PLAYERUNKNOWN'S BATTLEGROUNDS

FINISH PLACEMENT PREDICTION

Team 7 - Mengchen Zheng, Keshuo Liu,
Tingting Cao & Niming Wang

Why PUBG?

- Profitable

  > $6 Billion

- Large user base

  > 2 billion users

Why predict players' finish placement?

- Declining user base due to poor player level matching
- Placement prediction based on players' historical performance can be used for players matching.

Objectives

- How to accurately predict the finish placement in PUBG?
- Which model gives the best result of machine learning, and what's its performance on test data?

20283 Rows, 29 variables

Sampled SOLO data from the original Kaggle PUBG Finish Placement Prediction dataset

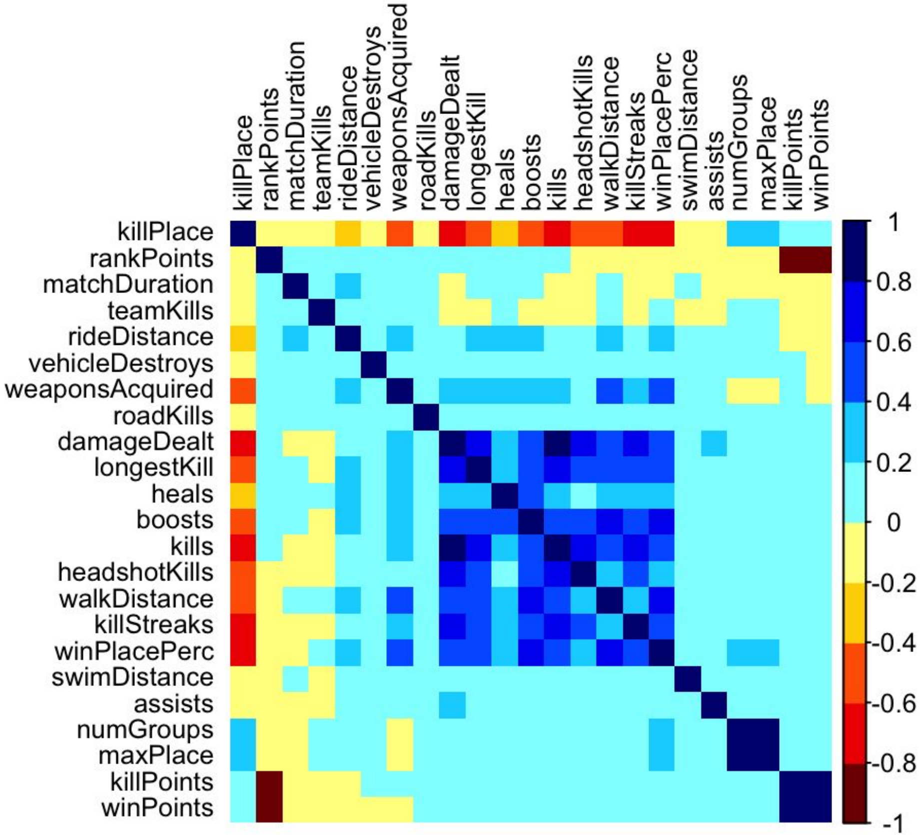
Data cleaning:

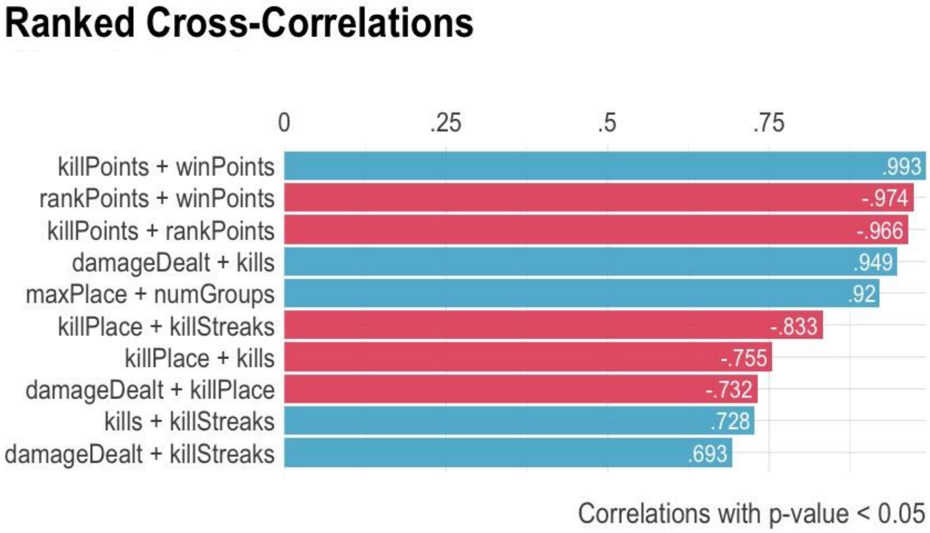
1. No Missing values
2. Remove features representing ID information: Id, groupId, matchId
3. Remove Variables with zero standard Deviation : assists, DBNOs, matchType, revives

