CSE – 454 Project
Analysis and Prediction of Football Statistics using Data Mining Techniques

Alper YAŞAR – 151044072

ABSTRACT

This project explored different data mining techniques used for predicting the match outcomes where the target class is win, draw, lose and total goals over/under of 2.5. The main objective of this research is to find the most accurate data mining technique that fits the nature of football data. The techniques tested are Logistic Regression, Support Virtual Machine and Bayesian Network.

1. Introduction

The reason that prediction approaches have been widely used in many companies around the world today is because prediction is the heart of remarkable disciplines in Science. This approach enables companies and organizations to predict and prepare reasonable future plans. Nowadays, many people, companies, and clubs want to predict matches before they are played. Many reasons affect the scores of the matches. But in this project, I am trying to predict the results of the next week based on the scores of the remaining matches of one league in one league.

2. Dataset

The dataset chosen to be implement in this research was the data first 13 weeks of Turkish Super League of 2020-2021 season and predict of 14. week. There is 138 match in 12 weeks and I took it from https://www.tff.org . There 4 column has home team, away team, match result and stadium of match. I added winner (home, away, draw) of the match and total goals (over, under) of match.

3. Evaluation metrics

The performance measures used in this paper is the accuracy. Overall, accuracy measures how often is the classifier or the classification techniques (Decision Trees, Neural Networks, Bayesian Networks, *k*-Nearest Neighbor) getting correct target class of classification. Equation 1 shows how accuracy is measured.

$$\text{accuracy} = \frac{TP + TN}{P + N} \tag{1}$$

From the equation, *TP* represents true positives or equivalent to hit, *TN* represents true negatives or equivalent to correct rejection, *P* is the total number of real positive cases in the data, and finally *N* is the total number of real negative cases in the data. In context of football match results, its refer to win, draw or lose target class. For example, if the predicted result of a match hit win but the actual result is lose the result will not be counted in *TP* and *TN*. However, it will counted in *P* which it is actually false positive. Literally, its can be explained that the total number of correct predicted match results (win, draw or lose) are divided with total number of matches.

4. Results

4.1. Naïve Bayes

I implement naïve bayes and read data from 'lig.csv' file. Firstly compare who win the match, and I calculate confusion matrix and calculate accuracy. Read 'tahmin.csv' file as test file.

```
Tested class :    ['Home', 'Home', 'Home', 'Home', 'Away', 'Home', 'Away', 'Draw', 'Home', 'Draw']
Predicted Class : ['Away', 'Home', 'Home', 'Draw', 'Home', 'Home', 'Home', 'Away', 'Away', 'Away']

Confussion Matrix :
[[0 0 2]
 [2 0 0]
 [2 1 3]]

Accuracy of naive Bayes : % 30.0

****************************************************************************************************
****************************************************************************************************

Tested class :    ['Under', 'Over', 'Over', 'Over', 'Over', 'Over', 'Over', 'Under', 'Under', 'Under']
Predicted Class : ['Under', 'Under', 'Under', 'Under', 'Over', 'Over', 'Over', 'Under', 'Over', 'Over']

Confussion Matrix :
[[3 3]
 [2 2]]

Accuracy of naive Bayes : % 50.0

****************************************************************************************************
old Naive Bayes algorithm accurance average :  40.0
```

Totally test cluster has 20 element and average of naïve bayes is 40%

I reduced data and just take football match from test and create new data. This data just including 2 football clubs all match and get result. When data size shrinks result I took to bad. When train set larger than getting better results. So I took match many times and train it.

```
Tested class :    ['Home', 'Home', 'Home', 'Home', 'Away', 'Home', 'Away', 'Draw', 'Home', 'Draw']
Predicted Class : ['Away', 'Away', 'Away', 'Draw', 'Home', 'Away', 'Home', 'Away', 'Away', 'Away']

Confussion Matrix :
[[0 0 2]
 [2 0 0]
 [5 1 0]]

Accuracy of naive Bayes : % 0.0

****************************************************************************************************
****************************************************************************************************

Tested class :    ['Under', 'Over', 'Over', 'Over', 'Over', 'Over', 'Over', 'Under', 'Under', 'Under']
Predicted Class : ['Over', 'Under', 'Under', 'Under', 'Under', 'Over', 'Over', 'Over', 'Over', 'Under']

Confussion Matrix :
[[2 4]
 [3 1]]

Accuracy of naive Bayes : % 30.0

****************************************************************************************************
Improved Naive Bayes algorithm accurance average :  15.0
****************************************************************************************************
```

The result worse than I expect and I calculate match results reduced data :

Average of prediction is 15%.

