Random Forest and Decision Tree Prediction of Career Batting Average

In [1]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
```

In [ ]:
```
```

In [2]:
```
# read in the data
master_df = pd.read_csv("./data/Master.csv")
batting_df = pd.read_csv("./data/Batting.csv")
batting_df
```

Out[2]:

	playerID	yearID	stint	teamID	lgID	G	AB	R	H	2B	...	RBI	SB	CS	BB	SO	IBB	HBP	SH	SF	GIDP
0	abercda01	1871	1	TRO	NaN	1	4.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN	NaN
1	addybo01	1871	1	RC1	NaN	25	118.0	30.0	32.0	6.0	...	13.0	8.0	1.0	4.0	0.0	NaN	NaN	NaN	NaN	NaN
2	allisar01	1871	1	CL1	NaN	29	137.0	28.0	40.0	4.0	...	19.0	3.0	1.0	2.0	5.0	NaN	NaN	NaN	NaN	NaN
3	allisdo01	1871	1	WS3	NaN	27	133.0	28.0	44.0	10.0	...	27.0	1.0	1.0	0.0	2.0	NaN	NaN	NaN	NaN	NaN
4	ansonca01	1871	1	RC1	NaN	25	120.0	29.0	39.0	11.0	...	16.0	6.0	2.0	2.0	1.0	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
101327	zitoba01	2015	1	OAK	AL	3	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
101328	zobribe01	2015	1	OAK	AL	67	235.0	39.0	63.0	20.0	...	33.0	1.0	1.0	33.0	26.0	2.0	0.0	0.0	3.0	5.0
101329	zobribe01	2015	2	KCA	AL	59	232.0	37.0	66.0	16.0	...	23.0	2.0	3.0	29.0	30.0	1.0	1.0	0.0	2.0	3.0
101330	zuninmi01	2015	1	SEA	AL	112	350.0	28.0	61.0	11.0	...	28.0	0.0	1.0	21.0	132.0	0.0	5.0	8.0	2.0	6.0
101331	zychto01	2015	1	SEA	AL	13	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

101332 rows × 22 columns

In [3]:
```
# process data that we want to use
batting_df2 = batting_df.drop(['yearID', 'stint', 'teamID', 'lgID'], axis=1)
batting_df2 = batting_df2.fillna(0)
batting_df2 = batting_df2.groupby(['playerID']).sum()
batting_averages = (batting_df2['H']/batting_df2['AB']).fillna(0) # compute the career batting averages
batting_df2.reset_index(inplace=True)
batting_df2 = batting_df2.drop(['playerID','H', 'AB'], axis=1)
batting_df2
```

Out[3]:

	G	R	2B	3B	HR	RBI	SB	CS	BB	SO	IBB	HBP	SH	SF	GIDP
0	331	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0
1	3298	2174.0	624.0	98.0	755.0	2297.0	240.0	73.0	1402.0	1383.0	293.0	32.0	21.0	121.0	328.0
2	437	102.0	42.0	6.0	13.0	94.0	9.0	8.0	86.0	145.0	3.0	0.0	9.0	6.0	36.0
3	448	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0
4	15	1.0	0.0	0.0	0.0	0.0	0.0	1.0	4.0	5.0	0.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18654	16	3.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	6.0	0.0	0.0	0.0	0.0	0.0
18655	209	41.0	17.0	2.0	2.0	20.0	2.0	0.0	34.0	50.0	1.0	2.0	18.0	0.0	8.0
18656	266	5.0	2.0	1.0	0.0	7.0	0.0	1.0	9.0	39.0	0.0	0.0	16.0	0.0	3.0
18657	366	167.0	76.0	15.0	30.0	202.0	46.0	0.0	128.0	139.0	0.0	4.0	31.0	0.0	0.0
18658	13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

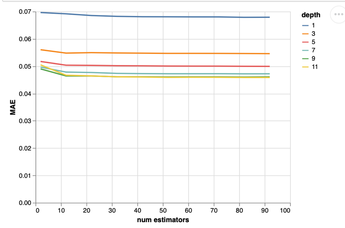
18659 rows × 15 columns

In [4]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

sc = StandardScaler()
columns = batting_df2.columns
batting_scaled = pd.DataFrame(sc.fit_transform(batting_df2))
batting_scaled.columns = columns
batting_scaled
X_train, X_test, y_train, y_test = train_test_split(batting_df2, batting_averages, test_size=0.2, random_state=0)
```

In [5]:
```
from sklearn.ensemble import RandomForestRegressor

nepochs = 101
max_depth = 12

epoch_lst = []
depth_lst = []
MAE_lst = []

for depth in range(1, max_depth, 2):
    for epoch in range(1, nepochs, 10):
        regressor = RandomForestRegressor(n_estimators=epoch, max_depth=depth, random_state=0)
        regressor.fit(X_train, y_train)
        y_pred = regressor.predict(X_test)

        epoch_lst.append(epoch+1)
        depth_lst.append(depth)
        MAE_lst.append(metrics.mean_absolute_error(y_test, y_pred))

results = pd.DataFrame({
    'num estimators' : epoch_lst,
    'depth' : depth_lst,
    'MAE' : MAE_lst
})
```

In [7]:
```
import altair as alt
alt.data_transformers.disable_max_rows()

alt.Chart(results).mark_line().encode(
    x='num estimators',
    y='MAE',
    color='depth:N'
)
```

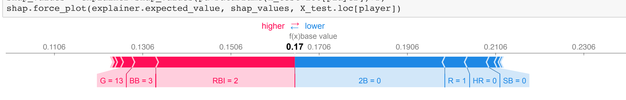
Out[7]:



In [8]:
```
import shap
shap.initjs()
```

In [9]:
```
player = X_test.index[2]
regressor = RandomForestRegressor(n_estimators=11, max_depth=11, random_state=0)
regressor.fit(X_train, y_train)

explainer = shap.TreeExplainer(regressor, X_test)
shap_values = explainer.shap_values(pd.DataFrame(X_test.loc[player]).T)
shap.force_plot(explainer.expected_value, shap_values, X_test.loc[player])
```

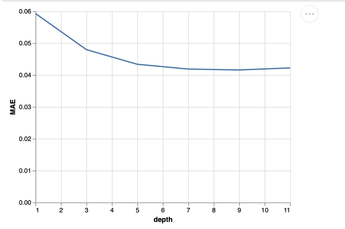
Out[9]:



In [10]:
```
from sklearn.tree import DecisionTreeRegressor

max_depth = 12

epoch_lst = []
depth_lst = []
MAE_lst = []

for depth in range(1, max_depth, 2):
#     for epoch in range(1, nepochs, 10):
    regressor = DecisionTreeRegressor(criterion='mae', max_depth=depth, random_state=0)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)

    epoch_lst.append(epoch+1)
    depth_lst.append(depth)
    MAE_lst.append(metrics.mean_absolute_error(y_test, y_pred))

results = pd.DataFrame({
    'depth' : depth_lst,
    'MAE' : MAE_lst
})
```

In [11]:
```
alt.Chart(results).mark_line().encode(
    x='depth',
    y='MAE'
)
```

Out[11]:



In [12]:
```
player = X_test.index[2]
regressor = DecisionTreeRegressor(criterion='mae', max_depth=depth, random_state=0)
regressor.fit(X_train, y_train)

explainer = shap.TreeExplainer(regressor, X_test)
shap_values = explainer.shap_values(pd.DataFrame(X_test.loc[player]).T)
shap.force_plot(explainer.expected_value, shap_values, X_test.loc[player])
```

Out[12]:

In [ ]:
```
```