Model 1. (B)-N-BT-0.2-2

Trains model 1 based on:

- Bootstrapped Linear Regression
- Normalised Data
- Both Teams data
- FS_Val 0.2
- FS_Rule 2

Author: Lang (Ron) Chen 2021.12-2022.1

---

0. Import Libraries

In [1]:
```
import pandas as pd
import os
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pickle

from BrownlowPredictorTools.predict import predict
from BrownlowPredictorTools.test import test
from BrownlowPredictorTools.return_tp import return_tp
from BrownlowPredictorTools.wholeseason import wholeseason
from BrownlowPredictorTools.feature_selection2 import feature_selection2
```

In [2]:
```
choice = 'NormalisedData'
```

In [3]:
```
filelist = os.listdir(f'./Data/{choice}')[1:]
filelist.sort()
filelist = filelist[1:]
# Remove the first file (an ipynb checkpoint file)
```

1. Feature Selection

In [4]:
```
# As we need to perform tests to evaluate this final model, still need to use do this

# Gets list of emperical test games (full 2021 season)
final_test_games = [file for file in filelist if '2021' in file]

# Gathers full games list (except 2021) and performs a single Train-Test Split (note diffe
test_train_games = [file for file in filelist if '2021' not in file]
train_games, test_games = train_test_split(test_train_games, train_size = 0.8, test_size =
```

In [5]:
```
# Read in pre-prepared sample data of trained data only
# (the same rows as if we used concatenated all the data from the train_games list)
train_data = pd.read_csv('./Models/TrainingData/M1_Data.csv')
```

Bootstrap

In [6]:
```
# Bootstraps data

# Picks out data labelled 1 vote, 2 votes, 3 votes
zero = train_data[train_data['Brownlow Votes'] == 0]
one = train_data[train_data['Brownlow Votes'] == 1]
two = train_data[train_data['Brownlow Votes'] == 2]
three = train_data[train_data['Brownlow Votes'] == 3]

# Sample them so each has same number as 0 votes
new_one = one.sample(n = len(zero), replace = True, random_state = 42)
new_two = two.sample(n = len(zero), replace = True, random_state = 42)
new_three = three.sample(n = len(zero), replace = True, random_state = 42)

# Add the sampled dataframes back onto zero
bootstrapped_data = pd.concat([zero, new_one, new_two, new_three], axis = 0)
```

In [7]:
```
# Select Columns of Both Teams Stats only
cols = [col for col in train_data.columns if ('BTN' in col or 'Winloss' in col)]

# Select Columns with correlation higher than 0.2 only
corr = dict()
for col in cols:
    corr[col] = train_data[[col, 'Brownlow Votes']].corr(method = 'pearson').loc[col]['Bro

corr = list(corr.items())

selected_features = [col[0] for col in corr if col[1] > 0.2]
selected_features
```

Out[7]:
```
['Kicks BTN',
 'Handballs BTN',
 'Disposals BTN',
 'Goals BTN',
 'Inside 50s BTN',
 'Clearances BTN',
 'Contested Possessions BTN',
 'Uncontested Possessions BTN',
 'Effective Disposals BTN',
 'Centre Clearances BTN',
 'Stoppage Clearances BTN',
 'Score Involvements BTN',
 'Metres Gained BTN',
 'Behind Assists BTN',
 'Ineffective Disposals BTN']
```

2. Trains Model using rule 2

In [8]:
```
selected_features = feature_selection2(cols, 2, False)
selected_features
```

Out[8]:
```
['Kicks BTN',
 'Handballs BTN',
 'Marks BTN',
 'Goals BTN',
 'Behinds BTN',
 'Tackles BTN',
 'Hitouts BTN',
 'Goal Assists BTN',
 'Inside 50s BTN',
 'Clearances BTN',
 'Clangers BTN',
 'Rebound 50s BTN',
 'Frees For BTN',
```

```
'Frees Agains BTN',
'Contested Possessions BTN',
'Uncontested Possessions BTN',
'Effective Disposals BTN',
'One Percenters BTN',
'Bounces BTN',
'Metres Gained BTN',
'Turnovers BTN',
'Intercepts BTN',
'Time On Ground % BTN',
'Winloss',
'Behind Assists BTN',
'Ineffective Disposals BTN']
```

In [9]:
```
traindata_x = train_data[selected_features]
traindata_x.index = range(0,len(traindata_x))
traindata_y = train_data['Brownlow Votes']
traindata_y.index = range(0,len(traindata_y))

lm = linear_model.LinearRegression()
traindata_x = traindata_x.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
model = lm.fit(traindata_x, traindata_y)
```

In [10]:
```
predictions, testdata_y = predict(test_games, lm, selected_features, choice)
```

In [11]:
```
result1, result2 = test(predictions, testdata_y, 4)
```

In [12]:
```
result1
```

Out[12]:
```
[[0.9725210352540207,
  0.013313451911811694,
  0.010011715837682395,
  0.004153796996485249],
 [0.5938864628820961,
  0.18340611353711792,
  0.11353711790393013,
  0.1091703056768559],
 [0.3624454148471616,
  0.14410480349344978,
  0.26200873362445415,
  0.2314410480349345],
 [0.1703056768558952,
  0.12663755458515283,
  0.21397379912663755,
  0.4890829694323144]]
```

In [13]:
```
result2
```

Out[13]:
```
[[0.9725210352540207,
  0.014485035680051123,
  0.008840132069442966,
  0.004153796996485249],
 [0.5458515283842795,
  0.18340611353711792,
  0.14410480349344978,
  0.12663755458515283],
 [0.4104803493449782,
  0.11353711790393013,
  0.26200873362445415,
  0.21397379912663755],
```

```
[0.1703056768558952,
 0.1091703056768559,
 0.2314410480349345,
 0.4890829694323144]]
```

In [14]:
```
return_tp(result1)
```

Out[14]:
```
(0.9725210352540207,
 0.18340611353711792,
 0.26200873362445415,
 0.4890829694323144)
```

3. Summary Observations

1. Emperical Experiment

In [15]:
```
# Runs the season 2021 data onto predictor and gets top players
leaderboard = wholeseason(final_test_games, lm, selected_features, choice)
```

In [16]:
```
leaderboard[0:15]
```

Out[16]:
```
[('Jack Steele', 39),
 ('Oliver Wines', 34),
 ('Clayton Oliver', 29),
 ('Marcus Bontempelli', 29),
 ('Christian Petracca', 28),
 ('Darcy Parish', 28),
 ('Jarryd Lyons', 26),
 ('Luke Parker', 24),
 ('Jackson Macrae', 21),
 ('Rory Laird', 21),
 ('Tom Mitchell', 20),
 ('Travis Boak', 20),
 ('Touk Miller', 19),
 ('Sam Walsh', 18),
 ('Jake Stringer', 18)]
```

1. Predictor's r scores

In [17]:
```
print(lm.score(traindata_x, traindata_y))
```

```
0.2465957489737035
```

4. Picklising

In [18]:
```
with open('./Models/M1.pickle', 'wb') as f:
    pickle.dump([lm, selected_features, choice], f)
```