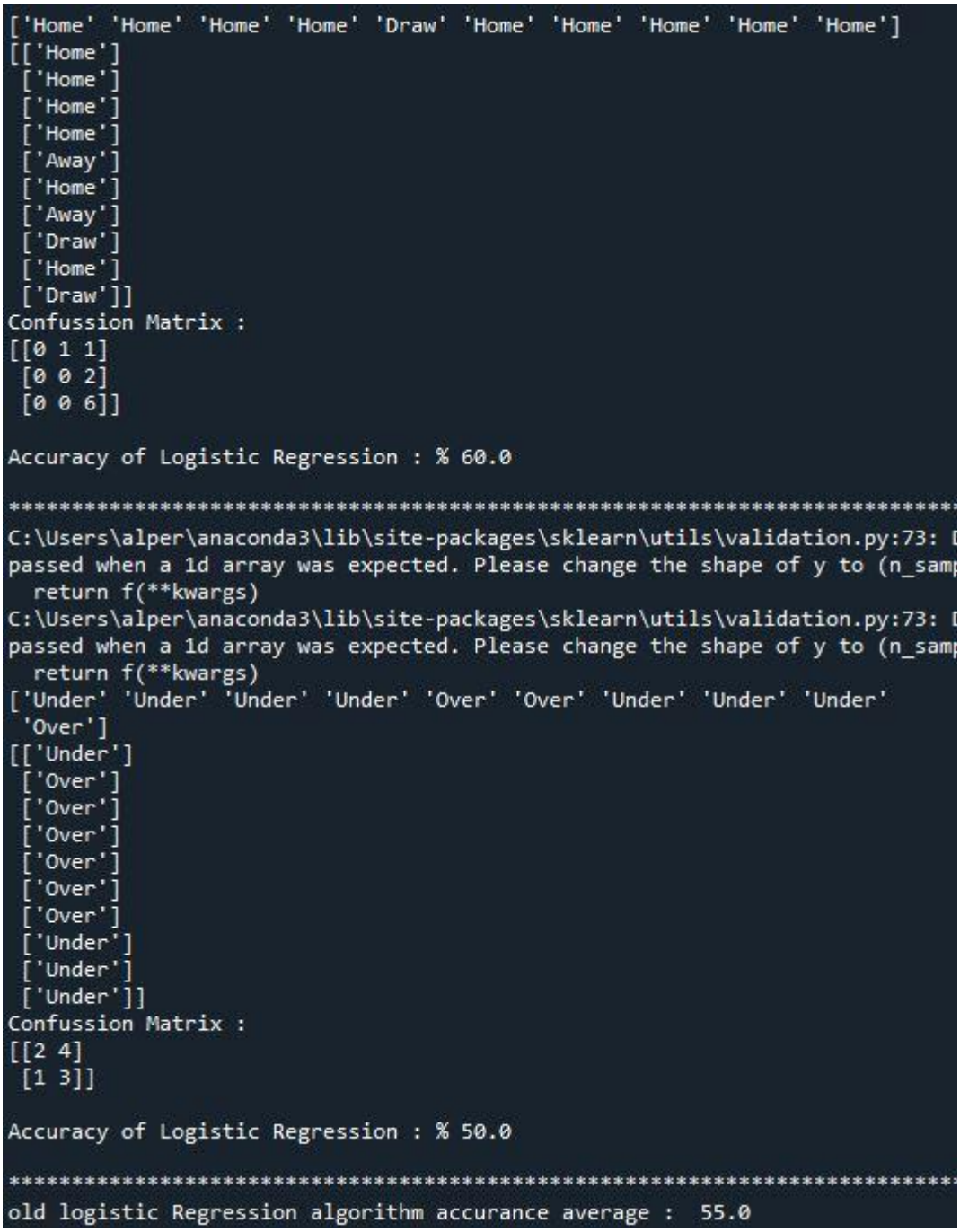
```
Tested class :    ['Home', 'Home', 'Home', 'Home', 'Away', 'Home', 'Away', 'Draw', 'Home', 'Draw']
Predicted Class : ['Away', 'Away', 'Away', 'Draw', 'Home', 'Home', 'Home', 'Draw', 'Away', 'Away']

Confussion Matrix :
[[0 0 2]
 [1 1 0]
 [4 1 1]]

Accuracy of naive Bayes : % 20.0

****************************************************************************************************
****************************************************************************************************

Tested class :    ['Under', 'Over', 'Over', 'Over', 'Over', 'Over', 'Over', 'Under', 'Under', 'Under']
Predicted Class : ['Over', 'Under', 'Under', 'Under', 'Over', 'Under', 'Under', 'Under', 'Over', 'Under']

Confussion Matrix :
[[1 5]
 [2 2]]

Accuracy of naive Bayes : % 30.0

****************************************************************************************************
Improved and reduced data Naive Bayes algorithm accurance average :  25.0
****************************************************************************************************
```

