model_batting

April 27, 2020

[1]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
```

[2]:
```
df = pd.read_csv('../core/output/batters.csv')
indexer = df.reset_index()[['index', 'retroID']].to_dict()['retroID']
y = df['Batting'].values
to_drop = ['debutYear', 'finalYear', 'G', '1B', 'AB', 'RBI', 'wOBA']
df.drop(columns=to_drop, inplace=True)
```

[3]:
```
df
```

[3]:

	retroID	weight	height	pos_1B	pos_2B	pos_3B	pos_C	pos_OF	\
0	aardd001	0.569672	0.60	0	0	0	0	0	
1	aaroh101	0.426230	0.45	0	0	0	0	1	
2	aarot101	0.467213	0.60	1	0	0	0	0	
3	aased001	0.467213	0.60	0	0	0	0	0	
4	abada001	0.442623	0.50	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	
15288	zupcb001	0.590164	0.65	0	0	0	0	1	
15289	zupof101	0.434426	0.40	0	0	0	1	0	
15290	zuveg101	0.487705	0.65	0	0	0	0	0	
15291	zuvep001	0.397541	0.45	0	0	0	0	0	
15292	zycht001	0.467213	0.60	0	0	0	0	0	

	pos_P	pos_SS	...	SO	IBB	HBP	SH	SF	GIDP	NL	wRC+	WAR	\
0	1	0	...	2	0	0	1	0	0	1	-100	-0.1	
1	0	0	...	1383	293	32	21	121	328	1	153	136.3	
2	0	0	...	145	3	0	9	6	36	1	76	-1.7	
3	1	0	...	3	0	0	0	0	0	1	-100	-0.1	
4	0	0	...	5	0	0	0	0	1	1	0	-0.4	
...	...	...	...	...	...	...	..	...	...	..	...	...	
15288	0	0	...	137	3	6	20	8	15	0	74	-0.9	
15289	0	0	...	6	0	0	0	0	0	0	37	-0.2	
15290	1	0	...	39	0	0	16	0	3	1	0	-0.3	
15291	0	1	...	50	1	2	18	0	8	1	52	-2.2	
15292	1	0	...	0	0	0	0	0	0	0	0	0.0	

```
        Batting
0      0.000358
1      0.350195
2      0.157131
3      0.000358
4      0.090001
...         ...
15288  0.156062
15289  0.123425
15290  0.090088
15291  0.135118
15292  0.090195

[15293 rows x 31 columns]
```

Building the Model

[4]:
```
from sklearn.model_selection import train_test_split
```

[5]:
```
X = df.drop(columns=['Batting']).values
y = df[['retroID', 'Batting']].values
```

When we do our train-test split, since it's random in how it splits up the data, we need to keep track of the appropriate keys (retro IDs) for each data point.

[6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=101)
X_train_keys = np.asarray([x[0] for x in X_train])
X_train = np.asarray([x[1:] for x in X_train])
X_test_keys = np.asarray([x[0] for x in X_test])
X_test = np.asarray([x[1:] for x in X_test])
y_train_keys = np.asarray([y[0] for y in y_train])
y_train = np.asarray([y[1] for y in y_train])
y_test_keys = np.asarray([y[0] for y in y_test])
y_test = np.asarray([y[1] for y in y_test])
```

[7]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import regularizers
```

[8]:
```
X_train.shape
```

[8]:
```
(12234, 29)
```

[9]:
```
from sklearn.preprocessing import MinMaxScaler
```

[10]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[11]:
```
def to_tensor_input(player):
    return scaler.transform(player.values.reshape(-1,29))[0]
```

[12]:
```
tensor = df.drop(columns=['retroID', 'Batting'])
player_tensor_inputs = tensor.apply(lambda player: to_tensor_input(player),␣
 ↪axis=1)
```

[13]:
```
player_tensor_inputs
```

[13]:
```
0         [0.5696720000000001, 0.6, 0.0, 0.0, 0.0, 0.0, ...
1         [0.42623, 0.45, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, ...
2         [0.467213, 0.6, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
3         [0.467213, 0.6, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, ...
4         [0.44262299999999993, 0.5, 1.0, 0.0, 0.0, 0.0,...
                                ...
15288     [0.590164, 0.65, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,...
15289     [0.43442600000000003, 0.4, 0.0, 0.0, 0.0, 1.0,...
15290     [0.487705, 0.65, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,...
15291     [0.397541, 0.45, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...
15292     [0.467213, 0.6, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, ...
Length: 15293, dtype: object
```

[14]:
```
tensor = pd.DataFrame(player_tensor_inputs.values.tolist())
```

[15]:
```
tensor.to_csv('../core/tensors/t_batting.csv', index=False, float_format='%g')
```

[16]:
```
epochs = 400
batch_size = 128
loss_param = 'mse'
optimizer_param = 'adam'
stop_monitor = 'val_loss'
stop_patience = 6
```

[17]:
```
model = Sequential()

