Data Warehousing & Data Mining Report

Student Name(s)	Connel McGovern	Eoin Murphy
Student Number(s)	11379746	11487358
Programme	Computer Applications (CASE4)	Computer Applications (CASE4)
Module Name	Data Warehousing & Data Mining	Data Warehousing & Data Mining
Module Code	CA4010	CA4010
Submission Date	06/12/2015	06/12/2015

Name: Connel McGovern Date: 05/12/2015

Name: Eoin Murphy Date: 05/12/2015

1. Idea & Dataset Description

For our Data Mining & Data Warehousing (CA4010) project we were set the task to obtain dataset and using it, make a prediction. As a team we initially searched for a dataset that we felt would allow us to make an interesting prediction. As we collaboratively looked at various datasets we started to identify possible ideas. The idea that we felt was best suited to our teams was to identify how money in football affects a team's overall league performance. To make this prediction we realised we required a dataset consisting of various teams performances and their spending figures. We soon realised that there was not a suitable dataset available for our project.

Due to a dataset not being readily available to make our prediction we decided that it would be of benefit for us to scrape the data that we required from the internet. Scraping the data allowed us to choose exactly what we felt we required to make the prediction. We had to identify specifically what data was required from the data available to make our overall prediction. We decided that we had to adapt the project idea for what was accessible to us. We decided to base our prediction on English league teams over the past ten years. The league's consisted of the Premier League, Championship, League One & League Two.

In scraping the data we used the python programming language and made use of the open-source Scrapy framework. The websites we identified that had the data required were http://www.espnfc.com/ and http://www.footballdatabase.eu/ . The ESPN website contained historic data for all league tables for the four league tables mentioned above over the past ten years, from the 2005-2006 season up until the 2014-2015 season. The footballdatabase website contained all individual incoming and outgoing transfers for England over the past ten years.

Collecting our data

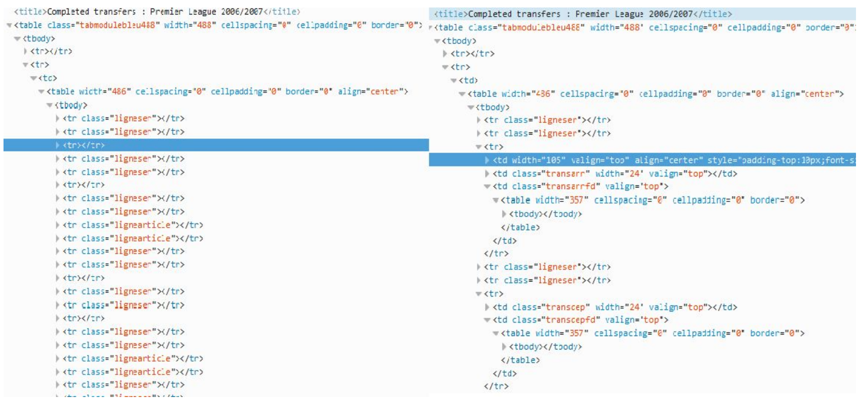
In retrieving the data we first had to identify what specifically we required to make our prediction. Using our prediction of “how money in football affects a team's overall league performance” we realised that the data we required could be comprised of two components. These two components included transfer details and associated figures and teams annual performances as depicted in league tables.

Transfers data

The first component of data we required for our project was the football transfers. In searching for data on the football transfers in England we noticed that many websites only displayed the financial figures as a net value for each team and would only display a maximum of ten years of data for teams. This posed an issue to our overall ability to make our desired prediction. We required to differentiate between expenditure and income and this could not be done with one net figure. The historic data available would also prove to be an issue as there was a distinct lack of volume.

After extensively searching the web we eventually found the website football-database.eu. This website contained all historic records of teams, player arrivals (spend) and player departures (income) with their associated transfer figures for many leagues across Europe, including the four English league that we required. However, this website gave us considerable problems when it came to scraping the required data.

The first issue related to the design of the website, specifically the HTML code. It was immediately apparent that the site was guarded for scraping data using separate tables for associated transfers, as seen below (Figure 1.1). With the obscured web formatting, data extracted was corrupt with invalid characters. There was also an issue retrieving team names for player departure's. Again this was down to the table layout within the website's HTML source code. To allow for a solution we had to implement a method of adding row ID's to the outer table in which our spiders scraper program was based. Applying ID's was again used for rows within the nested table. The missing values would then be filled in using these ID's during our data cleaning phase. The value attribute would then be required to be parsed so as to remove any invalid characters.

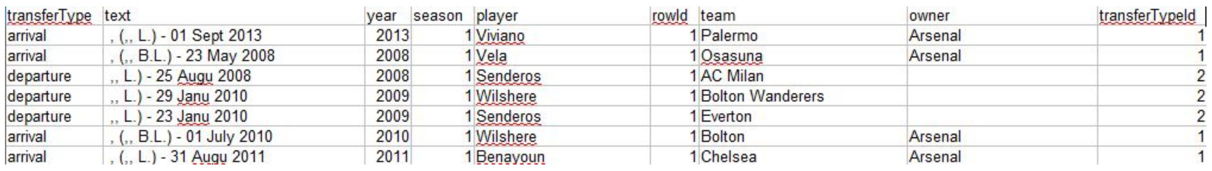


```
<title>Completed transfers : Premier League 2006/2007</title>
<table class="tabmodulebleu488" width="488" cellspacing="0" cellpadding="0" border="0">
  <tbody>
    <tr></tr>
    <tr>
      <td>
        <table width="486" cellspacing="0" cellpadding="0" border="0" align="center">
          <tbody>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr></tr>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr></tr>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr class="lignearticle"></tr>
            <tr class="lignearticle"></tr>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr></tr>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr></tr>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr class="lignearticle"></tr>
            <tr class="lignearticle"></tr>
            <tr class="ligneser"></tr>
```

```
<title>Completed transfers : Premier League 2006/2007</title>
<table class="tabmodulebleu488" width="488" cellspacing="0" cellpadding="0" border="0"
  <tbody>
    <tr></tr>
    <tr>
      <td>
        <table width="436" cellspacing="0" cellpadding="0" border="0" align="center">
          <tbody>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr>
              <td width="105" valign="top" align="center" style="padding-top:10px;font-s
              <td class="transarr" width="24" valign="top"></td>
              <td class="transarrfd" valign="top">
                <table width="357" cellspacing="0" cellpadding="0" border="0">
                  <tbody></tbody>
                </table>
              </td>
            </tr>
            <tr class="ligneser"></tr>
            <tr class="ligneser"></tr>
            <tr>
              <td class="transcep" width="24" valign="top"></td>
              <td class="transcepfd" valign="top">
                <table width="357" cellspacing="0" cellpadding="0" border="0">
                  <tbody></tbody>
                </table>
              </td>
            </tr>
```

<u>Figure 1.1</u> Screenshot depicting how the html caused problems in scraping data (no identifiers & departures being in completely separate tables to the team name etc.)

