Prediction du nombre de buts inscrit par joueur / Prediction of the number of goals scored per player

Table of contents

Introduction : Présentation du jeu de données et du sujet / Introducing the dataset and the subject

Cet ensemble de données contient les statistiques des joueurs de la saison 2024-2025 dans les cinq principaux championnats européens, provenant de FBref. / This dataset features player statistics from the 2024-2025 season across the top five European leagues, sourced from FBref.

Veuillez trouver ci-dessous la description des colonnes de ce jeu de données regroupées par catégorie / Please find below a description of the columns in this dataset grouped by category :

- Information de base par joueur / Basic Player Information

`Player`– Nom du joueur / Player's name, `Nation` – Nationalité du joueur / Player's nationality, `Pos` – Position / Position (FW, MF, DF, GK), `Squad` – Nom du club / Club name, `Comp` –

Championnat / League, `Age` – Âge du joueur / Age of the player, `Born` – Année de naissance / Year of birth

- Temps de jeu et apparitions / Playing time and Appearances

`MP` – Matches jouées / Matches played, `Starts` – Parties commencées / Games started, `Min` – Minutes jouées / Minutes played, `90s` – Nombre de matchs jouées en entier / Number of full 90-minute matches played

- Stats offensives / Attacking stats

`Gls` – Buts marquées / Goals scored, `Ast` – Passes décisives apportées / Assists provided, `G+A` – Buts + Passes décisives / Goals + Assists, `xG` – Buts Attendus / Expected goals
`xAG` – Passes décisives attendus / Expected assists, `npxG` –Buts attendus sans les penaltys / Non-penalty expected goals, `G-PK` – Buts saufs penaltys / Goals excluding penalties

- Stats défensives / Defensive stats

`Tkl` – Total de tacles / Total tackles, `TklW` – Tacles gagnés /Tackles won, `Blocks` – Blocs réalisées / Blocks made, `Int` – Interceptions / Interceptions, `Tkl+Int` – Tacles et interceptions combinés / Combined tackles and interceptions, `Clr` – Dégagements / Clearances, `Err` – Erreurs menant à un but / Errors leading to goals

- Stats de passes et de créativité / Passing and Creativity stats

`PrgP` – Passes progressive / Progressive passes, `PrgC` – Phase de portage du ballon progressive / Progressive carries, `KP` – Passes clés (passes menant à un tir) / Key passes (passes leading to a shot), `Cmp%_stats_passing` – Pourcentage de passes réussis / Pass completion percentage, `Ast_stats_passing` – Passes décisives / Assists, `xA` – Passes décisives attendus / Expected assists, `PPA` – Passes dans la surface de réparation / Passes into the penalty area

- Stats du gardien / Goalkeeping stats

`GA` – Buts concédés / Goals conceded, `Saves` – Arrêts effectués / Saves made, `Save%` – Pourcentage d'arrêt / Save percentage, `CS` – Pas de buts pris dans le match / Clean sheets
`CS%` – Pourcentage de matchs sans buts encaissés / Clean sheet percentage, `PKA` – Penaltys rencontrés / Penalties faced, `PKsv` – Penalties arrétés / Penalty saves

- Possession et Contrôle du ballon / Possession and Ball control

`Touches` – Total de touches de balles / Total touches of the ball, `Carries` – Total de ballon portés / Total ball carries, `PrgR` – Courses progressive (en portant le ballon vers l'avant de façon significative) / Progressive runs (carries moving the ball forward significantly), `Mis` – Mauvais contrôle du ballon / Miscontrols, `Dis` – Perte du ballon / Times dispossessed

- Statistiques diverses / Miscellaneous stats

`CrdY` – Cartons jaunes / Yellow cards, `CrdR` – Carton rouge / Red cards, `PKwon` – Penalties gagnées / Penalties won, `PKcon` – Penalties concédés / Penalties conceded, `Recov` – Ballon récupérés / Ball recoveries

On va dans un 1er temps importer les librairies puis les données. / First, we'll import the libraries, then the data.

On va ensuite prédire le nombre de buts inscrits par joueur / We will then predict the number of goals scored per player.

