Projet

Kenza SKALLI / Andy LAM

Chess Endgame Database for White King and White Rook against Black King (KRK) - **Black-to-move** Positions Drawn or Lost in N Moves

Notre base de données est de taille : **28 056 positions avec labels**

Auteur de la base de données: ***Michael Bain and Arthur van Hoff at the Turing Institute, Glasgow, UK***

Display :

```
import pandas as pd
data =  pd.read_csv('krkopt.data', sep=",",engine='python', header=None)
print(data)
```

```
       0  1  2  3  4  5        6
0      a  1  b  3  c  2     draw
1      a  1  c  1  c  2     draw
2      a  1  c  1  d  1     draw
3      a  1  c  1  d  2     draw
4      a  1  c  2  c  1     draw
...   .. .. .. .. .. ..      ...
28051  b  1  g  7  e  5  sixteen
28052  b  1  g  7  e  6  sixteen
28053  b  1  g  7  e  7  sixteen
28054  b  1  g  7  f  5  sixteen
28055  b  1  g  7  g  5  sixteen

[28056 rows x 7 columns]
```

Remarques :

- Nombre de positions totales possible : **223 944 positions**
- **[0 ,1] : Roi Blanc, [2 ,3] : Tour Blanche, [4 ,5] : Roi Noir,**
- Base de données **qualitative**
- **la profondeur pour gagner** = nombre de coups pour gagner

Description :

	Number	pourcent
fourteen	4553	16.228258
thirteen	4194	14.948674
twelve	3597	12.820787
eleven	2854	10.172512
draw	2796	9.965783
fifteen	2166	7.720274
ten	1985	7.075135
nine	1712	6.102082
eight	1433	5.107642
seven	683	2.434417
six	592	2.110066
five	471	1.678785
sixteen	390	1.390077
two	246	0.876818
four	198	0.705731
three	81	0.288708
one	78	0.278015
zero	27	0.096236

Première transformation :

	WhiteKing	WhiteRook	BlackKing	Results
0	a1	b3	c2	draw
1	a1	c1	c2	draw
2	a1	c1	d1	draw
3	a1	c1	d2	draw
4	a1	c2	c1	draw
...	...	...	...	...
28051	b1	g7	e5	sixteen
28052	b1	g7	e6	sixteen
28053	b1	g7	e7	sixteen
28054	b1	g7	f5	sixteen
28055	b1	g7	g5	sixteen

28056 rows × 4 columns

Deuxième description :

Roi Blanc et Tour Blanche :

```
c1    3596
b1    3596
d1    3596
d2    3410
d3    3410
c2    3410
a1    1878
c3    1720
d4    1720
b2    1720
Name: WhiteKing, dtype: int64
10
```

```
e2    455
e3    455
e1    454
e4    454
e5    453
     ...
d2    403
d3    402
c1    400
b1    399
d1    398
Name: WhiteRook, Length: 64, dtype: int64
64
```

Roi Noir :

```
h2    620
f3    620
h6    620
g4    620
f4    620
     ...
d2    248
c1    248
b2    220
c3    220
c2    124
Name: BlackKing, Length: 64, dtype: int64
64
```

Des probabilités :

Analyse de la position A1 du Roi Blanc

	Nombre	Probabilité
draw	200	0.106496
zero	0	0.000000
one	0	0.000000
two	0	0.000000
three	0	0.000000
four	0	0.000000
five	0	0.000000
six	0	0.000000
seven	12	0.006390
eight	19	0.010117
nine	22	0.011715
ten	74	0.039404
eleven	137	0.072950
twelve	118	0.062833
thirteen	219	0.116613
fourteen	329	0.175186
fifteen	470	0.250266
sixteen	278	0.148030

Analyse de la position C6 de la Tour Blanche

	Nombre	Probabilité
draw	48	0.110345
zero	0	0.000000
one	1	0.002299
two	4	0.009195
three	1	0.002299
four	0	0.000000
five	5	0.011494
six	9	0.020690
seven	2	0.004598
eight	23	0.052874
nine	44	0.101149
ten	43	0.098851
eleven	44	0.101149
twelve	31	0.071264
thirteen	56	0.128736
fourteen	40	0.091954
fifteen	57	0.131034
sixteen	27	0.062069

A/ Premier Modèle avec Sklearn :