features:

"boosts"	"damageDealt"	"headshotKills"
"heals"	"killPlace"	"killPoints"
"kills"	"killStreaks"	"longestKill"
"matchDuration"	"maxPlace"	"numGroups"
"rankPoints"	"rideDistance"	"roadKills"
"swimDistance"	"teamKills"	"vehicleDestroys"
"walkDistance"	"weaponsAcquired"	"winPoints"

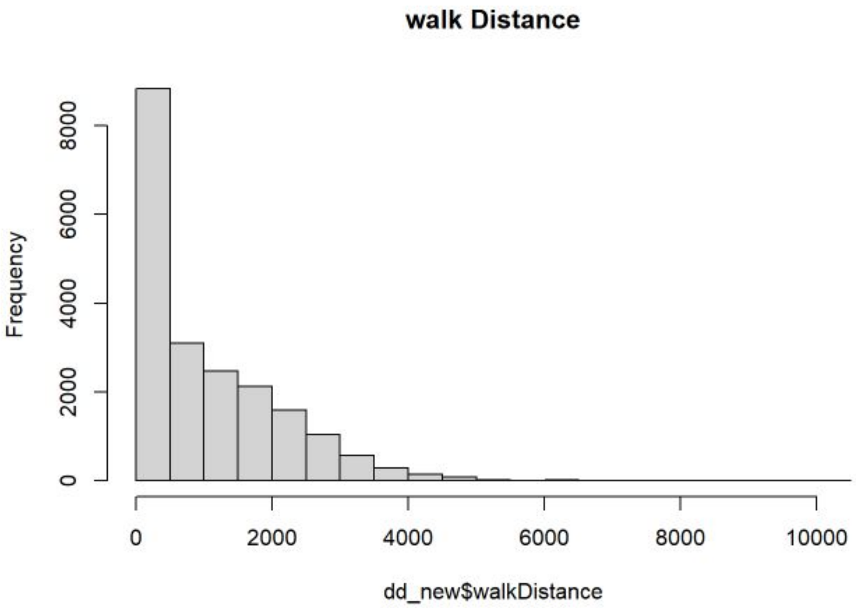
Y "winPlacePerc"





Feature Engineering Based On Multicollinearity and Domain Knowledge

Data Transformation



Data Transformation:

Log (Walk Distance+1)

Log (Ride Distance+1)

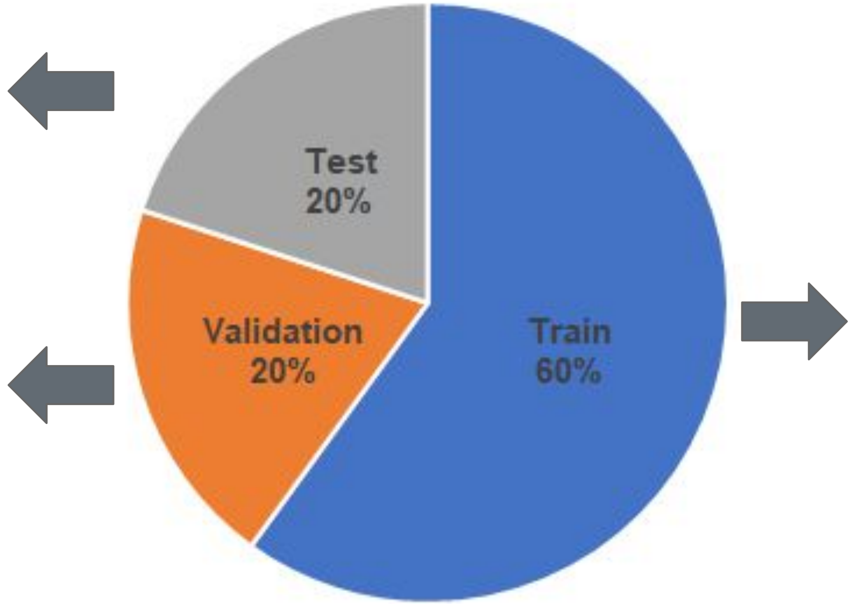
Log (Swim Distance+1)

