot of workers. I think it would be possible to predict this in a given game with more specific game statistics, but since these are average player stats it is almost impossible.

In [ ]:

```
1
```