The result worse than I expect and I calculate match results reduced data :

Average of prediction is 25%.

4.1. Logistic Regression

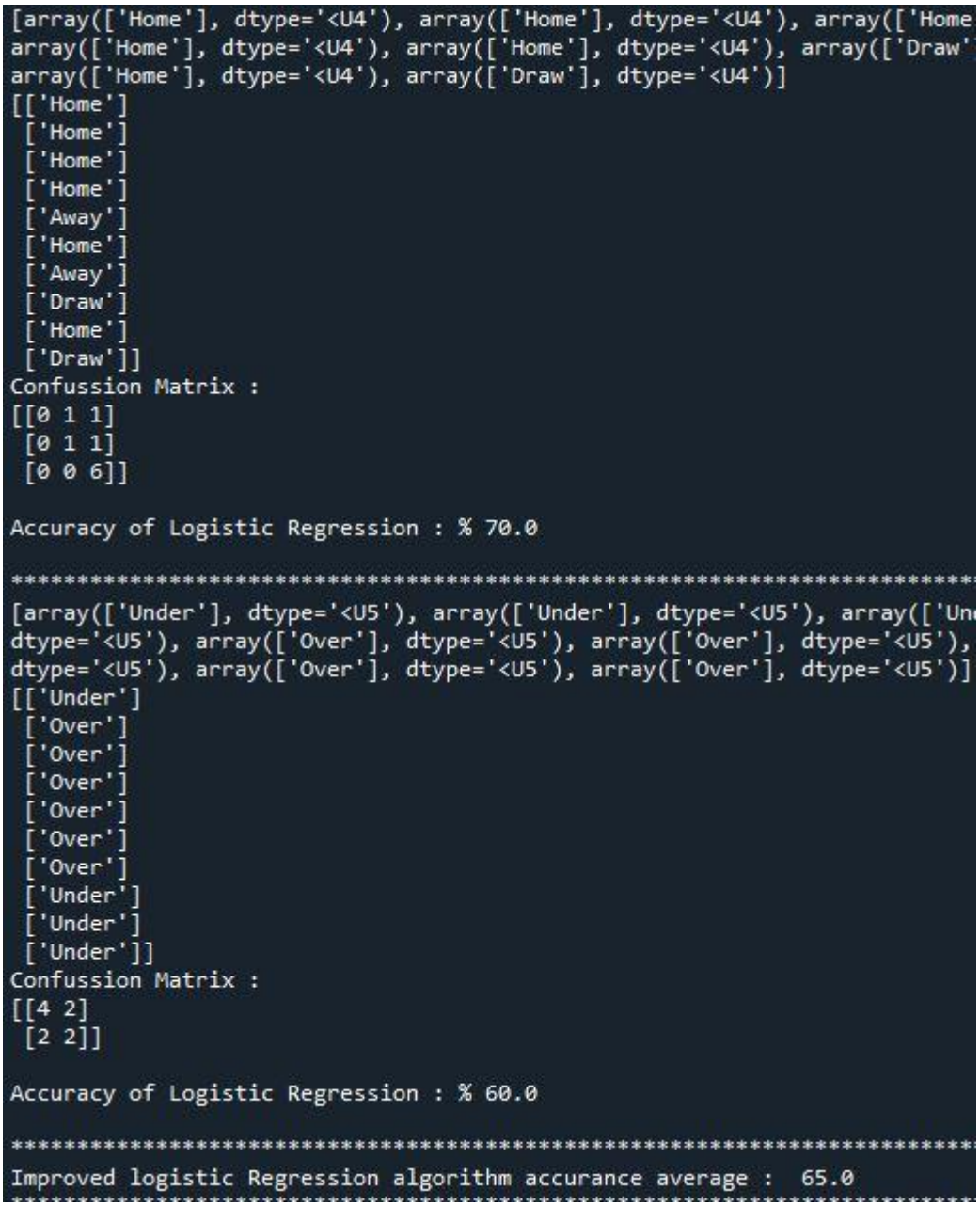
I implement naïve bayes and read data from ‘lig.csv’ file. Firstly compare who win the match, and I calculate confusion matrix and calculate accuracy. Read ‘tahmin.csv’ file as test file.



