Capstone: Choose Your Own

Artjoms Formulevics

03.04.2020

Contents

1 Executive Summary

Motivation: Author is a big fan of both real football as sports and as a digital game (FIFA series). Therefore, it was interesting topic to take for the analysis.

Purpose of the project: Sometimes it is hard to determine, what place on the football field is the most suitable for the player to play on. Therefore the goal of this project is to create player classification algorithm for players from FIFA 19 game. Classification of players is done by predicting their most suitable position on the field (role) during games.

Dataset used: FIFA 19 complete player dataset" is used from Kaggle. Dataset includes detailed attributes for every player registered in the latest edition of FIFA 19 database.

This dataset contains ~18k rows (players) and 89 columns (attributes).

The project is divided in the several parts:

1. Data import and data exploration;
2. Data pre-processing;
3. Building several prediction models;
4. Evaluating results and drawing conclusions.

Classification: All players except for Goalkeepers will be divided in 3 categories - Defender, Midfielder and Attacker. Models will be predicting to which of these 3 positions belongs the player.

Criteria: Models are going to be evaluated with multiple metrics available in Caret package, from which Accuracy metric will be one of the most important.

As the best model, the **GLMNET** model was selected. The Accuracy of this model was ~ **0.895** and Mean Balanced Accuracy ~ **0.913**.

2 Exploratory Data Analysis

2.1 Data Tidying and Pre-processing

The main dataset is loaded as `fifa`. There are 89 columns. The set includes players attributes like Age, Nationality, Overall, Potential, Club, Value, Wage, Preferred Foot, International Reputation, Weak Foot, Skill Moves, Work Rate, etc.

Table 1: Dimensions of the dataset

Rows	Columns
18207	89

Table 2: The structure of the data

variable_name	class	first_values
X	integer	0, 1, 2
ID	integer	158023, 20801, 190871
Name	character	L. Messi, Cristiano Ronaldo, Neymar Jr
Age	integer	31, 33, 26
Photo	character	https://cdn.sofifa.org/players/4/19/158023.png https://cdn.sofifa.org/players/4/19/20801.png, https://cdn.sofifa.org/players/4/19/190871.png
Nationality	character	Argentina, Portugal, Brazil
Flag	character	https://cdn.sofifa.org/flags/52.png, https://cdn.sofifa.org/flags/38.png, https://cdn.sofifa.org/flags/54.png
Overall	integer	94, 94, 92
Potential	integer	94, 94, 93
Club	character	FC Barcelona, Juventus, Paris Saint-Germain
Club.Logo	character	https://cdn.sofifa.org/teams/2/light/241.png, https://cdn.sofifa.org/teams/2/light/45.png, https://cdn.sofifa.org/teams/2/light/73.png
Value	character	€110.5M, €77M, €118.5M
Wage	character	€565K, €405K, €290K
Special	integer	2202, 2228, 2143
Preferred.Foot	character	Left, Right, Right
International.Reputation	integer	5, 5, 5
Weak.Foot	integer	4, 4, 5
Skill.Moves	integer	4, 5, 5
Work.Rate	character	Medium/ Medium, High/ Low, High/ Medium
Body.Type	character	Messi, C. Ronaldo, Neymar
Real.Face	character	Yes, Yes, Yes
Position	character	RF, ST, LW
Jersey.Number	integer	10, 7, 10
Joined	character	Jul 1, 2004, Jul 10, 2018, Aug 3, 2017

Table 2: The structure of the data *(continued)*

variable_name	class	first_values
Loaned.From	character	, ,
Contract.Valid.Until	character	2021, 2022, 2022
Height	character	5'7, 6'2, 5'9
Weight	character	159lbs, 183lbs, 150lbs
LS	character	88+2, 91+3, 84+3
ST	character	88+2, 91+3, 84+3
RS	character	88+2, 91+3, 84+3
LW	character	92+2, 89+3, 89+3
LF	character	93+2, 90+3, 89+3
CF	character	93+2, 90+3, 89+3
RF	character	93+2, 90+3, 89+3
RW	character	92+2, 89+3, 89+3
LAM	character	93+2, 88+3, 89+3
CAM	character	93+2, 88+3, 89+3
RAM	character	93+2, 88+3, 89+3
LM	character	91+2, 88+3, 88+3
LCM	character	84+2, 81+3, 81+3
CM	character	84+2, 81+3, 81+3
RCM	character	84+2, 81+3, 81+3
RM	character	91+2, 88+3, 88+3
LWB	character	64+2, 65+3, 65+3
LDM	character	61+2, 61+3, 60+3
CDM	character	61+2, 61+3, 60+3
RDM	character	61+2, 61+3, 60+3
RWB	character	64+2, 65+3, 65+3
LB	character	59+2, 61+3, 60+3
LCB	character	47+2, 53+3, 47+3
CB	character	47+2, 53+3, 47+3
RCB	character	47+2, 53+3, 47+3
RB	character	59+2, 61+3, 60+3
Crossing	integer	84, 84, 79
Finishing	integer	95, 94, 87
HeadingAccuracy	integer	70, 89, 62
ShortPassing	integer	90, 81, 84
Volleys	integer	86, 87, 84
Dribbling	integer	97, 88, 96
Curve	integer	93, 81, 88
FKAccuracy	integer	94, 76, 87
LongPassing	integer	87, 77, 78
BallControl	integer	96, 94, 95
Acceleration	integer	91, 89, 94
SprintSpeed	integer	86, 91, 90
Agility	integer	91, 87, 96
Reactions	integer	95, 96, 94
Balance	integer	95, 70, 84

Table 2: The structure of the data *(continued)*

variable_name	class	first_values
ShotPower	integer	85, 95, 80
Jumping	integer	68, 95, 61
Stamina	integer	72, 88, 81
Strength	integer	59, 79, 49
LongShots	integer	94, 93, 82
Aggression	integer	48, 63, 56
Interceptions	integer	22, 29, 36
Positioning	integer	94, 95, 89
Vision	integer	94, 82, 87
Penalties	integer	75, 85, 81
Composure	integer	96, 95, 94
Marking	integer	33, 28, 27
StandingTackle	integer	28, 31, 24
SlidingTackle	integer	26, 23, 33
GKDiving	integer	6, 7, 9
GKHandling	integer	11, 11, 9
GKKicking	integer	15, 15, 15
GKPositioning	integer	14, 14, 15
GKReflexes	integer	8, 11, 11
Release.Clause	character	€226.5M, €127.1M, €228.1M

As can be seen from the data structure table, there are numerous variables that must be pre-processed, in order to be incorporated into algorithm:

- Value, Wage and Release Clause are represented as characters with K or M sign, and will be converted to numerical;
- Height and Weigth are are represented as characters in Imperial System, and will be converted to numerical (+ converted to Metric System);
- There are multiple variables that are not meaningful and therefore need to be excluded (e.g. link to Player Photo or some parameters that are not any way related to player's position, e.g. Date when Joined the Club);
- Goalkeepers will be excluded from the analysis, as they have completely different set of important characterisitcs. They differ to much from field players;
- There are 26 columns (from nr. 29 to 54) that represent player's fit to each position. Since the goal of this project is to predict the fit for position, these columns will be also deleted to make the predictions clean.

```
# Removing Poistion scores that won't be used
fifa[, 29:54] <- NULL

# Removing not meaningful variables
fifa <-
  subset(
    fifa,
    select = -c(
      ID,
      X,
      Photo,
```

```
    Flag,
    Club.Logo,
    Real.Face,
    Joined,
    Loaned.From,
    Contract.Valid.Until,
    GKDiving,
    GKHandling,
    GKKicking,
    GKPositioning,
    GKReflexes,
    ValueLast,
    WageLast,
    Release.Clause.Last,
    Special
  )
)
```

In addition: Player Positions must be grouped to just three - Defender (Def), Midfielder (Mid) and Attacker (Att). These three position groups will be later predicted by the algorithms. Current positions will be grouped to new as follows:

Table 3: Original vs New Positions

Original	New
CAM	Mid
CDM	Mid
CM	Mid
LAM	Mid
LCM	Mid
LDM	Mid
LM	Mid
RAM	Mid
RCM	Mid
RDM	Mid
RM	Mid
CB	Def
LB	Def
LCB	Def
LWB	Def
RB	Def
RCB	Def
RWB	Def
CF	Att
LF	Att
LS	Att
LW	Att
RF	Att
RS	Att
RW	Att
ST	Att

2.2 Data Exploration

Taking a look at the dataset after initial processing:

Table 4: New Dimensions of the dataset

Rows	Columns
16122	49

Table 5: New Dataset Structure

variable_name	class	first_values
Name	character	L. Messi, Cristiano Ronaldo, Neymar Jr
Age	integer	31, 33, 26
Nationality	character	Argentina, Portugal, Brazil
Overall	integer	94, 94, 92
Potential	integer	94, 94, 93
Club	character	FC Barcelona, Juventus, Paris Saint-Germain
Value	double	1.1e+08, 7.7e+07, 1.18e+08
Wage	double	565000, 405000, 290000
Preferred.Foot	character	Left, Right, Right
International.Reputation	integer	5, 5, 5
Weak.Foot	integer	4, 4, 5
Skill.Moves	integer	4, 5, 5
Work.Rate	character	Medium/ Medium, High/ Low, High/ Medium
Body.Type	character	Messi, C. Ronaldo, Neymar
Position	character	RF, ST, LW
Jersey.Number	integer	10, 7, 10
Height	double	170.18, 187.96, 175.26
Weight	double	72.121128, 83.007336, 68.0388
Crossing	integer	84, 84, 79
Finishing	integer	95, 94, 87
HeadingAccuracy	integer	70, 89, 62
ShortPassing	integer	90, 81, 84
Volleys	integer	86, 87, 84
Dribbling	integer	97, 88, 96
Curve	integer	93, 81, 88
FKAccuracy	integer	94, 76, 87
LongPassing	integer	87, 77, 78
BallControl	integer	96, 94, 95
Acceleration	integer	91, 89, 94
SprintSpeed	integer	86, 91, 90
Agility	integer	91, 87, 96
Reactions	integer	95, 96, 94
Balance	integer	95, 70, 84
ShotPower	integer	85, 95, 80
Jumping	integer	68, 95, 61
Stamina	integer	72, 88, 81