Encodage :

```
# Lettre = {'a':10,'b':20,'c':30,'d':40,'e':50,'f':60,'g':70,'h':80}
Lettre = {'a':0.1,'b':0.2,'c':0.3,'d':0.4,'e':0.5,'f':0.6,'g':0.7,'h':0.8}
```

Avant

	WhiteKing	WhiteRook	BlackKing	Results
0	a1	b3	c2	draw
1	a1	c1	c2	draw
2	a1	c1	d1	draw
3	a1	c1	d2	draw
4	a1	c2	c1	draw
...	...	...	...	...
28051	b1	g7	e5	sixteen
28052	b1	g7	e6	sixteen
28053	b1	g7	e7	sixteen
28054	b1	g7	f5	sixteen
28055	b1	g7	g5	sixteen

28056 rows × 4 columns

Après

	WhiteKing	WhiteRook	BlackKing	Results
0	1.1	3.2	2.3	draw
1	1.1	1.3	2.3	draw
2	1.1	1.3	1.4	draw
3	1.1	1.3	2.4	draw
4	1.1	2.3	1.3	draw
...	...	...	...	...
28051	1.2	7.7	5.5	sixteen
28052	1.2	7.7	6.5	sixteen
28053	1.2	7.7	7.5	sixteen
28054	1.2	7.7	5.6	sixteen
28055	1.2	7.7	5.7	sixteen

28056 rows × 4 columns

Meilleur modèle : Arbre de décisions

```
X = data2.iloc[:, [0,1,2]]
y = data2.iloc[:,3]
```

```
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=0)
```

Arbre de décisions avec critère Gini	Arbre de décisions avec critère Entropy	Random Forest avec critère Gini
tree = DecisionTreeClassifier(criterion='gini', splitter='best') tree.fit(X_train, y_train) y_pred = tree.predict(X_test)	tree = DecisionTreeClassifier(criterion='entropy') tree.fit(X_train, y_train) y_pred = tree.predict(X_test)	rdforst = RandomForestClassifier() rdforst = rdforst.fit(X_train, y_train) y_pred = tree.predict(X_test)
Accuracy: 0.68 Erreur quadratique: 1.18	Accuracy: 0.67 Erreur quadratique: 0.22	Accuracy: 0.67 Erreur quadratique: 0.22

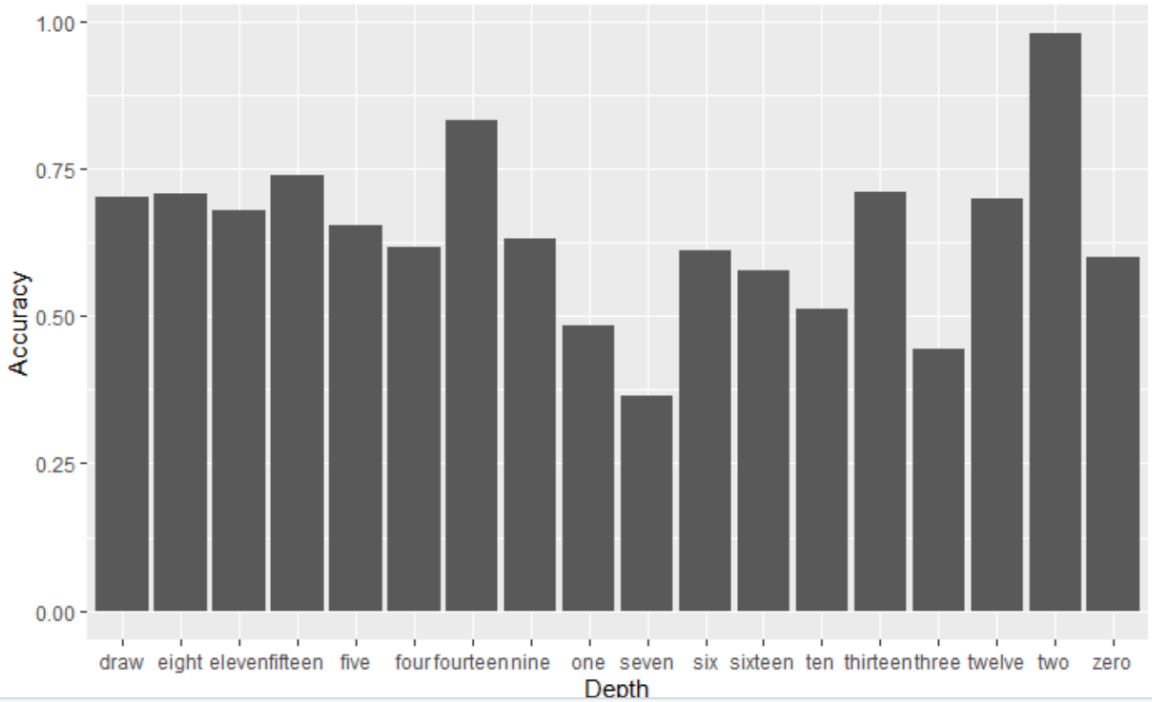
Autre modèles :