Totally test cluster has 20 element and average of logistic regression is 55%

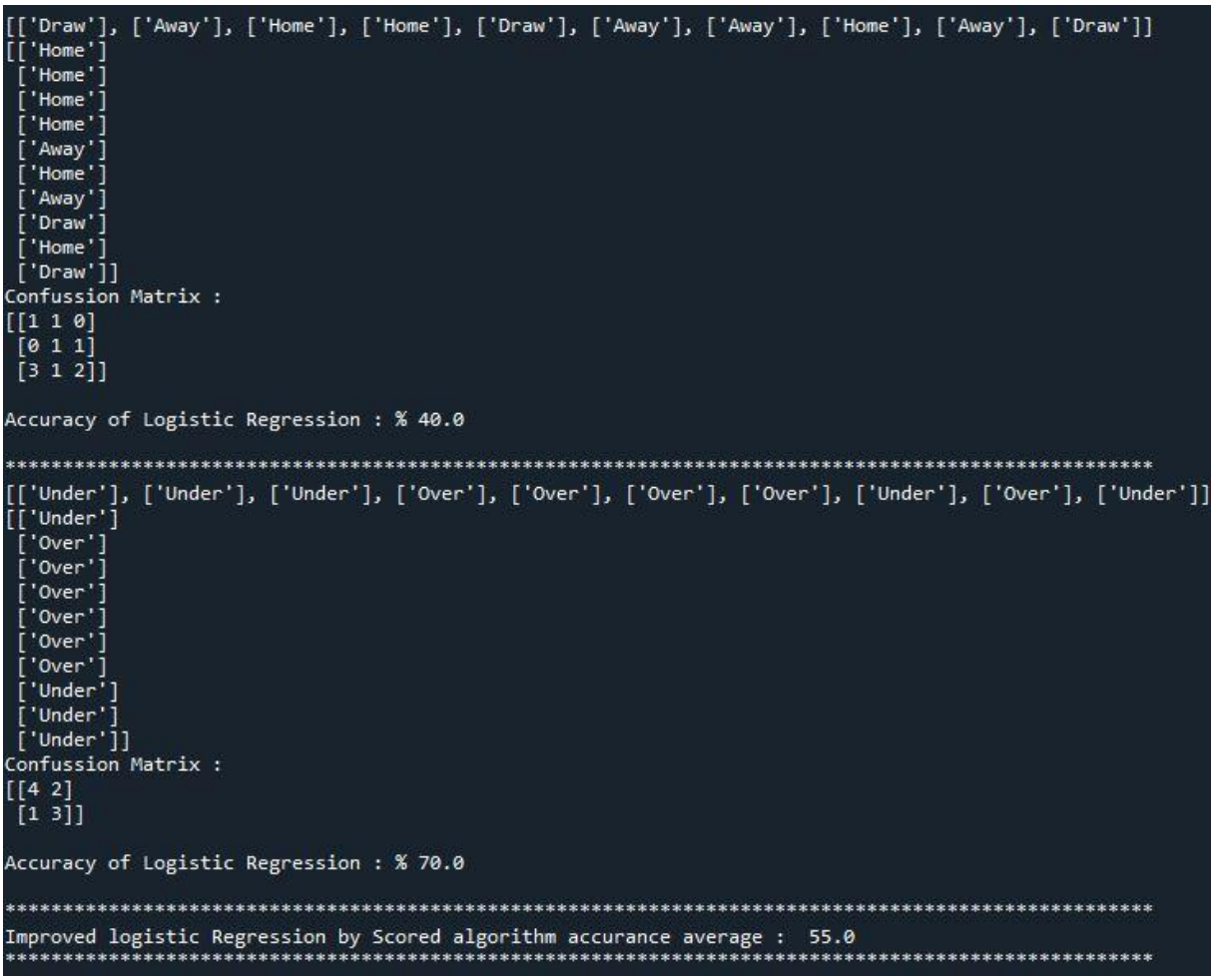
I reduced data and just take football match from test and create new data. This data just including 2 football clubs all match and get result. When data size shrinks result I took to bad. When train set larger than getting better results. When data size is reduced I took better winner result. Previous result is 60% but now the result 70%. Algorithm predict 1 more true result. Total goal increase also to 60% from 50%.