Traitement des données / Data processing

```
# Nettoyage des noms de colonnes / Cleaning up column names
names(data) <- make.names(names(data), unique = TRUE)

# Exclure les gardiens / Exclude goalkeepers
data <- data %>% filter(Pos != "GK")

# Remplacer les NA dans les colonnes numériques par 0 / Replace NA in numeric columns
# with 0
data <- data %>%
  mutate(across(where(is.numeric), ~replace_na(.x, 0)))

# Colonnes numériques à utiliser sauf exclusions / Numerical columns to be
# used unless excluded
cols_to_exclude <- c("Rk", "Age", "Born", "G.A", "G.PK", "G.A.PK", "G.Sh",
                     "G.xG","PK","np.G.xG", "G.SoT", "PK_stats_shooting")
num_cols <- data %>%
  select(where(is.numeric)) %>%
  select(-any_of(cols_to_exclude)) %>%
  colnames()
```

Méthode : Modèles de régression linéaire / Method : Linear regression models

```
# Créer le jeu de données pour le modèle / Create the dataset for the model
data_model <- data %>%
  select(Gls, all_of(num_cols)) %>%
  filter(!is.na(Gls))

# Créer le modèle / Create the model
```

```r
model <- lm(Gls ~ ., data = data_model)

# Préparer les données pour prédiction (même structure que data_model sans Gls)
# Prediction for the entire dataset (same structure as data_model without Gls)
predict_data <- data %>% select(all_of(num_cols))

# Créer un nouveau dataset avec les infos utiles + prédiction
# Create a new dataset with useful information + prediction
data_with_preds <- data %>%
  mutate(Gls_pred_linear = predict(model, newdata = predict_data),
         Gls_pred_linear = round(Gls_pred_linear, 2)) %>%
  select(Player, Nation, Pos, Squad, Gls, Gls_pred_linear)

# Afficher les premières lignes / Display first rows
head(data_with_preds)
```

```
# A tibble: 6 x 6
  Player            Nation  Pos   Squad           Gls Gls_pred_linear
  <chr>             <chr>   <chr> <chr>         <dbl>           <dbl>
1 Max Aarons        eng ENG DF    Bournemouth       0            0.1
2 Max Aarons        eng ENG DF,MF Valencia          0            0.13
3 Rodrigo Abajas    es ESP  DF    Valencia          0            0
4 James Abankwah    ie IRL  DF,MF Udinese           0           -0.03
5 Keyliane Abdallah fr FRA  FW    Marseille         0            0.16
6 Yunis Abdelhamid  ma MAR  DF    Saint-Étienne     0            0.12
```

Interprétation : Le modèle a une capacité de prédiction satisfaisant comme le montre le R^2 ajusté de 0.90. Des variables significatives semblent se dégager comme le nombre de tirs cadrés (SoT : 0.23), celui du total des penalties obtenus (PKWon : -0.16) et le nombre de passes décisives (Ast : -0.14). Néanmoins, certaines variables semblent fortement liées entre elles ce qui peut biaiser la prédiction du modèle. De plus, un nombre important de variable semblent non significatives. De ce fait, on va tester des modèles plus adaptés dans la régularisation de ce type de données afin d'observer si ces algorithmes sont plus efficaces.

Interpretation: The model has a satisfactory predictive capacity, as shown by the adjusted R^2 of 0.90. Significant variables seem to emerge, such as the number of shots on target (SoT: 0.23), total penalties won (PKWon: -0.16) and the number of assists (Ast: -0.14). Nevertheless, some variables appear to be strongly interrelated, which may bias the model's prediction. In addition, a significant number of variables appear to be insignificant. As a result, we'll be testing more suitable models for regularizing this type of data, to see whether.