Table 5: New Dataset Structure *(continued)*

variable_name	class	first_values
Strength	integer	59, 79, 49
LongShots	integer	94, 93, 82
Aggression	integer	48, 63, 56
Interceptions	integer	22, 29, 36
Positioning	integer	94, 95, 89
Vision	integer	94, 82, 87
Penalties	integer	75, 85, 81
Composure	integer	96, 95, 94
Marking	integer	33, 28, 27
StandingTackle	integer	28, 31, 24
SlidingTackle	integer	26, 23, 33
Release.Clause	double	2.26e+08, 1.27e+08, 2.28e+08
Position_gr	integer	Att, Att, Att

Table 6: Missing Values

	x
Name	0
Age	0
Nationality	0
Overall	0
Potential	0
Club	0
Value	0
Wage	0
Preferred.Foot	0
International.Reputation	0
Weak.Foot	0
Skill.Moves	0
Work.Rate	0
Body.Type	0
Position	0
Jersey.Number	0
Height	0
Weight	0
Crossing	0
Finishing	0
HeadingAccuracy	0
ShortPassing	0
Volleys	0
Dribbling	0
Curve	0
FKAccuracy	0
LongPassing	0
BallControl	0

Table 6: Missing Values *(continued)*

	x
Acceleration	0
SprintSpeed	0
Agility	0
Reactions	0
Balance	0
ShotPower	0
Jumping	0
Stamina	0
Strength	0
LongShots	0
Aggression	0
Interceptions	0
Positioning	0
Vision	0
Penalties	0
Composure	0
Marking	0
StandingTackle	0
SlidingTackle	0
Release.Clause	1379
Position_gr	0

It can be seen that there are lots of missing values in Release Clause variable. Which is understandable, since it is optional clause in player's contract. It does mean that player if any club is ready to pay the sum listed in the Release Clause for this player, they do not need to agree with the current player's club. In other words, it is the sum for which club is consent to sell the player. It is not mandatory to have this caluse in player's contract - in this case there is no pre-determined sum and this player can be sold only if two clubs agree on the certain amount. That means not having the release clause is somewhat equal to have it as infinite number. Therefore, there is no good way to deal with `NA` values and it will be removed from the analysis in order to leave in consistent.

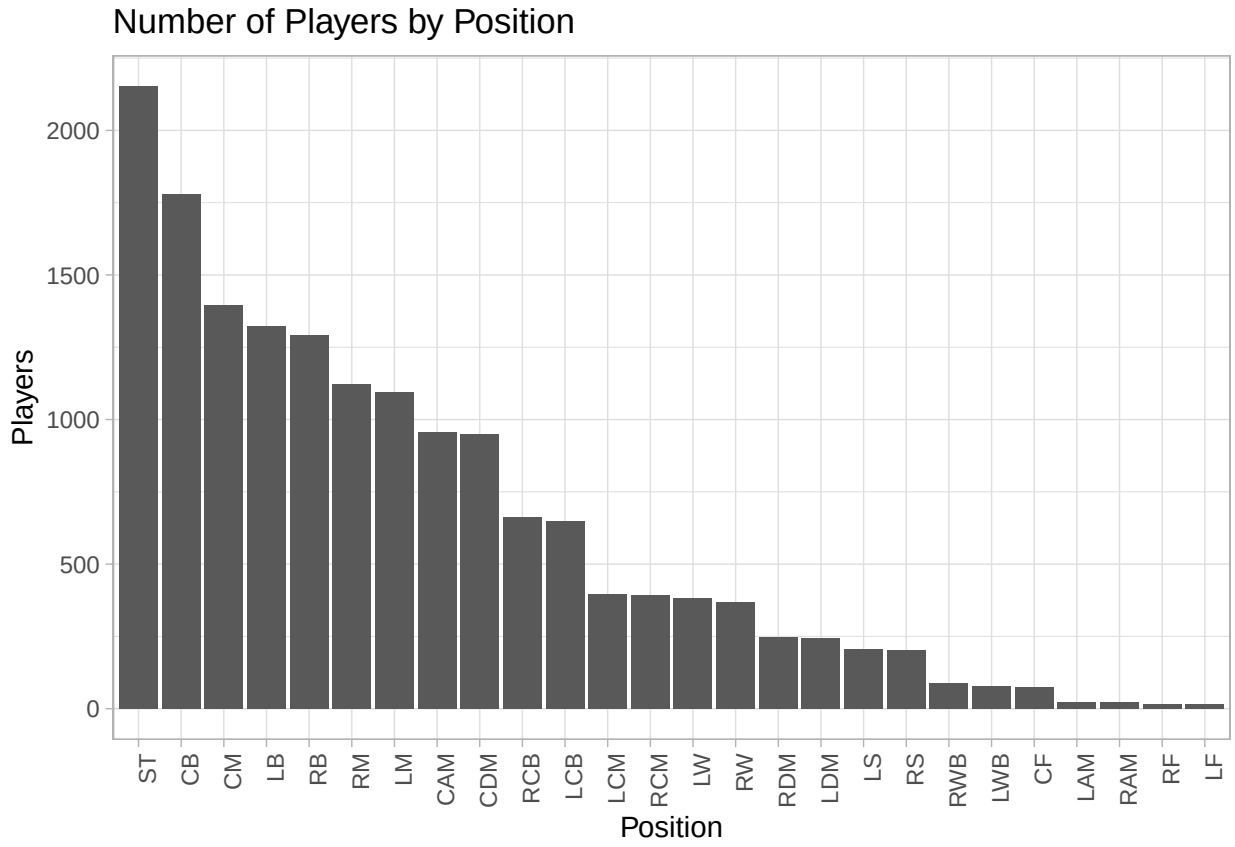
```
# Removing Release Clause from analysis
fifa$Release.Clause <- NULL
```

2.3 Position Analysis

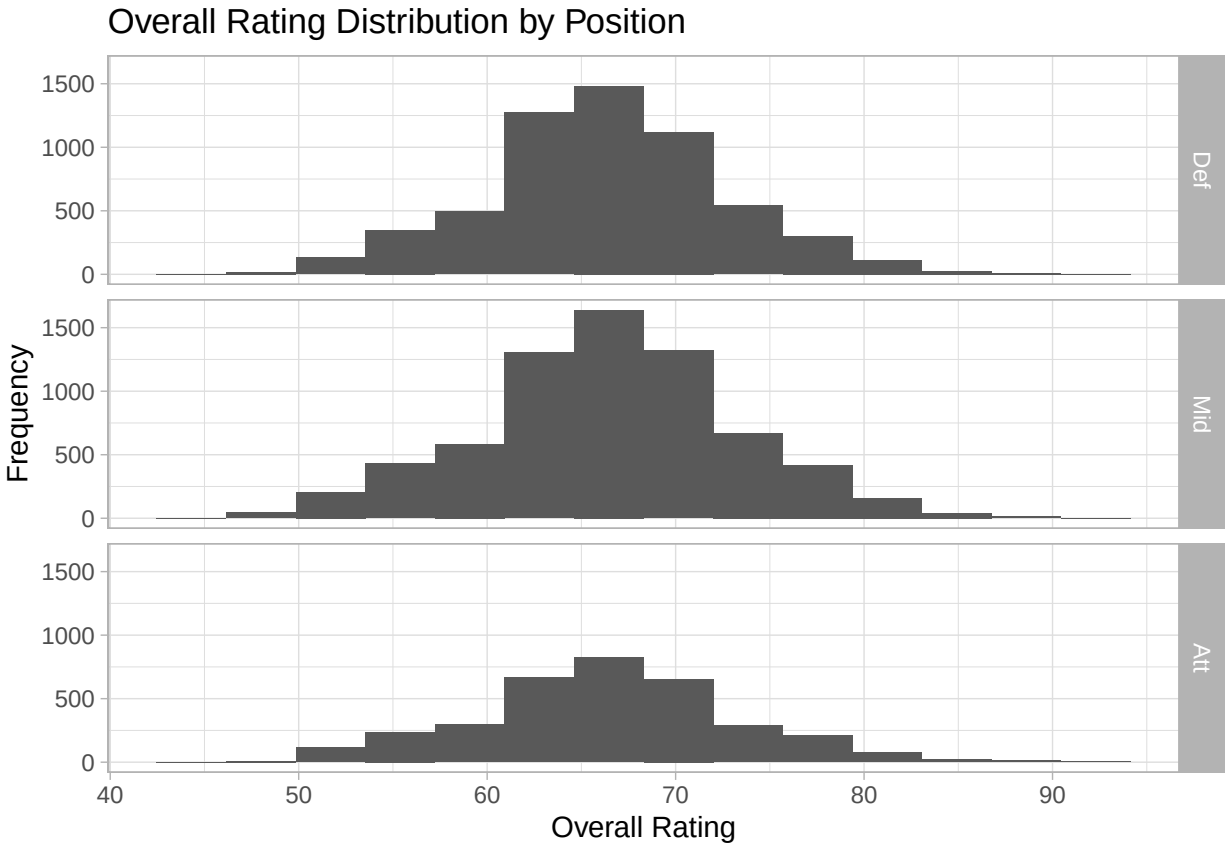
Chart below shows the number of players by each position (as originally in the dataset). It can be seen that there are many Strikers (ST) and Centre-Backs (CB). The table, however, shows how many players fall in each of three new defined groups - Defenders, Midfielders and Attackers. Despite there are lots of Strikers, overall nubmer of Attacking Players is lower than for Midfileders and Defenders.

Table 7: Number of Players by Position

Position	count
Def	5866
Mid	6838
Att	3418



Following chart shows the distribution of Player's ratings by Position. It can be seen, that all three groups follow normal distribution.



2.4 Player Analysis

Following table shows 15 top players in the game by their Overall raiting. Can be seen that there are just few playes that have scores above 90.

