Questions:
Can I use all the features in the data while testing?
How will you test the result against testing data?

Data Analysis:

1. There is an extra referee feature in test data.
2. There are empty rows in between training data.
3. “Index” column is absent in train data
4. Number of unique home teams and unique away teams = 159
5. Number of unique ‘HomeTeam - AwayTeam’ pairs = 4168
6. For now ignore ‘league’ column
7. Correlation between 14 features on average over all matches - assumes all data points (matches) are independent which is not the case. Their performance may have a time series relation.
8. Scatter plot Visualization between every pair of the 14 features
    a. From correlation matrix and scatter plot: strongest positive relation is found between AS-AST, HS-HST (~0.6), AS-AC, HS-HC(~0.45), AF-AY, HF-HY (~0.34). So I can choose to skip either AS or AST in the data while training.

Data Preprocessing:

1. Removed rows with no feature data expect “league” name
2. Kept the rows with fewer features (Home team, Away team and FTR)
3. Deleted the Referee col in Test data to match the number of features in train and test. Also referee col was null.
4. Make a dictionary of {'team name': 1, 'team name' : 2 ... 'FTR' : np.asarray([1, 0, 0])
   Initialize 12786x16x159 nparray: all_train
   Initialize 12786x1x3 nparray: all_valid
   For every row in df:
   row 1: 1 on home team and 1 on away team and rest 0s
   row 2: 1 on home team and rest 0s (special reference to home team)
   For all 14 features(rows) except FTR : 159 cols each with corresponding number in home and away team and rest 0s.

   all_valid will take [100], [010], [001] for (win, loose or draw) from df['FTR'].

Technique:
This is a 3 class classification problem : H, A, D being the three classes.
Model 1: Logistic regression

a) From the confusion matrix, most samples are classified as class 1 (H). That could be because of skewness in the training data between samples -
   Training data: 4152, 2497, 2298
   Validation data: 1779, 1071, 985
   Precision: class 1: 84%, Class II: 21%, Class III: 6%
   Average Precision: 37%

Recall: Class I: 49.8%, Class II: 37.7%, Class III: 28.6%
Recall: 38.7%

b) With class weights: accuracy of other classes significantly improved.
   Weight for Home team = log(total_samples/count_hometeam_wins)
   class_weights = {1: 0.76, 2: 1.27, 3: 1.35}
   Precision: Class 1: 48%, Class II: 45%, Class III: 27.7%
   Average Precision: 40%
   Recall: Class 1: 58%, Class II: 35.7%, Class III: 27.1%
   Average Recall: 40.2%
   Accuracy = **40%**

Model 2: Random Forest

a) With equal class weights, no_of_estimators = 700, max_features = sqrt (auto): Similar numbers as logistic regression
b) With number of estimators trees = 100 (ditto)
c) With max features = log2
d) With class weights: {1: 1, 2: 2, 3: 2}
   Precision: Class 1: 43%, Class 2: 37%, Class 3: 50%
   Average Precision: 43.3%
   Accuracy = **41.8%**

Model 3: XGBoost

a) With equal class weights: Worser numbers than above
b) After parameter turning also bad performance
c) With sample weights: {1, 2, 2}
   Precision: Class 1: 32%, Class 2: 41%, Class 3: 30%
   Average Precision: 34.3%
d) {1.5, 2, 2.5}
   51.1, 13.6, 43.4
   Average Precision: 36%
   Accuracy = **41%**
e) {1.5, 2.5, 2.5}
   45, 36, 19
   Average precision: 33%
f) {1.5, 2.5, 3}
   38, 22, 49
   Average : 36.3

Model 4: SVM

a) Same as above

Model 5, 6, 7: Neural network models: SimpleRNN, LSTM, Bidirectional LSTM

Using Titan V and Geforce 1080. For all the three neural networks above, accuracy is ~46%