OtherKills = kills - headshotKills

Train / Validation / Test Data Split (6/2/2)

Test the performance of the best fit model

Compare Performance of all models and find the best fit model



Build up different types of models and optimize the models using 5-fold CV

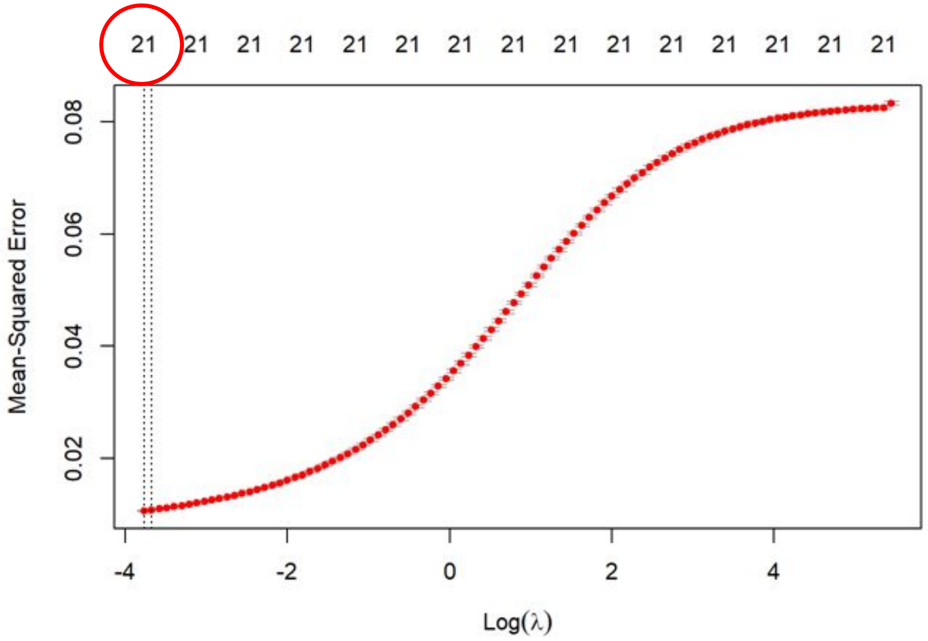
Model Comparison

What we learn:

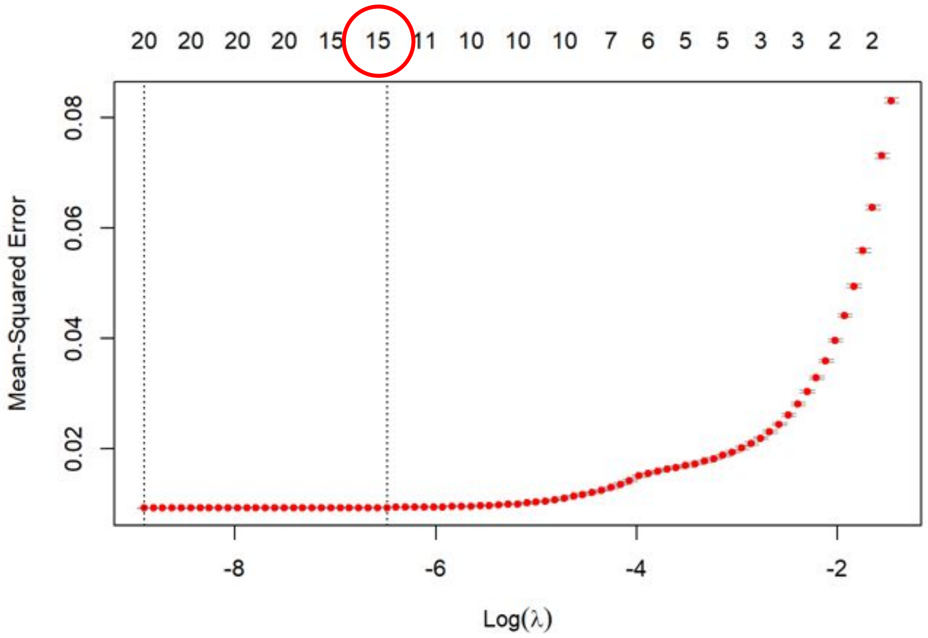
- Feature Engineering does not always improve model Performance.
- Low flexible models could outperform high flexible models.