model.add(Dense(29, activation='relu', kernel_regularizer=regularizers.l2(0.
 ↪0001)))
model.add(Dropout(0.5))

model.add(Dense(58, activation='relu', kernel_regularizer=regularizers.l2(0.
 ↪0001)))
model.add(Dropout(0.5))
```

```
model.add(Dense(116, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(232, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(116, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(58, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(29, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(14, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(7, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(units=1, activation='sigmoid'))

model.compile(loss=loss_param, optimizer=optimizer_param)
```

[18]:
```
from tensorflow.keras.callbacks import EarlyStopping
```

[19]:
```
early_stop = EarlyStopping(monitor=stop_monitor, patience=stop_patience)
```

[20]:
```
results = model.fit(x=X_train, y=y_train,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(X_test, y_test),
                        callbacks=[early_stop]
                    )
```

```
Train on 12234 samples, validate on 3059 samples
Epoch 1/400
12234/12234 [==============================] - 2s 135us/sample - loss: 0.1964 -
val_loss: 0.1657
```

```
Epoch 2/400
12234/12234 [==============================] - 0s 37us/sample - loss: 0.1357 -
val_loss: 0.0755
Epoch 3/400
12234/12234 [==============================] - 0s 34us/sample - loss: 0.0871 -
val_loss: 0.0368
Epoch 4/400
12234/12234 [==============================] - 0s 33us/sample - loss: 0.0676 -
val_loss: 0.0287
Epoch 5/400
12234/12234 [==============================] - 0s 34us/sample - loss: 0.0567 -
val_loss: 0.0233
Epoch 6/400
12234/12234 [==============================] - 0s 35us/sample - loss: 0.0485 -
val_loss: 0.0191
Epoch 7/400
12234/12234 [==============================] - 0s 38us/sample - loss: 0.0424 -
val_loss: 0.0161
Epoch 8/400
12234/12234 [==============================] - 1s 42us/sample - loss: 0.0372 -
val_loss: 0.0136
Epoch 9/400
12234/12234 [==============================] - 1s 52us/sample - loss: 0.0328 -
val_loss: 0.0120
Epoch 10/400
12234/12234 [==============================] - 0s 37us/sample - loss: 0.0294 -
val_loss: 0.0108
Epoch 11/400
12234/12234 [==============================] - 0s 33us/sample - loss: 0.0272 -
val_loss: 0.0098
Epoch 12/400
12234/12234 [==============================] - 0s 38us/sample - loss: 0.0246 -
val_loss: 0.0091
Epoch 13/400
12234/12234 [==============================] - 0s 34us/sample - loss: 0.0229 -
val_loss: 0.0087
Epoch 14/400
12234/12234 [==============================] - 0s 33us/sample - loss: 0.0210 -
val_loss: 0.0081
Epoch 15/400
12234/12234 [==============================] - 0s 34us/sample - loss: 0.0193 -
val_loss: 0.0078
Epoch 16/400
12234/12234 [==============================] - 1s 43us/sample - loss: 0.0178 -
val_loss: 0.0076
Epoch 17/400
12234/12234 [==============================] - 1s 44us/sample - loss: 0.0168 -
val_loss: 0.0073
```

