

Machine Learning Engineer Nanodegree

Capstone Project

Chess (King-Rook vs. King)

Pothuru Yaswanth

February 4th 2019

I. Definition

Project overview

Rook endgames are the most common type of endgames there is in the game of chess. These endgames take place in about in 50% of all the games. The main idea is checkmate of black king with white king and white rook. This should be done in less than 16 steps.

Chess endgames are complex domains which are enumerable. The game theoretic values stored denote whether or not positions are won for either side, or include also depth of wins (number of moves).

The relevant paper related to this M. Bain. "Learning optimal chess strategies", ILP 92: ICOT TM-1182, S. Muggleton, Institute for New Generation Computer Technology, Tokyo, Japan. [\[Web Link\]](#) .

Problem Statement:

The main aim of my project is to predict the outcome of the king-rook vs. king endgame. For this I selected the dataset from UCI

(<https://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King%29>)

So my goal is to predict the outcome of the game whether the white king will win the game in less than 16 moves or not. Here I am using classification models to find the accuracy of each model and select the model which will have high accuracy. Here the input parameter is given in the form of training data.

The tasks involved in it are:

1. Download the data
2. Cleaning the data and removing the null data points, duplicated data rows.
3. Visualizing the data.

4. Split the data and train and test which classifier performs good on the dataset based on the evaluation metric we have chosen.
5. Choose the best classifier that gives the best accuracy scores.

Metrics:

Here I will use accuracy score as evaluation metric. I will be predicting the accuracy score for the selected model. We will select accuracy because accuracy gives us the better and easy way to understand how the chosen model is predicting the output that is how accurately the chosen model is predicting the output. From the information of the classes that are given below in the attributes section it is clear that the classes are distributed over a very wide range which makes accuracy the metric that can be better used here. The model with high accuracy score will be the best model out of the chosen models. The selected model's accuracy should also be more than the accuracy of the benchmark model. Finally we will select the model whose accuracy is more than the all the other models.

For evaluating the accuracy we have the following formula:

Accuracy Score = $\frac{TP+TN}{TP+FP+FN+TN}$

True Positives: are the values which are correctly predicted as positives

True Negatives: are the values which are correctly classified as negatives.

False Positives: are the values which are wrongly classified as positives. These are also type-1 errors.

False Negatives: are the values which are wrongly classified as negatives. These are also called as type-2 errors.

II. Analysis

Data Exploration

The dataset will comprise of multivariate data. Here the dataset will have categorical values and integer values. There will be total of 28056 instances, and the total number of attributes is 6. The dataset here is a multivariate.

For the best result we will split the data into training set and testing set. On a whole we will assign 70% of the data to training set and 30% of the data to testing set.

Attribute information:

White king file (column)

White king rank (row)

White rook file

White rook rank

Black king file

Black king rank

Optimal depth-of-win for white in 0 to 16 moves otherwise drawn

{Draw, 0,1,2...16 }

The number of examples for each class labels is as follows:

Fourteen 4553

Thirteen 4194

Twelve 3597

Eleven 2854

Draw 2796

Fifteen 2166

Ten 1985

Nine 1712

Eight 1433

Seven 683

Six 592

Five 471

Sixteen 390

Two 246

Four 198

Three 81

One 78

Zero 27

The head of the data is as follows:

```
In [4]: # using head() print the first five columns of the data
df.head()
```

Out[4]:

	wkf	wkr	wrf	wrr	bkf	bkr	class
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

The statistical information regarding the dataset is as follows

	wkr	wrr	bkr
count	28056.000000	28056.000000	28056.000000
mean	1.854006	4.512404	4.451811
std	0.926414	2.282723	2.248387
min	1.000000	1.000000	1.000000
25%	1.000000	3.000000	3.000000
50%	2.000000	5.000000	4.000000
75%	2.000000	6.000000	6.000000
max	4.000000	8.000000	8.000000

Exploratory Visualization

Heat map:

