DATA Science project

Ibrahima Ndiaye

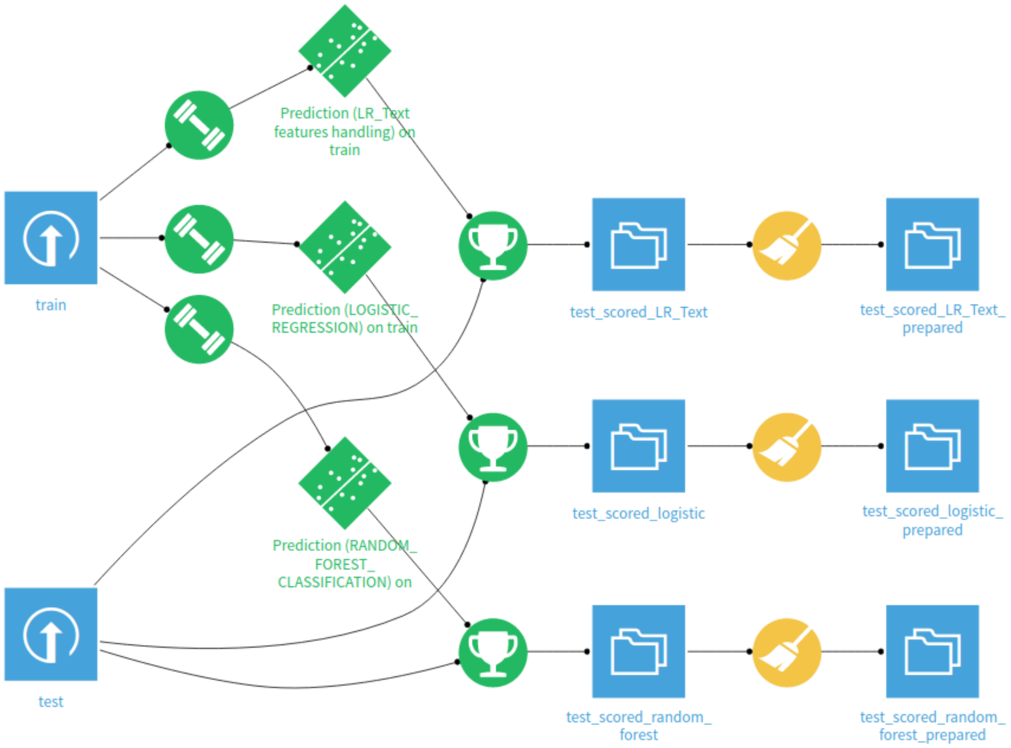
Football Data Challenge on Kaggle

Project description

The football data challenge is a kaggle competition found on their website. We are provided with data of the past 8 years of football games and their results in the Italian football league called Serie A .The objective of this project is to be able to predict the result of future football matches with the data provided.
My goal is to be able to predict the result of the games with an accuracy of at least 50%. If the goal is achieved, the project will be considered a success.

I have done 3 submissions on kaggle to be able to know what the score will be in the hopes of getting a score of at least 0.5

This is what my flow looks like at the end of my project:


Prediction (LR_Text features handling) on train
train
Prediction (LOGISTIC_ REGRESSION) on train
Prediction (RANDOM_ FOREST_ CLASSIFICATION) on
test
test_scored_LR_Text
test_scored_LR_Text_ prepared
test_scored_logistic
test_scored_logistic_ prepared
test_scored_random_ forest
test_scored_random_ forest_prepared

The dataset

The football challenge dataset is the following:

Train dataset

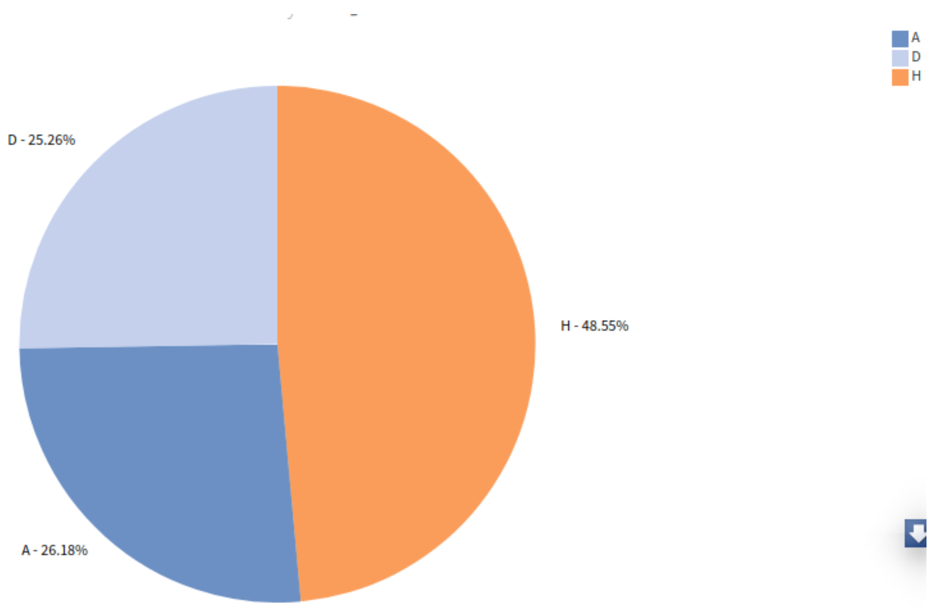
- ID = An anonymous ID unique to a given match
- Date = Date of the match
- FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win). This is the target variable

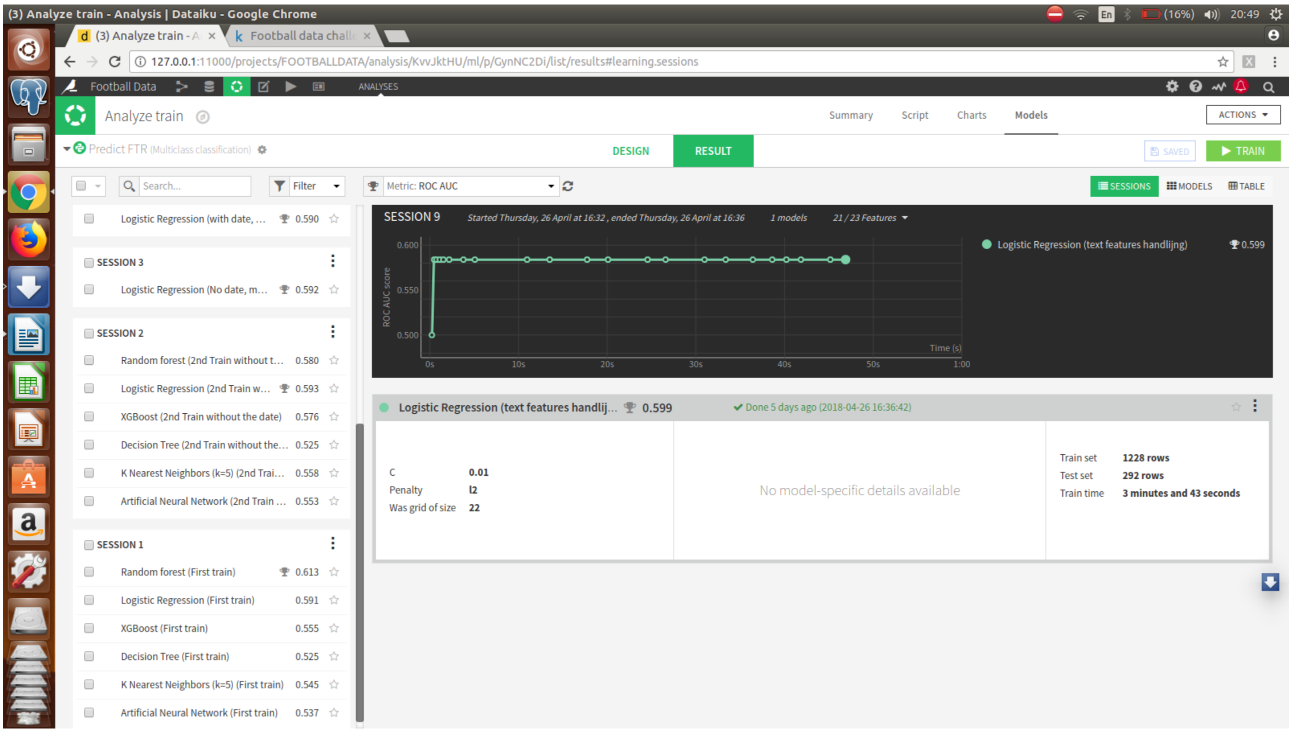
- HomeTeam = Home Team
- AwayTeam = Away Team
- B365H = Bet365 home win odds
- B365D = Bet365 draw odds
- B365A = Bet365 away win odds
- BWH = Bet&Win home win odds
- BWD = Bet&Win draw odds
- BWA = Bet&Win away win odds
- IWH = Interwetten home win odds
- IWD = Interwetten draw odds
- IWA = Interwetten away win odds
- LBH = Ladbrokes home win odds
- LBD = Ladbrokes draw odds
- LBA = Ladbrokes away win odds
- VCH = VC Bet home win odds
- VCD = VC Bet draw odds
- VCA = VC Bet away win odds
- WHH = William Hill home win odds
- WHD = William Hill draw odds
- WHA = William Hill away win odds

With all of the information provided above, our goal is to be able to predict what the full time result of future games will be on the test data set which will have all of the information, except the full time result which we will of course predict.