K Plus Proches Voisins	Perceptron	Régression logistique
Accuracy: 0.49 Erreur quadratique: 803.63	Accuracy: 0.21 Erreur quadratique: 12356.33	Accuracy: 0.25 Erreur quadratique: 5816.52

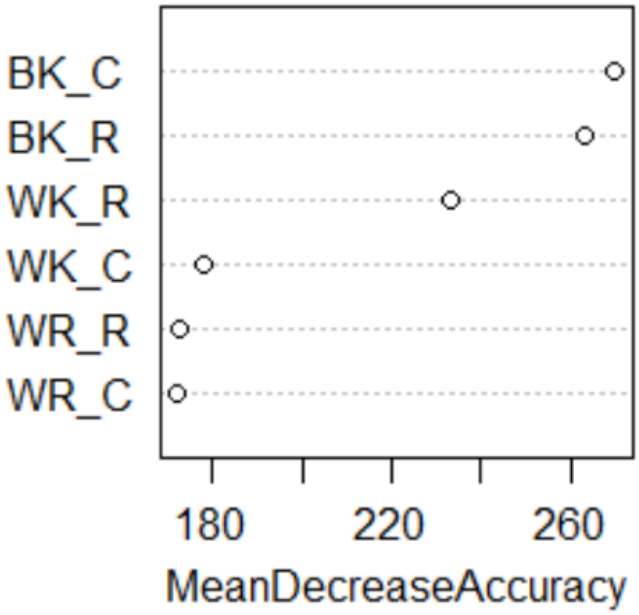
Support vector machines		
Accuracy: 0.33 Erreur quadratique: 2762.05		

B/ RandomForest

B.1/Précision

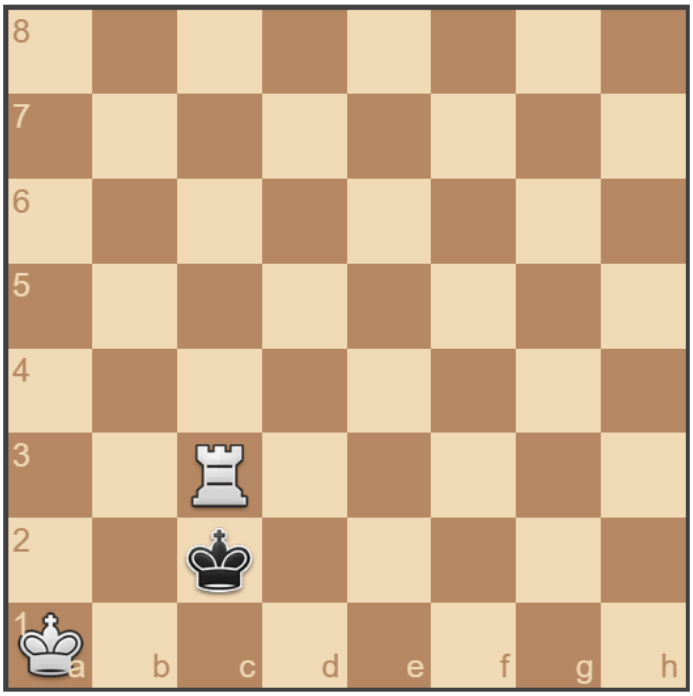


B.2/ Importance des features



Représentation fen : utilisation du package rchess

WK_C <chr>	WK_R <int>	WR_C <chr>	WR_R <int>	BK_C <chr>	BK_R <int>
a	1	c	3	c	2




```
[1] "codage fen: 8/8/8/8/8/2R5/2k5/0K7 w KQkq - 0 1"
[1] "D'après l'algorithme randomforest le joueur blanc peut gagner en:
ten coups"
[1] "Il peut réellement gagner en: draw"
```