In [24]:
```
%load_ext autoreload
%autoreload 2

%matplotlib inline
```
```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

In [25]:
```
from fastai.imports import *
from fastai.structured import *
import numpy as np
import pandas as pd
from pandas_summary import DataFrameSummary
import sklearn.model_selection
from IPython.display import display
import math
import random
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import collections
```

In [26]:
```
PATH = "data/Cristano_Ronaldo_Final_v1/"
```

In [27]:
```
!ls {PATH}
```
```
amit_dubey_190199_code_4.csv  data.csv  sample_submission.csv
amit_dubey_190199_code_5.csv  __MACOSX
```

Required Functions

In [28]:
```
def imae(x,y):
    return 1/(1+(abs(x-y)).mean())
```

In [29]:
```
def print_score(m):
    res = [
        imae(m.predict(X_train.drop(['Unnamed: 0'],axis=1)), y_train),
        imae(m.predict(X_valid.drop(['Unnamed: 0'],axis=1)), y_valid),
        m.score(X_train.drop(['Unnamed: 0'],axis=1), y_train),
        m.score(X_valid.drop(['Unnamed: 0'],axis=1), y_valid),
          ]
    if hasattr(m, 'oob_score_'): res.append(m.oob_score_)
    print(res)
```

In [30]:
```
def display_all(df):
    with pd.option_context("display.max_rows", 1000, "display.max_columns", 10
        display(df)
```

Data Pre-processing

In [31]:
```
df_i = pd.read_csv(f'{PATH}sample_submission.csv')
df_i.shot_id_number = df_i.shot_id_number-1
df_i=df_i.drop(['is_goal'], axis=1)
```

In [32]:
```
df_raw = pd.read_csv(f'{PATH}data.csv', low_memory=False, parse_dates=['date_of
```

In [33]:
```
df_raw.is_goal.value_counts()
```

Out[33]:
```
0.0    13550
1.0    10879
Name: is_goal, dtype: int64
```

```
In [34]: display_all(df_raw.T)
```

	0	1	2	3	4	5	
Unnamed: 0	0	1	2	3	4	5	
match_event_id	10	12	35	43	155	244	
location_x	167	-157	-101	138	0	-145	
location_y	72	0	135	175	0	-11	
remaining_min	10	10	7	6	NaN	9	
power_of_shot	1	1	1	1	2	3	
knockout_match	0	0	0	0	0	0	
game_season	2000-01	2000-01	2000-01	2000-01	2000-01	NaN	20
remaining_sec	27	22	45	52	19	32	
distance_of_shot	38	35	36	42	20	34	
is_goal	NaN	0	1	0	1	0	
area_of_shot	Right Side(R)	Left Side(L)	Left Side Center(LC)	Right Side Center(RC)	Center(C)	Left Side(L)	Cer
shot_basics	Mid Range	Mid Range	Mid Range	Mid Range	Goal Area	Mid Range	Goa
range_of_shot	16-24 ft.	8-16 ft.	16-24 ft.	16-24 ft.	Less Than 8 ft.	8-16 ft.	Less T
team_name	Manchester United	Manchester United	Manchester United	Manchester United	NaN	Manchester United	Manc
date_of_game	2000-10-31 00:00:00	2000-10-31 00:00:00	2000-10-31 00:00:00	2000-10-31 00:00:00	2000-10-31 00:00:00	2000-10-31 00:00:00	2000 00
home/away	MANU @ POR	MANU @ POR	NaN	MANU @ POR	MANU @ POR	MANU @ POR	MA
shot_id_number	1	2	3	4	5	6	
lat/lng	45.539131, -122.651648	45.539131, -122.651648	45.539131, -122.651648	45.539131, -122.651648	45.539131, -122.651648	45.539131, -122.651648	45.5 -122.6
type_of_shot	shot - 30	shot - 45	shot - 25	NaN	NaN	shot - 17	
type_of_combined_shot	NaN	NaN	NaN	shot - 3	shot - 1	NaN	s
match_id	20000012	20000012	20000012	20000012	20000012	20000012	200
team_id	1610612747	1610612747	1610612747	1610612747	1610612747	1610612747	16106
remaining_min.1	10	10	92.64	NaN	42.64	9	
power_of_shot.1	1	1	1	1	2	3	
knockout_match.1	50.608	28.8	0	122.608	0	0	
remaining_sec.1	54.2	22	63.7216	52	19	NaN	
distance_of_shot.1	38	35	54.4	42	20	34	

28 rows × 30697 columns

In [35]:
```
df_raw['date_of_game'] = pd.to_datetime(df_raw.date_of_game)
df_raw=df_raw.sort_values('date_of_game')
display_all(df_raw.T)
```

	22901	22903	22904	22905	22906	22907
Unnamed: 0	22901	22903	22904	22905	22906	22907
match_event_id	102	124	144	151	157	226
location_x	-140	-142	NaN	-10	75	-64
location_y	116	181	0	138	177	223
remaining_min	0	8	6	5	7	2
power_of_shot	1	2	2	2	2	2
knockout_match	0	0	0	0	0	0
game_season	1996-97	1996-97	1996-97	1996-97	1996-97	NaN
remaining_sec	42	37	34	27	18	16
distance_of_shot	38	43	20	33	39	43
is_goal	0	1	0	1	NaN	1

In [72]:
```
"""
I tried this but it lead to worse r^2 score :/ so its commented now
lst = [
    'is_goal',
    'knockout_match',
    'game_season',
    'shot_basics',
    'team_name',
    'home/away',
    'lat/lng',
    'type_of_combined_shot',
    'match_id',
    'team_id',
    'knockout_match.1',
]
for col in lst:
    df_raw[col].interpolate(method='nearest',inplace=True)
    """;
```

train_cats

It change any columns of strings in a panda's dataframe to a column of categorical values. This applies the changes inplace.