Another problem that was also encountered was for every 6-9 connections made by our spider program to the website the site would send an automatic redirection to a different landing page. The connections appeared to be counted on a 24 hour basis. Due to this we decided to use another spider to scrape the entire content of each URL link for all the required transfers. We would then have the ability to manipulate and scrape the data as we required working offline from a local copy. This web page spider was set to run daily and to collect the material which would include between 6-9 HTML files. Once all the pages required were scraped the necessary testing could be executed. Once these pages were validated the individual transfer spider could run as required offline to allow for the extraction of data for our first version of the transfers dataset (Figure 1.2).



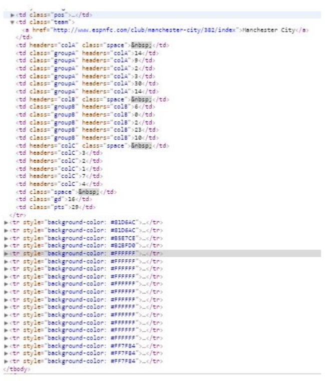
transferType	text	year	season	player	rowId	team	owner	transferTypeId
arrival	, (,, L.) - 01 Sept 2013	2013	1	Viviano	1	Palermo	Arsenal	1
arrival	, (,, B.L.) - 23 May 2008	2008	1	Vela	1	Osasuna	Arsenal	1
departure	,, L.) - 25 Augu 2008	2008	1	Senderos	1	AC Milan		2
departure	,, L.) - 29 Janu 2010	2009	1	Wilshere	1	Bolton Wanderers		2
departure	,, L.) - 23 Janu 2010	2009	1	Senderos	1	Everton		2
arrival	, (,, B.L.) - 01 July 2010	2010	1	Wilshere	1	Bolton	Arsenal	1
arrival	, (,, L.) - 31 Augu 2011	2011	1	Benayoun	1	Chelsea	Arsenal	1

Figure 1.2 Original uncleaned transfers dataset

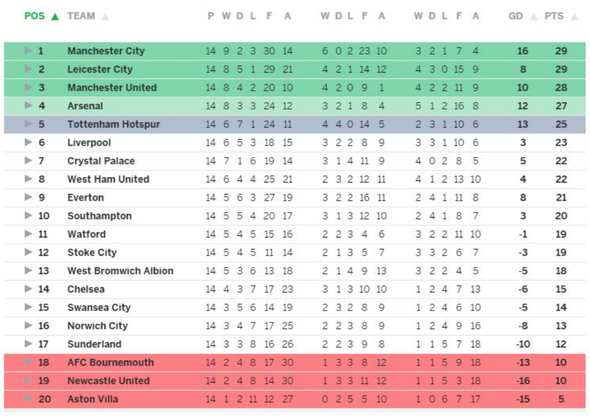
League Table Data

The second component for our overall dataset that we required was information in relation to team's annual performances. We decided the best way to analyse a team's performance over a season was to use the data from the league tables to allow us to see a variety of information. This information included the team's league standing in terms of position within the league, a teams goals scored, number of goals conceded by a team and their overall points haul for a season.

Similar to the transfer data, we again scraped the football league table's data. The data was scraped from ESPNFC.com. This was a far easier scrape then the initial transfers data scrape as the site was extremely well designed and accessible. This allowed us to cater our scrape specifically for what we required as all data was separated into separate table rows for each individual team and each element was stored in an individual cell within the HTML code. This can be seen from the below screenshot(s) (Figure 2.1).



```
▶<td class="pos">…</td>
▼<td class="team">
   <a href="http://www.espnfc.com/club/manchester-city/382/index">Manchester City</a>
 </td>
 <td headers="colA" class="space">&nbsp;</td>
 <td class="groupA" headers="colA">14</td>
 <td class="groupA" headers="colA">9</td>
 <td class="groupA" headers="colA">2</td>
 <td class="groupA" headers="colA">3</td>
 <td class="groupA" headers="colA">30</td>
 <td class="groupA" headers="colA">14</td>
 <td headers="colB" class="space">&nbsp;</td>
 <td class="groupB" headers="colB">6</td>
 <td class="groupB" headers="colB">0</td>
 <td class="groupB" headers="colB">2</td>
 <td class="groupB" headers="colB">23</td>
 <td class="groupB" headers="colB">10</td>
 <td headers="colC" class="space">&nbsp;</td>
 <td headers="colC">3</td>
 <td headers="colC">2</td>
 <td headers="colC">1</td>
 <td headers="colC">7</td>
 <td headers="colC">4</td>
 <td class="space">&nbsp;</td>
 <td class="gd">16</td>
 <td class="pts">29</td>
</tr>
▶<tr style="background-color: #81D6AC">…</tr>
▶<tr style="background-color: #81D6AC">…</tr>
▶<tr style="background-color: #85E7CE">…</tr>
▶<tr style="background-color: #82BFD0">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FFFFFF">…</tr>
▶<tr style="background-color: #FF7F84">…</tr>
▶<tr style="background-color: #FF7F84">…</tr>
▶<tr style="background-color: #FF7F84">…</tr>
</tbody>
```