Table 8: Number of Players by Position

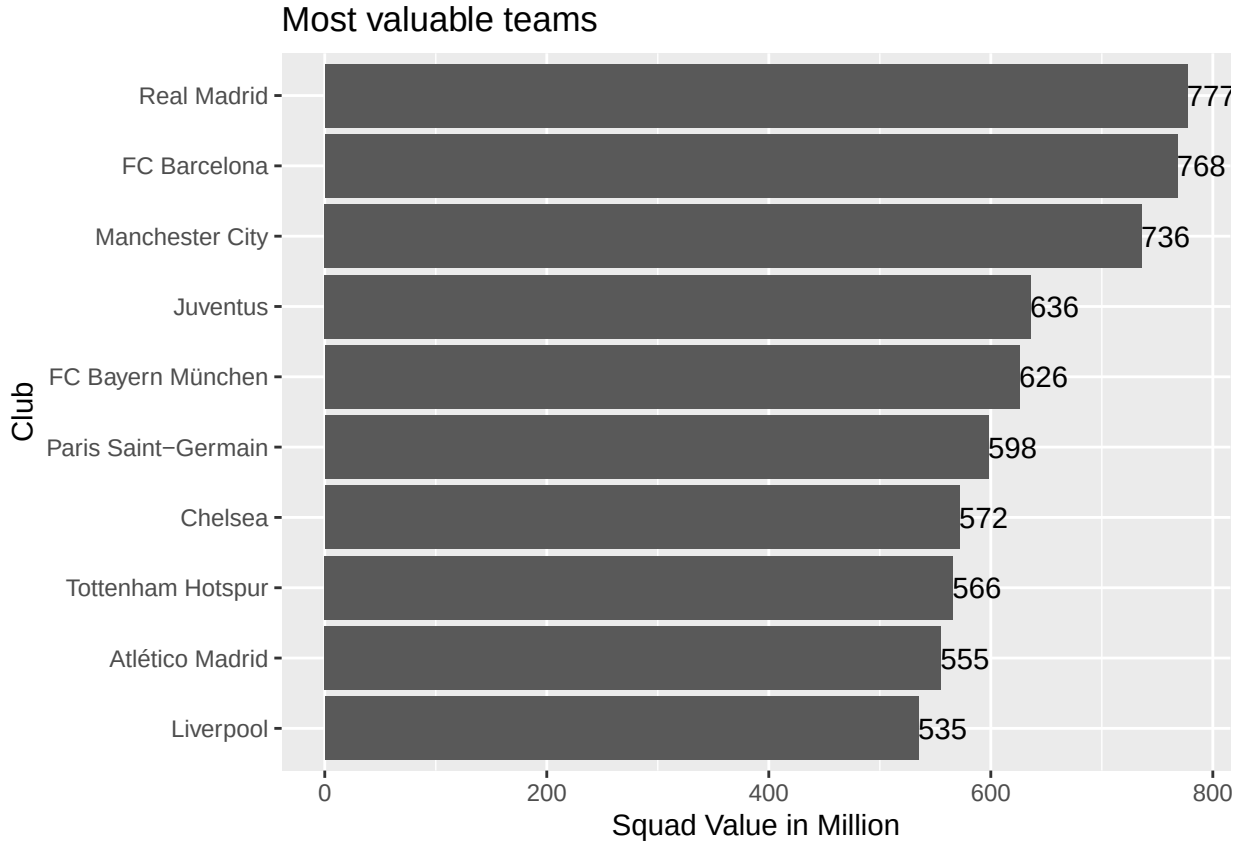
Position_gr	Name	Overall	Potential
Att	Cristiano Ronaldo	94	94
Att	L. Messi	94	94
Att	Neymar Jr	92	93
Def	Sergio Ramos	91	91
Mid	K. De Bruyne	91	92
Mid	L. Modrić	91	91
Att	E. Hazard	91	91
Att	L. Suárez	91	91
Def	D. Godín	90	90
Mid	David Silva	90	90
Mid	T. Kroos	90	90
Att	R. Lewandowski	90	90
Def	G. Chiellini	89	89
Mid	A. Griezmann	89	90
Mid	N. Kanté	89	90

Following table shows top 15 highest valued players and their wages. Many players in this table previously appeared in table from above, which means that high paid players are usually amongst the best by rating.

Table 9: Top 15 players by Value

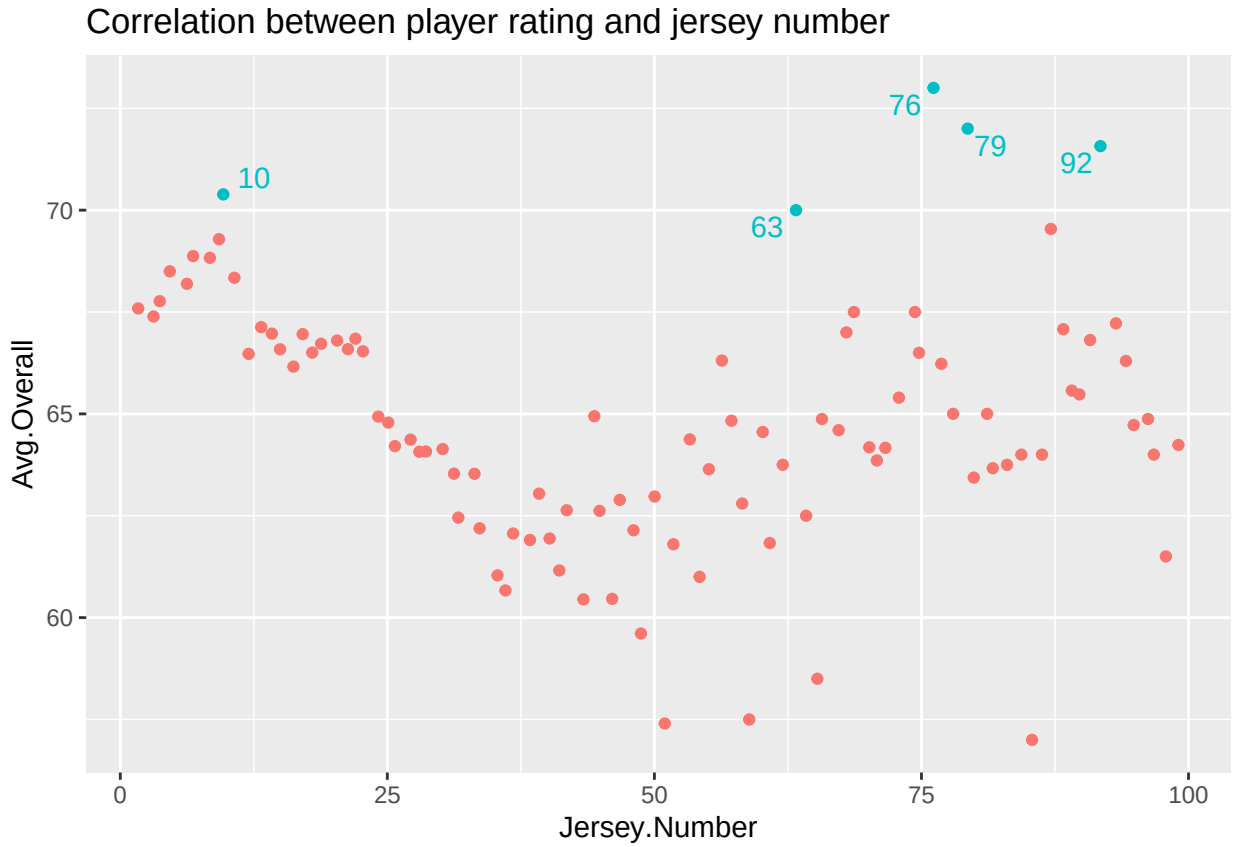
Position_gr	Name	Value	Wage
Att	Neymar Jr	1.18e+08	290000
Att	L. Messi	1.10e+08	565000
Mid	K. De Bruyne	1.02e+08	355000
Att	E. Hazard	9.30e+07	340000
Att	P. Dybala	8.90e+07	205000
Att	H. Kane	8.30e+07	205000
Mid	K. Mbappé	8.10e+07	100000
Att	L. Suárez	8.00e+07	455000
Mid	A. Griezmann	7.80e+07	145000
Att	Cristiano Ronaldo	7.70e+07	405000
Att	R. Lewandowski	7.70e+07	205000
Mid	T. Kroos	7.60e+07	355000
Mid	C. Eriksen	7.30e+07	205000
Att	Isco	7.30e+07	315000
Mid	J. Rodríguez	6.90e+07	315000

Following chart shows most valuable tems by total player value. As expected, most renowed and popular clubs in real life are the most valuable in the game as well.



2.5 Jersey Number Analysis

Following plot shows correlation between player jersey number and player's overall rating. It can be seen that only few nubers have high-skilled players on average. Plus, the numbers '63, 76, 79, 92' are not popular, so probably owned only by few young talents. Jersey number '10' which is traditionally worn by great playmakers or attackers stands out as scoring average above 70. Also, small numbers (2-11) are tending having better ratings, as they are usually picked by first-team (best) players.



The following table shows the most popular jersey numbers for each group of positions. Not surprisingly, small numbers (2-11) are most popular.

Table 10: Most popular jerseys by posiitons

Position_gr	Jersey.Number	count
	3	503
	2	478
Def	4	435
	5	433
	15	257
	8	523
	10	400
Mid	7	393
	6	325
	11	295
	9	488

Table 10: Most popular jerseys by posiitons *(continued)*

Position_gr	Jersey.Number	count
	11	259
Att	19	199
	10	183
	7	164

3 Model Building

3.1 Preparing Dataset

Before creating algorithms, last preparation steps are neccessarry. Additional dataset `fifa_names` is created to hold the names and positions of players that then can be compared with the predictions. This dataset will be split to test and training set in the same way as the main dataset. In addition, from the dataset will be removed non-meaningful for algorithm training variables like Name, Club, Nationality (each Club and Country needs all positions, so they are not meaningful) and old Position variable (it is redundant).

```
# Creating dataset with names for testing
fifa_names <-
  subset(fifa, select = c(Name, Overall, Position, Position_gr))

# Removing Names and ungroped positions and unwanted variables
fifa$Position <- fifa$Position_gr
fifa <-
  subset(fifa, select = -c(Name, Position_gr, Club, Nationality))
```

In the following step, remaining data needds to be pre-processed:

- All numberical variables are going to be centered and scaled;
- All categorical variables will be converted to binary numbers (0 or 1) using `dummyVars` (except Position variable which will be predicted);
- Seed will be set to 1 for reproducability and assigned to variable;
- `train.control` function will be introduced with 10-fold cross-validation repeated 3 times (`repeatedcv`) for training algorithms - a balance between number of repeats and computing resources needed;
- Dataset will be divided to `train` set and `test` set. The ratio used is 90% to train and 10% to test, as the dataset's size is not big and more observations are required to train the models.