In order to get varying results, I have made 3 different training models which were individually submitted to Kaggle to see what the best score would be.

Before diving into the procedure of how we have completed our project, there was one important analysis to be made. If we look at the image below, we get a percentage distribution of the results of the games at full time. What is impressive is the amount of games which were won by the home team as opposed to the draws and away team wins. The home team wins about 48.55 percent of the games that they play and this could be due to any number of factors. Football experts usually put it down to the fact that they are playing on their own soil which they are used to and also the support of the home team’s fans and the atmosphere they provide. This gives the home team an advantage when playing football games. Also, the same could be said about the away team which are not used to the field of the home team, and may perhaps be intimated by the home team’s fans and the overwhelming atmosphere and ambiance created by them which thus affects their performance.





SESSION 9 Started Thursday, 26 April at 16:32 , ended Thursday, 26 April at 16:36 1 models 21 / 23 Features

Logistic Regression (text features handlijng) 0.599

Logistic Regression (with date, … 0.590

SESSION 3

Logistic Regression (No date, m… 0.592

SESSION 2

Random forest (2nd Train without t…	0.580
Logistic Regression (2nd Train w…	0.593
XGBoost (2nd Train without the date)	0.576
Decision Tree (2nd Train without the…	0.525
K Nearest Neighbors (k=5) (2nd Trai…	0.558
Artificial Neural Network (2nd Train …	0.553

SESSION 1

Random forest (First train)	0.613
Logistic Regression (First train)	0.591
XGBoost (First train)	0.555
Decision Tree (First train)	0.525
K Nearest Neighbors (k=5) (First train)	0.545
Artificial Neural Network (First train)	0.537

Logistic Regression (text features handlij… 0.599 ✓ Done 5 days ago (2018-04-26 16:36:42)

C	0.01
Penalty	l2
Was grid of size	22

No model-specific details available

Train set	1228 rows
Test set	292 rows
Train time	3 minutes and 43 seconds

We have done multiple sessions to determine which algorithms and settings would be the best to use in order to get the best prediction results. 9 sessions were made, but we will focus on the main ones which yielded the best results.

Before starting this project, we knew that logistic regression was the most suitable algorithm for this type of situation. But for curiosity purposes and to be able to see the results by myself, I have used 6 different algorithms for the first 2 sessions to better determine which algorithm would be best suited for our goals.

Algorithms

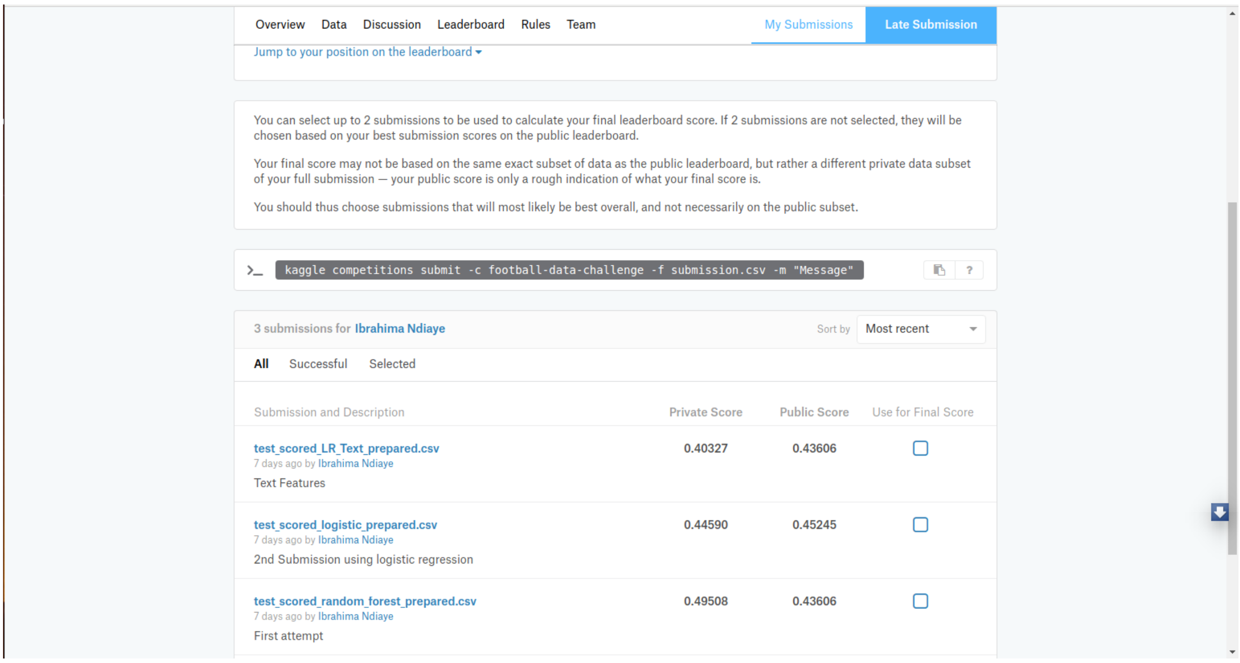
- **Random Forest**

- **Logistic Regression**
- **XGBoost**
- **Decision Tree**
- **K Nearest Neighbors**
- **Artificial Neural Network**

These different algorithms provided different results, and the 2 best algorithms were the Random Forest and the Logistic Regression algorithms.
In our first session, the Random Forest was the winner with an ROC AUC score of 0.613.
After playing around with the different features of the Logistic Regression algorithm, I wasn't able to get more than 0.599 and that is only by changing the settings in features handling to Text.
Other than that, the normal score for Logistic Regression is 0.593

Submissions to Kaggle

In this section, I will explain how I obtained the different scores in the kaggle competition.



You can select up to 2 submissions to be used to calculate your final leaderboard score. If 2 submissions are not selected, they will be chosen based on your best submission scores on the public leaderboard.

Your final score may not be based on the same exact subset of data as the public leaderboard, but rather a different private data subset of your full submission — your public score is only a rough indication of what your final score is.

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.

```
kaggle competitions submit -c football-data-challenge -f submission.csv -m "Message"
```