POS	TEAM	P	W	D	L	F	A	W	D	L	F	A	W	D	L	F	A	GD	PTS
1	Manchester City	14	9	2	3	30	14	6	0	2	23	10	3	2	1	7	4	16	29
2	Leicester City	14	8	5	1	29	21	4	2	1	14	12	4	3	0	15	9	8	29
3	Manchester United	14	8	4	2	20	10	4	2	0	9	1	4	2	2	11	9	10	28
4	Arsenal	14	8	3	3	24	12	3	2	1	8	4	5	1	2	16	8	12	27
5	Tottenham Hotspur	14	6	7	1	24	11	4	4	0	14	5	2	3	1	10	6	13	25
6	Liverpool	14	6	5	3	18	15	3	2	2	8	9	3	3	1	10	6	3	23
7	Crystal Palace	14	7	1	6	19	14	3	1	4	11	9	4	0	2	8	5	5	22
8	West Ham United	14	6	4	4	25	21	2	3	2	12	11	4	1	2	13	10	4	22
9	Everton	14	5	6	3	27	19	3	2	2	16	11	2	4	1	11	8	8	21
10	Southampton	14	5	5	4	20	17	3	1	3	12	10	2	4	1	8	7	3	20
11	Watford	14	5	4	5	15	16	2	2	3	4	6	3	2	2	11	10	-1	19
12	Stoke City	14	5	4	5	11	14	2	1	3	5	7	3	3	2	6	7	-3	19
13	West Bromwich Albion	14	5	3	6	13	18	2	1	4	9	13	3	2	2	4	5	-5	18
14	Chelsea	14	4	3	7	17	23	3	1	3	10	10	1	2	4	7	13	-6	15
15	Swansea City	14	3	5	6	14	19	2	3	2	8	9	1	2	4	6	10	-5	14
16	Norwich City	14	3	4	7	17	25	2	2	3	8	9	1	2	4	9	16	-8	13
17	Sunderland	14	3	3	8	16	26	2	2	3	9	8	1	1	5	7	18	-10	12
18	AFC Bournemouth	14	2	4	8	17	30	1	3	3	8	12	1	1	5	9	18	-13	10
19	Newcastle United	14	2	4	8	14	30	1	3	3	11	12	1	1	5	3	18	-16	10
20	Aston Villa	14	1	2	11	12	27	0	2	5	5	10	1	0	6	7	17	-15	5

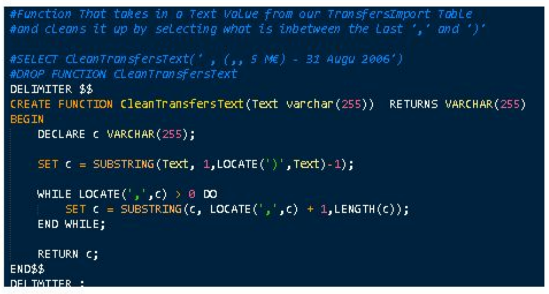
Figure 1.3 HTML code & Tranfer table on ESPNFC.com

S2: Data Preparation

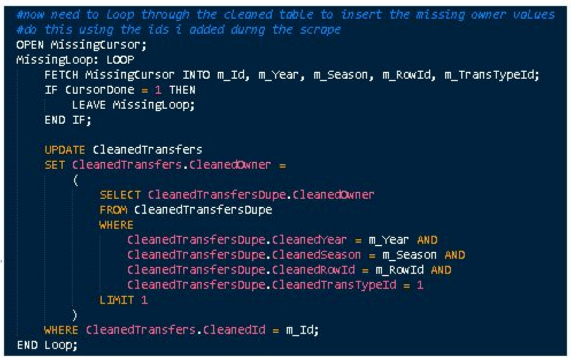
In getting our scraped data into a usable state for our project we had to implement a cleaning process. Initially we had to identify what issues were associated to our two scraped data tables. We identified simple string formatting issues, missing or null values and inconsistent numerical representation relating to the data within the transfer table and naming differences between the two tables. During the cleaning process we designed everything with scalability and automation in mind so that our solutions could be applied to other english leagues and previous years scaped with our spider.

Firstly we dealt with the formatting issues associated to the transfer table. This issue related to corruption of some data, with it all being contained within an individual string for each transfer. In dealing with this data we wrote a procedure in SQL. The procedure allowed us to import the original CSV file into a table while also cleaning the text column. This allowed for the extraction of the transfers values using a function to split the field (Figure 2.1).

The procedure also filled in missing team names for player departures. This was done by using the ID's (Figure 2.2) as described in the initial scraping of the data. Following the scrape these transfer values were represented in string format (eg, 10M, 100k etc). This was to be represented as numerical values. This was done with just a simple function to check if the string represented a million figure or a thousand figure and then just multiplying it accordingly, to get the real value. Also associated to the transfers were loans. These player loans were identified by either a back-loan (BL) or loan (L). It was required that we set flags for these values as the associated value would have been zero spend. This was again done by reading the string value and setting a flag column accordingly.

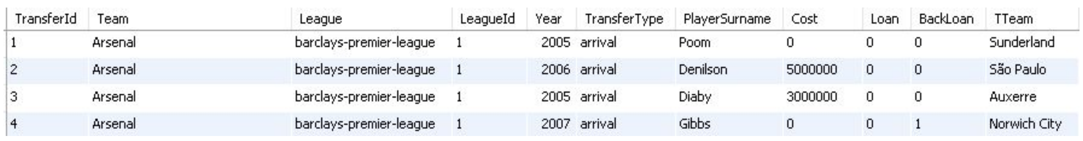


```sql
#Function That takes in a Text Value from our TransfersImport Table
#and cleans it up by selecting what is inbetween the Last ',' and ')'

#SELECT CleanTransfersText(' , (,, 5 M€) - 31 Augu 2006')
#DROP FUNCTION CleanTransfersText
DELIMITER $$
CREATE FUNCTION CleanTransfersText(Text varchar(255))  RETURNS VARCHAR(255)
BEGIN
    DECLARE c VARCHAR(255);

    SET c = SUBSTRING(Text, 1,LOCATE(')',Text)-1);

    WHILE LOCATE(',',c) > 0 DO
        SET c = SUBSTRING(c, LOCATE(',',c) + 1,LENGTH(c));
    END WHILE;

    RETURN c;
END$$
DELIMITER ;
```



```sql
#now need to Loop through the cleaned table to insert the missing owner values
#do this using the ids i added durng the scrape
OPEN MissingCursor;
MissingLoop: LOOP
    FETCH MissingCursor INTO m_Id, m_Year, m_Season, m_RowId, m_TransTypeId;
    IF CursorDone = 1 THEN
        LEAVE MissingLoop;
    END IF;

    UPDATE CleanedTransfers
    SET CleanedTransfers.CleanedOwner =
        (
            SELECT CleanedTransfersDupe.CleanedOwner
            FROM CleanedTransfersDupe
            WHERE
                CleanedTransfersDupe.CleanedYear = m_Year AND
                CleanedTransfersDupe.CleanedSeason = m_Season AND
                CleanedTransfersDupe.CleanedRowId = m_RowId AND
                CleanedTransfersDupe.CleanedTransTypeId = 1
            LIMIT 1
        )
    WHERE CleanedTransfers.CleanedId = m_Id;
END Loop;
```