Abovementioned steps:

```
# Centering and scaling
preProc <- preProcess(fifa, method = c("center", "scale"))
fifa <- predict(preProc, fifa)

# Convering factors to numbers (dummy variables)
dmy <- dummyVars("~ .", data = fifa, fullRank = F)
fifa <- data.frame(predict(dmy, newdata = fifa))

# Putting Position back as the first variable
fifa$Position <- fifa_pos
fifa <- fifa[, c(61, 1:60)]

# Setting seed for reproducability
seed = 1
set.seed(seed)

# Train Control function for repeated cross-validation
train.control <-
  trainControl(method = "repeatedcv",
               number = 10,
```

```
                repeats = 3)

# Splitting into train and test sets 0.9 vs 0.1 (also for names)
sample <- sample.split(fifa, SplitRatio = 0.9)
train <- subset(fifa, sample == TRUE)
test <- subset(fifa, sample == FALSE)

train_names <- subset(fifa_names, sample == TRUE)
test_names <- subset(fifa_names, sample == FALSE)
```

3.2 Modelling

There are 10 popular machine learning algorithms of different complexity that are going to be used for modelling:

1. Linear Discriminant Analysis - "lda"
2. Regularized Discriminant Analysis - "rda"
3. Naive Bayes Algorithm - "nb"
4. K-Nearest Neighbours - "knn"
5. RPART: Classification And Regression Tree (CART) - "rpart"
6. C5.0 Decision Tree Algorithm - "C5.0"
7. Random Forest Algorithm - "rf"
8. GLMNET: Generalized Linear Model (Lasso and Elastic-Net Regularized) - "glmnet"
9. Gradient Bossting Machine - "gbm"
10. XGBoost Linear - "xgbLinear"

Each model with previously defined `seed` and `train.control` values will be applied to `train` set and saved to named list `fit`. Then each model predictions for `test` set observations will be saved into `pred` list. In addition, probability outcomes of each model will be saved to a list `pred.prob`.

The whole modelling cycle:

```
# Setting list and names
fit <- list()
pred <- list()
pred.prob <- list()

models <-
  c("lda",
    "rda",
    "nb",
    "knn",
    "rpart",
    "C5.0",
    "rf",
    "glmnet",
    "gbm",
    "xgbLinear")

# Creating each model and predictions & probabilities for each class
for (i in c(1:length(models))) {
  set.seed(seed)
  if(models[i] == "gbm") {
```

```
  fit[[i]] <-
    train(Position ~ .,
          data = train,
          method = models[i],
          trControl = train.control,
          verbose = F) # supress iterations output for gbm model
  } else {
    fit[[i]] <-
    train(Position ~ .,
          data = train,
          method = models[i],
          trControl = train.control)
}
  pred[[i]] <- predict(fit[[i]], test, type = "raw")
  pred.prob[[i]] <- predict(fit[[i]], test, type = "prob")
}

# Adding names to models in list
names(fit) <- models
names(pred) <- models
names(pred.prob) <- models
```

4 Model Evaluation

4.1 Examine Incorrectly Guessed Players

Following tables will show top 10 players (by rating) which were incorrectly classified by each model.

Table 11: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
21	K. Mbappé	88	RM	Mid	Att	0.00	0.07	0.93	0	lda
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.00	1.00	0	lda
65	M. Özil	86	CAM	Mid	Att	0.00	0.20	0.80	0	lda
67	Iniesta	86	LF	Att	Mid	0.00	1.00	0.00	0	lda
122	José Callejón	84	RM	Mid	Att	0.00	0.07	0.93	0	lda
187	Javi Martínez	83	CDM	Mid	Def	0.99	0.01	0.00	0	lda
311	T. Hazard	81	RW	Att	Mid	0.00	0.95	0.05	0	lda
312	K. Bellarabi	81	RM	Mid	Att	0.00	0.16	0.84	0	lda
387	Rafa	80	RW	Att	Mid	0.00	0.81	0.19	0	lda
431	A. Yarmolenko	80	RM	Mid	Att	0.00	0.08	0.92	0	lda

Table 12: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
4	K. De Bruyne	91	RCM	Mid	Att	0.00	0.00	1.00	0	rda
6	L. Modrić	91	RCM	Mid	Att	0.00	0.00	1.00	0	rda
21	K. Mbappé	88	RM	Mid	Att	0.00	0.00	1.00	0	rda
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.00	1.00	0	rda
65	M. Özil	86	CAM	Mid	Att	0.00	0.00	1.00	0	rda
67	Iniesta	86	LF	Att	Mid	0.00	0.98	0.02	0	rda
122	José Callejón	84	RM	Mid	Att	0.00	0.00	1.00	0	rda
128	D. Payet	84	CAM	Mid	Att	0.00	0.10	0.90	0	rda
183	Luiz Gustavo	83	LCB	Def	Mid	0.35	0.65	0.00	0	rda
187	Javi Martínez	83	CDM	Mid	Def	1.00	0.00	0.00	0	rda

Table 13: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
7	L. Suárez	91	RS	Att	Mid	0.00	1.00	0.00	0	nb
61	L. Bonucci	86	RCB	Def	Mid	0.00	1.00	0.00	0	nb
65	M. Özil	86	CAM	Mid	Att	0.00	0.01	0.99	0	nb
67	Iniesta	86	LF	Att	Mid	0.00	0.99	0.01	0	nb
122	José Callejón	84	RM	Mid	Att	0.00	0.09	0.91	0	nb
128	D. Payet	84	CAM	Mid	Att	0.00	0.04	0.96	0	nb
140	Josué Chiamulera	83	LCB	Def	Mid	0.08	0.92	0.00	0	nb
183	Luiz Gustavo	83	LCB	Def	Mid	0.00	1.00	0.00	0	nb
212	João Cancelo	82	RB	Def	Mid	0.00	1.00	0.00	0	nb
244	S. Mustafi	82	RCB	Def	Mid	0.00	1.00	0.00	0	nb

Table 14: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
4	K. De Bruyne	91	RCM	Mid	Att	0.00	0.33	0.67	0	knn
6	L. Modrić	91	RCM	Mid	Att	0.00	0.33	0.67	0	knn
21	K. Mbappé	88	RM	Mid	Att	0.00	0.44	0.56	0	knn
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.33	0.67	0	knn
65	M. Özil	86	CAM	Mid	Att	0.00	0.44	0.56	0	knn
67	Iniesta	86	LF	Att	Mid	0.00	0.89	0.11	0	knn
82	A. Sánchez	85	RW	Att	Mid	0.00	0.67	0.33	0	knn
187	Javi Martínez	83	CDM	Mid	Def	0.67	0.33	0.00	0	knn
273	Dani Alves	82	RB	Def	Mid	0.44	0.56	0.00	0	knn
311	T. Hazard	81	RW	Att	Mid	0.00	0.56	0.44	0	knn

Table 15: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
6	L. Modrić	91	RCM	Mid	Def	0.62	0.37	0.01	0	rpart
21	K. Mbappé	88	RM	Mid	Att	0.01	0.25	0.74	0	rpart
29	P. Aubameyang	88	LM	Mid	Att	0.01	0.25	0.74	0	rpart
67	Iniesta	86	LF	Att	Def	0.62	0.37	0.01	0	rpart
90	R. Nainggolan	85	CAM	Mid	Def	0.62	0.37	0.01	0	rpart
122	José Callejón	84	RM	Mid	Att	0.01	0.25	0.74	0	rpart
126	E. Banega	84	CDM	Mid	Def	0.62	0.37	0.01	0	rpart
128	D. Payet	84	CAM	Mid	Att	0.01	0.25	0.74	0	rpart
143	N. Keïta	83	CM	Mid	Def	0.62	0.37	0.01	0	rpart
151	L. Goretzka	83	CM	Mid	Def	0.62	0.37	0.01	0	rpart

Table 16: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
21	K. Mbappé	88	RM	Mid	Att	0.00	0.41	0.59	0	C5.0
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.21	0.79	0	C5.0
67	Iniesta	86	LF	Att	Mid	0.00	0.97	0.03	0	C5.0
183	Luiz Gustavo	83	LCB	Def	Mid	0.34	0.66	0.00	0	C5.0
187	Javi Martínez	83	CDM	Mid	Def	0.77	0.23	0.00	0	C5.0
311	T. Hazard	81	RW	Att	Mid	0.00	0.63	0.37	0	C5.0
312	K. Bellarabi	81	RM	Mid	Att	0.00	0.36	0.64	0	C5.0
387	Rafa	80	RW	Att	Mid	0.00	0.87	0.13	0	C5.0
431	A. Yarmolenko	80	RM	Mid	Att	0.00	0.32	0.68	0	C5.0
445	Susaeta	80	RW	Att	Mid	0.00	0.68	0.32	0	C5.0

Table 17: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
21	K. Mbappé	88	RM	Mid	Att	0.00	0.21	0.79	0	rf
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.20	0.80	0	rf

Table 17: Top 10 players by rating incorrectly guessed *(continued)*

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
67	Iniesta	86	LF	Att	Mid	0.00	0.92	0.08	0	rf
128	D. Payet	84	CAM	Mid	Att	0.00	0.49	0.51	0	rf
183	Luiz Gustavo	83	LCB	Def	Mid	0.30	0.70	0.00	0	rf
187	Javi Martínez	83	CDM	Mid	Def	0.65	0.35	0.00	0	rf
311	T. Hazard	81	RW	Att	Mid	0.00	0.62	0.37	0	rf
387	Rafa	80	RW	Att	Mid	0.00	0.61	0.39	0	rf
431	A. Yarmolenko	80	RM	Mid	Att	0.00	0.30	0.70	0	rf
445	Susaeta	80	RW	Att	Mid	0.01	0.71	0.28	0	rf

Table 18: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
21	K. Mbappé	88	RM	Mid	Att	0.00	0.23	0.77	0	glmnet
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.01	0.99	0	glmnet
67	Iniesta	86	LF	Att	Mid	0.00	0.99	0.01	0	glmnet
122	José Callejón	84	RM	Mid	Att	0.00	0.28	0.72	0	glmnet
187	Javi Martínez	83	CDM	Mid	Def	0.93	0.07	0.00	0	glmnet
311	T. Hazard	81	RW	Att	Mid	0.00	0.88	0.12	0	glmnet
312	K. Bellarabi	81	RM	Mid	Att	0.00	0.44	0.56	0	glmnet
387	Rafa	80	RW	Att	Mid	0.00	0.80	0.20	0	glmnet
431	A. Yarmolenko	80	RM	Mid	Att	0.00	0.36	0.64	0	glmnet
445	Susaeta	80	RW	Att	Mid	0.00	0.97	0.03	0	glmnet