Models [Before Feature Engineering]	RMSE	Models [After Feature Engineering]	RMSE
Linear Regression	0.09549229	Linear Regression	0.1136815
Ridge Regression	0.1047128	Ridge Regression	0.1149708
Lasso Regression	0.09552912	Lasso Regression	0.1137235
Elastic Net	0.09554557	Elastic Net	0.1137238
Decision Tree	0.08445499	Decision Tree	0.08465836
Gradient Boosting	0.07970851	Gradient Boosting	0.08129965
Bagging	0.07260767	Bagging	0.07389403
Random Forest	**0.07032698**	Random Forest	0.07226445

Ridge V.S. Lasso



RMSE: 0.1047128



RMSE: 0.09552912

Hyperparameter Tuning for Tree Based Models

Expand Grid Cross Validation

Gradient Boosting

- Depth = 3
- Learning rate=0.01
- Ntrees = 1000

RMSE <dbl>	n.trees <dbl>
0.15261738	100
0.11129026	200
0.08628056	500
0.07754863	1000

Random Forest

- Mtry = 10

(Mtry: number of predictors sampled for splitting at each node)

	mtry	OOBError
5	5	0.005310172
7	7	0.005127089
10	10	0.004987805
15	15	0.005040988

Bagging

- Complexity Parameter = 1e-04
- Nbagg [100,200,500]

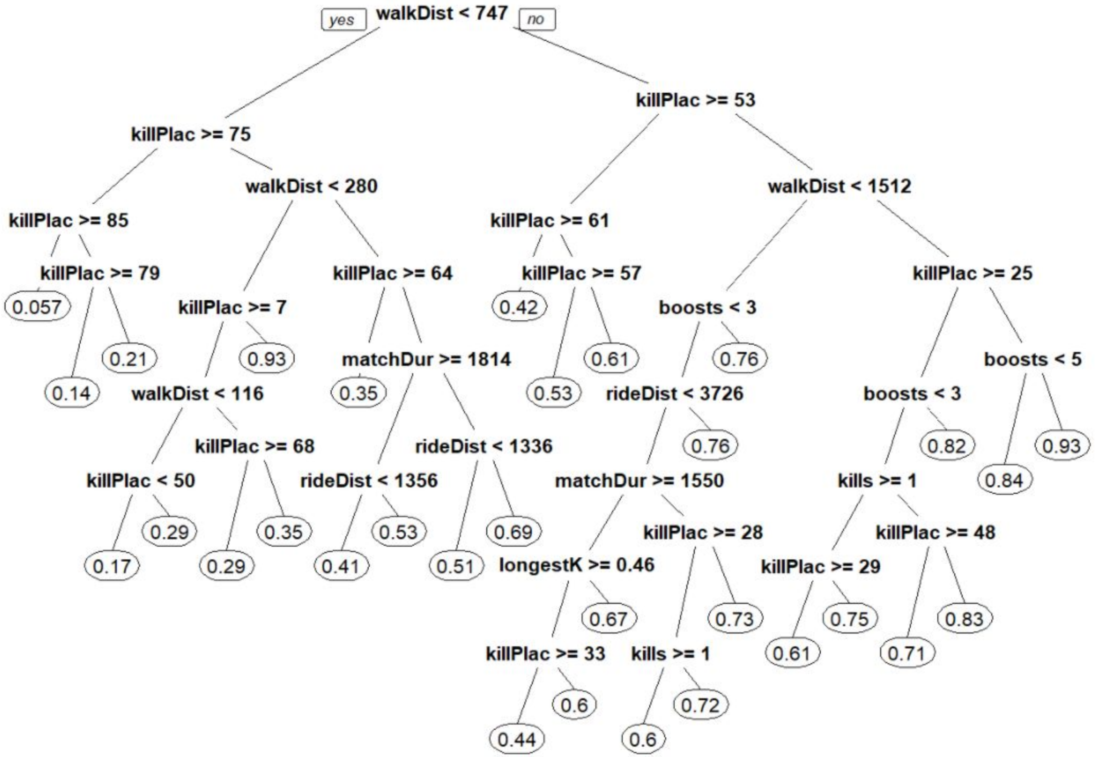
cp <dbl>	RMSE <dbl>
1e-01	0.16627019
1e-02	0.11173671
1e-03	0.08349359
1e-04	0.06104832