<u>Figure 2.1</u> CleanTransfersText Function & <u>Figure 2.2</u> Cursor to fill in missing values



TransferId	Team	League	LeagueId	Year	TransferType	PlayerSurname	Cost	Loan	BackLoan	TTeam
1	Arsenal	barclays-premier-league	1	2005	arrival	Poom	0	0	0	Sunderland
2	Arsenal	barclays-premier-league	1	2006	arrival	Denilson	5000000	0	0	São Paulo
3	Arsenal	barclays-premier-league	1	2005	arrival	Diaby	3000000	0	0	Auxerre
4	Arsenal	barclays-premier-league	1	2007	arrival	Gibbs	0	0	1	Norwich City

<u>Figure 2.3</u> Imported transfer history after cleaning

After the initial cleaning of data we then had to identify any possible outliers that could cause a skew in our results. To do this we plotted our data using scatter plots (Figure 2.4). This allowed us visualise the collection of our data and to simply see where the outliers were located in a visual representation. Once identified we could then work to deal with this data. We excluded team transfer values for spend and income when both equalled zero.

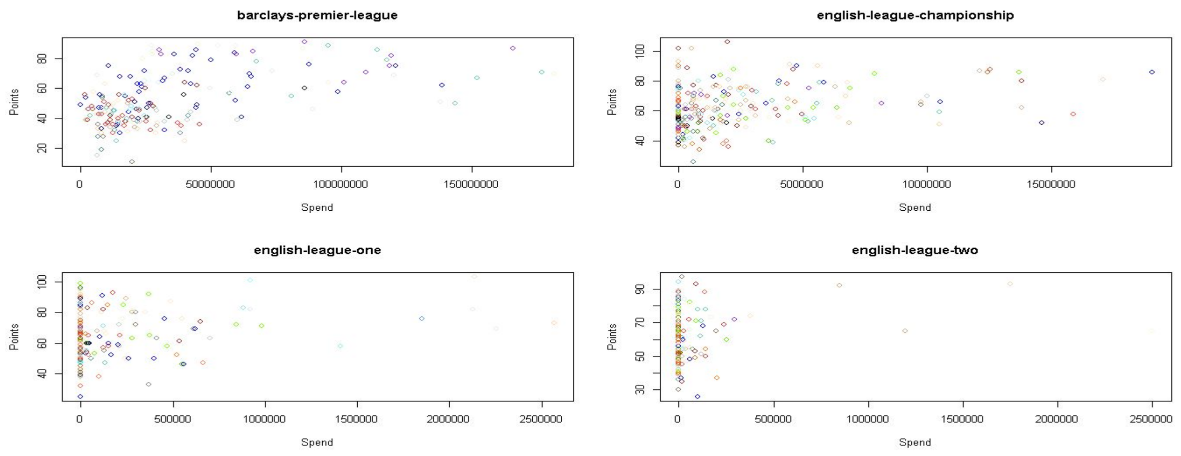
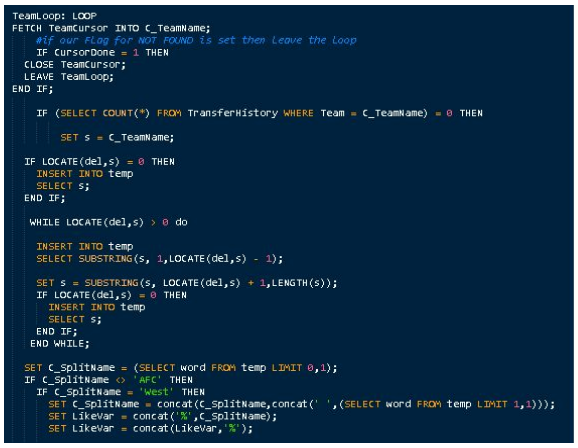


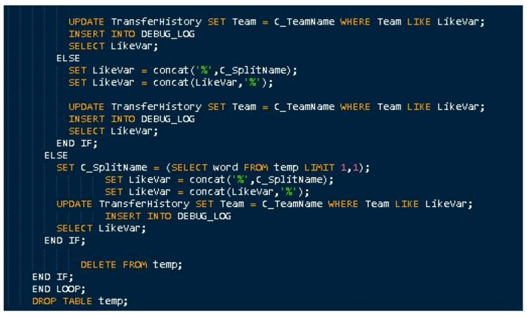
Figure 2.4 Transfers data on scatter plot

The next step in our data cleaning process was to identify the attributes in both our transfers table and points table that we felt were necessary for our prediction. We then joined the two tables using these attributes. When we initially joined the two we found that it resulted in null values. This was caused by discrepancies in the team names between the two tables. With the team name being one of the keys on the join it was vital to our dataset to fix this we wrote a procedure to parse the team names extracting values that were unique to the team names and match them using the LIKE function (Figure 2.5).

NOTE: Non-unique prefixes were hard coded in as there was only two to deal with ("AFC" & "West"). The optimal solution would have been to have a procedure to first generate a table of words that appear in teams more than once. This would allow for the script to be scalable to international team transfer histories.