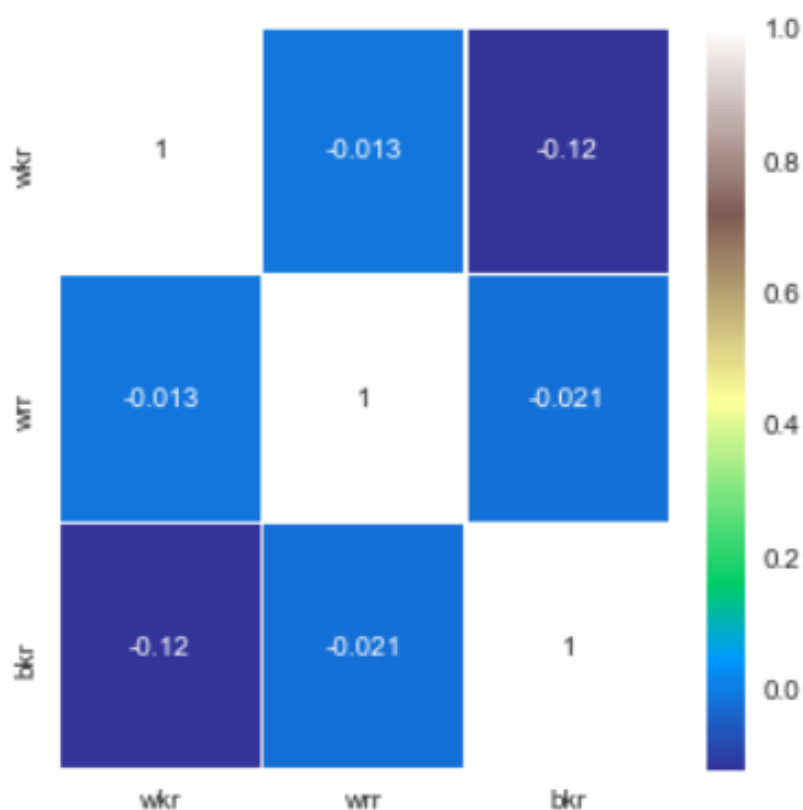
The heat map is a 2-D representation of data in which values are represented by colours. A simple heat map provides the immediate visual summary of information. More elaborate heat maps allow the user to understand complex data.

Here we will know the correlation the attributes like wkr (white king rank), wrr (white rook rank), bkr (Black king rank)

By using the below code we will generate the heat map between the attributes and try to deduce the correlation between the attributes. We will use the following code to do so.

```
In [32]: # import matplotlib.pyplot and seaborn
import matplotlib.pyplot as plt
import seaborn as sns
#heatmap to find the correlation.
sns.heatmap(df.corr(),annot=True,cmap='terrain',linewidths=0.1)
fig=plt.gcf()
fig.set_size_inches(5,5)
plt.show()
```

By executing the above code we get the following heat map:



(For a more clear heat map refer to the Code shown in the above that was executed in the project)

From the above heat map we can clearly say that there is no much correlation between the attributes.

Algorithms and techniques:

Here mainly we will use three algorithms.

1. Logistic Regression.(Benchmark Model)
2. Random Forest.
3. Support Vector Machine.

Logistic Regression:

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts $P(Y=1)$ as a function of X .

Advantages:

Because of its efficient and straightforward nature, doesn't require high computation power, easy to implement, easily interpretable, used widely by data analyst and scientist. Also, it doesn't require scaling of features. Logistic regression provides a probability score for observations.

Disadvantages:

Logistic regression is not able to handle a large number of categorical features/variables. It is vulnerable to over fitting. Also, can't solve the non-linear problem with the logistic regression that is why it requires a transformation of non-linear features. Logistic regression will not perform well with independent variables that are not correlated to the target variable and are very similar or correlated to each other.

Parameters:

```
Class sklearn.linear_model.LogisticRegression (penalty='l2', dual=False,
tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None,
random_state=None, solver='warn', max_iter=100, multi_class='warn',
verbose=0, warm_start=False, n_jobs=None)
```

The application is classification oriented . So, techniques that are used are taken from Classification techniques.

Random Forest:

Tree models are known to be high variance, low bias models. In consequence, they are prone to over fit the training data. This is catchy if we recapitulate what a tree model does if we do not prune it or introduce early stopping criteria like a minimum number of instances per leaf node. Well, it tries to split the data along the features until the instances are pure regarding the value of the target feature, there are no data left, or there are no features left to spit the dataset on. If one of the above holds true, we grow a leaf node. The consequence is that the tree model is grown to the maximal depth and therewith tries to reshape the training data as precise as possible which can easily lead to over fitting. Another drawback of classical tree models like the (ID3 or CART) is that they are relatively unstable. This instability can lead to the situation that a small change in the composition of the dataset leads to a completely different tree model.