3 submissions for Ibrahima Ndiaye

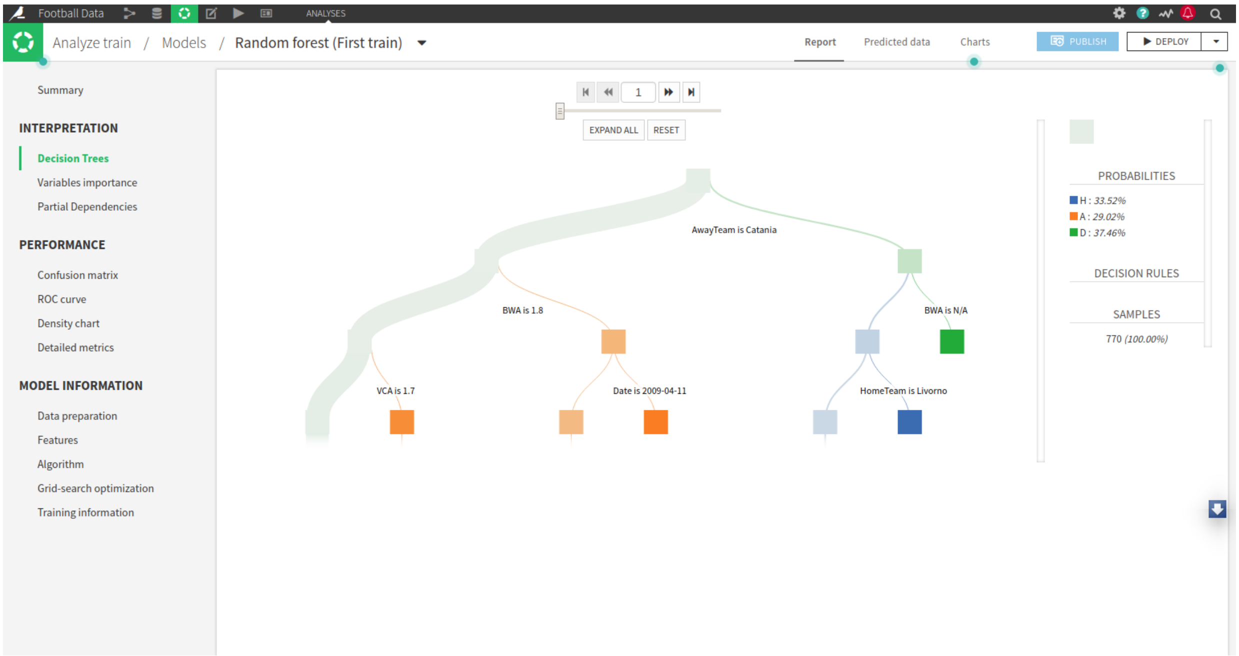
Sort by Most recent

All Successful Selected

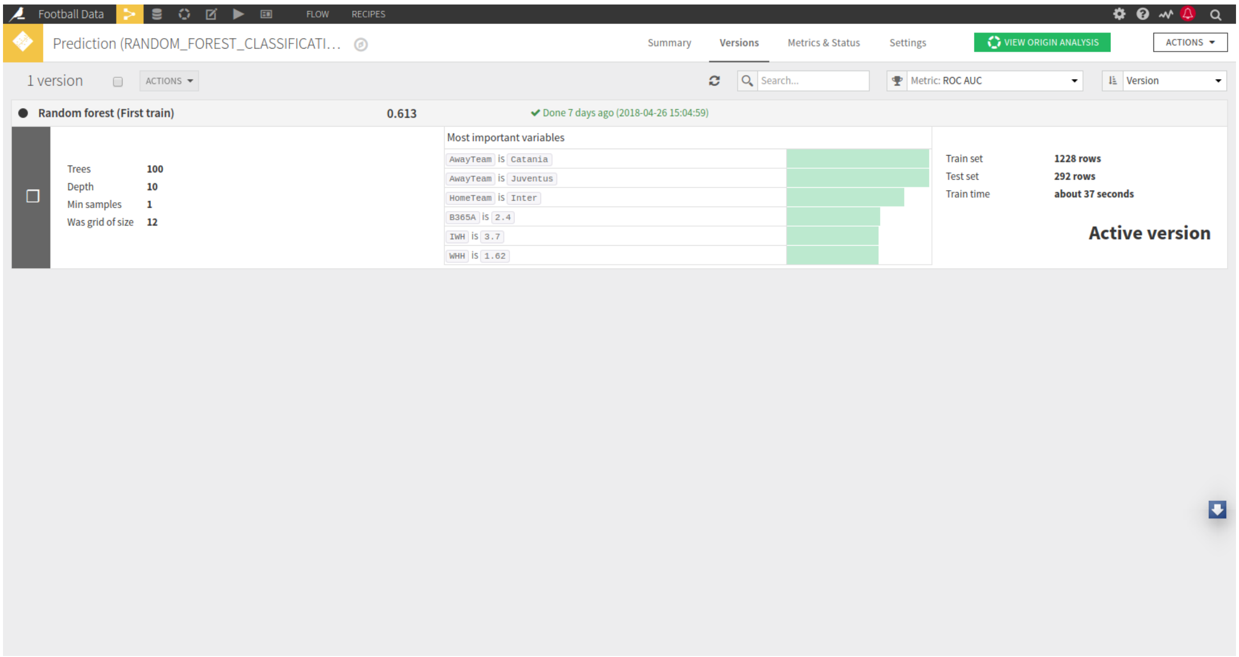
Submission and Description	Private Score	Public Score	Use for Final Score
test_scored_LR_Text_prepared.csv 7 days ago by Ibrahima Ndiaye Text Features	0.40327	0.43606	□
test_scored_logistic_prepared.csv 7 days ago by Ibrahima Ndiaye 2nd Submission using logistic regression	0.44590	0.45245	□
test_scored_random_forest_prepared.csv 7 days ago by Ibrahima Ndiaye First attempt	0.49508	0.43606	□

Submission 1 - Random Forest Algorithm

In this first submission, I have used the Random Forest Algorithm as it gave me a pretty good score when I was working with different algorithms in the model prediction section. Also I was curious to find out how it would fare once submitted to Kaggle.



I have left the default settings for the Random Forest algorithm and it gave me a ROC AUC score of 0.613 which was the highest achieved in DSS.



● Random forest (First train) 0.613 ✓ Done 7 days ago (2018-04-26 15:04:59)

Trees	100
Depth	10
Min samples	1
Was grid of size	12

Most important variables

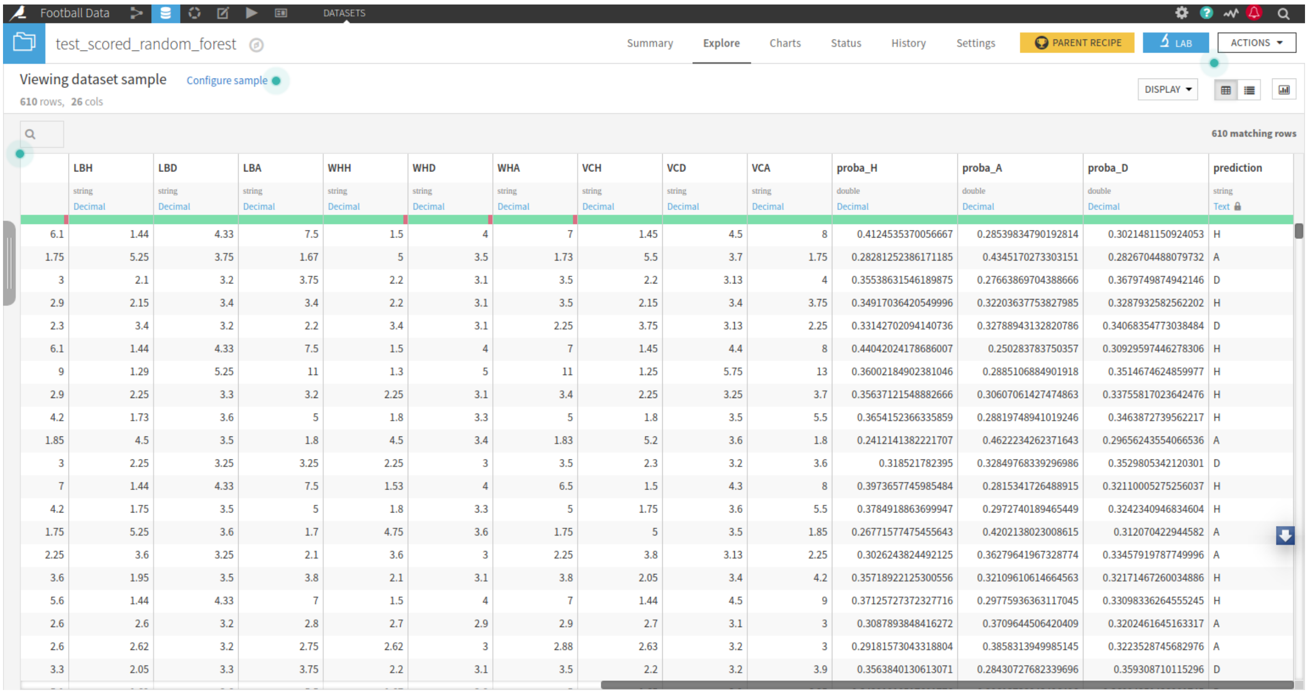
- AwayTeam is Catania
- AwayTeam is Juventus
- HomeTeam is Inter
- B365A is 2.4
- IWH is 3.7
- WHH is 1.62

Train set	1228 rows
Test set	292 rows
Train time	about 37 seconds

Active version

After scoring this last model, we got 4 extra columns :

- Probability that the home team wins – proba_H
- Probability that the away team wins – proba_A
- Probability that the score is a draw - proba_D
- The predicted result of the match – prediction



