Project Status Report

CS 6350 Big Data Management and Analytics

Anurag Nagar

Chandani Madnani- cxm152930
Roshan Ravikumar -rxr151330
Salil Kansal – sxk150430
Twinkle Sharma – txs151730

Table of Contents

I. Design

Problem Definition

To predict the outcomes of this year's US men's college basketball tournament using a combination of rich historical data and computing power. This competition consists of two stages. For this project, we have chosen to work on the first stage.

In the first stage of the competition, Kagglers will rely on results of past tournaments to build and test models. Using the constructed models the objective is to predict probabilities for every possible matchup in the past 4 NCAA tournaments (2013-2016).

Dataset Details

Below we describe the format and fields of the "essential" data files required for our project.

1. TourneySeeds: This file consists of 3 fields. The season, the seed of the team in that seasons tournament, and the team ID.
2. RegularSeasonCompactResults: This file identifies the game-by-game results for 32 seasons of historical data, from 1985 to 2016. Each row in the file represents a single game played.
    - “season”-year of associated entry in seasons file in which final tournament occurs
    - “wteam/lteam”-team id of the winning/losing tem
    - “wscore/lscore”-points scored by winning/losing team
    - “numot”-number of overtime periods
    - “wloc”-home team or visiting team
3. RegularSeasonDetailedResults: This file is a more detailed set of game results, covering seasons 2003-2016. This includes team-level total statistics for each game (total field goals attempted, offensive rebounds, etc.) The column names are self-explanatory to basketball fans (as above, "w" or "l" refers to the winning or losing team):
    - Wfgm/ Wfga-field goals made/attempted
    - Wfgm3/Wfga3-three pointers made/attempted
    - Wftm/Wfta-free throws made/attempted
    - Wor/wdr-offensive/defensive rebounds
    - Wast-assists
    - Wto-turnovers
    - Wstl-steals
    - Wblk-blocks
    - Wpf-personal fouls
4. TourneyCompactResults: This file identifies the game-by-game NCAA tournament results for all seasons of historical data. The data is formatted exactly like the regular_season_compact_results.csv data. Note that these games also include the play-in games (which always occurred on day 134/135) for those years that had play-in games.

File Name	No. of Instances	No. of Features
TourneySeeds.csv	2150	3
RegularSeasonCompactResults.csv	150684	7
RegularSeasonDetailedResults.csv	76636	13
TourneyCompactResults.csv	2050	7

Algorithm and Pseudo-code

One of the important phases of the project is to come up the ideal set of predictors that give us the best result. In layman's terms, from the data provided by Kaggle, we need to be able to identify what are the indicators for any given team winning a game in March Madness.

Based on our initial discussions, we have identified the following as the key indicators that capture if a team will do well in each year's NCAA tournament:
(Please note that all indicators are specific to each season)

- Win percentage of the team in Regular Season.
- Seeding in the NCAA tournament.
- Average point difference with which they won their games in the Regular Season.
- Away games win percentage in Regular Season.
- Their record in the last 10 games of the Regular season (Shows the teams momentum coming into the NCAA Tournament).

Also, please understand that these indicators only need to be calculated for the teams that made it to the NCAA tournament that season. There is no point calculating the indicators for teams that didn't make the tournament that year as they would not be helpful in training our model. In addition, the model has been trained only on the matchups that happened in that year's NCAA tournament.

After arriving at the best model, we can start to predict results for every possible matchup in the past 4 NCAA tournaments (2013-2016) and calculate the accuracy by comparing with the actual outcome of the game.

Example: Predicting probabilities for every possible matchup in 2013
This will be the testing or validation phase where we use the trained model to predict outcomes. We must create the above-mentioned indicators for all the teams based on their 2013 Regular Season performance. We will then pass these indicators for every possible matchup through our trained model. The model will predict the winning team when a team with certain type of indicators plays a team with a certain type of indicators. The model can do this because it has been trained using historical data, where its might have seen 2 teams with similar Regular season indicators and it knows who won when they met in the NCAA tournament.