```
Epoch 18/400
12234/12234 [==============================] - 1s 58us/sample - loss: 0.0158 -
val_loss: 0.0072
Epoch 19/400
12234/12234 [==============================] - 1s 58us/sample - loss: 0.0148 -
val_loss: 0.0070
Epoch 20/400
12234/12234 [==============================] - 1s 42us/sample - loss: 0.0138 -
val_loss: 0.0068
Epoch 21/400
12234/12234 [==============================] - 1s 47us/sample - loss: 0.0127 -
val_loss: 0.0067
Epoch 22/400
12234/12234 [==============================] - 1s 42us/sample - loss: 0.0122 -
val_loss: 0.0066
Epoch 23/400
12234/12234 [==============================] - 0s 36us/sample - loss: 0.0114 -
val_loss: 0.0065
Epoch 24/400
12234/12234 [==============================] - 1s 51us/sample - loss: 0.0109 -
val_loss: 0.0064
Epoch 25/400
12234/12234 [==============================] - 1s 73us/sample - loss: 0.0105 -
val_loss: 0.0064
Epoch 26/400
12234/12234 [==============================] - 1s 66us/sample - loss: 0.0098 -
val_loss: 0.0063
Epoch 27/400
12234/12234 [==============================] - 1s 58us/sample - loss: 0.0093 -
val_loss: 0.0062
Epoch 28/400
12234/12234 [==============================] - 1s 50us/sample - loss: 0.0091 -
val_loss: 0.0062
Epoch 29/400
12234/12234 [==============================] - 1s 47us/sample - loss: 0.0086 -
val_loss: 0.0061
Epoch 30/400
12234/12234 [==============================] - ETA: 0s - loss: 0.008 - 1s
55us/sample - loss: 0.0083 - val_loss: 0.0061
Epoch 31/400
12234/12234 [==============================] - 0s 33us/sample - loss: 0.0082 -
val_loss: 0.0060
Epoch 32/400
12234/12234 [==============================] - 1s 55us/sample - loss: 0.0078 -
val_loss: 0.0060
Epoch 33/400
12234/12234 [==============================] - 1s 46us/sample - loss: 0.0076 -
val_loss: 0.0059
```

```
Epoch 34/400
12234/12234 [==============================] - 1s 42us/sample - loss: 0.0074 -
val_loss: 0.0059
Epoch 35/400
12234/12234 [==============================] - 1s 46us/sample - loss: 0.0072 -
val_loss: 0.0058
Epoch 36/400
12234/12234 [==============================] - 1s 45us/sample - loss: 0.0070 -
val_loss: 0.0058
Epoch 37/400
12234/12234 [==============================] - 1s 44us/sample - loss: 0.0070 -
val_loss: 0.0058
Epoch 38/400
12234/12234 [==============================] - 0s 38us/sample - loss: 0.0067 -
val_loss: 0.0058
Epoch 39/400
12234/12234 [==============================] - 0s 38us/sample - loss: 0.0066 -
val_loss: 0.0058
Epoch 40/400
12234/12234 [==============================] - 0s 31us/sample - loss: 0.0065 -
val_loss: 0.0058
Epoch 41/400
12234/12234 [==============================] - 0s 32us/sample - loss: 0.0064 -
val_loss: 0.0057
Epoch 42/400
12234/12234 [==============================] - 0s 30us/sample - loss: 0.0063 -
val_loss: 0.0057
Epoch 43/400
12234/12234 [==============================] - 0s 30us/sample - loss: 0.0062 -
val_loss: 0.0057
Epoch 44/400
12234/12234 [==============================] - 0s 31us/sample - loss: 0.0061 -
val_loss: 0.0057
Epoch 45/400
12234/12234 [==============================] - 0s 35us/sample - loss: 0.0061 -
val_loss: 0.0057
Epoch 46/400
12234/12234 [==============================] - 1s 55us/sample - loss: 0.0059 -
val_loss: 0.0056
Epoch 47/400
12234/12234 [==============================] - 1s 41us/sample - loss: 0.0060 -
val_loss: 0.0056
Epoch 48/400
12234/12234 [==============================] - 0s 38us/sample - loss: 0.0058 -
val_loss: 0.0056
Epoch 49/400
12234/12234 [==============================] - 0s 36us/sample - loss: 0.0058 -
val_loss: 0.0056
```

