Alperen OZKURT s1037565
Ege SARI s1034535

Online Game Win Factors Research Report

Abstraction

Our project is concerned about a "problem" and aimed to get results from that problem based on a domain. Our domain is 3 data sets that includes the data for an online competitive game and differs with the experience of the players. Our research problem is to find what are the parameters that affects the teams for winning in this game. We argued what parameter is more advantageous than others and what affects the win rates. For this problem, basically, we have used classification and dimension reduction approaches and methods. For these approaches, we have used decision trees, principal component analysis, covariance matrix and XGBClassifier. Also we have learned some important results from our research. One result that we have is, we can say XGBClassifier is better than decision trees, and about the data we have understand the ranking's different strategies and priorities for winning the game.

1. Application Domain and Research problem

Our domain consists of 3 data sets which are about a game called League of Legends (LOL). . Before explaining our research problem, I should explain this game and its rules, the purpose of the players and winning conditions briefly. There are two teams of 5 players each. These teams have a base tower called Nexus. Each team tries to protect it from the other team. The team that destroys the other team’s Nexus wins. Also there are towers that protects Nexus. All the players of a team have different roles. Since they want to destroy the Nexus, they have to be powerful and control the map. They can do this by earning gold in the game. To earn gold, the player can kill another player in the opposite team, or help a player for killing. This is the main resource of gold. Also there are jungle sides in the map where there are monsters that can be killed for money and also some of them, gives some perks like shield protection or attack power. These are namely Baron, Drake, Blue and Red. This was the game that we have data for briefly.
Now we can explain our domain and its data set. We have found a data set collection of 3 data set in Kaggle, namely “League of Legends High Elo Ranked Games 2020”. The data sets differs from each other as their names suggests, the players ranking which is about their experience and skillfulness. The name of data sets are “Master Rank”, “Challenger Rank” and “Grand Master Rank”. Each data set contains 50 columns that has values for important things that related to game we have been informed for example how many Drakes blue team killed, how many gold earned by red team or how many kills got by one player average in blue team and also which team won. And in data sets, each row is an instance of a game that played between the blue and red team. The data set “Master Rank” contains 107125 rows, the data set “Challenger Rank” contains 26905 rows and the data set “Grand Master Rank” contains 65897 rows. So basically every data set is a collection of game statistics that played by players have a specific ranking and tells us the details of the game with the accomplishments of each team.

As we have introduced our data sets, it can be interfered that there are lots of attributes. Some of them differ from the others, or affect inversely proportional. Also some of them are close and affect each other directly proportional or also become a cause and result of each other. In addition, there must be some attributes that do not affect the result of match, or affect less but also there must be some attributes that affect more the result of game, or even cause it directly. They are actually criterions that indicates the result of the game. So our main motivation for research problem lies in here. We have to find the criterion(s) or the path that the teams should follow to win the game. Also we should analyze the data and we should get some results. We should find answer the question like which criteria is more important or which one does not affect the game. Or can we make a generalization for some of the attributes, or not? If we can what are they? In this research we tried to find answers to these questions.
For this research, also we have another motivation. We both love playing this game, and we wanted to apply our knowledge and skills from practical sessions and lectures with data mining techniques to this data sets, so that we can see an application of data mining in real life and we thought it will be very good for our development in both data science and the game.

2. Related Work

We have found some work that related to our research problem and indeed, one of them[1] is inspired us. But it has major flaws. Below, we have compared that research with its data set and our research problem and data set :

- The related data set has less columns, specifically 40. And we find that they are not much related to winning option as we know game. Our data sets contains 50 columns and they are more related to game.
- The related data set far less rows, so we it seems it does not countable since it has not data enough, but our data sets are much bigger and we thought it could be more secure to classify the data.
- The related data set is just one and it belongs to diamond elo, which is lower ranking than our lowest ranking master elo. So in this way, we thought the data contains game statistics played by inexperienced players so that it will not reflect the real strategies or tactical movements. Also our data sets are three which gives us to compare each other among them. But the related data set is only one and we thought it would be not enough.
- The related data set has data for only first 10 minutes of the game, a normal game lasts for 30-40 minutes, so we can say that related work only interested in %25-%30 of the game and tries to predict the outcome of the game. We find it very insufficient and we thought we should have the data of the whole game.
- The related work also uses dimension reduction as we do, but it only reduces nearly 20 columns, but we can reduce more than 25 columns. So we can eliminate the unnecessary data and get the core of the data, but the related work is not able to do it properly.

