removed\_cols = ['timestamp', 'fixture\_code', 'kickoff\_time', 'opposition', 'event\_id']

for col in removed\_cols:

del train\_data[col]

del test\_data[col]

[**We are removing these unnecessary column ['timestamp', 'fixture\_code', 'kickoff\_time', 'opposition', 'event\_id'**]

plt.figure(figsize=(10, 10))

sns.histplot(x=train\_data.response, hue=train\_data.status, binwidth=1)

[ **we are using figure for graphical representation size of 10,10. From the data.csv, we training the response and status data and then into a graphically representation and set the binwidth 1.0.]**

played\_match\_1 = train\_data[train\_data.status == 'a']["minutes\_1"] > 0

played\_match\_2 = train\_data[train\_data.status == 'a']["minutes\_2"] > 0

played\_match\_3 = train\_data[train\_data.status == 'a']["minutes\_3"] > 0

**[ If a player is wounded or unavailable, they are very certain to receive zero points, however if they are completely fit and available, they are far more likely to receive points.**

**As this shows that if a player's status is not 'available', then they are very likely to score 0 points, it is therefore only interesting to explore the data for those whose status is available. This is based around the number of minutes a player plays. How else can we predict minutes? Well, we have the data for minutes for the last three matches.]**

available\_players = train\_data[train\_data.status == 'a'].copy()

[**for available players, here train data from training dataset, train data.status= available, we are creating a new available players data copy in new data frame]**

sns.histplot(x=available\_players.response, hue=available\_players.is\_home, binwidth=1)

[ **We are checking those available players response is in home or away. Then we train the data info to get dummy info. From new available players data frame, we get new training and test dummy data info from the available players data frame.**].

def fill\_values(df):

df['chance\_of\_playing\_this\_round'] = df['chance\_of\_playing\_this\_round'].fillna(100)

df.fillna(0, inplace=True)

fill\_values(train\_data)

fill\_values(test\_data)

train\_data.head()

**[ The fillna() method replaces the NULL values with a specified value. Here, we are filling up the null value with the chance of playing this round into 100.**

**The fillna() method returns a new DataFrame object unless the inplace parameter is set to True , in that case the fillna() method does the replacing in the original DataFrame instead.]**

**train\_data.info() [then we train the data info]**