Parameters:

```
Class sklearn.ensemble.RandomForestClassifier (n_estimators='warn',  
criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1,  
min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,  
min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True,  
oob_score=False, n_jobs=None, random_state=None, verbose=0,  
warm_start=False, class_weight=None
```

Advantages:

1. Reduction in over fitting: by averaging several trees, there is a significantly lower risk of over fitting.
2. Less variance: By using multiple trees, you reduce the chance of stumbling across a classifier that doesn't perform well because of the relationship between the train and test data.

Disadvantages:

1. It takes more time to train samples.

Support vector machine (svm):

The objective of the support vector machine algorithm is to find a hyper plane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyper planes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.


Advantages:

SVM's are very good when we have no idea on the data. Works well with even unstructured and semi structured data like text, Images and trees. The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem. Unlike in neural networks, SVM is not solved for local optima. It scales relatively well to high dimensional data. SVM models have generalization in practice; the risk of over fitting is less in SVM.

Disadvantages:

Choosing a good kernel function is not easy. Long training time for large datasets. Difficult to understand and interpret the final model, variable weights and individual impact. Since the final model is not so easy to see, we can not do small calibrations to the model hence it's tough to incorporate our business logic.

Parameters:

```
class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=1, decision_function_shape='ovr', random_state=None) 
```

Benchmark Model:

For this problem we will choose Logistic Regression model as the benchmark model.

By applying this Gaussian Logistic Regression we achieved an accuracy of 0.358 that is 35.8%.

Now we will try and achieve better accuracy than this model by using the above mentioned classification models.

III. Methodology

Data Pre-processing:

In this step we will pre process the data. Data pre-processing is considered to be the first and foremost step that is to be done before starting any process. We will read the data by using `read_csv`. Then we will know the shape of the data. After that we will know the unique class attributes by using `unique ()`. And by using the `info ()` we will know the information of the attributes. Then we will check whether there are any null values by using `isnull ()`.

`LabelEncoder` can be used to normalize labels. We will import `LabelEncoder` from `sklearn.preprocessing` we will also use `fit_transform(y)` for Fit label encoder and return encoded labels. After doing that we will use `le.transform ()` for Transform labels to normalized encoding.

After that we will divide the whole data into training and testing data. We will assign 70% of the data to the training data and the remaining 30% of the data into testing data. We will do this by using `train_test_split` from `sklearn.model_selection`.

Implementation

Out of the chosen algorithms we will first take Random Forest classification model. We will take a classifier and fit the training data. After that we will predict that by using `predict (X_train)`. Now we will predict the accuracy of the testing data by using `accuracy_score (y_test, pred)`. By doing so for the Logistic regression which is the benchmark will give us the accuracy of 0.358.

We will now choose the algorithm which will give us the better score than the benchmark model out of Random Forest and SVM.

By following the same procedure above that is fitting, predicting and finding the accuracy score we will get the accuracy score as bellow.

Random Forest: 0.737

SVM: 0.654

From the above reports Random Forest seems to be performing well.

Complications:

While building the Random Forest Classifier I faced a complication while tuning the parameters. Firstly we should come to a conclusion about the parameters we are going to tune. Then by varying the range of values of the chosen parameter we should be able to get the parameter value which gives us the better accuracy than the untuned value. So it is complicated to choose the parameter that is to be tuned in this case `n_estimator`.

Refinement

I found out random forest as the best classifier out of the chosen classifiers. Now we will perform tuning of random forest classifier in order to achieve the better accuracy. For this we will use `GridSearchCV`.

For doing refinement we will just tune the parameter. Here we will assign `n_estimators=[450,500]` and `'criterion': ['gini', 'entropy']`. Now will find out the new accuracy. The new accuracy will be 0.7397. This tuned accuracy will be a little bit more than the untuned accuracy.

IV. Result

Model evaluation and validation

The final model we have chosen is tuned random forest which gave us more accuracy that is 0.7397. In order to achieve this accuracy we assigned `n_estimators=[450,500]` and `'criterion': ['gini', 'entropy']` but without using the `n_estimators` and `criterion` we achieved the accuracy of only 0.737 which is a bit less than the tuned value. Here we can say that the solution is reasonable because we are getting much more less accuracy while using other models.