This related work has inspired us and gave us an idea, also gave us hint how to approach to our data, but it does seem it is only a small research compared to ours.

3. Data Sets and Pre-processing Methods

3.1 About data collection

Our data collection contains three data sets which is taken from this Kaggle website:

https://www.kaggle.com/gyejr95/league-of-legends-challenger-ranked-games2020

Each data set represents games played in different player skill groups. And their number of rows changes according to that. From less skilled to the most skilled, it is named as Master, Grand Master and Challenger. And the played games (rows in our case) are lower in high elo games in contrast to low elo games.

-107125 rows and 50 columns (Master Rank)

-65897 rows and 50 columns (Grand Master Rank)

-26905 rows and 50 columns (Challenger Rank)

3.2 Pre-processing methods

We removed and combined some of the columns according to these main reasons:

3.2.1 Removed columns because of being unrelated

gameId, gameDuraton

gameId is a label type of these recorded game in order to find it in database.

gameDuraton is more like an unnecessary data to work on it because it doesn't affect the result of the game by itself.

1 The work has the data set in this link : https://www.kaggle.com/bobbyscience/league-of-legends-diamond-ranked-games-10-min/tasks?taskId=2381. Also there was a work related to our problem in this work : https://www.kaggle.com/maheshpatapalli/lol-ranked-1.

3.2.2 Removed columns because of information is derivable from opponent team

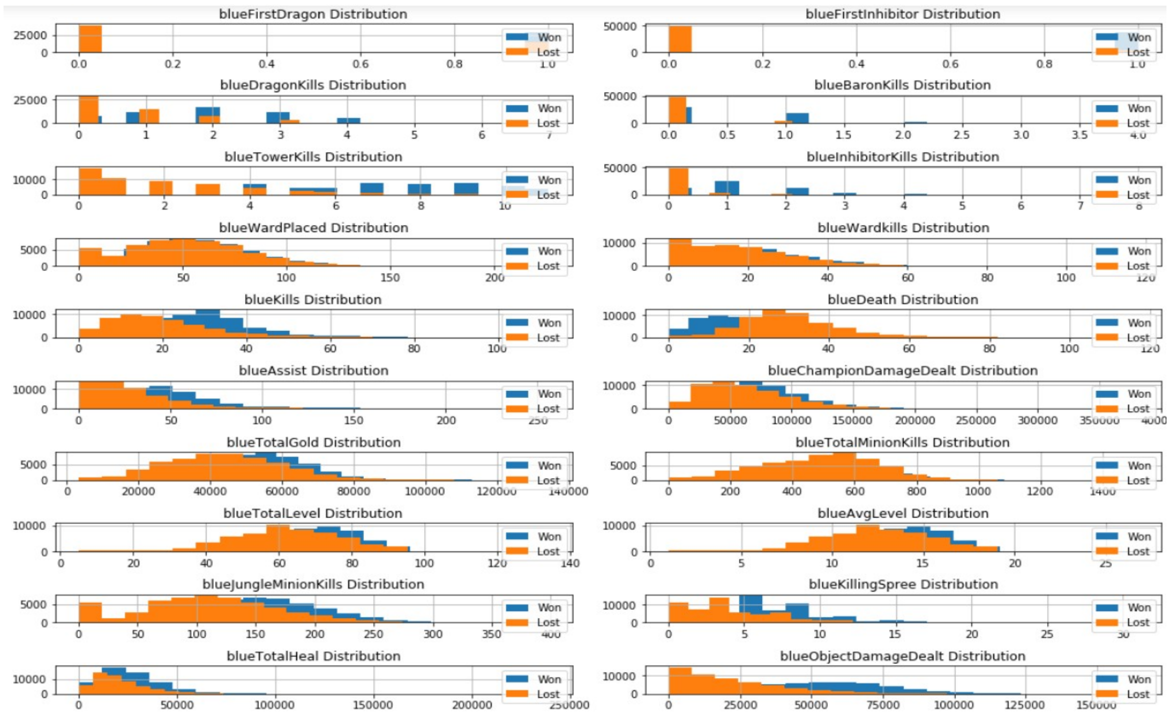
redFirstBlood, redFirstTower, redFirstDragon, redFirstInhibitor

If blueFirstBlood is 1 then redFirstBlood is 0 automatically and this applies for other columns as well.