test_scored_random_forest

Viewing dataset sample Configure sample

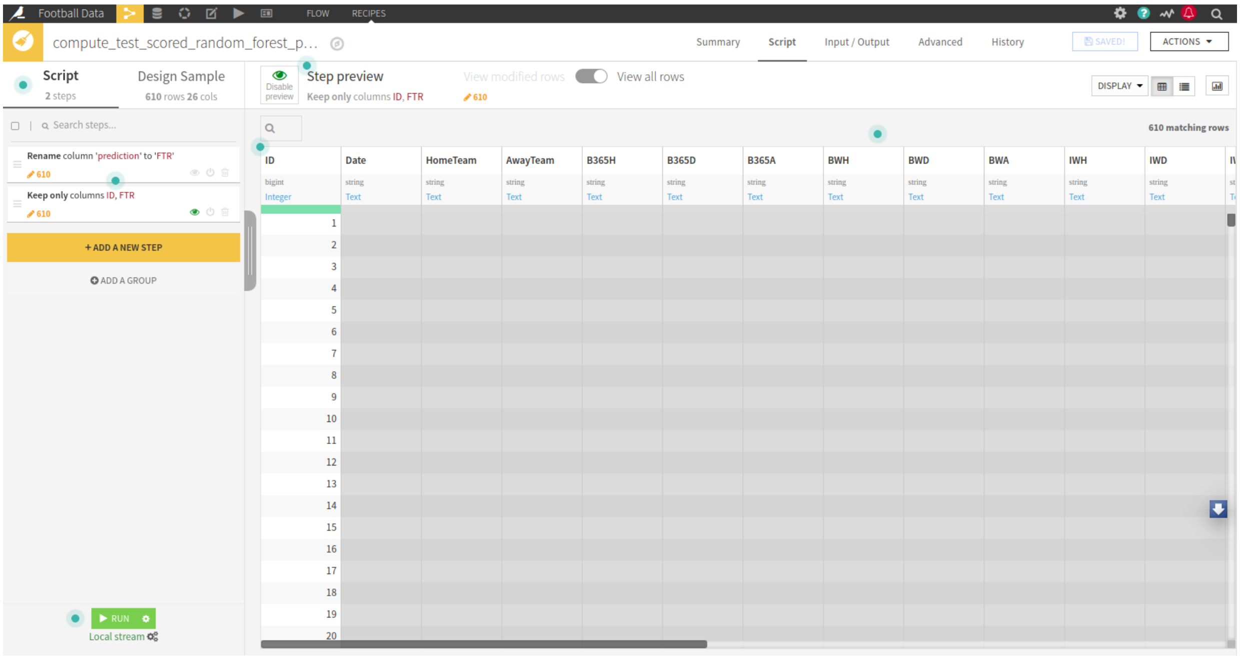
610 rows, 26 cols

610 matching rows

	LBH	LBD	LBA	WHH	WHD	WHA	VCH	VCD	VCA	proba_H	proba_A	proba_D	prediction
	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	double Decimal	double Decimal	double Decimal	string Text
6.1	1.44	4.33	7.5	1.5	4	7	1.45	4.5	8	0.4124535370056667	0.28539834790192814	0.3021481150924053	H
1.75	5.25	3.75	1.67	5	3.5	1.73	5.5	3.7	1.75	0.28281252386171185	0.4345170273303151	0.2826704488079732	A
3	2.1	3.2	3.75	2.2	3.1	3.5	2.2	3.13	4	0.35538631546189875	0.27663869704388666	0.3679749874942146	D
2.9	2.15	3.4	3.4	2.2	3.1	3.5	2.15	3.4	3.75	0.34917036420549996	0.32203637753827985	0.3287932582562202	H
2.3	3.4	3.2	2.2	3.4	3.1	2.25	3.75	3.13	2.25	0.33142702094140736	0.32788943132820786	0.34068354773038484	D
6.1	1.44	4.33	7.5	1.5	4	7	1.45	4.4	8	0.44042024178686007	0.250283783750357	0.30929597446278306	H
9	1.29	5.25	11	1.3	5	11	1.25	5.75	13	0.36002184902381046	0.2885106884901918	0.3514674624859977	H
2.9	2.25	3.3	3.2	2.25	3.1	3.4	2.25	3.25	3.7	0.35637121548882666	0.30607061427474863	0.33755817023642476	H
4.2	1.73	3.6	5	1.8	3.3	5	1.8	3.5	5.5	0.3654152366335859	0.28819748941019246	0.346387273956217	H
1.85	4.5	3.5	1.8	4.5	3.4	1.83	5.2	3.6	1.8	0.2412141382221707	0.4622234262371643	0.29656243554066536	A
3	2.25	3.25	3.25	2.25	3	3.5	2.3	3.2	3.6	0.318521782395	0.32849768339296986	0.3529805342120301	D
7	1.44	4.33	7.5	1.53	4	6.5	1.5	4.3	8	0.3973657745985484	0.2815341726488915	0.32110005275256037	H
4.2	1.75	3.5	5	1.8	3.3	5	1.75	3.6	5.5	0.3784918863699947	0.2972740189465449	0.3242340946834604	H
1.75	5.25	3.6	1.7	4.75	3.6	1.75	5	3.5	1.85	0.26771577475455643	0.4202138023008615	0.312070422944582	A
2.25	3.6	3.25	2.1	3.6	3	2.25	3.8	3.13	2.25	0.3026243824492125	0.36279641967328774	0.33457919787749996	A
3.6	1.95	3.5	3.8	2.1	3.1	3.8	2.05	3.4	4.2	0.35718922125300556	0.32109610614664563	0.32171467260034886	H
5.6	1.44	4.33	7	1.5	4	7	1.44	4.5	9	0.37125727372327716	0.29775936363117045	0.33098336264555245	H
2.6	2.6	3.2	2.8	2.7	2.9	2.9	2.7	3.1	3	0.3087893848416272	0.3709644506420409	0.3202461645163317	A
2.6	2.62	3.2	2.75	2.62	3	2.88	2.63	3.2	3	0.29181573043318804	0.3858313949985145	0.3223528745682976	A
3.3	2.05	3.3	3.75	2.2	3.1	3.5	2.2	3.2	3.9	0.3563840130613071	0.28430727682339696	0.359308710115296	D

In order to submit our results to Kaggle, we had to follow the format of having only 2 columns ; the match ID and the FTR which is the predicted result of the match. It was thus necessary to prepare the dataset to conform to this format.

All we had to do was to keep the “ID” column and the “prediction” column, and then proceed by renaming the “prediction” column to “FTR” :



compute_test_scored_random_forest_p…

Script
2 steps

Design Sample
610 rows 26 cols

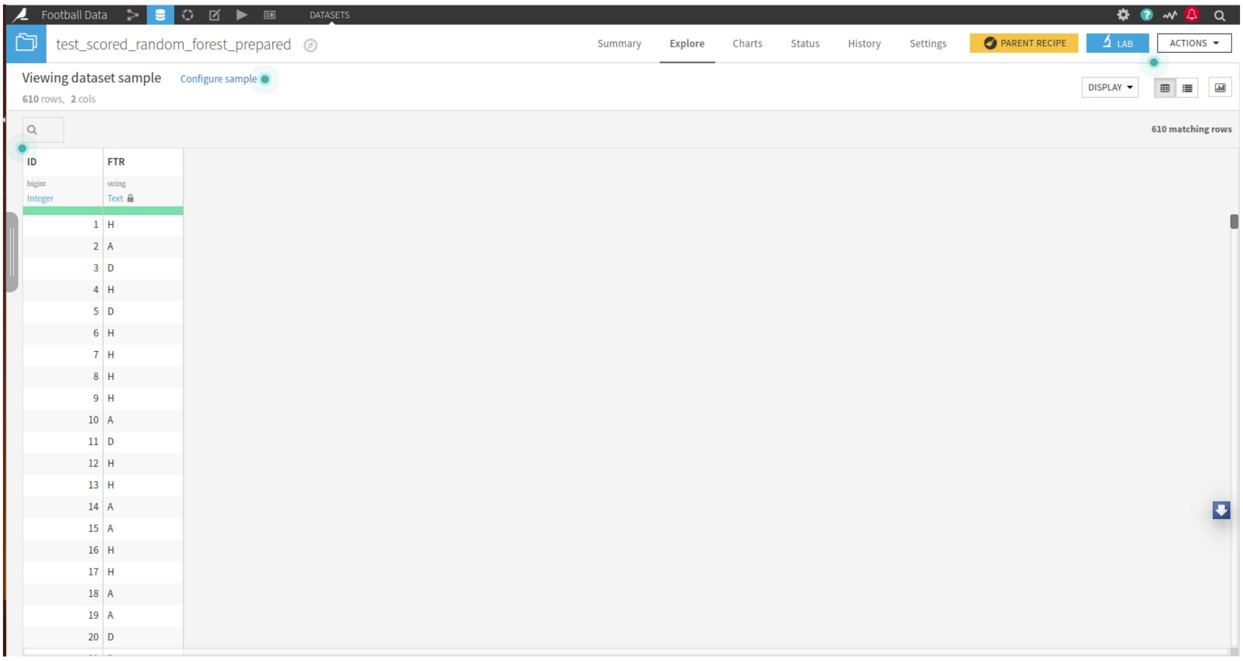
Step preview
Keep only columns ID, FTR
610

Rename column 'prediction' to 'FTR'
610

Keep only columns ID, FTR
610

610 matching rows

ID	Date	HomeTeam	AwayTeam	B365H	B365D	B365A	BWH	BWD	BWA	IWH	IWD
bigint Integer	string Text	string Text	string Text	string Text	string Text	string Text	string Text	string Text	string Text	string Text	string Text
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											



test_scored_random_forest_prepared

Viewing dataset sample

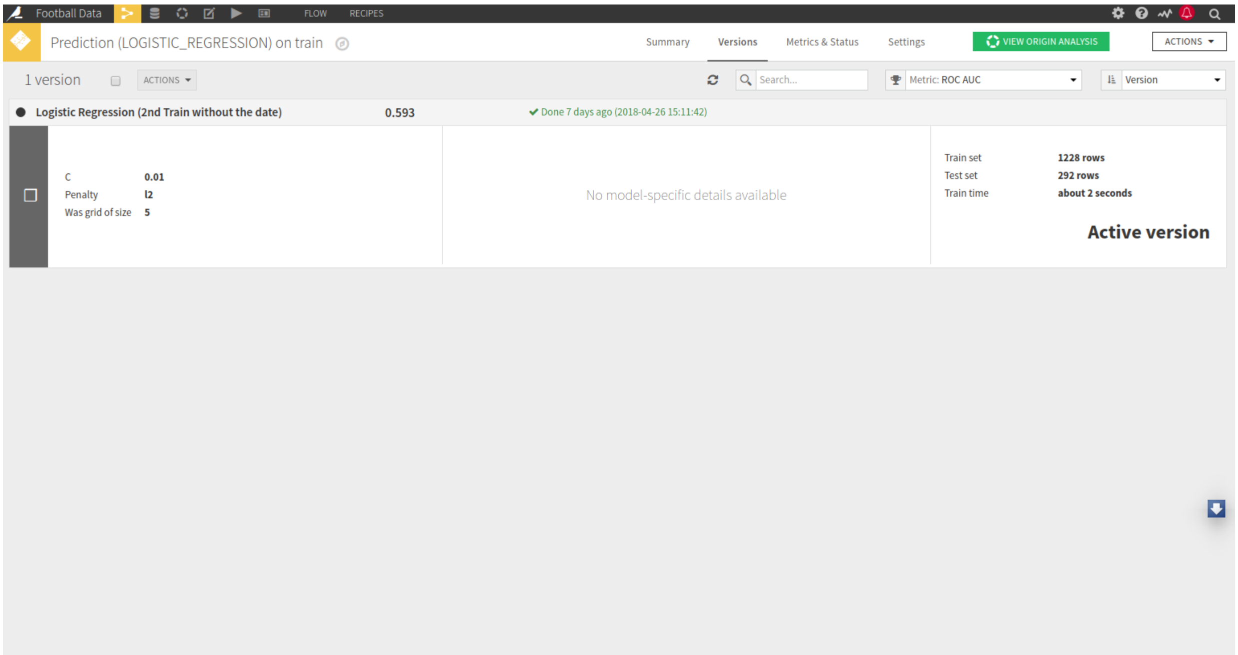
610 rows, 2 cols

610 matching rows

ID bigint Integer	FTR string Text
1	H
2	A
3	D
4	H
5	D
6	H
7	H
8	H
9	H
10	A
11	D
12	H
13	H
14	A
15	A
16	H
17	H
18	A
19	A
20	D

This final format was then submitted to Kaggle and the scores obtained were a private score of 0.49508 and a public score 0.43606.