Méthode : Modèles de régression linéaire régularisée / Method : Regularized linear regression models

On va créer un fonction pour les 3 types de modèle de régression régularisée : Ridge, Lasso, Elastic Net. / We will create a function for the 3 types of regularized regression models: Ridge, Lasso, Elastic Net.

```
add_glmnet_prediction <- function(data_with_preds, data, y_var, x_vars,
                                  alpha_val, pred_col_name, seed = 123) {

  # Variable cible / Target variable
  y <- data[[y_var]]

  # Variables explicatives sous forme de matrice
  # Explanatory variables as matrix
  X <- data %>% select(all_of(x_vars)) %>% as.matrix()

  # Sélection du lambda optimal par validation croisée
  # Cross-validation to find optimal lambda
  set.seed(seed)  # Pour reproductibilité / For reproducibility
  cv_model <- cv.glmnet(X, y, alpha = alpha_val)

  # Lambda optimal / Optimal lambda
  best_lambda <- cv_model$lambda.min
  cat("Lambda optimal pour", pred_col_name, ":", best_lambda, "\n")

  # Réentraînement du modèle avec le meilleur lambda
  # Retrain model with best lambda
  best_model <- glmnet(X, y, alpha = alpha_val, lambda = best_lambda)

  # Prédictions sous forme de vecteur et arrondi
  # Predictions as numeric vector, rounded
  predictions <- predict(best_model, newx = X) %>%
    as.numeric() %>%
    round(2)

  # Ajout des prédictions au dataset / Add predictions to the dataset
  data_with_preds[[pred_col_name]] <- predictions

  # Retourner le nouveau tableau enrichi / Return the updated dataset
  return(data_with_preds)
}
```

On va ensuite effectuer ces prédictions. / We will then make these predictions.

```
# Ajouter les prédictions Ridge / Add Ridge predictions
data_with_preds <- add_glmnet_prediction(
  data_with_preds = data_with_preds,
  data = data,
  y_var = "Gls",
  x_vars = num_cols,
  alpha_val = 0,              # alpha = 0 → Ridge
  pred_col_name = "Gls_pred_ridge"
)
```

```
Lambda optimal pour Gls_pred_ridge : 0.3098882
```

```
# Ajouter les prédictions Lasso / Add Lasso predictions
data_with_preds <- add_glmnet_prediction(
  data_with_preds,
  data,
  "Gls",
  num_cols,
  alpha_val = 1,              # alpha = 1 → Lasso
  pred_col_name = "Gls_pred_lasso"
)
```

```
Lambda optimal pour Gls_pred_lasso : 0.09033406
```

```
# Ajouter les prédictions Elastic Net / Add Elastic Net predictions
data_with_preds <- add_glmnet_prediction(
  data_with_preds,
  data,
  "Gls",
  num_cols,
  alpha_val = 0.5,            # alpha = 0.5 → Elastic Net (mix Ridge + Lasso)
  pred_col_name = "Gls_pred_elastic"
)
```

```
Lambda optimal pour Gls_pred_elastic : 0.06492875
```

```
# Afficher les premières lignes / Display first rows
head(data_with_preds)
```

```
# A tibble: 6 x 9
  Player  Nation Pos   Squad   Gls Gls_pred_linear Gls_pred_ridge Gls_pred_lasso
  <chr>   <chr>  <chr> <chr> <dbl>           <dbl>          <dbl>          <dbl>
1 Max Aa~ eng E~ DF    Bour~     0            0.1            0.02           0.05
2 Max Aa~ eng E~ DF,MF Vale~     0            0.13          -0.01           0.05
3 Rodrig~ es ESP DF    Vale~     0            0             -0.07           0.05
4 James ~ ie IRL DF,MF Udin~     0           -0.03          -0.04           0.05
5 Keylia~ fr FRA FW    Mars~     0            0.16           0.06           0.05
6 Yunis ~ ma MAR DF    Sain~     0            0.12           0.09           0.05
# i 1 more variable: Gls_pred_elastic <dbl>
```

Interprétation : Les valeurs de lambda optimales indiquent une régularisation modérée : Ridge applique une réduction générale des coefficients, tandis que Lasso et Elastic Net utilisent une pénalisation plus légère, suggérant peu de variables redondantes et un faible risque de surapprentissage.

Interpretation : Optimal lambda values indicate moderate regularization: Ridge applies a general reduction in coefficients, while Lasso and Elastic Net use lighter penalization, suggesting few redundant variables and a low risk of overlearning.