```
Epoch 50/400
12234/12234 [==============================] - 0s 31us/sample - loss: 0.0058 -
val_loss: 0.0056
Epoch 51/400
12234/12234 [==============================] - 1s 50us/sample - loss: 0.0057 -
val_loss: 0.0056
Epoch 52/400
12234/12234 [==============================] - 0s 37us/sample - loss: 0.0058 -
val_loss: 0.0056
Epoch 53/400
12234/12234 [==============================] - 0s 36us/sample - loss: 0.0057 -
val_loss: 0.0056
Epoch 54/400
12234/12234 [==============================] - 0s 40us/sample - loss: 0.0057 -
val_loss: 0.0056
Epoch 55/400
12234/12234 [==============================] - 1s 47us/sample - loss: 0.0057 -
val_loss: 0.0056
Epoch 56/400
12234/12234 [==============================] - 1s 43us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 57/400
12234/12234 [==============================] - 1s 60us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 58/400
12234/12234 [==============================] - 1s 75us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 59/400
12234/12234 [==============================] - 1s 64us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 60/400
12234/12234 [==============================] - 1s 63us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 61/400
12234/12234 [==============================] - 1s 63us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 62/400
12234/12234 [==============================] - 1s 68us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 63/400
12234/12234 [==============================] - 1s 66us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 64/400
12234/12234 [==============================] - 0s 38us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 65/400
12234/12234 [==============================] - 0s 31us/sample - loss: 0.0056 -
val_loss: 0.0056
```

```
Epoch 66/400
12234/12234 [==============================] - 0s 32us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 67/400
12234/12234 [==============================] - 0s 32us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 68/400
12234/12234 [==============================] - 0s 31us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 69/400
12234/12234 [==============================] - 0s 31us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 70/400
12234/12234 [==============================] - 0s 32us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 71/400
12234/12234 [==============================] - 0s 31us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 72/400
12234/12234 [==============================] - 0s 33us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 73/400
12234/12234 [==============================] - 0s 35us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 74/400
12234/12234 [==============================] - 1s 44us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 75/400
12234/12234 [==============================] - 1s 42us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 76/400
12234/12234 [==============================] - 0s 40us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 77/400
12234/12234 [==============================] - 0s 38us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 78/400
12234/12234 [==============================] - 0s 40us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 79/400
12234/12234 [==============================] - 1s 51us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 80/400
12234/12234 [==============================] - 0s 35us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 81/400
12234/12234 [==============================] - 0s 33us/sample - loss: 0.0056 -
val_loss: 0.0056
```

```
Epoch 82/400
12234/12234 [==============================] - 0s 32us/sample - loss: 0.0056 -
val_loss: 0.0056
Epoch 83/400
12234/12234 [==============================] - 0s 33us/sample - loss: 0.0056 -
val_loss: 0.0056
```

[21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                multiple                  870
_________________________________________________________________
dropout (Dropout)            multiple                  0
_________________________________________________________________
dense_1 (Dense)              multiple                  1740
_________________________________________________________________
dropout_1 (Dropout)          multiple                  0
_________________________________________________________________
dense_2 (Dense)              multiple                  6844
_________________________________________________________________
dropout_2 (Dropout)          multiple                  0
_________________________________________________________________
dense_3 (Dense)              multiple                  27144
_________________________________________________________________
dropout_3 (Dropout)          multiple                  0
_________________________________________________________________
dense_4 (Dense)              multiple                  27028
_________________________________________________________________
dropout_4 (Dropout)          multiple                  0
_________________________________________________________________
dense_5 (Dense)              multiple                  6786
_________________________________________________________________
dropout_5 (Dropout)          multiple                  0
_________________________________________________________________
dense_6 (Dense)              multiple                  1711
_________________________________________________________________
dropout_6 (Dropout)          multiple                  0
_________________________________________________________________
dense_7 (Dense)              multiple                  420
_________________________________________________________________
dropout_7 (Dropout)          multiple                  0
_________________________________________________________________
dense_8 (Dense)              multiple                  105
_________________________________________________________________
dropout_8 (Dropout)          multiple                  0
```

```
_________________________________________________________________
dense_9 (Dense)              multiple                  8
=================================================================
Total params: 72,656
Trainable params: 72,656
Non-trainable params: 0
_________________________________________________________________
```

[22]:
```
import os
```

[23]:
```
losses = model.history.history
losses['loss'] = np.asarray(losses['loss'])
losses['val_loss'] = np.asarray(losses['val_loss'])
final_number_of_epochs = len(losses['loss'])
min_loss = losses['loss'].min()
mean_loss = losses['loss'].mean()
final_loss = losses['loss'][-1]
min_val_loss = losses['val_loss'].min()
mean_val_loss = losses['val_loss'].mean()
final_val_loss = losses['val_loss'][-1]

def get_model_summary():
    output = []
    model.summary(print_fn=lambda line: output.append(line))
    return str(output).strip('[]')

summary = get_model_summary()

record = {
    'Epochs': final_number_of_epochs,
    'Batch_Size': batch_size,
    'Loss_Func': loss_param,
    'Optimizer': optimizer_param,
    'Early_Stop_Monitor': stop_monitor,
    'Early_Stop_Patience': stop_patience,
    'Min_Loss': min_loss,
    'Mean_Loss': mean_loss,
    'Final_Loss': final_loss,
    'Min_Val_Loss': min_val_loss,
    'Mean_Val_Loss': mean_val_loss,
    'Final_Val_Loss': final_val_loss,
    'Model': summary
}

new_data = pd.DataFrame(record, index=[0])

if os.path.exists('../core/records/batting_results.csv'):
```