Table 19: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
21	K. Mbappé	88	RM	Mid	Att	0.00	0.07	0.93	0	gbm
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.17	0.83	0	gbm
67	Iniesta	86	LF	Att	Mid	0.01	0.98	0.02	0	gbm
122	José Callejón	84	RM	Mid	Att	0.01	0.46	0.53	0	gbm
183	Luiz Gustavo	83	LCB	Def	Mid	0.29	0.70	0.01	0	gbm
311	T. Hazard	81	RW	Att	Mid	0.00	0.56	0.43	0	gbm
326	J. Corona	81	RM	Mid	Att	0.00	0.31	0.68	0	gbm
387	Rafa	80	RW	Att	Mid	0.00	0.85	0.14	0	gbm
431	A. Yarmolenko	80	RM	Mid	Att	0.00	0.14	0.86	0	gbm
445	Susaeta	80	RW	Att	Mid	0.00	0.87	0.13	0	gbm

Table 20: Top 10 players by rating incorrectly guessed

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
21	K. Mbappé	88	RM	Mid	Att	0.00	0.09	0.91	0	xgbLinear
29	P. Aubameyang	88	LM	Mid	Att	0.00	0.01	0.99	0	xgbLinear
67	Iniesta	86	LF	Att	Mid	0.00	1.00	0.00	0	xgbLinear

Table 20: Top 10 players by rating incorrectly guessed *(continued)*

	Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
183	Luiz Gustavo	83	LCB	Def	Mid	0.19	0.81	0.00	0	xgbLinear
187	Javi Martínez	83	CDM	Mid	Def	0.77	0.23	0.00	0	xgbLinear
387	Rafa	80	RW	Att	Mid	0.00	0.74	0.26	0	xgbLinear
431	A. Yarmolenko	80	RM	Mid	Att	0.00	0.25	0.75	0	xgbLinear
445	Susaeta	80	RW	Att	Mid	0.00	0.88	0.12	0	xgbLinear
506	S. Bergwijn	79	LW	Att	Mid	0.00	0.77	0.23	0	xgbLinear
509	C. Pavón	79	RW	Att	Mid	0.00	0.95	0.05	0	xgbLinear

As can be seen, there are many high-rated famous players that were incorrectly classified by the algorithms. However, that has a logical explanation - despite all players have their formal role (Defender, Midfielder or Attacker), their real role on the pitch may highly differ. There are midfielders who are focused on the attack which might be (and were) classified as attackers due to prevalence of their attacking traits. And vice-versa.

In the tables above can be seen that there are players that are repeating in every or almost every algorithm. Theregore, the results were binded together to create summary of which top players were incorrectly classified by multiple algorithms.

Table 21: Highest-rated incorrectly classified players

Name	Overall	Position	Position_gr	pred	count
K. De Bruyne	91	RCM	Mid	Att	2
L. Modrić	91	RCM	Mid	Def	1
L. Modrić	91	RCM	Mid	Att	2
L. Suárez	91	RS	Att	Mid	1
K. Mbappé	88	RM	Mid	Att	9
P. Aubameyang	88	LM	Mid	Att	9
Iniesta	86	LF	Att	Def	1
Iniesta	86	LF	Att	Mid	9
L. Bonucci	86	RCB	Def	Mid	1
M. Özil	86	CAM	Mid	Att	4
A. Sánchez	85	RW	Att	Mid	1
R. Nainggolan	85	CAM	Mid	Def	1
D. Payet	84	CAM	Mid	Att	4
E. Banega	84	CDM	Mid	Def	1
José Callejón	84	RM	Mid	Att	6
Cesc Fàbregas	83	CM	Mid	Def	1
Cesc Fàbregas	83	CM	Mid	Att	1
Javi Martínez	83	CDM	Mid	Def	8
Josué Chiamulera	83	LCB	Def	Mid	1
L. Goretzka	83	CM	Mid	Def	1

By taking a closer look, it is possible to make several explanations:

- Kevin De Bruyne was classified as Attacker, while being a Midfielder, but he is well-known for being one of the best attacking midfileders and one of the best by the number of assists or created chances in English Premier League;

- Luka Modric is one of the best universal midfielders in the game, Ballon d'Or 2018 winner, that excels in both defending and attacking, so some algorithms marked him incorrectly. Moreover, as it will be shown later, he was classified incorrectly by the worst algorithms;

Table 22: Incorrect guesses for Luka Modric

Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
L. Modrić	91	RCM	Mid	Att	0.00	0.00	1.00	0	rda
L. Modrić	91	RCM	Mid	Att	0.00	0.33	0.67	0	knn
L. Modrić	91	RCM	Mid	Def	0.62	0.37	0.01	0	rpart

- Kylian Mbappe and Pierre-Emerick Aubameyang are well-known for their attacking abilities and therefore most algorithms classified them as Attackers;
- Leonardo Bonucci, while being a Defender is known for having a good passing ability and often starts attacks for his team, therefore was incorrectly marked as Midfielder.

Table 23: Incorrect guesses for Leonardo Bonucci

Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
L. Bonucci	86	RCB	Def	Mid	0	1	0	0	nb

And following table shows which players (sorted by rating) were incorrectly classfied by most (all) algorithms.

Table 24: Highest-rated incorrectly classified players by all algorithms

Name	Overall	Position	Position_gr	count
Iniesta	86	LF	Att	10
Susaeta	80	RW	Att	10
C. Pavón	79	RW	Att	10
S. Kalou	79	LM	Mid	10
E. Giaccherini	77	LW	Att	10
J. McClean	71	LW	Att	10
M. Sylla	71	RM	Mid	10
D. Zuma	70	RW	Att	10
M. Pisano	70	RW	Att	10
Arthur Caike	69	LM	Mid	10
S. Villa	69	RW	Att	10
Nando	68	RW	Att	10
Iñigo Muñoz	67	RW	Att	10
J. Opoku	67	RW	Att	10
K. Miyoshi	67	RW	Att	10
M. Carretta	67	LW	Att	10
S. Sararer	67	LM	Mid	10
F. Sakala	66	LAM	Mid	10
F. Wadja	66	CDM	Mid	10
I. Durmuş	65	LW	Att	10

Only two players with ratings of 80 and more were classified incorrectly by all algorithms. For example, Andreas Iniesta was playing Midfielder while being in Barcelona and only recently switched role to Attacker, but still has more Midfield-based traits (and one algorithms was just bad). And as can be seen from the original dataset, even in the game itself, he is more suitable to be Midfielder.

Table 25: Incorrect guesses for Andreas Iniesta

Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
Iniesta	86	LF	Att	Mid	0.00	1.00	0.00	0	lda
Iniesta	86	LF	Att	Mid	0.00	0.98	0.02	0	rda
Iniesta	86	LF	Att	Mid	0.00	0.99	0.01	0	nb
Iniesta	86	LF	Att	Mid	0.00	0.89	0.11	0	knn
Iniesta	86	LF	Att	Def	0.62	0.37	0.01	0	rpart
Iniesta	86	LF	Att	Mid	0.00	0.97	0.03	0	C5.0
Iniesta	86	LF	Att	Mid	0.00	0.92	0.08	0	rf
Iniesta	86	LF	Att	Mid	0.00	0.99	0.01	0	glmnet
Iniesta	86	LF	Att	Mid	0.01	0.98	0.02	0	gbm
Iniesta	86	LF	Att	Mid	0.00	1.00	0.00	0	xgbLinear

Table 26: Poisiton traits for Iniesta

	X77
LS	74+3
ST	74+3
RS	74+3
LW	82+3
LF	81+3
CF	81+3
RF	81+3
RW	82+3
LAM	85+3
CAM	85+3
RAM	85+3
LM	82+3
LCM	83+3
CM	83+3
RCM	83+3
RM	82+3
LWB	71+3
LDM	73+3
CDM	73+3
RDM	73+3
RWB	71+3
LB	68+3
LCB	63+3
CB	63+3
RCB	63+3
RB	68+3

In the following table can be seen how many incorrect predictions was for each model in descending order out of total 1852 observations in test set.

Table 27: Total number of players incorrectly predicted by each model

Model	Missed
rpart	623
nb	450
rda	262
knn	246
C5.0	216
lda	215
gbm	209
rf	206
xgbLinear	205
glmnet	194

Already from this table can be seen that more complex models like GLMNET, XGBoost, Gradient Boosting Machine and Random Forest are coping with the task better that the others.