Submission 2 – Logistic Regression Algorithm



Prediction (LOGISTIC_REGRESSION) on train

1 version

● Logistic Regression (2nd Train without the date) 0.593 ✓ Done 7 days ago (2018-04-26 15:11:42)

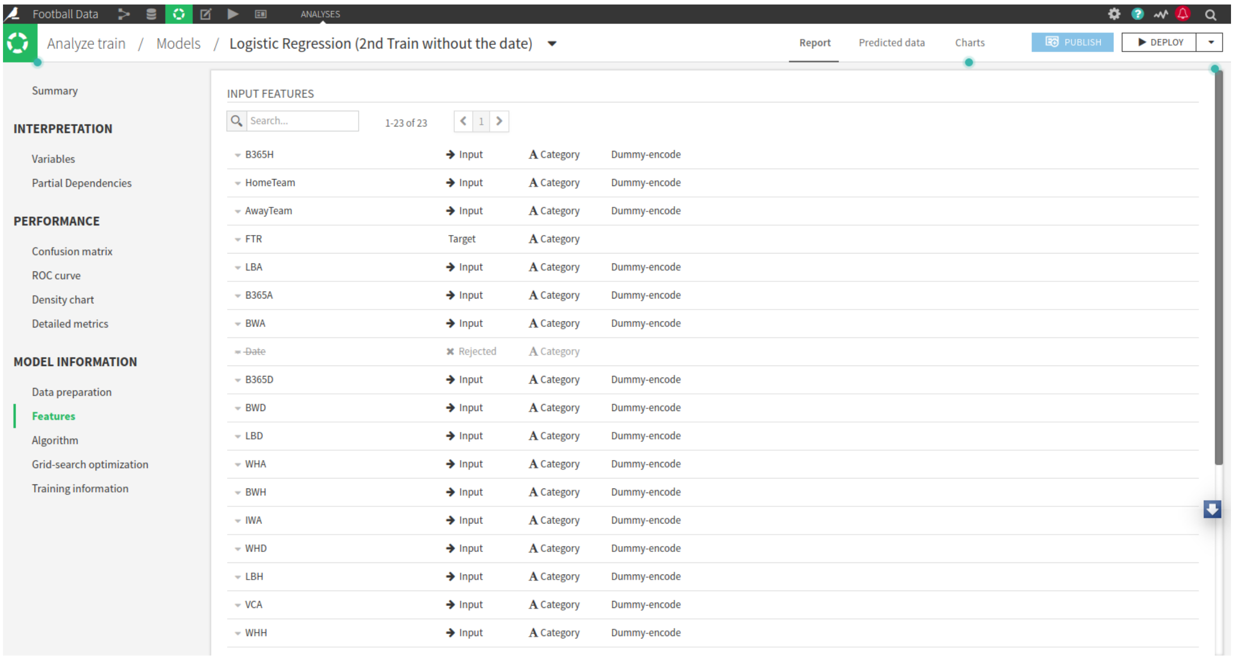
C	0.01
Penalty	l2
Was grid of size	5

No model-specific details available

Train set	1228 rows
Test set	292 rows
Train time	about 2 seconds

Active version

The second submission was with a Logistic Regression algorithm that yielded at ROC AUC score of 0.593, which was decent but not as good as the previous one achieved with the Random forest.

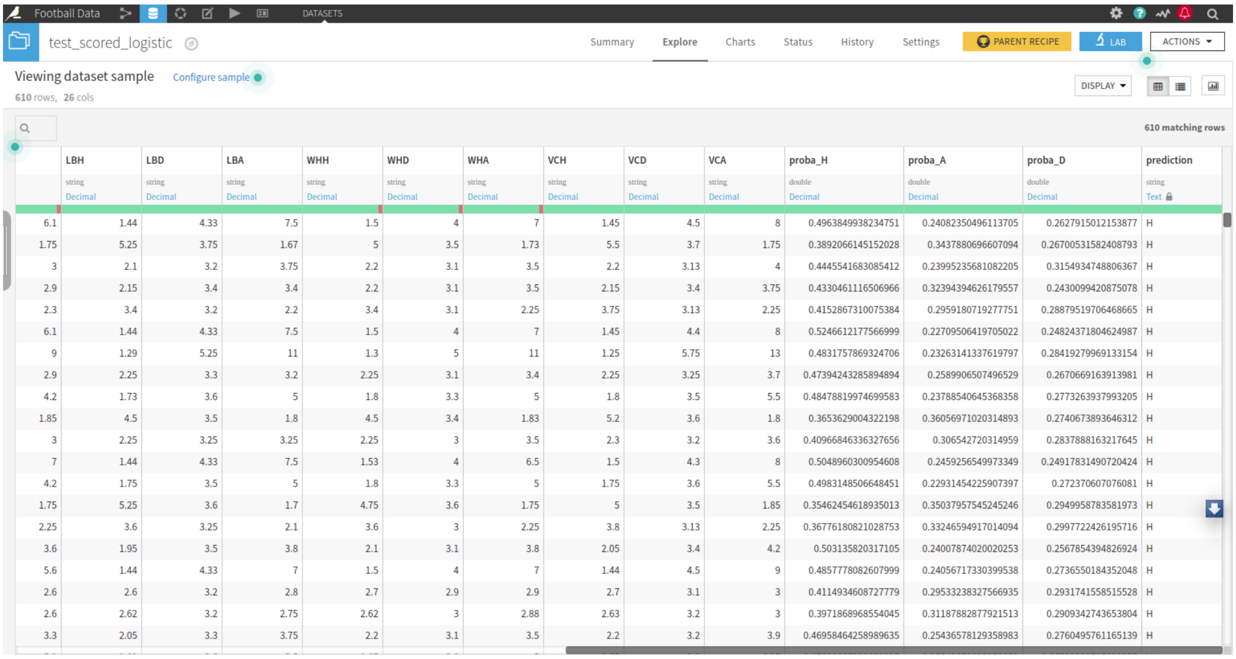


INPUT FEATURES

1-23 of 23

B365H	→ Input	A Category	Dummy-encode
HomeTeam	→ Input	A Category	Dummy-encode
AwayTeam	→ Input	A Category	Dummy-encode
FTR	Target	A Category	
LBA	→ Input	A Category	Dummy-encode
B365A	→ Input	A Category	Dummy-encode
BWA	→ Input	A Category	Dummy-encode
~~Date~~	✖ Rejected	A Category	
B365D	→ Input	A Category	Dummy-encode
BWD	→ Input	A Category	Dummy-encode
LBD	→ Input	A Category	Dummy-encode
WHA	→ Input	A Category	Dummy-encode
BWH	→ Input	A Category	Dummy-encode
IWA	→ Input	A Category	Dummy-encode
WHD	→ Input	A Category	Dummy-encode
LBH	→ Input	A Category	Dummy-encode
VCA	→ Input	A Category	Dummy-encode
WHH	→ Input	A Category	Dummy-encode

In the features handling we could see that the defaut category variable type was kept, as we wanted to first see what kind of result we would be getting by just keeping all of the default settings.