In [37]:
```
train_cats(df_raw)
```

In [38]:
```
cols = ['knockout_match','match_event_id', 'game_season', 'area_of_shot','shot_
       'home/away', 'shot_id_number',  'type_of_shot', 'type_of_combined_shot',
```

In [39]:
```
for col in cols:
    df_raw[col] = df_raw[col].astype('category').cat.codes
```

In [40]:
```
df_raw['year'] = df_raw['date_of_game'].dt.year
df_raw['month']=df_raw['date_of_game'].dt.month
df_raw=df_raw.drop(['date_of_game'],axis=1)
```

In [41]:
```
df_raw          = proc_df(df_raw)
```

proc_df

It takes a data frame df and splits off the response variable, and changes the df into an entirely numeric dataframe. For each column of df which is not in skip_flds nor in ignore_flds, na values are replaced by the median value of the column.

```
In [42]: df_raw.match_event_id.value_counts()
```

```
Out[42]: -1      1563
          0       128
          2       102
          9        92
          276      88
          6        87
          15       86
          10       85
          4        85
          316      83
          247      82
          265      82
          335      82
          7        81
          24       81
          255      80
          22       80
          311      79
          100      79
          254      79
          11       79
          237      79
          86       79
          25       78
          71       78
          236      78
          240      78
          301      77
          14       77
          269      77
                 ...
          572       2
          581       2
          612       2
          596       2
          595       2
          579       2
          604       2
          594       2
          573       2
          577       2
          586       2
          588       1
          617       1
          603       1
          587       1
          606       1
          602       1
          592       1
          585       1
          616       1
          615       1
          599       1
          608       1
          614       1
          597       1
