Model 2. LR-N-BT-0.2-1

Trains model 2 based on:

- Regular Linear Regression
- Normalised Data
- Both Teams data
- FS_Val 0.2
- FS_Rule 1

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}')
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/M2_Data.csv')
```

In [6]:
```
# 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[6]:
```
['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 1

No need to run feature_selection2 because this utilises rule 1

In [7]:
```
# Trains LR model
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 [8]:
```
# Get predictions and observations
predictions, testdata_y = predict(test_games, lm, selected_features, choice)
```

In [9]:
```
# Get True Positive/True Negative results
result1, result2 = test(predictions, testdata_y, 4)
```

In [10]:
```
# TP/TN based on what was predicted
result1
```

Out[10]:
```
[[0.9697518372563638,
  0.013845989988284162,
  0.01107679199062733,
  0.005325380764724678],
 [0.6331877729257642,
  0.17467248908296942,
  0.09170305676855896,
  0.10043668122270742],
```

```
 [0.39737991266375544,
  0.11790393013100436,
  0.2183406113537118,
  0.2663755458515284],
 [0.2096069868995633,
  0.13973799126637554,
  0.23580786026200873,
  0.4148471615720524]]
```

In [11]:
```
# TP/TN based on what was observed
result2
```

Out[11]:
```
[[0.9697518372563638,
  0.015443604217701567,
  0.009692192991798914,
  0.00511236553413569],
 [0.5676855895196506,
  0.17467248908296942,
  0.11790393013100436,
  0.13973799126637554],
 [0.45414847161572053,
  0.09170305676855896,
  0.2183406113537118,
  0.23580786026200873],
 [0.2183406113537118,
  0.10043668122270742,
  0.2663755458515284,
  0.4148471615720524]]
```

In [12]:
```
# Only the True Positive Values
return_tp(result1)
```

Out[12]:
```
(0.9697518372563638,
 0.17467248908296942,
 0.2183406113537118,
 0.4148471615720524)
```

3. Summary Observations

1. Emperical Experiment

In [13]:
```
# Runs the season 2021 data onto predictor and gets top players
leaderboard = wholeseason(final_test_games, lm, selected_features, choice)
```

In [14]:
```
leaderboard[0:15]
```

Out[14]:
```
[('Oliver Wines', 34),
 ('Jack Steele', 33),
 ('Christian Petracca', 30),
 ('Clayton Oliver', 28),
 ('Darcy Parish', 28),
 ('Jarryd Lyons', 27),
 ('Tom Mitchell', 27),
 ('Jackson Macrae', 27),
 ('Marcus Bontempelli', 26),
 ('Rory Laird', 23),
 ('Jake Stringer', 21),
 ('Cameron Guthrie', 20),
 ('Touk Miller', 20),
 ('Luke Parker', 20),
 ('Sam Walsh', 19)]
```

1. Predictor's r scores

In [15]:
```
print(lm.score(traindata_x, traindata_y))
```

```
0.23298806127226634
```

4. Picklising

In [16]:
```
with open('./Models/M2.pickle', 'wb') as f:
    pickle.dump([lm, selected_features, choice], f)
```