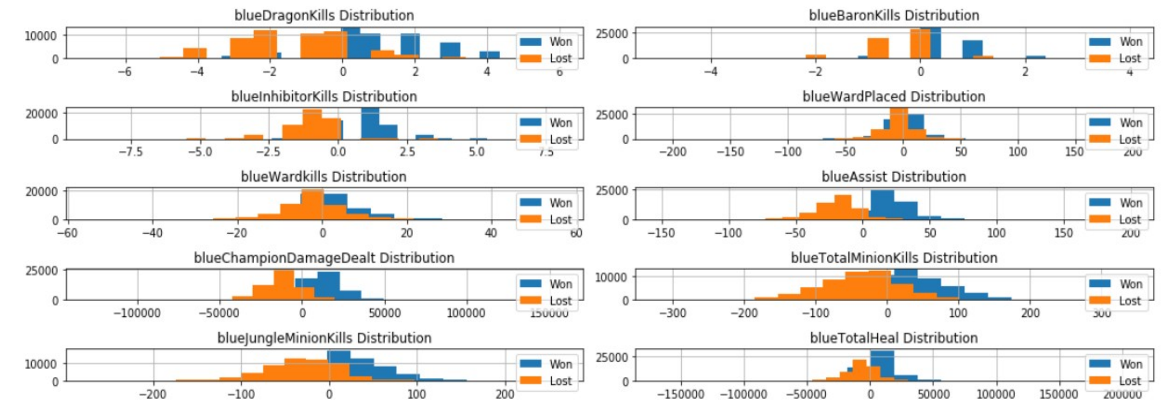
3.2.3 Combined columns because of their difference makes much more sense

DragonKillsBaronKills, InhibitorKills, WardPlaced, WardKills, Assist, ChampionDamageDealt, JungleMinionKills, TotalHeal

Game winner is mostly related with making gaps and being advantageous team. That’s what we were thinking before starting this investigation. As we were investigating through the data with graphs and statistics (which you can find in the notebooks), we also saw that the distribution of these columns contains outliers and don’t have normal distribution. By their getting difference as our new columns, you will see how we fixed it(not entirely have normal distribution but it is better to investigate at this point). Figure 3.2.3.A represents original data distribution and 3.2.3.B represents the processed version’s distribution.



3.2.3.A



3.2.3.B

3.2.4 Needs to be deleted columns because they are really co-related with win condition

TotalGold, TotalMinionKills, TotalTowerKills, TotalLevel, AvgLevel, Kills, Death, ObjectDamageDealt

These were the columns we thought we should get rid of them after data co-relation statistics with their variance on their data. These were basically macro information about on-going game:

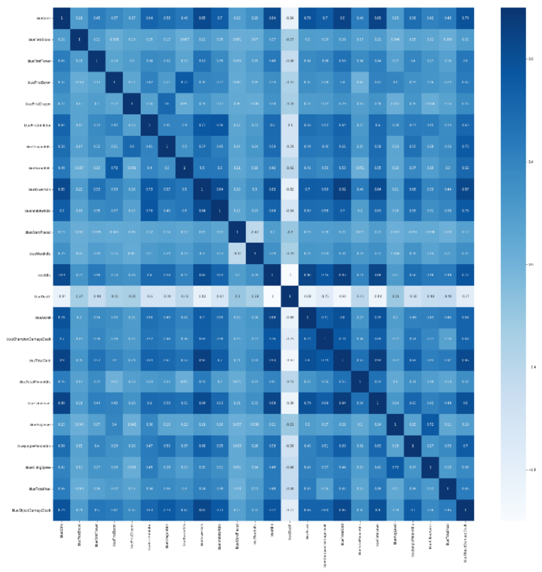
Winning and losing, both seek to secure objectives.

Differences in dragon, baron, tower, and inhibitors destruction are greater than the differences in gold/creep score (CS)/ wards placed/wards destroyed

Overall, the differences in macro management are greater than the differences in micro management.

Dataset will benefit from a transformation to move closer to a normal distribution.