Average of prediction is 65%.



When I reduce data for found match goal results total prediction is less again. Winner prediction fell to %40 but goal prediction increase to 70%.

Average of prediction 55%.



4.1. Support Vector Machine

I implement svm and read data from ‘lig.csv’ file. Firstly compare who win the match, and I calculate confusion matrix and calculate accuracy. Read ‘tahmin.csv’ file as test file.

```
['Home' 'Home' 'Home' 'Home' 'Home' 'Home' 'Home' 'Home' 'Home' 'Home']
[['Home']
 ['Home']
 ['Home']
 ['Home']
 ['Away']
 ['Home']
 ['Away']
 ['Draw']
 ['Home']
 ['Draw']]
Confussion Matrix :
[[0 0 2]
 [0 0 2]
 [0 0 6]]

Accuracy of SVM : % 60.0

****************************************************************************************************
['Over' 'Under' 'Over' 'Over' 'Over' 'Over' 'Under' 'Under' 'Under' 'Over']
[['Under']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Under']
 ['Under']
 ['Under']]
Confussion Matrix :
[[4 2]
 [2 2]]

Accuracy of SVM : % 60.0

****************************************************************************************************
old SVM algorithm accurance average :  60.0
****************************************************************************************************
```

Totally test cluster has 20 element and average of svm is 60%

I reduced data and just take football match from test and create new data. This data just including 2 football clubs all match and get result. When data size shrinks results not change.

Average of prediction is 60%.

```
[array(['Away'], dtype='<U4'), array(['Draw'], dtype='<U4'), array(['Home'], dtype='<U4'), array(['Home'], dtype='<U4'),
array(['Home'], dtype='<U4'), array(['Home'], dtype='<U4'), array(['Away'], dtype='<U4'), array(['Home'], dtype='<U4'),
array(['Home'], dtype='<U4'), array(['Draw'], dtype='<U4')]
[['Home']
 ['Home']
 ['Home']
 ['Home']
 ['Away']
 ['Home']
 ['Away']
 ['Draw']
 ['Home']
 ['Draw']]
Confussion Matrix :
[[1 0 1]
 [0 1 1]
 [1 1 4]]

Accuracy of SVM : % 60.0

****************************************************************************************************
C:\Users\alper\anaconda3\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
C:\Users\alper\anaconda3\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
[array(['Under'], dtype='<U5'), array(['Under'], dtype='<U5'), array(['Under'], dtype='<U5'), array(['Under'],
dtype='<U5'), array(['Over'], dtype='<U5'), array(['Over'], dtype='<U5'), array(['Over'], dtype='<U5'), array(['Under'],
dtype='<U5'), array(['Over'], dtype='<U5'), array(['Under'], dtype='<U5')]
[['Under']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Under']
 ['Under']
 ['Under']]
Confussion Matrix :
[[3 3]
 [1 3]]

Accuracy of SVM : % 60.0

****************************************************************************************************
Improved SVM algorithm accurance average :  60.0
****************************************************************************************************
```

When I reduce data for found match goal results total prediction is better. Winner prediction fell to %50 but goal prediction increase 80%.

Average of prediction 65%.

```
[['Away'], ['Away'], ['Home'], ['Draw'], ['Home'], ['Home'], ['Away'], ['Home'], ['Home'], ['Draw']]
[['Home']
 ['Home']
 ['Home']
 ['Home']
 ['Away']
 ['Home']
 ['Away']
 ['Draw']
 ['Home']
 ['Draw']]
Confussion Matrix :
[[1 0 1]
 [0 1 1]
 [2 1 3]]

Accuracy of SVM : % 50.0

****************************************************************************************************
[['Under'], ['Under'], ['Over'], ['Under'], ['Over'], ['Over'], ['Over'], ['Under'], ['Under'], ['Under']]
[['Under']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Over']
 ['Under']
 ['Under']
 ['Under']]
Confussion Matrix :
[[4 2]
 [0 4]]

Accuracy of SVM : % 80.0

****************************************************************************************************
Improved SVM by Scored algorithm accurance average :  65.0
```