          611       1
          610       1
          609       1
          593       1
          607       1
         Name: match_event_id, Length: 619, dtype: int64
```

```
In [43]: df_tst = df_raw[df_raw['Unnamed: 0'].isin(df_i['shot_id_number'])]
         df_trn = df_raw[~df_raw['Unnamed: 0'].isin(df_i['shot_id_number'])]
```

In [44]:
```
display_all(df_trn.T)
```

	22901	22903	22904	22905	22907	22909	22910	22911	22912	22913	22914
Unnamed: 0	22901	22903	22904	22905	22907	22909	22910	22911	22912	22913	22914
match_event_id	100	122	142	149	224	332	335	350	378	382	105
location_x	-140	-142	0	-10	-64	-79	-103	0	-155	0	0
location_y	116	181	0	138	223	177	207	0	175	0	0
remaining_min	0	8	6	5	2	1	1	0	9	8	1
power_of_shot	1	2	2	2	2	3	3	3	4	4	1
knockout_match	0	0	0	0	0	-1	0	0	0	0	0
game_season	0	0	0	0	-1	0	0	0	-1	0	-1
remaining_sec	42	37	34	27	16	53	14	2	9	36	10
distance_of_shot	38	43	20	33	43	39	43	20	43	20	20
is_goal	0	1	0	1	1	0	1	0	0	0	0

In [45]:
```
df_trn.describe()
```

Out[45]:

	Unnamed: 0	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout
count	25697.000000	25697.000000	25697.000000	25697.000000	25697.000000	25697.000000	25697
mean	15327.166946	235.127602	7.105421	90.453438	4.891116	2.545122	0
std	8860.462397	155.817575	107.559386	85.810451	3.365993	1.128151	0
min	1.000000	-1.000000	-250.000000	-44.000000	0.000000	1.000000	-1
25%	7645.000000	90.000000	-59.000000	7.000000	2.000000	2.000000	0
50%	15335.000000	241.000000	0.000000	74.000000	5.000000	3.000000	0
75%	22975.000000	358.000000	90.000000	156.000000	8.000000	3.000000	0
max	30696.000000	616.000000	248.000000	791.000000	11.000000	7.000000	1

8 rows × 29 columns

In [46]:
```
 X_train, X_valid, y_train, y_valid  = sklearn.model_selection.train_test_split
```

In [47]:
```
X_train.shape, X_valid.shape, y_train.shape, y_valid.shape
```

Out[47]:
```
((20557, 42), (5140, 42), (20557,), (5140,))
```

Model Selection & Analysis

set_rf_samples(n)

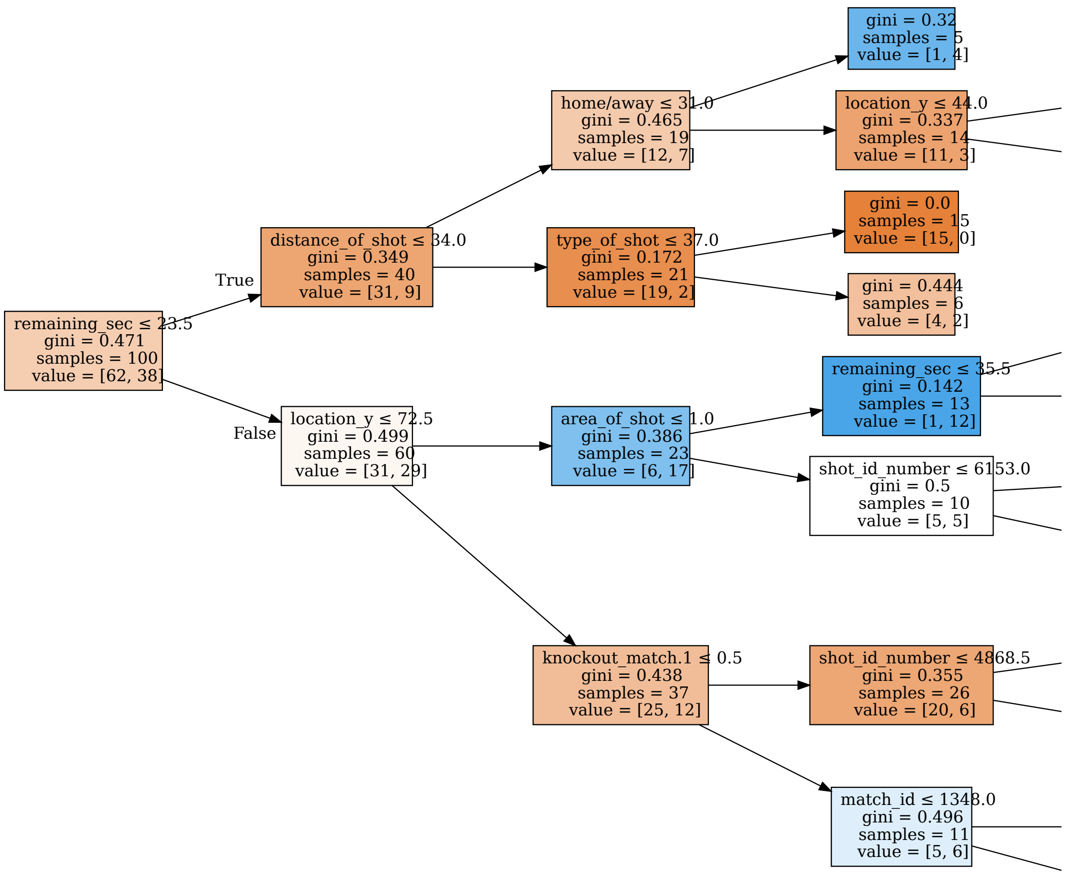
Changes Scikit learn's random forests to give each tree a random sample of n random rows.