Comparatif / Comparative

````
::: {.cell}

```{.r .cell-code}
# Calcul des erreurs pour chaque modèle / Error calculation for each model

# Modèle linéaire / Linear Model
mae_linear <- mae(data_with_preds$Gls, data_with_preds$Gls_pred_linear)
rmse_linear <- rmse(data_with_preds$Gls, data_with_preds$Gls_pred_linear)
r2_linear <- R2(data_with_preds$Gls_pred_linear, data_with_preds$Gls)

# Régression Ridge / Ridge Regression
mae_ridge <- mae(data_with_preds$Gls, data_with_preds$Gls_pred_ridge)
rmse_ridge <- rmse(data_with_preds$Gls, data_with_preds$Gls_pred_ridge)
r2_ridge <- R2(data_with_preds$Gls_pred_ridge, data_with_preds$Gls)

# Régression Lasso / Lasso Regression
mae_lasso <- mae(data_with_preds$Gls, data_with_preds$Gls_pred_lasso)
rmse_lasso <- rmse(data_with_preds$Gls, data_with_preds$Gls_pred_lasso)
r2_lasso <- R2(data_with_preds$Gls_pred_lasso, data_with_preds$Gls)
```
````

```
# Régression Elastic Net / Elastic Net Regression
mae_elastic <- mae(data_with_preds$Gls, data_with_preds$Gls_pred_elastic)
rmse_elastic <- rmse(data_with_preds$Gls, data_with_preds$Gls_pred_elastic)
r2_elastic <- R2(data_with_preds$Gls_pred_elastic, data_with_preds$Gls)

#  Résumé des performances / Summary table
performance_comparison <- data.frame(
  Modèle = c("Linéaire", "Ridge", "Lasso", "Elastic Net"),
  MAE = c(mae_linear, mae_ridge, mae_lasso, mae_elastic),
  RMSE = c(rmse_linear, rmse_ridge, rmse_lasso, rmse_elastic),
  R2 = c(r2_linear, r2_ridge, r2_lasso, r2_elastic)
)

# Affichage du tableau de comparaison / Display comparison table
print(performance_comparison)
```

::: {.cell-output .cell-output-stdout}

```
       Modèle        MAE       RMSE        R2
1    Linéaire 0.61585338 0.96544596 0.9029388
2       Ridge 0.28911798 0.45108566 0.9800869
3       Lasso 0.05806033 0.09024848 0.9999997
4 Elastic Net 0.05886598 0.09574072 0.9994226
```

:::

```{.r .cell-code}
# Réutiliser ou créer le tableau de performances / Reuse or create the performance chart
performance_comparison <- data.frame(
  Modèle = c("Linéaire", "Ridge", "Lasso", "Elastic Net"),
  MAE = c(mae_linear, mae_ridge, mae_lasso, mae_elastic),
  RMSE = c(rmse_linear, rmse_ridge, rmse_lasso, rmse_elastic),
  R2 = c(r2_linear, r2_ridge, r2_lasso, r2_elastic)
)

#  Convertir en format long pour ggplot2 / Convert to long format for ggplot2
performance_long <- performance_comparison %>%
  pivot_longer(cols = c(MAE, RMSE, R2), names_to = "Métrique", values_to = "Valeur")
```

```
# Graphique comparatif / Comparative performance plot
ggplot(performance_long, aes(x = Modèle, y = Valeur, fill = Modèle)) +
  geom_col(position = "dodge") +
  facet_wrap(~ Métrique, scales = "free_y") +
  labs(title = "Comparaison des performances des modèles / Model performance comparison",
       x = "Modèle / Models",
       y = "Valeur de la métrique / Metric value") +
  theme_minimal() +
  theme(legend.position = "none",
        text = element_text(size = 8))
```

::: {.cell-output-display}

:::
:::

Interprétation : Les modèles Lasso et Elastic Net offrent les meilleures performances, avec des erreurs très faibles (MAE < 0.06, RMSE < 0.1) et un R^2 quasi parfait, indiquant un excellent pouvoir prédictif. Le modèle Ridge est également performant, tandis que le modèle linéaire est nettement moins précis. Néanmoins, les performances très élevées de Lasso et Elastic Net suggèrent un possible surapprentissage. Une validation sur un jeu de test séparé est nécessaire pour confirmer leur capacité de généralisation.