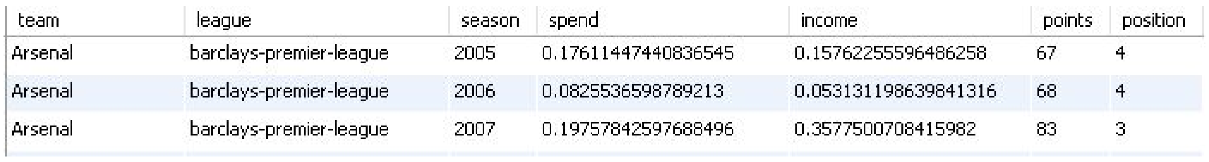
```
TeamLoop: LOOP
FETCH TeamCursor INTO C_TeamName;
    #if our Flag for NOT FOUND is set then Leave the Loop
    IF CursorDone = 1 THEN
  CLOSE TeamCursor;
  LEAVE TeamLoop;
END IF;

    IF (SELECT COUNT(*) FROM TransferHistory WHERE Team = C_TeamName) = 0 THEN

        SET s = C_TeamName;

  IF LOCATE(del,s) = 0 THEN
    INSERT INTO temp
    SELECT s;
  END IF;

   WHILE LOCATE(del,s) > 0 do

    INSERT INTO temp
    SELECT SUBSTRING(s, 1,LOCATE(del,s) - 1);

    SET s = SUBSTRING(s, LOCATE(del,s) + 1,LENGTH(s));
    IF LOCATE(del,s) = 0 THEN
      INSERT INTO temp
      SELECT s;
    END IF;
   END WHILE;

  SET C_SplitName = (SELECT word FROM temp LIMIT 0,1);
  IF C_SplitName <> 'AFC' THEN
    IF C_SplitName = 'West' THEN
      SET C_SplitName = concat(C_SplitName,concat(' ',(SELECT word FROM temp LIMIT 1,1)));
      SET LikeVar = concat('%',C_SplitName);
      SET LikeVar = concat(LikeVar,'%');
```



```
      UPDATE TransferHistory SET Team = C_TeamName WHERE Team LIKE LikeVar;
      INSERT INTO DEBUG_LOG
      SELECT LikeVar;
    ELSE
      SET LikeVar = concat('%',C_SplitName);
      SET LikeVar = concat(LikeVar,'%');

      UPDATE TransferHistory SET Team = C_TeamName WHERE Team LIKE LikeVar;
      INSERT INTO DEBUG_LOG
      SELECT LikeVar;
    END IF;
  ELSE
    SET C_SplitName = (SELECT word FROM temp LIMIT 1,1);
            SET LikeVar = concat('%',C_SplitName);
            SET LikeVar = concat(LikeVar,'%');
    UPDATE TransferHistory SET Team = C_TeamName WHERE Team LIKE LikeVar;
            INSERT INTO DEBUG_LOG
    SELECT LikeVar;
  END IF;

      DELETE FROM temp;
END IF;
END LOOP;
DROP TABLE temp;
```

Figure 2.5 Snippet of procedure scrubbing the team names, parsing and then match the different team names.

Once our procedure had finished matching team names we were able to join the two tables. The join was successful and due to the naming issues being dealt with sufficiently it did not result in any missing values. One of the outliers we identified that we felt would affect the precision of our prediction were teams who were not ever present within the top four league structure for the entire ten years. For that reason we decided to only include only ever-present teams in our final dataset. The final step we took was to normalise our spend and income values using min/max normalization. In preparation for our workshop we implemented binning on our positions attribute. These bins included a categorical hierarchy of ten even tiers however during the implementation of our algorithm we reverted this,the reasoning is described in more detail in the algorithm description.



team	league	season	spend	income	points	position
Arsenal	barclays-premier-league	2005	0.17611447440836545	0.15762255596486258	67	4
Arsenal	barclays-premier-league	2006	0.0825536598789213	0.053131198639841316	68	4
Arsenal	barclays-premier-league	2007	0.19757842597688496	0.3577500708415982	83	3

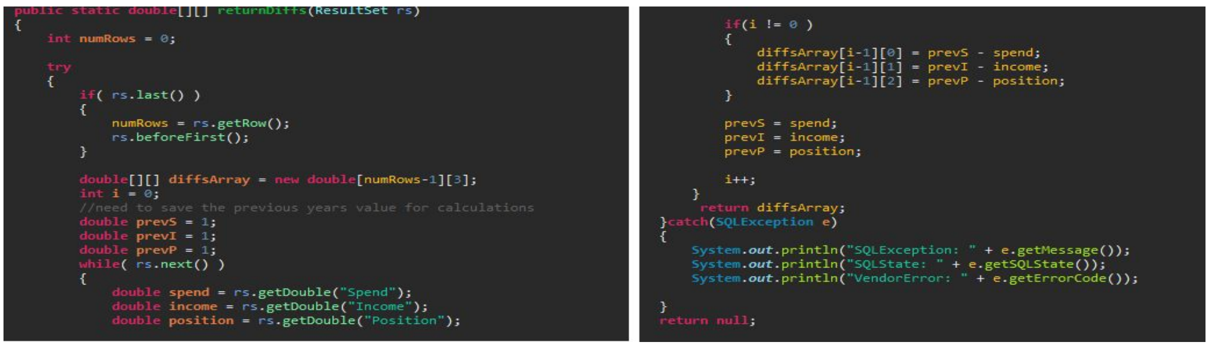
<u>Figure 2.6</u> Final workable dataset with normalization.

S3: Algorithm description

For our predictions we decided to use the nearest neighbor algorithm using the euclidean distance as our distance function. We choose this algorithm as following our data preparation it was apparent we only had two strong attributes to work off. This highlighted to us that algorithms such as decision trees would not be possible to apply to our data. Nearest neighbor stood also out among Bayes classification as we could very nicely represent the two attributes as points in a 2D space.

In adapting nearest neighbor to our dataset we felt that just simply returning the class for the team of the closest spend and income would not work. This method would not take into account the current composition of a team's squad. We also felt that the prediction would be skewed by outliers such as teams that spend very little compared to other teams in their league and teams of a higher league table positions (eg. Arsenal). The issue would also then arise for teams that did not have any spend or income. The way we dealt with this was by calculating the difference of the predicted teams spend and income to the previous year and then selecting the the four nearest neighbors to the training set. Where the training set is now the difference in each team of the same leagues spending to the previous year. The classification that is returned by the nearest neighbor algorithm is the difference in position that the spend and income resulted in. This is done using a java program as follows:

Initially our java program reads in the team name to make the prediction, their spend, income and the league that they are playing in for the prediction year. The spend and income read in is normalised with our dataset. The difference with their spend and income to their previous year is calculated. This value is the value that we use to make the prediction. Next we use the training data. This is done by querying our MySQL database using the java connector which returns a resultset. This resultset is a subset of our dataset where the data is made same league of the team it is predicting. The program then calculates the difference in spending for each team in the result set and the difference in position to their previous year. These differences are returned as a 2D array to base our prediction from (Figure 3.1).