In [67]:
```
set_rf_samples(100)
```

In [68]:
```
clf = RandomForestClassifier(n_estimators=1000, max_depth=10, max_features=0.5,
clf.fit(X_train.drop(['Unnamed: 0'],axis=1), y_train)
print_score(clf)
```

```
[0.7244502396391317, 0.7206954570947841, 0.6196429440093398, 0.612451361867704
3]
```

In [50]:
```
draw_tree(clf.estimators_[0], X_train.drop(['Unnamed: 0'],axis=1), precision=3)
```

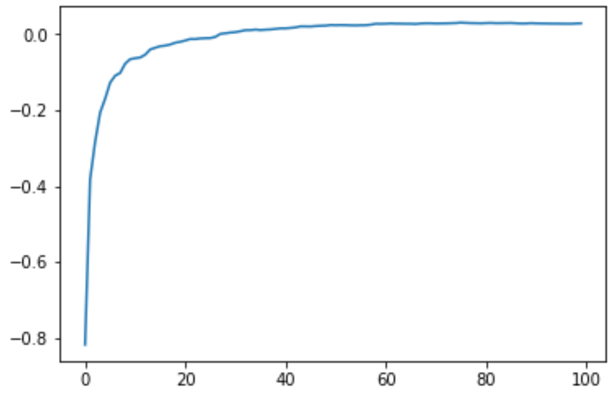


In [51]:
```
pred_valid =  clf.predict(X_train.drop(['Unnamed: 0'],axis=1))
collections.Counter(pred_valid)
```

Out[51]:
```
Counter({0.0: 15933, 1.0: 4624})
```

In [52]:
```
preds = np.stack([t.predict(X_valid.drop(['Unnamed: 0'],axis=1)) for t in clf.e
#preds[:,0], np.mean(preds[:,0]), y_valid[0]
```

In [65]:
```
plt.plot([metrics.r2_score(y_valid, np.mean(preds[:i+1], axis=0)) for i in rang
```



Feature Importance

In [54]:
```
fi = rf_feat_importance(clf, X_valid.drop(['Unnamed: 0'],axis=1))
fi[:]
```
Out[54]:

	cols	imp
2	location_y	0.068470
0	match_event_id	0.068415
8	distance_of_shot	0.065186
13	home/away	0.063609
23	remaining_sec.1	0.059963
1	location_x	0.057858
24	distance_of_shot.1	0.057441
7	remaining_sec	0.054528
18	match_id	0.051288
14	shot_id_number	0.049876
20	remaining_min.1	0.042990

Final model

In [55]:
```
clf.fit(df_trn.drop(['Unnamed: 0','is_goal'],axis=1), df_trn['is_goal'])
```
Out[55]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=10, max_features=0.5, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=1000,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```
In [56]:
```
df_tst = df_tst.drop(['is_goal'],axis=1)
```
In [57]:
```
df_trn.T
```
Out[57]:

	22901	22903	22904	22905	22907	22909	22910	22911	22912	22913	...	305
Unnamed: 0	22901	22903	22904	22905	22907	22909	22910	22911	22912	22913	...	305
match_event_id	100	122	142	149	224	332	335	350	378	382	...	1
location_x	-140	-142	0	-10	-64	-79	-103	0	-155	0	...	-1
location_y	116	181	0	138	223	177	207	0	175	0	...	
remaining_min	0	8	6	5	2	1	1	0	9	8	...	
power_of_shot	1	2	2	2	2	3	3	3	4	4	...	
knockout_match	0	0	0	0	0	-1	0	0	0	0	...	
game_season	0	0	0	0	-1	0	0	0	-1	0	...	
remaining_sec	42	37	34	27	16	53	14	2	9	36	...	
distance_of_shot	38	43	20	33	43	39	43	20	43	20	...	
is_goal	0	1	0	1	1	0	1	0	0	0	...	

In [58]:
```
pred_tst = clf.predict(df_tst.drop(['Unnamed: 0'],axis=1))
```
In [59]:
```
pred_tst
```
Out[59]:
```
array([0., 1., 0., ..., 0., 0., 1.])
```

In [60]:
```
df_ans=pd.DataFrame()
df_ans['id']=df_tst['Unnamed: 0']+1
```

In [61]:
```
df_ans['prediction']=pred_tst.astype('int')
```

In [62]:
```
df_ans=df_ans.sort_values('id')
```

In [63]:
```
#df_ans.prediction = df_ans.prediction.astype('int')
df_ans.to_csv(f'{PATH}/amit_dubey_190199_code_5.csv', index=False)
```

In [64]:
```
df_ans.T
```

Out[64]:

	0	**7**	**16**	**19**	**32**	**33**	**34**	**35**	**36**	**37**	**...**	**30646**	**30648**	**30655**	**30659**	**30664**	**30668**	**30680**	**30**
id	1	8	17	20	33	34	35	36	37	38	...	30647	30649	30656	30660	30665	30669	30681	30
prediction	0	0	1	1	0	0	1	0	0	0	...	0	0	0	0	0	0	1	

2 rows × 5000 columns