4.2 Evaluation Metrics

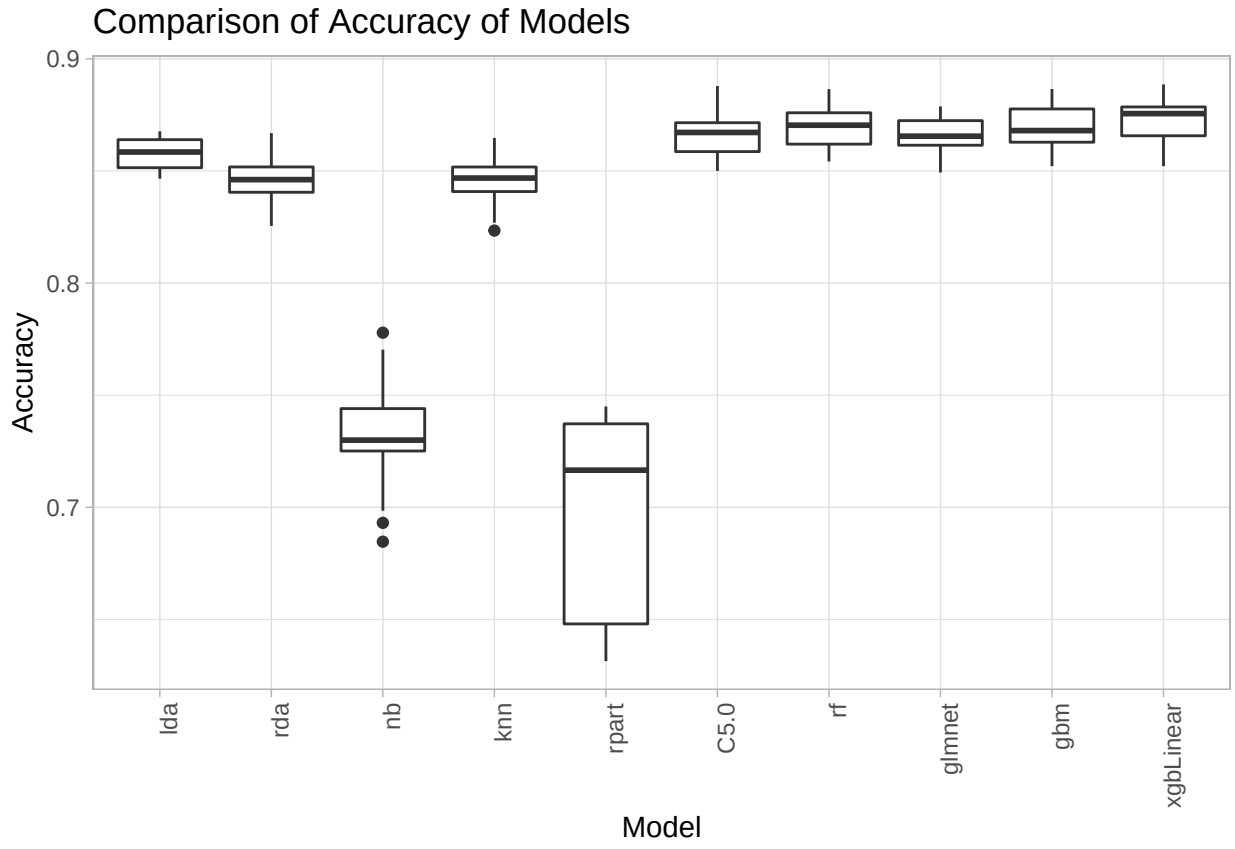
Below can be seen summary statistics for Accuracy and Kappa metrics for resamples of all algorithms.

```
# Creating summary for all resamples for all models with Accuracy and Kappa metrics
results_resamples <- resamples(fit)
summary(results_resamples, metric = c("Accuracy", "Kappa"))
```

```
##
## Call:
## summary.resamples(object = results_resamples, metric = c("Accuracy", "Kappa"))
##
## Models: lda, rda, nb, knn, rpart, C5.0, rf, glmnet, gbm, xgbLinear
## Number of resamples: 30
##
## Accuracy
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda       0.8465312 0.8514366 0.8584443 0.8567192 0.8639551 0.8676471   13
## rda       0.8255081 0.8404906 0.8461275 0.8458307 0.8517870 0.8668535    0
## nb        0.6846531 0.7250921 0.7299476 0.7327252 0.7439985 0.7778556    0
## knn       0.8234057 0.8408061 0.8468280 0.8459698 0.8517608 0.8647512    0
## rpart     0.6313945 0.6479764 0.7165382 0.6955640 0.7372109 0.7449194    0
## C5.0      0.8500350 0.8586195 0.8671570 0.8657793 0.8714986 0.8878767    0
## rf        0.8542397 0.8619481 0.8704028 0.8690028 0.8759414 0.8864751    0
## glmnet    0.8493343 0.8614712 0.8654989 0.8662229 0.8724373 0.8787666    0
## gbm       0.8521374 0.8628241 0.8679972 0.8692128 0.8776517 0.8865546    0
## xgbLinear 0.8521374 0.8656744 0.8755699 0.8732771 0.8785490 0.8885774    0
##
## Kappa
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
```

```
## lda       0.7604817 0.7682351 0.7784228 0.7764571 0.7877375 0.7934350   13
## rda       0.7255288 0.7501159 0.7580717 0.7580077 0.7673453 0.7915805    0
## nb        0.5296604 0.5789888 0.5867291 0.5892276 0.6072787 0.6466243    0
## knn       0.7224106 0.7500766 0.7591920 0.7582169 0.7672475 0.7878995    0
## rpart     0.4310199 0.4557435 0.5535817 0.5252032 0.5846415 0.6000080    0
## C5.0      0.7650344 0.7790097 0.7920800 0.7900742 0.7994482 0.8244102    0
## rf        0.7716602 0.7841654 0.7967353 0.7949451 0.8059768 0.8224171    0
## glmnet    0.7648530 0.7834238 0.7904542 0.7910901 0.8009560 0.8108923    0
## gbm       0.7685945 0.7861078 0.7936868 0.7956406 0.8096849 0.8228757    0
## xgbLinear 0.7685591 0.7897405 0.8050287 0.8018813 0.8097754 0.8255054    0
```

Accuracy metrics plotted as Box-Plot.



It can be seen that Naive Bayes and RPART algorithms are significantly lacking behind others.

Following code prepares summary statistics from `Multi-Class summary` from `Caret` package, including metrics like Accuracy, AUC, Precision, Recall, F1-score and others.

```
# Preparing tables for summary function (selecting observations, predictions and probabilities)
# Creating Multi-Class summary stats from caret package
summary_tabl <- list()
summary_stats <- list()

for (i in c(1:length(models))) {
  summary_tabl[[i]] <- test.model[[i]][, 4:8]
  colnames(summary_tabl[[i]]) <- c("obs", "pred", "Def", "Mid", "Att")
```

```
  summary_stats[[i]] <- multiClassSummary(data = summary_tabl[[i]],
                                          lev = levels(test$Position),
                                          model = fit[[i]])
}

# Adding model names
names(summary_tabl) <- models
names(summary_stats) <- models
```

All these metrics were ranked for each algorithm. Algorithm that has scored the best metric (e.g. the highest Accuracy) received 10 pts (inverted rank) for that, second - 9 pts, and up to last - 1 pts. The average of points was calculated and models were sorted in the descending order (the most "accurate" model by most metrics first)

Table 28: Algorithms and their metircs

model	glmnet	xgbLinear	rf	gbm	lda	C5.0	knn	rda	nb	rpart
score	9.50	8.21	8.07	7.29	6.29	5.57	4.00	3.00	1.93	1.14
logLoss	0.3588	0.5137	0.3165	0.3012	0.5817	0.4117	0.6613	0.8395	4.4994	0.6894
AUC	0.9697	0.9719	0.9733	0.9724	0.9660	0.9684	0.9586	0.9603	0.8793	0.8000
prAUC	0.8182	0.5140	0.8720	0.8903	0.6047	0.7769	0.5586	0.4740	0.2869	0.1861
Accuracy	0.8952	0.8893	0.8888	0.8871	0.8839	0.8834	0.8672	0.8585	0.7570	0.6636
Kappa	0.8351	0.8256	0.8247	0.8223	0.8179	0.8160	0.7904	0.7758	0.6255	0.4806
Mean_F1	0.8885	0.8806	0.8797	0.8777	0.8778	0.8740	0.8580	0.8508	0.7631	0.6580
Mean_Sensitivity	0.8830	0.8742	0.8732	0.8723	0.8757	0.8666	0.8506	0.8400	0.7821	0.6773
Mean_Specificity	0.9435	0.9403	0.9401	0.9395	0.9383	0.9369	0.9281	0.9222	0.8733	0.8238
Mean_Pos_Pred_Value	0.8951	0.8887	0.8879	0.8842	0.8805	0.8836	0.8676	0.8666	0.7715	0.6921
Mean_Neg_Pred_Value	0.9459	0.9431	0.9429	0.9419	0.9392	0.9401	0.9314	0.9273	0.8722	0.8486
Mean_Precision	0.8951	0.8887	0.8879	0.8842	0.8805	0.8836	0.8676	0.8666	0.7715	0.6921
Mean_Recall	0.8830	0.8742	0.8732	0.8723	0.8757	0.8666	0.8506	0.8400	0.7821	0.6773
Mean_Detection_Rate	0.2984	0.2964	0.2963	0.2957	0.2946	0.2945	0.2891	0.2862	0.2523	0.2212
Mean_Balanced_Accuracy	0.9132	0.9073	0.9066	0.9059	0.9070	0.9018	0.8894	0.8811	0.8277	0.7506

As previously, by comparing all the metrics, GLMNET is above all other algorothms for this task. The only metric where it was not on top is `prAUC`. XGBoost which may be awarded the second place is lacking behind in also `prAUC` and also in `logLoss` metric, which is much higher than for other top algorithms. However, Random Forest algorithm is consistently good algorithm across all metrics.

Confusion matrices for the best GLMNET algorithm can be seen below.

Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Def Mid Att
##        Def 619  34   2
##        Mid  45 730  69
##        Att   0  44 309
##
## Overall Statistics
##
##                Accuracy : 0.8952
##                  95% CI : (0.8804, 0.9088)
```

```
##     No Information Rate : 0.4363
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8351
##
##  Mcnemar's Test P-Value : 0.02847
##
## Statistics by Class:
##
##                      Class: Def Class: Mid Class: Att
## Sensitivity              0.9322     0.9035     0.8132
## Specificity              0.9697     0.8908     0.9701
## Pos Pred Value           0.9450     0.8649     0.8754
## Neg Pred Value           0.9624     0.9226     0.9526
## Prevalence               0.3585     0.4363     0.2052
## Detection Rate           0.3342     0.3942     0.1668
## Detection Prevalence     0.3537     0.4557     0.1906
## Balanced Accuracy        0.9510     0.8971     0.8916
```

Confusion Matrix Precision-Recall

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Def Mid Att
##        Def 619  34   2
##        Mid  45 730  69
##        Att   0  44 309
##
## Overall Statistics
##
##                Accuracy : 0.8952
##                  95% CI : (0.8804, 0.9088)
##     No Information Rate : 0.4363
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8351
##
##  Mcnemar's Test P-Value : 0.02847
##
## Statistics by Class:
##
##                      Class: Def Class: Mid Class: Att
## Precision                0.9450     0.8649     0.8754
## Recall                   0.9322     0.9035     0.8132
## F1                       0.9386     0.8838     0.8431
## Prevalence               0.3585     0.4363     0.2052
## Detection Rate           0.3342     0.3942     0.1668
## Detection Prevalence     0.3537     0.4557     0.1906
## Balanced Accuracy        0.9510     0.8971     0.8916
```

As can be seen form matrices, only two players that are Attackers were classified as Defenders. All other incorrect guesses were for neighbouring positions Def/Mid or Mid/Att.

Table 29: Attackers classified as Defenders for GLMNET

Name	Overall	Position	Position_gr	pred	Def	Mid	Att	correct	method
M. Diouf	74	ST	Att	Def	0.63	0.10	0.27	0	glmnet
J. McGarry	54	LW	Att	Def	0.53	0.43	0.04	0	glmnet

5 Conclusions

For this task, GLMNET model was able to get ahead of other algorithms scoring Accuracy of **0.8952**, Mean Balanced Accuracy of **0.9132** and Mean F1 score of **0.8885**. The best model parameters were:

```
##   alpha       lambda
## 4  0.55 0.0006530246
```

Other good-scoring models were XGBoost with parameters:

```
##    nrounds lambda alpha eta
## 27     150    0.1   0.1 0.3
```

And Random Forests with parameters:

```
##   mtry
## 2   30
```

The results were satisfying. However the work for sure can improved by diving deeper in few models like GLMNET or XGBoost and fine-tuning them to achieve more accurate results. For example to at least eliminate two Attacking players that were classified as Defenders. Also, the best models were the most time-consuming and required lots of computing resources. Althrough they were more accurate than Lindear Discriminant Analysis (LDA) model, it was much faster to compute LDA model and the results are not dramatically worse for that. Therefore, it is possible to conclude that it is possible to ahieve good results fast using simple models, but in order to bring it to perfection - one need to have a lot of resources and time to improve prediction bit by bit.