Viewing dataset sample

610 rows, 26 cols

610 matching rows

	LBH	LBD	LBA	WHH	WHD	WHA	VCH	VCD	VCA	proba_H	proba_A	proba_D	prediction
	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	string Decimal	double Decimal	double Decimal	double Decimal	string Text
6.1	1.44	4.33	7.5	1.5	4	7	1.45	4.5	8	0.4963849938234751	0.24082350496113705	0.2627915012153877	H
1.75	5.25	3.75	1.67	5	3.5	1.73	5.5	3.7	1.75	0.3892066145152028	0.3437880696607094	0.26700531582408793	H
3	2.1	3.2	3.75	2.2	3.1	3.5	2.2	3.13	4	0.4445541683085412	0.23995235681082205	0.3154934748806367	H
2.9	2.15	3.4	3.4	2.2	3.1	3.5	2.15	3.4	3.75	0.4330461116506966	0.32394394626179557	0.2430099420875078	H
2.3	3.4	3.2	2.2	3.4	3.1	2.25	3.75	3.13	2.25	0.4152867310075384	0.2959180719277751	0.28879519706468665	H
6.1	1.44	4.33	7.5	1.5	4	7	1.45	4.4	8	0.5246612177566999	0.22709506419705022	0.24824371804624987	H
9	1.29	5.25	11	1.3	5	11	1.25	5.75	13	0.4831757869324706	0.23263141337619797	0.28419279969133154	H
2.9	2.25	3.3	3.2	2.25	3.1	3.4	2.25	3.25	3.7	0.47394243285894894	0.2589906507496529	0.2670669163913981	H
4.2	1.73	3.6	5	1.8	3.3	5	1.8	3.5	5.5	0.48478819974699583	0.23788540645368358	0.2773263937993205	H
1.85	4.5	3.5	1.8	4.5	3.4	1.83	5.2	3.6	1.8	0.3653629004322198	0.36056971020314893	0.2740673893646312	H
3	2.25	3.25	3.25	2.25	3	3.5	2.3	3.2	3.6	0.40966846336327656	0.306542720314959	0.2837888163217645	H
7	1.44	4.33	7.5	1.53	4	6.5	1.5	4.3	8	0.5048960300954608	0.2459256549973349	0.24917831490720424	H
4.2	1.75	3.5	5	1.8	3.3	5	1.75	3.6	5.5	0.4983148506648451	0.22931454225907397	0.272370607076081	H
1.75	5.25	3.6	1.7	4.75	3.6	1.75	5	3.5	1.85	0.35462454618935013	0.35037957545245246	0.2949958783581973	H
2.25	3.6	3.25	2.1	3.6	3	2.25	3.8	3.13	2.25	0.36776180821028753	0.33246594917014094	0.2997722426195716	H
3.6	1.95	3.5	3.8	2.1	3.1	3.8	2.05	3.4	4.2	0.503135820317105	0.24007874020020253	0.2567854394826924	H
5.6	1.44	4.33	7	1.5	4	7	1.44	4.5	9	0.4857778082607999	0.24056717330399538	0.2736550184352048	H
2.6	2.6	3.2	2.8	2.7	2.9	2.9	2.7	3.1	3	0.4114934608727779	0.29533238327566935	0.2931741558515528	H
2.6	2.62	3.2	2.75	2.62	3	2.88	2.63	3.2	3	0.3971868968554045	0.31187882877921513	0.2909342743653804	H
3.3	2.05	3.3	3.75	2.2	3.1	3.5	2.2	3.2	3.9	0.46958464258989635	0.25436578129358983	0.2760495761165139	H

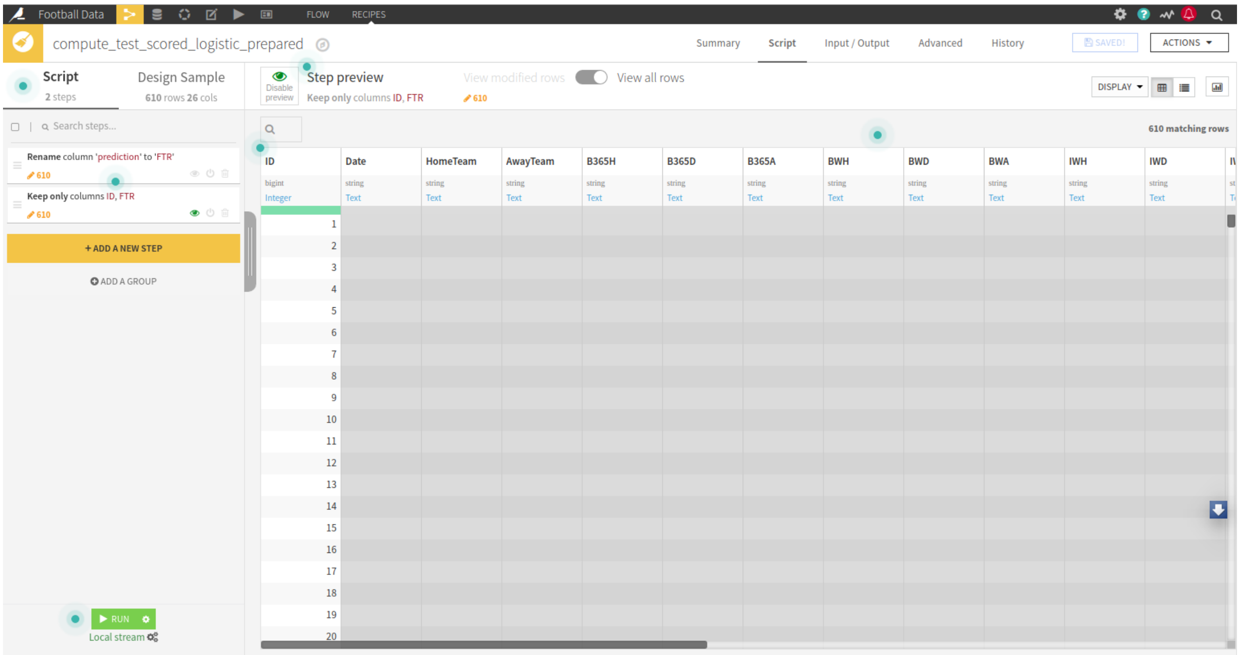
After scoring this second model, we got 4 extra columns :

- Probability that the home team wins – proba_H
- Probability that the away team wins – proba_A
- Probability that the score is a draw - proba_D

- The predicted result of the match – prediction

In order to submit our results to Kaggle, we had to follow the format of having only 2 columns ; the match ID and the FTR which is the predicted result of the match. It was thus necessary to prepare the dataset to conform to this format.

All we had to do was to keep the “ID” column and the “prediction” column, and then proceed by renaming the “prediction” column to “FTR” :



compute_test_scored_logistic_prepared

Script
2 steps

Design Sample
610 rows 26 cols

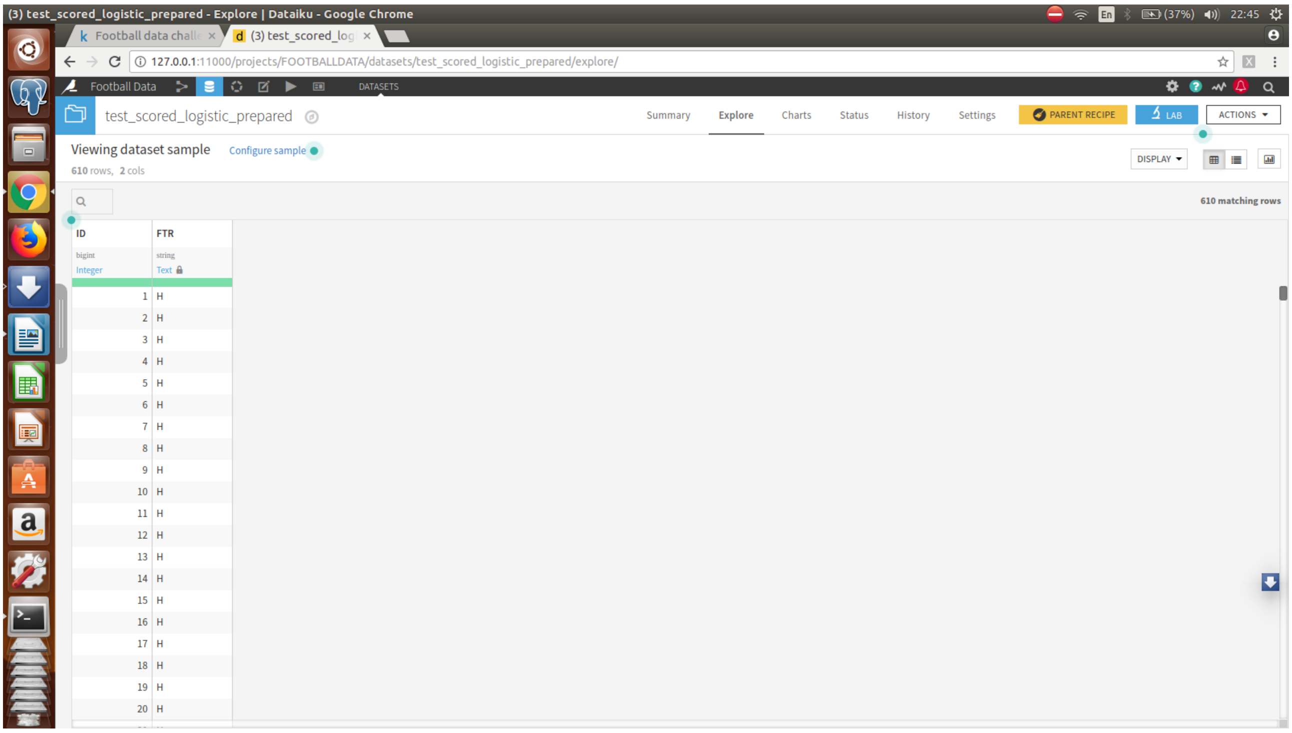
Step preview
Keep only columns ID, FTR
610