Calculating the Indicators per Season

Indicator	How to calculate?
Win percentage of the team in Regular Season	RegularSeasonCompactResults.csv can be used to calculate the win percentage of a team during each regular season. This csv captures all the matchups between teams in across the 32 divisions in the Regular season.
Seeding in the NCAA tournament.	TourneySeeds.csv captures the seeding of each team in their respective region. Each region consists of 16 teams and this is the starting point of the NCAA tournament. There are 4 regions.
Average point difference with which they won their games in the Regular Season / Average point difference with which they lose their games in the Regular Season.	RegularSeasonCompactResults.csv has the fields Wscore (winning teams score) and Lscore (Losing teams score). This can be used to calculate the average point difference they with which they win or lose by for each Regular Season.
Home games win percentage in Regular Season / Away games win percentage in Regular Season.	RegularSeasonCompactResults.csv has a field Wloc (Win Location). H is home, A is Away and N is Neutral. This field will help us in calculating this indicator. Sometimes it is unclear whether the site should be considered neutral, since it is near one team's home court.
Their record in the last 10 games of the Regular season	This also can be calculated from RegularSeasonCompactResults.csv, by just taking the last 10 records for each season per team.

Big Data Technologies used

We have used Spark to process our data, build our indicators and train our model. Scala is our language of choice. We are heavily reliant on Spark's MLlib to perform our predictions and derive conclusions.

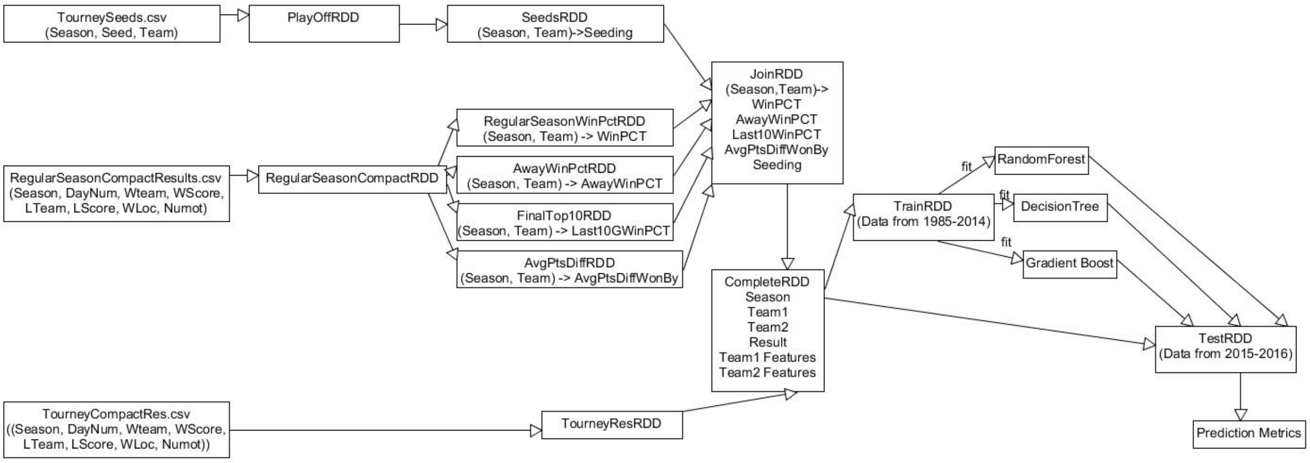
As for the machine learning techniques to be used, we have used the following classifiers to train our data:

- Random Forests
- Decision Trees
- Gradient Boosted Trees

Why Spark?

Since spark uses an in-memory design, it is known to be up to 100x faster than Hadoop MapReduce. It also has an excellent Machine Learning library, which made it the clear winner over other Big data stacks.

Data Flow Diagram



How we handle bad/missing data? How we prevent overfitting?

The dataset provided to us does not have any missing or bad data. We have still checked for the missing values and made sure that the final data that we are training the model on is free of any errors and bad values.