6 Appendix

6.1 1.1 - Used enviroment/session

```
## [1] "Environment:"

##                _
## platform       x86_64-apple-darwin15.6.0
## arch           x86_64
## os             darwin15.6.0
## system         x86_64, darwin15.6.0
## status
## major          3
## minor          6.1
## year           2019
## month          07
## day            05
## svn rev        76782
## language       R
## version.string R version 3.6.1 (2019-07-05)
## nickname       Action of the Toes

## [1] "Session Info:"

## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] glmnet_3.0-2      Matrix_1.2-18     gbm_2.1.5         klaR_0.6-15
##  [5] MASS_7.3-51.5     C50_0.1.3         rpart_4.1-15      xgboost_0.90.0.2
##  [9] MLmetrics_1.1.1   caTools_1.18.0    ggrepel_0.8.2     magrittr_1.5
## [13] purrr_0.3.3       readr_1.3.1       tibble_2.1.3      tidyverse_1.3.0
## [17] tidyr_1.0.0       stringr_1.4.0     knitr_1.27        kableExtra_1.1.0
## [21] reshape2_1.4.3    forcats_0.4.0     dplyr_0.8.3       data.table_1.12.8
## [25] caret_6.0-85      ggplot2_3.2.1     lattice_0.20-38
##
## loaded via a namespace (and not attached):
##  [1] Cubist_0.2.3         colorspace_1.4-1     ellipsis_0.3.0
##  [4] class_7.3-15         fs_1.3.1             rstudioapi_0.10
##  [7] farver_2.0.3         prodlim_2019.11.13   fansi_0.4.1
## [10] mvtnorm_1.1-0        lubridate_1.7.4      xml2_1.2.2
## [13] codetools_0.2-16     splines_3.6.1        libcoin_1.0-5
```

```
## [16] Formula_1.2-3        jsonlite_1.6.1       pROC_1.16.1
## [19] broom_0.5.3          dbplyr_1.4.2         shiny_1.4.0
## [22] compiler_3.6.1       httr_1.4.1           backports_1.1.5
## [25] assertthat_0.2.1     fastmap_1.0.1        lazyeval_0.2.2
## [28] cli_2.0.1            later_1.0.0          htmltools_0.4.0
## [31] tools_3.6.1          partykit_1.2-6       gtable_0.3.0
## [34] glue_1.3.1           Rcpp_1.0.3           cellranger_1.1.0
## [37] vctrs_0.2.2          gdata_2.18.0         nlme_3.1-143
## [40] iterators_1.0.12     timeDate_3043.102    inum_1.0-1
## [43] gower_0.2.1          xfun_0.12            rvest_0.3.5
## [46] mime_0.8             miniUI_0.1.1.1       lifecycle_0.1.0
## [49] gtools_3.8.1         scales_1.1.0         ipred_0.9-9
## [52] hms_0.5.3            promises_1.1.0       yaml_2.2.1
## [55] gridExtra_2.3        stringi_1.4.5        highr_0.8
## [58] randomForest_4.6-14  foreach_1.4.7        e1071_1.7-3
## [61] shape_1.4.4          lava_1.6.6           rlang_0.4.4
## [64] pkgconfig_2.0.3      bitops_1.0-6         evaluate_0.14
## [67] ROCR_1.0-7           labeling_0.3         recipes_0.1.9
## [70] tidyselect_0.2.5     plyr_1.8.5           R6_2.4.1
## [73] gplots_3.0.1.2       generics_0.0.2       combinat_0.0-8
## [76] DBI_1.1.0            pillar_1.4.3         haven_2.2.0
## [79] withr_2.1.2          survival_3.1-8       nnet_7.3-12
## [82] modelr_0.1.5         crayon_1.3.4         questionr_0.7.0
## [85] KernSmooth_2.23-16   rmarkdown_2.1        grid_3.6.1
## [88] readxl_1.3.1         ModelMetrics_1.2.2.1 reprex_0.3.0
## [91] digest_0.6.23        webshot_0.5.2        xtable_1.8-4
## [94] httpuv_1.5.2         stats4_3.6.1         munsell_0.5.0
## [97] viridisLite_0.3.0
```