Rename column 'prediction' to 'FTR'
610

Keep only columns ID, FTR
610

610 matching rows

ID	Date	HomeTeam	AwayTeam	B365H	B365D	B365A	BWH	BWD	BWA	IWH	IWD
bigint	string	string	string	string	string	string	string	string	string	string	string
Integer	Text	Text	Text	Text	Text	Text	Text	Text	Text	Text	Text
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											



test_scored_logistic_prepared

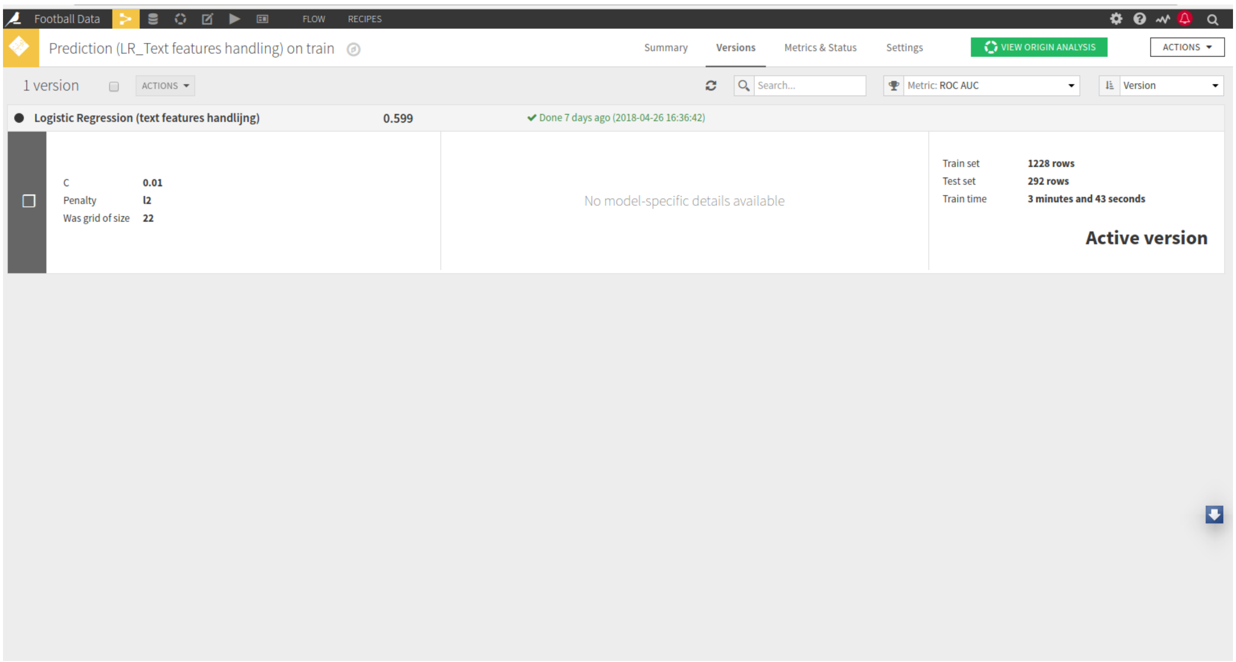
Viewing dataset sample
610 rows, 2 cols

610 matching rows

ID	FTR
bigint	string
Integer	Text
1	H
2	H
3	H
4	H
5	H
6	H
7	H
8	H
9	H
10	H
11	H
12	H
13	H
14	H
15	H
16	H
17	H
18	H
19	H
20	H

We now have the correct format for submission to kaggle. This second submission achieved a private score of 0.44590 and a public score of 0.45245. In my opinion this was a pretty balanced score and most probably reflected the most correct score out of all the submissions.

Submission 3 - Logistic Regression Algorithm with Text Variable Type



Prediction (LR_Text features handling) on train

Logistic Regression (text features handlijng) 0.599 ✓ Done 7 days ago (2018-04-26 16:36:42)

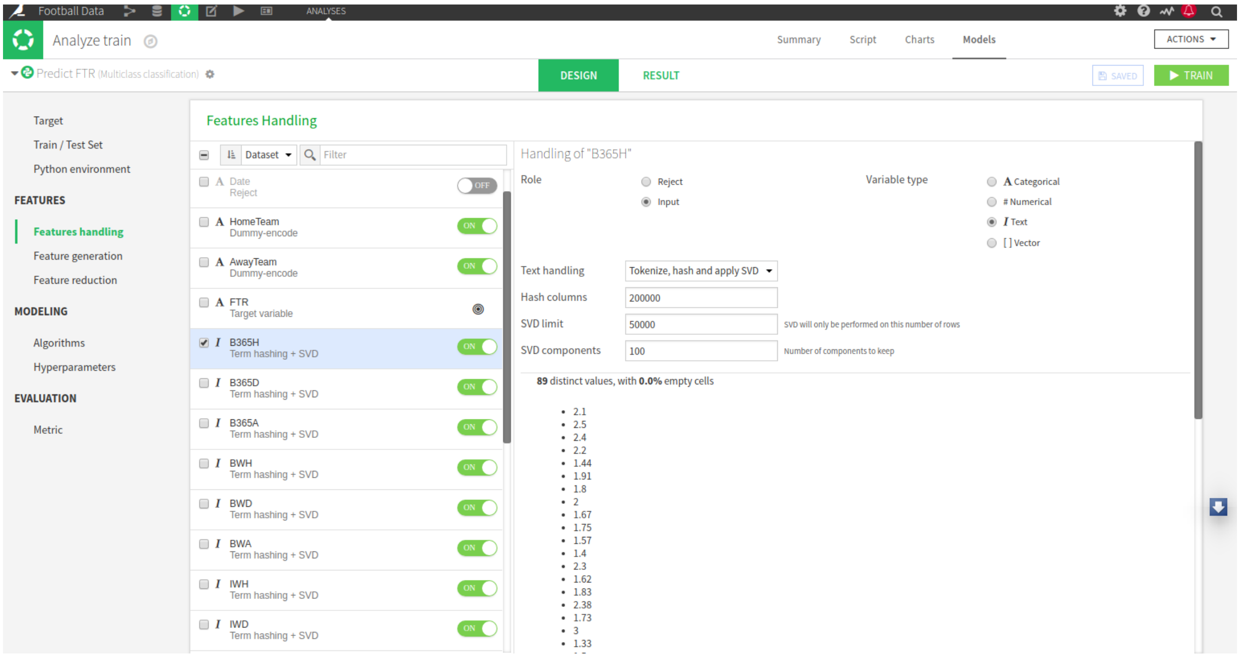
C	0.01
Penalty	l2
Was grid of size	22

No model-specific details available

Train set	1228 rows
Test set	292 rows
Train time	3 minutes and 43 seconds

Active version

The ROC AUC score that we have obtained using the Logisctic Regression algorithm while using the Text Variable Type in the features handling section was 0.599 as we can see above.



Features Handling

	Feature	Handling	
□	A Date	Reject	OFF
□	A HomeTeam	Dummy-encode	ON
□	A AwayTeam	Dummy-encode	ON
□	A FTR	Target variable	◉
☑	I B365H	Term hashing + SVD	ON
□	I B365D	Term hashing + SVD	ON
□	I B365A	Term hashing + SVD	ON
□	I BWH	Term hashing + SVD	ON
□	I BWD	Term hashing + SVD	ON
□	I BWA	Term hashing + SVD	ON
□	I IWH	Term hashing + SVD	ON
□	I IWD	Term hashing + SVD	ON

Handling of "B365H"

Role: ○ Reject ● Input

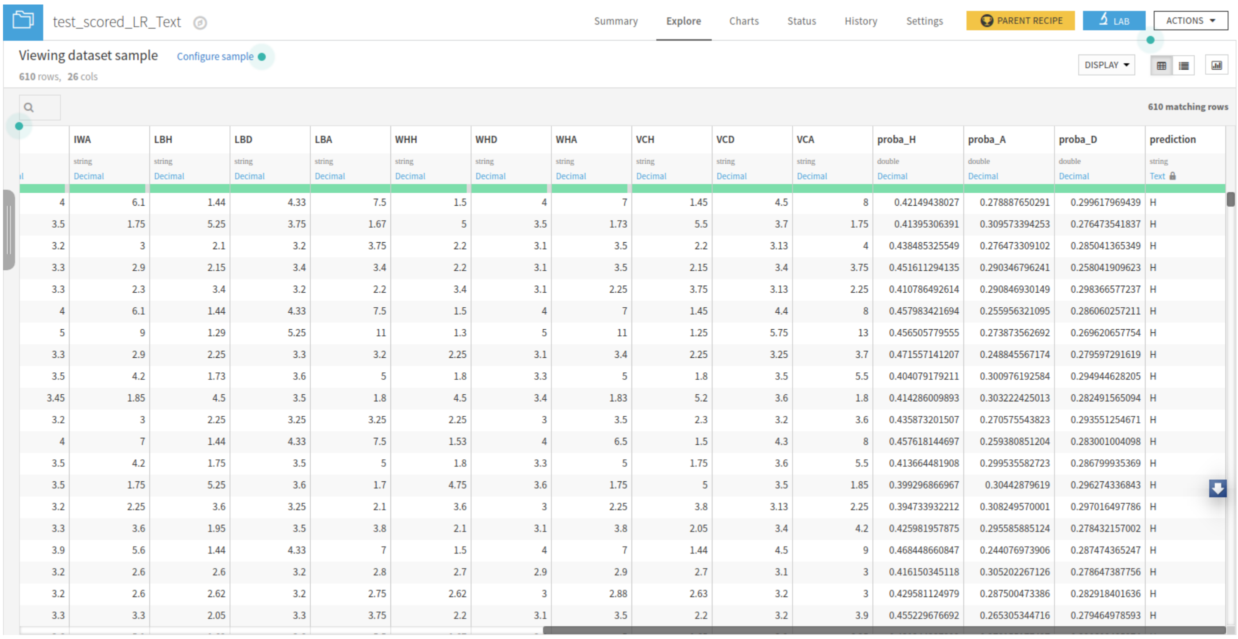
Variable type: ○ A Categorical ○ # Numerical ● I Text ○ [ ] Vector