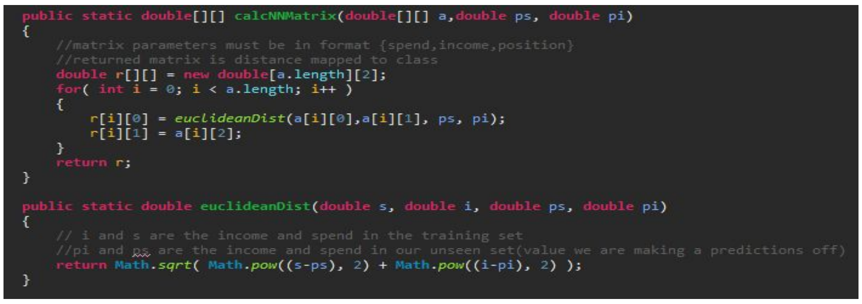


```java
public static double[][] returnDiffs(ResultSet rs)
{
    int numRows = 0;

    try
    {
        if( rs.last() )
        {
            numRows = rs.getRow();
            rs.beforeFirst();
        }

        double[][] diffsArray = new double[numRows-1][3];
        int i = 0;
        //need to save the previous years value for calculations
        double prevS = 1;
        double prevI = 1;
        double prevP = 1;
        while( rs.next() )
        {
            double spend = rs.getDouble("Spend");
            double income = rs.getDouble("Income");
            double position = rs.getDouble("Position");
```

```java
            if(i != 0 )
            {
                diffsArray[i-1][0] = prevS - spend;
                diffsArray[i-1][1] = prevI - income;
                diffsArray[i-1][2] = prevP - position;
            }

            prevS = spend;
            prevI = income;
            prevP = position;

            i++;
        }
         return diffsArray;
    }catch(SQLException e)
    {
        System.out.println("SQLException: " + e.getMessage());
        System.out.println("SQLState: " + e.getSQLState());
        System.out.println("VendorError: " + e.getErrorCode());

    }
    return null;
```

Figure 3.1 Function to calculate difference in spend & income

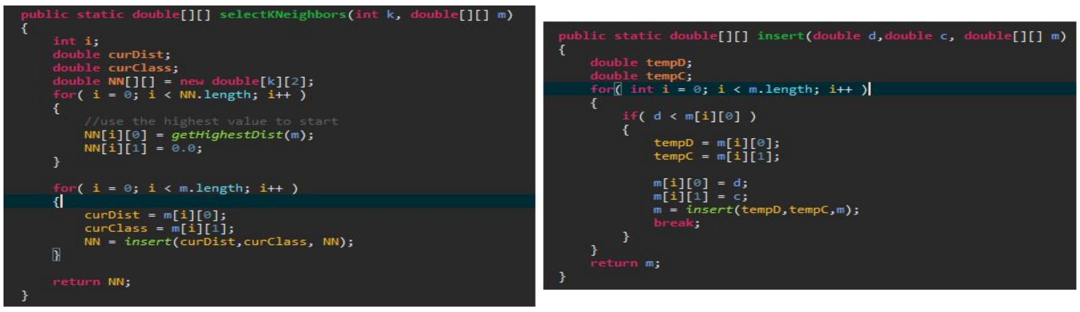
The next step is to calculate our nn-matrix. This is done by simply iterating through the previous 2D array applying the euclidean distance to spend and income against the prediction spend and income (Figure 3.2).



```java
public static double[][] calcNNMatrix(double[][] a,double ps, double pi)
{
    //matrix parameters must be in format {spend,income,position}
    //returned matrix is distance mapped to class
    double r[][] = new double[a.length][2];
    for( int i = 0; i < a.length; i++ )
    {
        r[i][0] = euclideanDist(a[i][0],a[i][1], ps, pi);
        r[i][1] = a[i][2];
    }
    return r;
}

public static double euclideanDist(double s, double i, double ps, double pi)
{
    // i and s are the income and spend in the training set
    //pi and ps are the income and spend in our unseen set(value we are making a predictions off)
    return Math.sqrt( Math.pow((s-ps), 2) + Math.pow((i-pi), 2) );
}
```

Figure 3.2 Calculating the nn matrix

We then select our k nearest neighbors in the matrix(Figure 3.3) and from these we chose the classifier that has highest count in the k-nn matrix, if all classifiers in the k-nn matrix have the same count we just select the closest classifier.



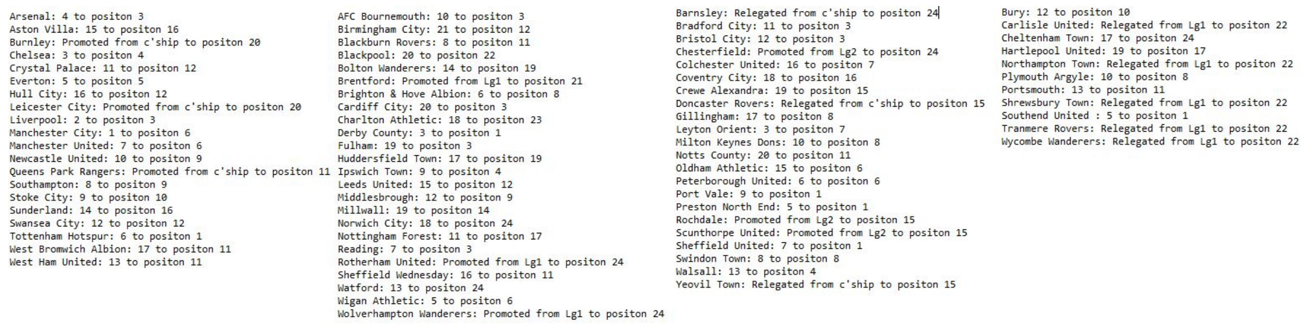
```java
public static double[][] selectKNeighbors(int k, double[][] m)
{
    int i;
    double curDist;
    double curClass;
    double NN[][] = new double[k][2];
    for( i = 0; i < NN.length; i++ )
    {
        //use the highest value to start
        NN[i][0] = getHighestDist(m);
        NN[i][1] = 0.0;
    }

    for( i = 0; i < m.length; i++ )
    {
        curDist = m[i][0];
        curClass = m[i][1];
        NN = insert(curDist,curClass, NN);
    }

    return NN;
}
```

```java
public static double[][] insert(double d,double c, double[][] m)
{
    double tempD;
    double tempC;
    for( int i = 0; i < m.length; i++ )
    {
        if( d < m[i][0] )
        {
            tempD = m[i][0];
            tempC = m[i][1];

            m[i][0] = d;
            m[i][1] = c;
            m = insert(tempD,tempC,m);
            break;
        }
    }
    return m;
}
```

Figure 3.3 Function to select k neighbors and making use of a recursive function to insert based of the lowest distance.