Interpretation : The Lasso and Elastic Net models offer the best performance, with very low errors (MAE < 0.06, RMSE < 0.1) and near-perfect R^2, indicating excellent predictive power. The Ridge model also performs well, while the linear model is considerably less accurate. Nevertheless, the very high performance of Lasso and Elastic Net suggests possible overlearning. Validation on a separate test set is required to confirm their generalizability.

5. Validation croisée / Cross-validation

On va d'abord séparer les données en jeu d'entrainement et test (80/20). / First, we'll separate the data into training and test sets (80/20).

```
set.seed(42)  # Reproductibilité / Reproducibility
train_index <- createDataPartition(data$Gls, p = 0.8, list = FALSE)

train_data <- data[train_index, ]
test_data  <- data[-train_index, ]

# Variables explicatives / Features matrix
X_train <- train_data %>% select(all_of(num_cols)) %>% as.matrix()
```

```r
X_test  <- test_data %>% select(all_of(num_cols)) %>% as.matrix()

# Cible / Target
y_train <- train_data$Gls
y_test  <- test_data$Gls
```

On va ensuite créer un fonction d'évaluation de ces modèles. / We'll then create an evaluation function for these models.

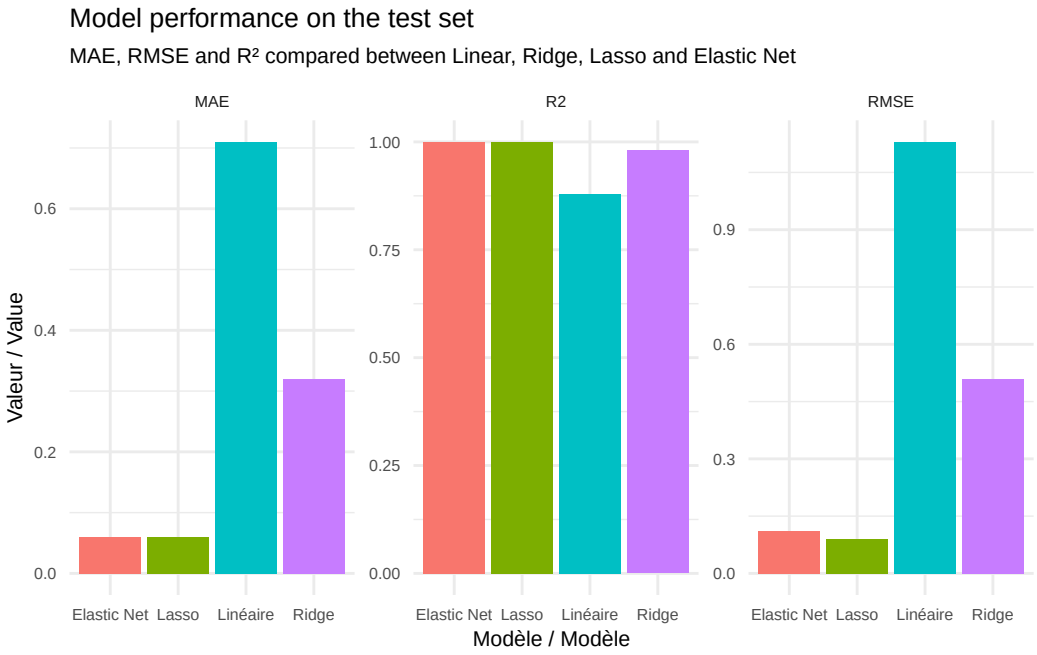
```r
eval_glmnet_model <- function(X_train, y_train, X_test,
                              y_test, alpha_val, model_name) {
  set.seed(123)
  cv_model <- cv.glmnet(X_train, y_train, alpha = alpha_val)
  best_lambda <- cv_model$lambda.min

  model <- glmnet(X_train, y_train, alpha = alpha_val, lambda = best_lambda)
  preds <- predict(model, newx = X_test) %>% as.numeric()

  data.frame(
    Modèle = model_name,
    MAE = mae(y_test, preds),
    RMSE = rmse(y_test, preds),
    R2 = R2(preds, y_test)
  )
}
```