Text handling	Tokenize, hash and apply SVD	
Hash columns	200000	
SVD limit	50000	SVD will only be performed on this number of rows
SVD components	100	Number of components to keep

89 distinct values, with 0.0% empty cells

- 2.1
- 2.5
- 2.4
- 2.2
- 1.44
- 1.91
- 1.8
- 2
- 1.67
- 1.75
- 1.57
- 1.4
- 2.3
- 1.62
- 1.83
- 2.38
- 1.73
- 3
- 1.33

This last submission was made using the Text Variable Type instead of Categorical Variable type which was used by default.



Viewing dataset sample Configure sample

610 rows, 26 cols

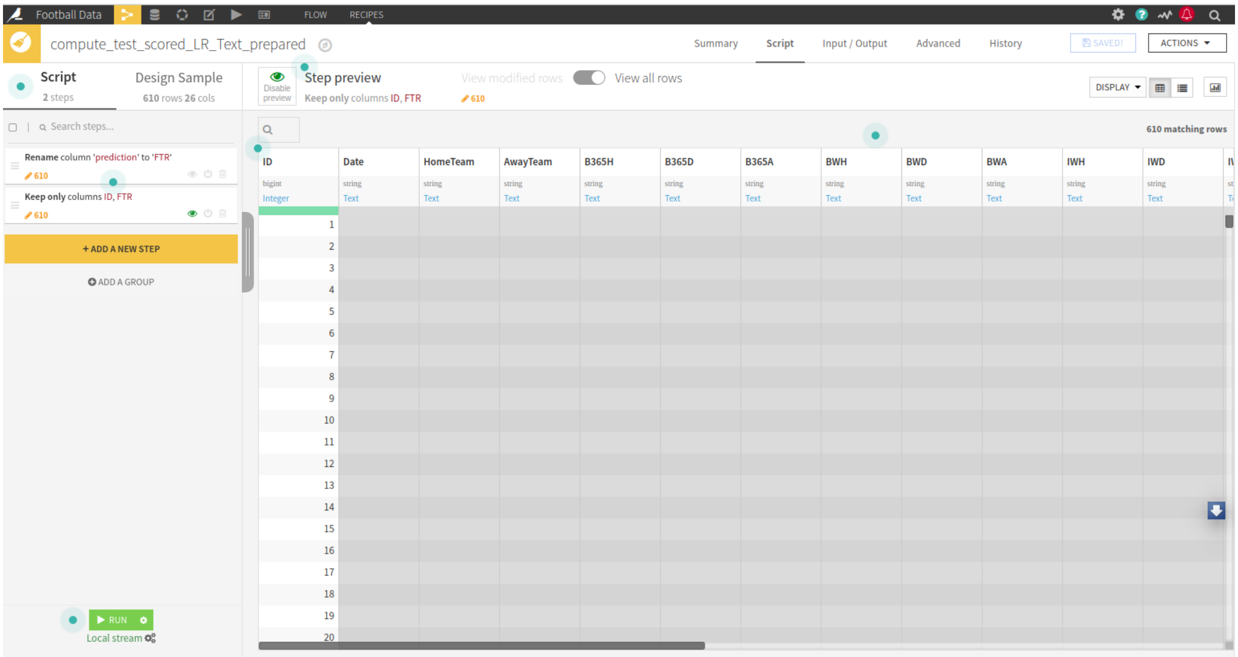
610 matching rows

	IWA	LBH	LBD	LBA	WHH	WHD	WHA	VCH	VCD	VCA	proba_H	proba_A	proba_D	prediction
	string	string	string	string	string	string	string	string	string	string	double	double	double	string
	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal	Text
4	6.1	1.44	4.33	7.5	1.5	4	7	1.45	4.5	8	0.42149438027	0.278887650291	0.299617969439	H
3.5	1.75	5.25	3.75	1.67	5	3.5	1.73	5.5	3.7	1.75	0.41395306391	0.309573394253	0.276473541837	H
3.2	3	2.1	3.2	3.75	2.2	3.1	3.5	2.2	3.13	4	0.438485325549	0.276473309102	0.285041365349	H
3.3	2.9	2.15	3.4	3.4	2.2	3.1	3.5	2.15	3.4	3.75	0.451611294135	0.290346796241	0.258041909623	H
3.3	2.3	3.4	3.2	2.2	3.4	3.1	2.25	3.75	3.13	2.25	0.410786492614	0.290846930149	0.298366577237	H
4	6.1	1.44	4.33	7.5	1.5	4	7	1.45	4.4	8	0.457983421694	0.255956321095	0.286060257211	H
5	9	1.29	5.25	11	1.3	5	11	1.25	5.75	13	0.456505779555	0.273873562692	0.269620657754	H
3.3	2.9	2.25	3.3	3.2	2.25	3.1	3.4	2.25	3.25	3.7	0.471557141207	0.248845567174	0.279597291619	H
3.5	4.2	1.73	3.6	5	1.8	3.3	5	1.8	3.5	5.5	0.404079179211	0.300976192584	0.294944628205	H
3.45	1.85	4.5	3.5	1.8	4.5	3.4	1.83	5.2	3.6	1.8	0.414286009893	0.303222425013	0.282491565094	H
3.2	3	2.25	3.25	3.25	2.25	3	3.5	2.3	3.2	3.6	0.435873201507	0.270575543823	0.293551254671	H
4	7	1.44	4.33	7.5	1.53	4	6.5	1.5	4.3	8	0.457618144697	0.259380851204	0.283001004098	H
3.5	4.2	1.75	3.5	5	1.8	3.3	5	1.75	3.6	5.5	0.413664481908	0.299535582723	0.286799935369	H
3.5	1.75	5.25	3.6	1.7	4.75	3.6	1.75	5	3.5	1.85	0.399296866967	0.30442879619	0.296274336843	H
3.2	2.25	3.6	3.25	2.1	3.6	3	2.25	3.8	3.13	2.25	0.394733932212	0.308249570001	0.297016497786	H
3.3	3.6	1.95	3.5	3.8	2.1	3.1	3.8	2.05	3.4	4.2	0.425981957875	0.295585885124	0.278432157002	H
3.9	5.6	1.44	4.33	7	1.5	4	7	1.44	4.5	9	0.468448660847	0.244076973906	0.287474365247	H
3.2	2.6	2.6	3.2	2.8	2.7	2.9	2.9	2.7	3.1	3	0.416150345118	0.305202267126	0.278647387756	H
3.2	2.6	2.62	3.2	2.75	2.62	3	2.88	2.63	3.2	3	0.429581124979	0.287500473386	0.282918401636	H
3.3	3.3	2.05	3.3	3.75	2.2	3.1	3.5	2.2	3.2	3.9	0.455229676692	0.265305344716	0.279464978593	H

After scoring this last model, we got 4 extra columns
- Probability that the home team wins – proba_H
- Probability that the away team wins – proba_A
- Probability that the score is a draw - proba_D
- The predicted result of the match – prediction

In order to submit our results to Kaggle, we had to follow the format of having only 2 columns ; the match ID and the FTR which is the predicted result of the match. It was thus necessary to prepare the dataset to conform to this format.

All we had to do was to keep the “ID” column and the “prediction” column, and then proceed by renaming the “prediction” column to “FTR” :



Step preview

Keep only columns ID, FTR 610

Rename column 'prediction' to 'FTR'

610

Keep only columns ID, FTR

610

610 matching rows

ID	Date	HomeTeam	AwayTeam	B365H	B365D	B365A	BWH	BWD	BWA	IWH	IWD
bigint	string	string	string	string	string	string	string	string	string	string	string
Integer	Text	Text	Text	Text	Text	Text	Text	Text	Text	Text	Text
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											

My private score was 0.40327 and my public score was 0.43606 which was the worst score I have achieved out of all 3 submissions.

CONCLUSION

This Football data challenge has helped me put into practice the different tools and knowledge that we have acquired about Data Science and Dataiku DSS during our course. It helped me analyze the results and to be able to tell which algorithm was the best to use and for what reasons. Unfortunately, I was not able to add addition features to further enhance and enrich my data as I had planned to do in the beginning of the project. This was due to my inability to find the appropriate data to work on. I wish to work in future projects where I will be able to add features on my own, in order to achieve better results, and more importantly to achieve the objectives which have been set. Even though the final score is not too far from the targeted score (~ 0.45 and 0.5 respectively), I would have been more satisfied to attain my initial goals.