That's why we dropped some of these macros and kept more micros. We were more interested in micro changes when we were starting to this actually and we are happy that our research directed us to the same point we wanted. Down below you can see the correlation matrix before applying this step as a visual but not the mathematical values. These can be found in notebook as well.



before appyling 3.2.4

3.2.5 Preparing our train set and test set

We took %70 of our data as train set and rest of it as test set. We used sklearn’s model_selection library for this with stratifying the data.

4. Our Approach and Methods

4.1 Helper algorithms that we used to improve our main approach

We used double cross validation in order to set up our attribute variables in classifiers. We used sklearn’s datasets, model_selection, ensemble and metrics libraries for this part. These attributes were n_estimartors(between 100,400 and 1000) and max_features(between 4, 7 and 10) that we tested with gridSearch on randomTreeClassifier. That takes quite a time to see the results (should be avoided to run in the notebook!!!) but we had to wait it and best results we got were with 100 estimators(400 is also close to that) and with 4 max_features.

4.2 Main approach

4.2.1 Applying XGB Tree Classifier

We put our evaluation sets and fit them into xgboost's classifier with our previously found n_estimators with learning rate 0.3.

4.2.2 Applying Decision Tree Classifier

We put our evaluation sets and fit them into Decision Tree Classifier from sklearn with our previously found max_features with minimum samples of 5000 for master games, 3000 for grand master games and 1000 for challenger games(Amount of minimum samples depends on data set size, it is bigger for large data sets). Also, quality of the split is measured by 'gini'.

4.2.3 Applying PCA

We apply PCA for getting rid of sharp results that we had previously. And fit them again to our classifiers to test their performance.

4.2.4 Rationale behind this approach

Double cross validation process is pretty straight forward, we pre-calculated the parameters for classifiers to get better accuracy results.

Before starting this project, we wanted to apply different classifying techniques other than applying second tree classifier. But while we were doing our research, we saw many tree classifiers so we wanted to test their differences. In which parts they differ attributes, do they prioritize branching attributes, which one is more trustable? These questions got answered after we see the metrics and the graphs that classifiers produced.

After the first results, we applied the PCA for get rid of noise in the data and dimension reduction with more general tree, for reducing overfitting. As you might remember we dropped most co-related features. That was also a good preparation for PCA process, otherwise PCA does it for you and it might end up badly (You will see the results and comments on them in sections 5 and 6). Also, we investigate the graphs and metrics of this both classifiers.

5. Main results of the project

5.1 Metrics results

ACCURACY	Master - OG	GrandMaster-OG	Challenger- OG	Master- UP	GrandMaster-UP	Challenger- UP
XGB-1	0.9911008774659282	0.9913500935808589	0.9907086223984143	0.9806148484659905	0.9789063685568314	0.9804261645193261
DEC-1	0.9163295786918912	0.9295867266933077	0.9390485629335976	0.919161117680005	0.9223531792199909	0.958374628344896
XGB-PCA	0.511170576887174	0.5044261217057009	0.5007433102081269	0.9468230754869624	0.9415751934847488	0.9501982160555005
DEC-PCA	0.510361565747713	0.499064191410079467	0.5038404360753221	0.9468853071130748	0.9420810359654004	0.9504459861248761

5.2 Tree Inspection on branch breaking points

Importance of the attributes goes down respectively, from start to the end. For example at down below, in **XGB Tree win condition strong breaking points**, redTeamToweKills classifies more data while redFirstInhibitor classifies less.

5.2.1.1 Master elo games – original table

XGB Tree win condition strong breaking points:

redTeamTowerKills, blueTowerKills, bluetotalMinionKills, redTotalMinionKills, redFirstInhibitor, blueInhibitorKills, blueTowerKills, blueKills, blueDeath, blueAssist, redFirstInhibitor

XGB Tree win condition weak breaking points:

redKillingSpree, redDragonKills, blueDragonKills, redFirstBaron, redAssist

Decision Tree win condition strong breaking points:

redTeamTowerKills, blueFirstTower, blueChampionDamageDealt, redFirstInhibitor, blueTowerKills, bluetotalMinionKills, redTotalMinionKills, redTotalHeal, blueTotalLevel, redKills, blueInhibitorKills, blueTowerKills, blueKills, blueDeath, blueAssist