6.2 1.2 - Source Code

```
# Author: Artjoms Formulevics
# Capstone: Your Own Project

#-----Data loading/initializing-----

# Installing libraries, if not already present

if (!require(caret))
  install.packages("caret")
if (!require(data.table))
  install.packages("data.table")
if (!require(dplyr))
  install.packages("dplyr")
if (!require(forcats))
  install.packages("forcats")
if (!require(ggplot2))
  install.packages("ggplot2")
if (!require(reshape2))
  install.packages("reshape2")
if (!require(kableExtra))
  install.packages("kableExtra")
if (!require(knitr))
```

```
  install.packages("knitr")
if (!require(stringr))
  install.packages("stringr")
if (!require(tidyr))
  install.packages("tidyr")
if (!require(tidyverse))
  install.packages("tidyverse")
if (!require(magrittr))
  install.packages("magrittr")
if (!require(ggrepel))
  install.packages("ggrepel")
if (!require(caTools))
  install.packages("caTools")
if (!require(MLmetrics))
  install.packages("MLmetrics")

# Loading libraries

library(caret)
library(data.table)
library(dplyr)
library(forcats)
library(ggplot2)
library(reshape2)
library(kableExtra)
library(knitr)
library(stringr)
library(tidyr)
library(tidyverse)
library(magrittr)
library(ggrepel)
library(caTools)
library(MLmetrics)

# Setting Knitr format

options(knitr.table.format = "html")

# Loading Data

fifa <-
  read.csv(
    "https://raw.githubusercontent.com/artjoms-formulevics/HarvardX/master/data.csv",
    header = T,
    stringsAsFactors = F
  )

# Saving original dataset

fifa_original <- fifa

#-----Tidying Data----

## Taking a look at the dataset structure
```

```
# Getting dimensions (size) of the dataset

data.frame("Rows" = nrow(fifa), "Columns" = ncol(fifa)) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Getting information about variables classes and values

data.frame(
  variable_name = names(fifa),
  class = sapply(fifa, typeof),
  first_values = sapply(fifa, function(x)
    paste0(head(x, n = 3),  collapse = ", ")),
  row.names = NULL
) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Converting player Value, Wage and Release Clause to numeric

fifa$ValueLast <-
  sapply(strsplit(as.character(fifa$Value), ""), tail, 1)
fifa$WageLast <-
  sapply(strsplit(as.character(fifa$Wage), ""), tail, 1)
fifa$Release.Clause.Last <-
  sapply(strsplit(as.character(fifa$Release.Clause), ""), tail, 1)

extract <- function(x) {
  regexp <- "[[:digit:]]+"
  str_extract(x, regexp)
}

temp <- sapply(fifa$Value, extract)
fifa$Value <- as.numeric(temp)
fifa$Value <-
  ifelse(fifa$ValueLast == "M", fifa$Value * 1000000, fifa$Value * 1000)

temp <- sapply(fifa$Wage, extract)
fifa$Wage <- as.numeric(temp)
fifa$Wage <-
  ifelse(fifa$WageLast == "M", fifa$Wage * 1000000, fifa$Wage * 1000)

temp <- sapply(fifa$Release.Clause, extract)
fifa$Release.Clause <- as.numeric(temp)
```

```
fifa$Release.Clause <- ifelse(
  fifa$Release.Clause.Last == "M",
  fifa$Release.Clause * 1000000,
  fifa$Release.Clause * 1000
)

# Converting Height to numeric & metric system

temp <- str_split(fifa$Height, "'")
for (i in 1:length(temp)) {
  temp[[i]] <- as.numeric(temp[[i]])
  temp[[i]] <- (temp[[i]][1] * 12) + temp[[i]][2]
}
temp <- as.numeric(unlist(temp)) * 2.54
fifa$Height <- temp

# Converting Weight to numeric & metric system

temp <- sapply(fifa$Weight, extract)
temp <- as.numeric(temp) * 0.453592
fifa$Weight <- temp

# Filtering out Goalkeepers

fifa <- filter(fifa, fifa$Position != "GK")
fifa <- filter(fifa, fifa$Position != "")

# Removing Poistion scores that won't be used

fifa[, 29:54] <- NULL

# Removing not meaningful variables

fifa <-
  subset(
    fifa,
    select = -c(
      ID,
      X,
      Photo,
      Flag,
      Club.Logo,
      Real.Face,
      Joined,
      Loaned.From,
      Contract.Valid.Until,
      GKDiving,
      GKHandling,
      GKKicking,
      GKPositioning,
      GKReflexes,
      ValueLast,
      WageLast,
      Release.Clause.Last,
```

```
      Special
    )
  )

# Show Position groups

temp <- data.frame(Original = unique(fifa$Position))
temp$New <- fct_collapse(
  temp$Original,
  Def = c("RCB", "CB", "LCB", "LB", "RB", "RWB", "LWB"),
  Mid = c(
    "RCM",
    "LCM",
    "LDM",
    "CAM",
    "CDM",
    "RM",
    "LAM",
    "LM",
    "RDM",
    "CM",
    "RAM"
  ),
  Att = c("RF", "ST", "LF", "RS", "LS", "RW", "CF", "LW")
)
temp %>%
  arrange(New, Original) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Group all positions into three categories

fifa$Position_gr <- fct_collapse(
  fifa$Position,
  Def = c("RCB", "CB", "LCB", "LB", "RB", "RWB", "LWB"),
  Mid = c(
    "RCM",
    "LCM",
    "LDM",
    "CAM",
    "CDM",
    "RM",
    "LAM",
    "LM",
    "RDM",
    "CM",
    "RAM"
  ),
  Att = c("RF", "ST", "LF", "RS", "LS", "RW", "CF", "LW")
```

```
)

fifa$Position_gr <-
  factor(fifa$Position_gr, levels = c("Def", "Mid", "Att"))

# Remove unwanted variables and temporarty variables

rm(temp, i, extract)

#-----Manipulations/Visualizations-----

# Getting dimensions (size) of the dataset

data.frame("Rows" = nrow(fifa), "Columns" = ncol(fifa)) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Getting information about variables classes and values

data.frame(
  variable_name = names(fifa),
  class = sapply(fifa, typeof),
  first_values = sapply(fifa, function(x)
    paste0(head(x, n = 3),  collapse = ", ")),
  row.names = NULL
) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Showing missing values

sapply(fifa, function(x)
  sum(is.na(x))) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Removing Release Clause from analysis

fifa$Release.Clause <- NULL
```

```
# Number of players by position in the dataset

fifa %>% group_by(Position) %>% summarise(count = n()) %>%
  ggplot(aes(reorder(Position, -count), count)) +
  theme_light()  +
  geom_col() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Number of Players by Position",
       x = "Position",
       y = "Players")

# Number of players by position after grouping in the dataset

fifa %>% group_by(Position = Position_gr) %>% summarise(count = n()) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Distribution of Overall Rating by Position

fifa %>%
  ggplot(aes(Overall)) +
  theme_light()  +
  geom_histogram(bins =  14) +
  labs(title = "Overall Rating Distribution by Position",
       x = "Overall Rating",
       y = "Frequency") +
  facet_grid(Position_gr ~ .)

# Top 15 players by overall rating

fifa %>%
  group_by(Position_gr, Name, Overall, Potential) %>%
  summarise() %>%
  arrange(desc(Overall)) %>%
  head(n = 15) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Top 15 players by Value

fifa %>%
  group_by(Position_gr, Name, Value, Wage) %>%
  summarise() %>%
```

```
  arrange(desc(Value)) %>%
  head(n = 15) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Finding the most valuable teams

fifa %>%
  group_by(Club) %>%
  summarise(Club.Squad.Value = round(sum(Value) / 1000000)) %>%
  arrange(-Club.Squad.Value) %>%
  head(10) %>%
  ggplot(aes(
    x = as.factor(Club) %>%
      fct_reorder(Club.Squad.Value),
    y = Club.Squad.Value,
    label = Club.Squad.Value
  )) +
  geom_text(hjust = 0.01,
            inherit.aes = T,
            position = "identity") +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("Club") +
  ylab("Squad Value in Million") +
  ggtitle("Most valuable teams")

# Correlation between player rating and jersey number

fifa %>%
  group_by(Jersey.Number) %>%
  summarise(
    Avg.Overall = sum(Overall) / length(Jersey.Number),
    Player.Count = sum(Jersey.Number)
  ) %>%
  arrange(-Avg.Overall) %>%
  ggplot(aes(
    x = Jersey.Number,
    y = Avg.Overall,
    col = ifelse(Avg.Overall < 70, "darkgrey", "Red")
  )) +
  geom_point(position = "jitter") +
  theme(legend.position = "none") +
  geom_text_repel(aes(label = ifelse(Avg.Overall >= 70, Jersey.Number, ""))) +
  ggtitle("Correlation between player rating and jersey number")

# Most popular Player Jersey by Position Group

bind_rows(
```

```
  fifa[fifa$Position_gr == "Def",] %>%
    group_by(Position_gr, Jersey.Number) %>%
    summarise(count = n()) %>%
    arrange(-count) %>%
    head(5),
  fifa[fifa$Position_gr == "Mid",] %>%
    group_by(Position_gr, Jersey.Number) %>%
    summarise(count = n()) %>%
    arrange(-count) %>%
    head(5),
  fifa[fifa$Position_gr == "Att",] %>%
    group_by(Position_gr, Jersey.Number) %>%
    summarise(count = n()) %>%
    arrange(-count) %>%
    head(5)
) %>%
  kable() %>% collapse_rows(columns = 1) %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

#----Preparing for Modelling-----

# Creating dataset with names for testing

fifa_names <-
  subset(fifa, select = c(Name, Overall, Position, Position_gr))

# Removing Names and ungroped positions and unwanted variables

fifa$Position <- fifa$Position_gr
fifa <-
  subset(fifa, select = -c(Name, Position_gr, Club, Nationality))

# Leaving position pout of preprocessing

fifa_pos <- fifa$Position
fifa$Position <- NULL

# Converting characters to factors

fifa[sapply(fifa, is.character)] <-
  lapply(fifa[sapply(fifa, is.character)],
         as.factor)

# Centering and scaling

preProc <- preProcess(fifa, method = c("center", "scale"))
fifa <- predict(preProc, fifa)

# Convering factors to numbers (dummy variables)
```

```
dmy <- dummyVars("~ .", data = fifa, fullRank = F)
fifa <- data.frame(predict(dmy, newdata = fifa))

# Putting Position back as the first variable

fifa$Position <- fifa_pos
fifa <- fifa[, c(61, 1:60)]

# Setting seed for reproducability

seed = 1
set.seed(seed)

# Train Control function for repeated cross-validation

train.control <-
  trainControl(method = "repeatedcv",
               number = 10,
               repeats = 3)

# Splitting into train and test sets 0.9 vs 0.1 (also for names)

sample <- sample.split(fifa, SplitRatio = 0.9)
train <- subset(fifa, sample == TRUE)
test <- subset(fifa, sample == FALSE)

train_names <- subset(fifa_names, sample == TRUE)
test_names <- subset(fifa_names, sample == FALSE)

#----Modelling-----

# Setting list and names

fit <- list()
pred <- list()
pred.prob <- list()

models <-
  c("lda",
    "rda",
    "nb",
    "knn",
    "rpart",
    "C5.0",
    "rf",
    "glmnet",
    "gbm",
    "xgbLinear")

# Creating each model and predictions & probabilities for each class

for (i in c(1:length(models))) {
  set.seed(seed)
```

```
  fit[[i]] <-
    train(Position ~ .,
          data = train,
          method = models[i],0
names(fit) <- models
names(pred) <- models
names(pred.prob) <- models

#------Testing Results-----

# Testing what players were guessed incorrectly
# Setting lists

test.model <- list()
n_miss <- list()

# For each model creating a table with actual vs predicted outcome & probabilities of classes
# Also creating a table with overall number of incorrect predictions
# For each algorithm showing 10 top players (by rating) which were incorrectly classified

for (i in c(1:length(models))) {
  test.model[[i]] <-
    cbind(test_names, pred = pred[[i]], round(pred.prob[[i]], 2))
  test.model[[i]]$correct <-
    ifelse(test.model[[i]]$Position_gr == test.model[[i]]$pred, 1, 0)
  test.model[[i]]$method <- models[i]

  n_miss[[i]] <-
    test.model[[i]] %>% filter(test.model[[i]]$correct == 0) %>% summarise(n())

  head(test.model[[i]][test.model[[i]]$correct == 0,], 10) %>%
    kable() %>%
    kable_styling(
      bootstrap_options = c("striped", "bordered", "hover", "responsive"),
      full_width = FALSE,
      position = "center",
      font_size = 11
    ) %>% print()
}

# Adding model names

names(test.model) <- models
names(n_miss) <- models

# Binding all results together

testing_results <- rbindlist(test.model)
n_missed <- rbindlist(n_miss, idcol = models)
colnames(n_missed) <- c("Model", "Missed")

# Showing how top players were incorrectly classified

testing_results %>% filter(correct == "0") %>% group_by(Name, Overall, Position, Position_gr, pred) %>%
```

```
  summarise(count = n()) %>%
  arrange(desc(Overall)) %>% head(20) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Showing some examples

testing_results[Name == "L. Modrić" & correct == 0, ] %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

testing_results[Name == "L. Bonucci" & correct == 0, ] %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Showing which players been wrongly classified by most algorithms

testing_results %>% filter(correct == "0") %>% group_by(Name, Overall, Position, Position_gr) %>%
  summarise(count = n()) %>%
  arrange(desc(Overall)) %>% arrange(desc(count)) %>% head(20) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Showing some examples

testing_results[Name == "Iniesta" & correct == 0, ] %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )
```

```
temp <- data.frame(fifa_original[fifa_original$Name == "Iniesta", 29:54])  %>% t() %>% data.frame()
sort(temp[,1], decreasing = T) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

rm(temp)

# Showing total number of missed predictions for each model in descending order

n_missed %>% arrange(desc(Missed)) %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Creating summary for all resamples for all models with Accuracy and Kappa metrics

results_resamples <- resamples(fit)
summary(results_resamples, metric = c("Accuracy", "Kappa"))

# Selecting Accuracy metric to plot with Box-Plot chart

results_acc <-
  results_resamples$values[, c(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)]
colnames(results_acc) <-
  gsub("~Accuracy", "", colnames(results_acc))

ggplot(data = melt(results_acc),
       aes(x = variable, y = value)) + geom_boxplot() + theme_light()  +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Comparison of Accuracy of Models",
       x = "Model",
       y = "Accuracy")

# Preparing tables for summary function (selecting observations, predictions and probabilities)
# Creating Multi-Class summary stats from caret package

summary_tabl <- list()
summary_stats <- list()

for (i in c(1:length(models))) {
  summary_tabl[[i]] <- test.model[[i]][, 4:8]
  colnames(summary_tabl[[i]]) <-
    c("obs", "pred", "Def", "Mid", "Att")

  summary_stats[[i]] <- multiClassSummary(data = summary_tabl[[i]],
```

```
                                        lev = levels(test$Position),
                                        model = fit[[i]])

}

# Adding model names

names(summary_tabl) <- models
names(summary_stats) <- models

# Creating data frame with stats for models

stats <-
  data.frame(model = models, round(t(data.frame(summary_stats[1:10])), 4))

# Creating table with ranks for each statistic

r <- data.frame(stats)
r[, 2] <- rank(-stats[, 2])
for (i in c(3:length(stats))) {
  r[, i] <- rank(stats[, i])
}

# Showing top algorithms sorted by mean of their rank

left_join(data.frame(model = models, score = round(rowMeans(
  subset(r, select = -1)
), 2)),
stats, by = "model") %>%
  arrange(desc(score)) %>% t() %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )

# Confusion Matrices for top algorithm
confusionMatrix(data = pred[["glmnet"]], reference = test$Position)
confusionMatrix(data = pred[["glmnet"]],
                reference = test$Position,
                mode = "prec_recall")

# Examples for GLMNET
testing_results[method == "glmnet" & Position_gr == "Att" & pred == "Def",] %>%
  kable() %>%
  kable_styling(
    bootstrap_options = c("striped", "bordered", "hover", "responsive"),
    full_width = FALSE,
    position = "center",
    font_size = 11
  )
```

```
# Best model parameters

fit[["glmnet"]]$bestTune
fit[["xgbLinear"]]$bestTune
fit[["rf"]]$bestTune
```