Overfitting is prevented by different methods in all the different classifiers. Following is the explanation of the methods adopted for each of the classifiers.

1. Random Forest: Since Random forest is based on decision tree, there is a parameter which we can tune called Maximum Depth. This max depth prevents further splitting of the data and just take the classification to be the majority value of the data till now.

2. Decision Tree: Same is the case with decision tree, where we have set the maxDepth parameter to be 5. This value was chosen after testing different values and getting the accuracy.

3. Gradient Boosting: In gradient boosting we are choosing the classifiers to be decision trees only. So here also the same maxDepth attribute is present and we have chosen the value to be 5. Here also this value was found after thorough testing on different values.

II. Analysis of Results

Classifier	Precision	Confusion Matrix
Random Forest	75%	58.0 19.0 13.0 39.0
Decision Tree	71%	56.0 22.0 15.0 36.0
Gradient Boosted Tree	70%	54.0 21.0 17.0 37.0

We obtained the highest accuracy using the Random Forest Classifier. We noticed that the winner of the competition this year had an accuracy of 76%. An accuracy of 75% for Random Forest was a satisfactory point to stop at.

III. Conclusion

In this section, you will present your conclusions. Following are key points:

- Explain how using Big Data helped you with this project? Explain how using Big Data helped you arrive at a better/faster/more efficient solution.
- Describe what you learned in this project.
- Describe how your technique/strategy can be improved.

How Big Data Helped?

The march madness is the biggest competitions of college basketball in the United States. There are hundreds of attributes and every data scientists tries to predict the tournament.

For creating a training data as input to the model we need to calculate a lot of attributes and many joins are involved between tables which have thousands of rows. Big Data, specifically spark and it's in memory computing helps us in efficiently executing the joins.

We have programmed the code in such a way that first we are creating five different tables (one table for each attribute). Joining the tables using native Hadoop map reduce code would have been a tedious task. Therefore, the choice of spark helped us prevent the code from getting too verbose. In addition, joins were sped by using a distributed map reduce framework like Spark. The same joins in a traditional relational database would have taken more time since, it is not a distributed environment.

Spark also has the lazy computation and interpreter mode which helped us in easy testing of the code and simplified the development process. Scala helped us in writing efficient code without worrying much about the syntax as it is very close to the pseudocode.

MLlib is de facto library available in spark which has all the classifiers one would ever need to create a machine learning project on big data. Documentation of machine learning and the rich user base helped us in the development process. It also made the training of data an ease which would not have been possible in a non-distributed environment.

What we learned?

All four of us were alien to the tournament and how it went through. Participating in the Kaggle Challenge helped us to get accustomed to the tournament. Finding the best parameters were a lot of fun and involved a lot of thinking as to what makes a team win a match. This tournament helped us create an end to end algorithm which can now be used on any of the future years to find out the winners of various stages of the tournament.

Areas of Improvement

We got a maximum of 75% accuracy by just 5 attributes. We are certain that by thinking of more parameters and by increasing our feature dimension space we can improve upon the accuracy of the result.

IV. Role of each Member

Name	Contribution
Chandani Madnani (cxm152930)	Calculating Last 10 games Win Percentage.
Roshan Ravikumar (rxr151330)	Constructing training and testing data from calculated features
Salil Kansal (sxk150430)	Calculating Win Percentage and Away Win Percentage.
Twinkle Sharma (txs151730)	Calculating Average Points a team wins by and seed of the team in that year's tournament.

Sections developed together as a Team

- Identifying the best features to describe a team.
- Choosing classification techniques that would give the best accuracy.
- Testing the accuracy of the model built by comparing the predicted results with the actual outcomes of the games in the playoffs.

V. References

- https://dzone.com/articles/applying-machine-learning-to-march-madness
- https://spark.apache.org/docs/2.1.0/mllib-evaluation-metrics.html
- https://en.wikipedia.org/wiki/NCAA_Division_I_Men%27s_Basketball_Tournament
- https://www.kaggle.com/c/march-machine-learning-mania-2017/data