On va logiquement les évaluer par la suite. / We will then logically evaluate them.

```r
#  Modèle linéaire / Linear Model
lm_model <- lm(Gls ~ ., data = train_data %>% select(Gls, all_of(num_cols)))
lm_preds <- predict(lm_model, newdata = test_data)
lm_metrics <- data.frame(
  Modèle = "Linéaire",
  MAE = mae(y_test, lm_preds),
  RMSE = rmse(y_test, lm_preds),
  R2 = R2(lm_preds, y_test)
)

#  Fonction d'évaluation pour glmnet /  Evaluation function for glmnet
eval_glmnet_model <- function(X_train, y_train, X_test,
                              y_test, alpha_val, model_name) {
```

```r
  set.seed(123)
  cv_model <- cv.glmnet(X_train, y_train, alpha = alpha_val)
  best_lambda <- cv_model$lambda.min
  model <- glmnet(X_train, y_train, alpha = alpha_val, lambda = best_lambda)
  preds <- predict(model, newx = X_test) %>% as.numeric()

  data.frame(
    Modèle = model_name,
    MAE = mae(y_test, preds),
    RMSE = rmse(y_test, preds),
    R2 = R2(preds, y_test)
  )
}

#  Évaluation des modèles pénalisés / Penalized models
ridge_metrics   <- eval_glmnet_model(X_train, y_train,
                                     X_test, y_test, alpha_val = 0,
                                     model_name = "Ridge")
lasso_metrics   <- eval_glmnet_model(X_train, y_train,
                                     X_test, y_test, alpha_val = 1,
                                     model_name = "Lasso")
elastic_metrics <- eval_glmnet_model(X_train, y_train,
                                     X_test, y_test, alpha_val = 0.5,
                                     model_name = "Elastic Net")

# Regrouper les résultats / Combine all results
results <- bind_rows(lm_metrics, ridge_metrics, lasso_metrics, elastic_metrics)

# Arrondir à 2 décimales / Round to 2 decimals
results <- results %>%
  mutate(across(c(MAE, RMSE, R2), ~ round(.x, 2)))

# Format long pour ggplot / Long format for ggplot
results_long <- results %>%
  pivot_longer(cols = c(MAE, RMSE, R2),
               names_to = "Métrique", values_to = "Valeur")

# Comparaison graphique / Visual model performance comparison
ggplot(results_long, aes(x = Modèle, y = Valeur, fill = Modèle)) +
  geom_col(position = "dodge") +
  facet_wrap(~ Métrique, scales = "free_y") +
  labs(
```

```
    title = "Model performance on the test set",
    subtitle = "MAE, RMSE and R² compared between Linear, Ridge, Lasso and Elastic Net",
    x = "Modèle / Modèle",
    y = "Valeur / Value"
  ) +
  theme_minimal() +
  theme(legend.position = "none", text = element_text(size = 8))
```



Conclusion

Les modèles régularisés, notamment Lasso et Elastic Net, présentent des performances nettement supérieures au modèle linéaire, avec une erreur minimale et un R^2 1 sur le jeu de test. Bien que ces résultats aient été obtenus après validation croisée et test indépendant, ce qui limite le risque de surapprentissage. Néanmoins, leur précision quasi parfaite invite à rester prudent. Ces modèles apparaissent toutefois comme les plus adaptés pour ce jeu de données, grâce à leur capacité à régulariser efficacement sans perte de performance.

Regularised models, notably Lasso and Elastic Net, performed significantly better than the linear model, achieving minimal error and an R^2 value of approximately 1 on the test set. These results were obtained after cross-validation and independent testing, which limits the risk of overlearning. Nevertheless, their near-perfect accuracy invites caution. These models appear to be the most suitable for this dataset thanks to their ability to regularise efficiently without loss of performance.