During our data preparation we found that outliers existed in our dataset. These outliers included teams that were either promoted or relegated to different leagues. To handle this we implemented a rule that if a team was promoted/relegated the previous year to start their position at the bottom of the league they were moved to and then apply our classification of the difference in position to that position. This is done within our function getPositionDiffPrevYear().

To test the accuracy of our algorithm we applied a function to iterate through each teams spend and income for 2014 passing it into our algorithm outputting the team's position change for examination (Figure 3.4).



```
Arsenal: 4 to positon 3
Aston Villa: 15 to positon 16
Burnley: Promoted from c'ship to positon 20
Chelsea: 3 to positon 4
Crystal Palace: 11 to positon 12
Everton: 5 to positon 5
Hull City: 16 to positon 12
Leicester City: Promoted from c'ship to positon 20
Liverpool: 2 to positon 3
Manchester City: 1 to positon 6
Manchester United: 7 to positon 6
Newcastle United: 10 to positon 9
Queens Park Rangers: Promoted from c'ship to positon 11
Southampton: 8 to positon 9
Stoke City: 9 to positon 10
Sunderland: 14 to positon 16
Swansea City: 12 to positon 12
Tottenham Hotspur: 6 to positon 1
West Bromwich Albion: 17 to positon 11
West Ham United: 13 to positon 11
```

```
AFC Bournemouth: 10 to positon 3
Birmingham City: 21 to positon 12
Blackburn Rovers: 8 to positon 11
Blackpool: 20 to positon 22
Bolton Wanderers: 14 to positon 19
Brentford: Promoted from Lg1 to positon 21
Brighton & Hove Albion: 6 to positon 8
Cardiff City: 20 to positon 3
Charlton Athletic: 18 to positon 23
Derby County: 3 to positon 1
Fulham: 19 to positon 3
Huddersfield Town: 17 to positon 19
Ipswich Town: 9 to positon 4
Leeds United: 15 to positon 12
Middlesbrough: 12 to positon 9
Millwall: 19 to positon 14
Norwich City: 18 to positon 24
Nottingham Forest: 11 to positon 17
Reading: 7 to positon 3
Rotherham United: Promoted from Lg1 to positon 24
Sheffield Wednesday: 16 to positon 11
Watford: 13 to positon 24
Wigan Athletic: 5 to positon 6
Wolverhampton Wanderers: Promoted from Lg1 to positon 24
```

```
Barnsley: Relegated from c'ship to positon 24
Bradford City: 11 to positon 3
Bristol City: 12 to positon 3
Chesterfield: Promoted from Lg2 to positon 24
Colchester United: 16 to positon 7
Coventry City: 18 to positon 16
Crewe Alexandra: 19 to positon 15
Doncaster Rovers: Relegated from c'ship to positon 15
Gillingham: 17 to positon 8
Leyton Orient: 3 to positon 7
Milton Keynes Dons: 10 to positon 8
Notts County: 20 to positon 11
Oldham Athletic: 15 to positon 6
Peterborough United: 6 to positon 6
Port Vale: 9 to positon 1
Preston North End: 5 to positon 1
Rochdale: Promoted from Lg2 to positon 15
Scunthorpe United: Promoted from Lg2 to positon 15
Sheffield United: 7 to positon 1
Swindon Town: 8 to positon 8
Walsall: 13 to positon 4
Yeovil Town: Relegated from c'ship to positon 15
```

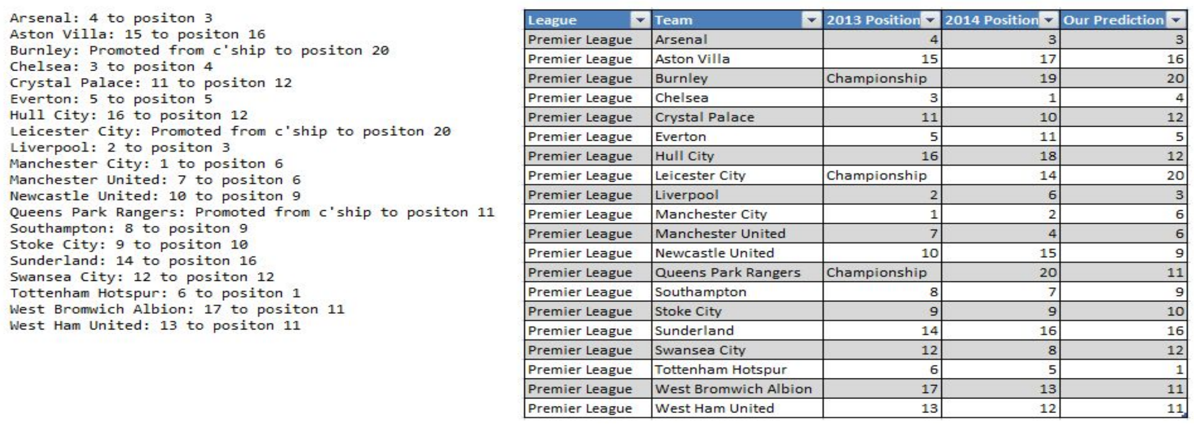
```
Bury: 12 to positon 10
Carlisle United: Relegated from Lg1 to positon 22
Cheltenham Town: 17 to positon 24
Hartlepool United: 19 to positon 17
Northampton Town: Relegated from Lg1 to positon 22
Plymouth Argyle: 10 to positon 8
Portsmouth: 13 to positon 11
Shrewsbury Town: Relegated from Lg1 to positon 22
Southend United : 5 to positon 1
Tranmere Rovers: Relegated from Lg1 to positon 22
Wycombe Wanderers: Relegated from Lg1 to positon 22
```

<u>Figure 3.4</u> Test Results for each team for 2014 season

S4: Results & Analysis

Results

For our results we first wished to validate how accurate our algorithm was in predicting the overall influence that money had in football. We excluded all the data for the Barclays Premier League 2014 from our dataset and applied the algorithm to use the nine previous years to make the following prediction for the table for 2014.