Decision Tree win condition weak breaking points:

redTotalGold, BlueWardPlaced, redTotalGold, redWardPlaced, redKillingSpree, redDragonKills, blueDragonKills, redFirstBaron, blueKillingSpree, redChampionDamageDealt, redAssist

5.2.1.2 Master elo games – Updated table

XGB Tree win condition strong breaking points:

assistDifference, inhibitorDifference, MinionKillsDifference, HealDifference

XGB Tree win condition weak breaking points:

jungleMinionDifference, championDamageDealtDifference

Decision Tree win condition strong breaking points:

assistDifference, inhibitorDifference, baronKillsDifference, MinionKillsDifference, HealDifference jungleMinionDifference, championDamageDealtDifference

Decision Tree win condition weak breaking points:

blueFirstBaron, blueFirstInhibitor, WardPlacedDifference

5.2.2.1 Grand Master elo games – Original table

Original table contains nearly the same breaking points for 3 data sets so we wanted to mention about main differences we saw.

Main breakpoint in XGB is redTowerKills while Decision Tree’s is redFirstBaron.

XGB Tree has much more dependencies in contrast to Master elo games. It mainly contains attributes that in master elo’s original table decision tree. And there are some new dependencies which is actually the micro game changers in the game, such as: blueWardPlaced, blueFirstInhibitor, blueTotalLevel, redChampionDamageDealt, firstInhis, redKillingSpree.

Decision Tree is quite similar to previous version itself but now such attributes like redDragonKills, redJungleMinionKills, blueAvgLevel, redDeath like such attributes occurs in the table.

5.2.2.2 Grand Master elo games – Updated table

XGB Tree win condition strong breaking points:

assistDifference, inhibitorDifference, MinionKillsDifference, dragonKillDifference, baronKillDifference, blueFirstBaron

XGB Tree win condition weak breaking points:

championDamageDealtDifference, HealDifference, wardPlacedDifference, wardKillsDifference

Decision Tree win condition strong breaking points:

championDamageDealt, blueFirstBaron, assistDifference, inhibitorDifference, dragonKillsDifference

Decision Tree win condition weak breaking points:

blueFirstDragon, blueWardPlaced, totalMinionKillsDifference

5.2.3.1 Challenger elo games – Original table

Original table contains nearly the same breaking points for 3 data sets so we wanted to mention about main differences we saw as we did before.

Main breakpoint is both the same: redTowerKills.

XGB tree is much more based on macro-objectives, such as tower, inhibitor, baron, total gold, minion kills, total level, jungle minion kills, object damage dealt, kills and less interested with ward placing and ward killing, killing spree or destroying any objective first

Decision tree is also based on macro-objectives but it focuses more on micros in contrast to XGB in its decisions: average levels, total healing, killing spree. Also, it has fewer breaking points in this elo.

5.2.3.2 Challenger elo games – Updated table

XGB Tree win condition strong breaking points:

assistDifference, inhibitorDifference, MinionKillsDifference, dragonKillDifference, baronKillDifference, wardKilledDifference, championDamageDealtDifference

XGB Tree win condition weak breaking points:

HealDifference, wardPlacedDifference, JungleMinionKillsDifference

Decision Tree win condition strong breaking points:

assistDifference, inhibitorDifference, championDamageDealt, dragonKillsDifference, baronKillDifference, healDifference, totalMinionKillsDifference, wardKilledDifference

Decision Tree win condition weak breaking points:

WardPlacedDifference, blueFirstBaron, JungleMinionKillsDifference

6. Analysis of results

This part consists of 3 main sections, analysis of classifiers' accuracy results, analysis of classifiers' choices on attributes and investigating dependencies on different skill rank groups and about gap and limitations on our project.

First of all, we want to mention about classifiers' accuracy results. For all original data analyses, XGBClassifier has proven itself to be a pretty good classifier and Decision Tree Classifier is somewhere near XGBClassifier but performs less accurate (max_features drops the accuracy but we kept it in that way in a purpose as you are about to read soon). Applying PCA on original data is a very bad idea while there are so many dimension which affects the win rate by having around same rate of positive and negative co-relation attributes(Having red and blue team's attributes while there is only one winner team). So, taking PCA1 in almost %50 %50 distributed data is nothing but a coin toss. So, PCA is meant for updated data sets.