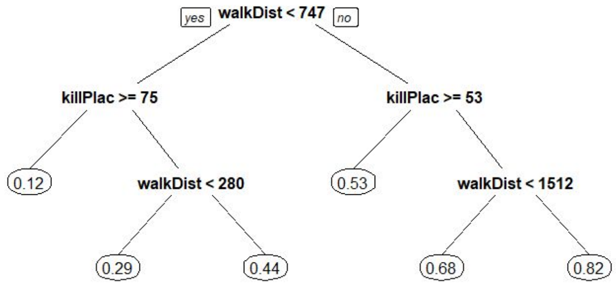
Decision Tree

- Complexity Parameter = 1e-08

cp <dbl>	RMSE <dbl>
1e-01	0.17311437
1e-02	0.11867681
1e-03	0.08951498
1e-04	0.06938891
1e-05	0.05599443
1e-06	0.05517040
1e-07	0.05514929
1e-08	0.05514927

Decision Tree Plot





Left: complexity parameter = 0.001
Right: complexity parameter = 0.01

Best Fit Random Forest Model

RMSE val: 0.07032698

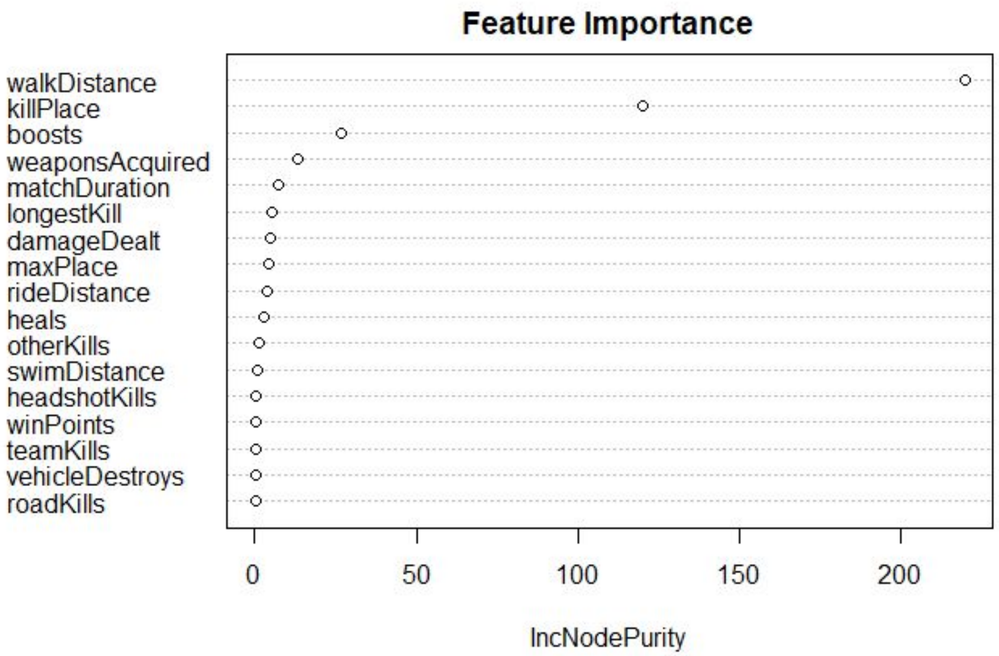
RMSE test: 0.07060608

Test Data Prediction Performance

Key Predictors of Finish Placement

- Based on Regression Models and Decision Trees

	Walk Distance	+
	Kill place	–
	# of boost items used	+
	Weapons Acquired	+
	Match Duration	–



Implications

For Consumers:

- Adopt the optimal strategies to maximize the chance of winning

For Game Designers:

- Improve user experience by using match algorithm to gather similar level players