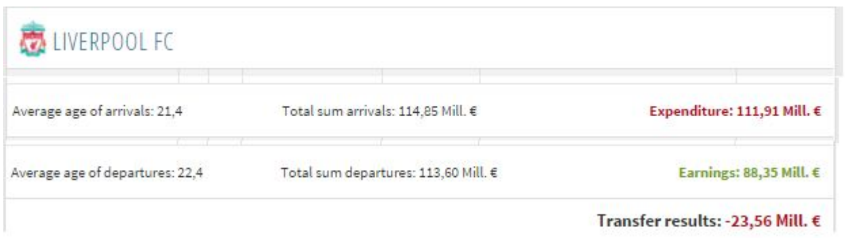
```
Arsenal: 4 to positon 3
Aston Villa: 15 to positon 16
Burnley: Promoted from c'ship to positon 20
Chelsea: 3 to positon 4
Crystal Palace: 11 to positon 12
Everton: 5 to positon 5
Hull City: 16 to positon 12
Leicester City: Promoted from c'ship to positon 20
Liverpool: 2 to positon 3
Manchester City: 1 to positon 6
Manchester United: 7 to positon 6
Newcastle United: 10 to positon 9
Queens Park Rangers: Promoted from c'ship to positon 11
Southampton: 8 to positon 9
Stoke City: 9 to positon 10
Sunderland: 14 to positon 16
Swansea City: 12 to positon 12
Tottenham Hotspur: 6 to positon 1
West Bromwich Albion: 17 to positon 11
West Ham United: 13 to positon 11
```

League	Team	2013 Position	2014 Position	Our Prediction
Premier League	Arsenal	4	3	3
Premier League	Aston Villa	15	17	16
Premier League	Burnley	Championship	19	20
Premier League	Chelsea	3	1	4
Premier League	Crystal Palace	11	10	12
Premier League	Everton	5	11	5
Premier League	Hull City	16	18	12
Premier League	Leicester City	Championship	14	20
Premier League	Liverpool	2	6	3
Premier League	Manchester City	1	2	6
Premier League	Manchester United	7	4	6
Premier League	Newcastle United	10	15	9
Premier League	Queens Park Rangers	Championship	20	11
Premier League	Southampton	8	7	9
Premier League	Stoke City	9	9	10
Premier League	Sunderland	14	16	16
Premier League	Swansea City	12	8	12
Premier League	Tottenham Hotspur	6	5	1
Premier League	West Bromwich Albion	17	13	11
Premier League	West Ham United	13	12	11

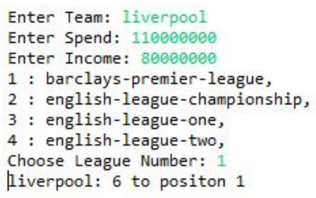
<u>Figure 4.1</u> Results for Barclays Premier League for 2014 season

The results from running our algorithm with our test data showed that our model can predict whether a team will increase or decrease in their league position with a seemingly high accuracy. Although it is unable to predict the exact position accurately. We found that using spend and income alone for our model restricts the algorithm. It is unable to determine how efficient the money spent was in terms of value for players. From analysing our results we found that some teams would sell one of their best players and use the money from the sale to buy a larger quantity of players of a lesser quality. These other players did not compensate for the loss of the individual sold (eg. Suarez, Bale, Ronaldo all leaving teams in the Barclays Premier League). Taking these results into consideration we reavaluated our dataset to try and determine what other attributes would be needed.

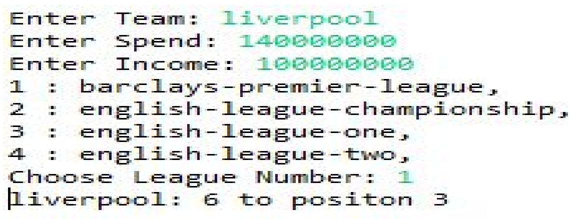
Once we had verified that our prediction was largely accurate we then looked to apply it to make the prediction for the 2015 season. Unfortunately as the January transfer window has not yet commenced we do not have the full data required to make this prediction. Within our Java program we wrote in the functionality to use the algorithm to input team details at run-time to make the prediction. This allows you to test data for a team based on estimated spending for a team (Figure 4.2 (Liverpool)).


LIVERPOOL FC

Average age of arrivals: 21,4	Total sum arrivals: 114,85 Mill. €	Expenditure: 111,91 Mill. €
Average age of departures: 22,4	Total sum departures: 113,60 Mill. €	Earnings: 88,35 Mill. €
		Transfer results: -23,56 Mill. €


```
Enter Team: liverpool
Enter Spend: 110000000
Enter Income: 80000000
1 : barclays-premier-league,
2 : english-league-championship,
3 : english-league-one,
4 : english-league-two,
Choose League Number: 1
liverpool: 6 to positon 1
```

Estimating that Liverpool will spend another €30,000,000 and earn another €20,000,000 from player sales, judging by spending patterns for previous years for the January Transfer Window. We see the projected position decreases.


```
Enter Team: liverpool
Enter Spend: 140000000
Enter Income: 100000000
1 : barclays-premier-league,
2 : english-league-championship,
3 : english-league-one,
4 : english-league-two,
Choose League Number: 1
liverpool: 6 to positon 3
```

It can be seen from the above test results for Liverpool F.C that the amount spent is not the only factor for the overall prediction. A similar weight is placed on a team's income for selling players. We feel that this was important to establishing accurate predictions.

We felt that our results did make a prediction for teams well but also highlighted which teams were more efficient in the amount of money spent. This can be seen by teams that underachieve or over achieve in relation to league position based on our model.

Conclusion

From this assignment the importance of strong data preparation was evident. The identification of outliers and how they would skew our results was vital to the overall prediction. The majority of our work was spent on cleaning our data and transforming it in such a way that it would greatly increase our algorithms strength. We feel this work was highlighted by the generation of our prediction using our defined model and the well managed data.

A slight drawback to our project is the amount of data we scraped in preparation. The fact that our data only spanned over ten years was a minor drawback. We feel that if we were to do this project again, more data would allow for a better comparison to be made.

We found that if we had individual player ratings for each team we could better define a tuple set to measure a team's strength. We began looking online and found that the video game FIFA has a database on their website for their game mode “Ultimate Team” which includes statistics on every player. Different statistics are defined for multiple different attributes such as attack, goalkeeping and defence etc. We feel that this data would be perfect for us to achieve much stronger results.

By spending the majority of our time on the preparation it allowed us to create what we feel is a strong model even with our lack of data. If we were to redo our assignment and add more historic data for the league tables and associated transfers and also to incorporate player statistical data as described above into our model we feel we could achieve even stronger results.