Moreover, just like in original data sets, in data sets with updated columns also have better accuracy results when it comes to XGBClassifer. We were thinking that XGB model might be overfitted and Decision Tree Classifier suffers from some outliers. By applying PCA, they end up with almost the same numbers(XGB's accuracy decreases and Decision Tree's accuracy increases). From these results, we concluded that XGB Classifier is a bit overkill for fitting data while Decision Tree is a naïve version of XGBC but overall, they both need post-processing at the end. Nobody is perfect.

From investigating breaking points of tree classifiers, we saw that decision tree is more sensitive on binary attributes like firstBaron, firstDragon… This sensitivity affects classification branching as well in trees. Also, there are attributes which haven't been used at all in breaking points, such as firstBlood and firstTower which we want to mention about them later on, about game skill ranking analysis.

Secondly, this classification technique proved that in every skill group, there are prioritization of objectives. Winning or losing a game is dependent on different attributes. In master elo games, having assists on enemy kills, heal difference, champion damage dealt and jungle minion kills steps forth. So, in master skill group, team fights took long part of the game or in lane phases damage tradeoffs is much more as we can understand from assists, champion damage dealt and heal difference. And we can derive healing supports or self healing-sustaining champions are more preferable in champion picks.

When it comes the grand master games, baron and dragon objectives starting to be more important and total heal loses its interest. Also, killing wards and placing wards becomes more important. From loss of interest to healing and ward importance, we can come up with the idea of the damage tradeoffs individually and team fights are less often in grand masters. So, games are much more dependent to taking advantage from another team's mistake(losing first baron objective can be called as a vital mistake in this skill group). In addition to that, dragon and baron objectives' importance supports our claim. From these assumptions with addition of less healing importance, we also think in GM elo, the assassin champions are more preferable.

In challenger games, all of these attributes we mentioned before put their importance also in challenger games but two attributes come forward in this elo which are total minion kills and jungle minion kills. We can derive that, these games are too intense that these people are trying to make gold advantage by killing neutral enemies instead of looking mistakes of opponent team champions. In our opinion, even just this proves how these players are expert and talented in this game. Getting the first dragon and baron killings loses its importance and this shows how they are capable of stabilizing the game even if they become disadvantageous team at first.

There are two attributes which are not that important for all data sets which are first blood and first tower. They only give +100 for first blood and +150 for first tower. For such skilled players, they know how to stay cool even if they don't get first blood or tower. And if they get the first blood or tower they know they should not be over-couraged. Because +100-150 is not that a big amount of gold

Finally, in our opinion, there is only one gap and two limitations on our project. We are not sure if classifiers are overfitted (but we are focusing the performance differences and main breakpoints in the trees, so this is not directly affecting our project. It can be called as a gap) and we couldn't find any other data to put them in our model (except we mine for it, which we didn't look forward to it because of time and effort). Other limitation was the speed of computation in the double cross validation (It takes absurd amount of time). We couldn't investigate all the values we wanted to test.

7. Improving Results

We have used the same ranking's plays as test data and build desicion trees and used XGBClassifier. But also we can use other ranking's play as test data to build another ranking's desicion tree. In this way, we can get results about antother similarities or differences about the rankings.

We can use more updated and fresh data from Riot Games database which is the database for game play information and put them in a format that we can use for our models.

In addition to these, we can try more classification methods and approach instead of using only XGBClassifier and building decision trees. Again in this way, maybe we can find more proper one for our data sets and we can get rid of "over-fitting classification" as well as "not good enough classification".

And finally we can create a mechanism such that it would use regression and the results will be updated as the new data , for instance the data from Riot Games database, comes as input. By this way, we can have much more clear results since our data set grows and we can be more precise.

8. Insight of code

Notebook consists of sections:

#Quick Investigation on Data : Giving general information about rows and columns, about the data set

#Co-relation Matrix : Showing co-relation matrix of attributes

#Necessary modifications on the main data(Only in updated versions): Self-explanatory by its name with additional heat co-relation maps

#Exploratory Data Analysis : Seeing distribution of values in win condition and lose condition

#Data Preprocessing : Setting up training data + testing data and contains parameter optimizer as double cross validation

#Modal Selection and Metrics with Trees and their visualization : Fitting data sets into classifiers and seeing their results

#Dimensionality Reduction by PCA + Modal Selection and Metrics with Trees and their visualization : Applying PCA and then fitting data sets into classifiers and seeing their results