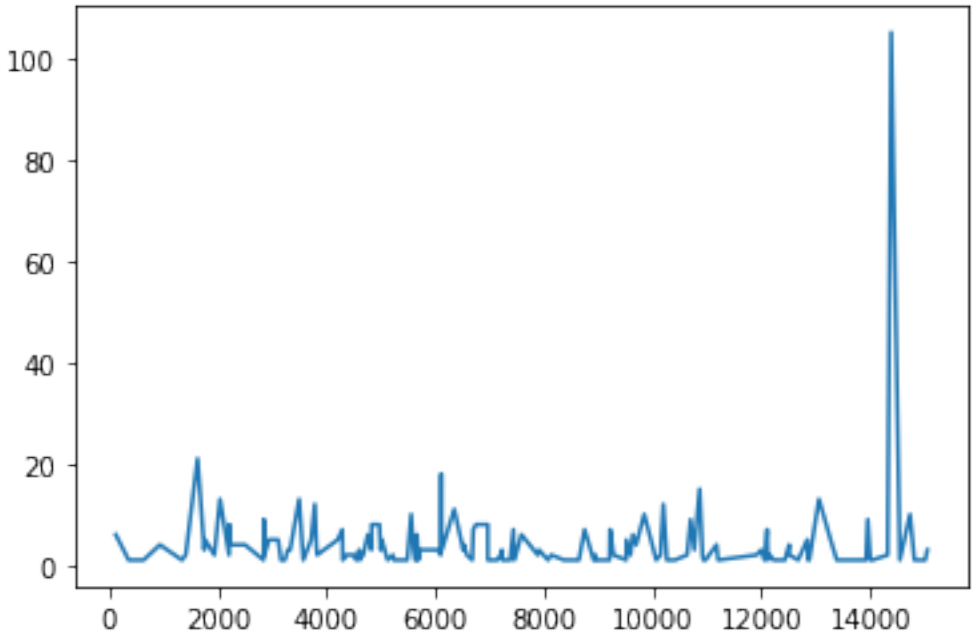
```
    df_records = pd.read_csv('../core/records/batting_results.csv')
    df_records.append(new_data)
else:
    df_records = pd.DataFrame(new_data)

df_records.to_csv('../core/records/batting_results.csv', index=False,␣
 ↪float_format='%g')
```

Model Evaluation

[24]:
```
losses = pd.DataFrame(model.history.history)
```

[25]:
```
losses.plot()
```

[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14011ce10>
```



[26]:
```
predictions = model.predict(X_test)
predictions = [pred for sublist in predictions for pred in sublist]
```

[27]:
```
test_player_ratings = dict(zip(X_test_keys, predictions))
```

[28]:
```
player_key = df['retroID']
```

[29]:
```
player_key
```

[29]:
```
0        aardd001
1        aaroh101
2        aarot101
3        aased001
4        abada001
           ...
15288    zupcb001
15289    zupof101
15290    zuveg101
15291    zuvep001
15292    zycht001
Name: retroID, Length: 15293, dtype: object
```

[30]:
```
results = model.predict(tensor.to_numpy())
```

[31]:
```
len(results)
```

[31]:
```
15293
```

[32]:
```
results.mean()
```

[32]:
```
0.12198973
```

[33]:
```
df['Batting'].shape
```

[33]:
```
(15293,)
```

[34]:
```
results.shape
```

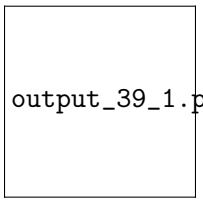
[34]:
```
(15293, 1)
```

[35]:
```
results = [pred for sublist in results for pred in sublist]
```

[36]:
```
diff = df['Batting'] - results
```

[37]:
```
diff.plot()
```

[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x140c786d0>
```



[38]:
```
diff.mean()
```

[38]:
```
-0.00040076900953220845
```

[ ]:
```

```