The final model that is tuned random forest has been tested with various inputs to evaluate whether the model generalises well. This model is also robust enough for the given problem. We can say this by testing it over different random sates. Here we used the random sates 1,2,3 and obtained nearly same accuracy for each model and the mean accuracy is equal to the accuracy when random sate is 1. From this we can say that Small changes in the training data will not affect the results greatly. So the results found from this model can be trusted.

Justification

My final model's solution is better than the benchmark model.

	Tuned random forest	Benchmark model
Accuracy	0.7397	0.358

From the above we can conclude that the results for the final model are stronger than the benchmark model.

Hence we can say that tuned random forest provides the significant to solve the problem of predicting the outcome of the endgame King-Rook vs. King.

V. Conclusion

Free-form Visualization

After studying the data and applying all the classification models we can visualize the following. We try to find out the important features in the model.

For this we will be using `feature_importances_`

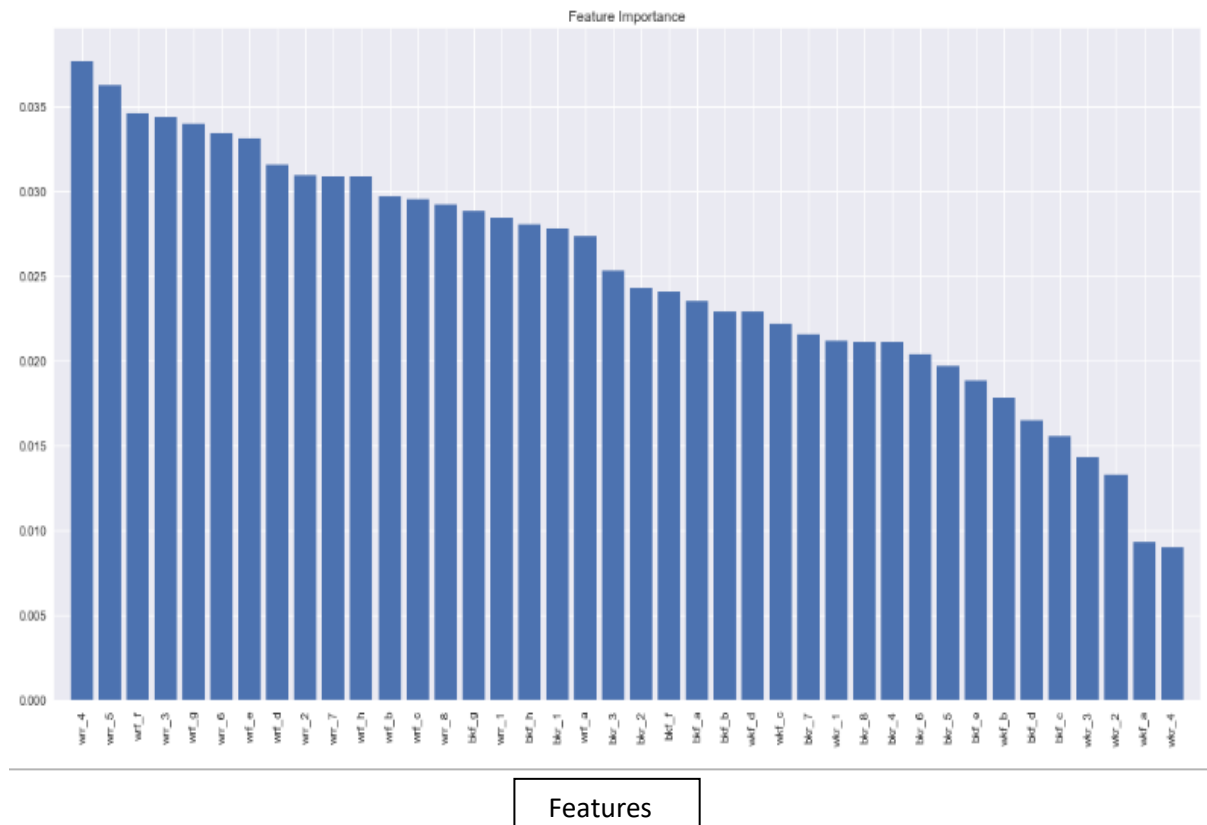
We will plot the important features according to their rank by using the bar graph.

```
In [28]: # plotting the graph for Important Features
plt.figure(figsize=(15, 8))
feat_imp = feat_importances()
feat_imp.get_feat_importances(X_train, y_train)
```

We will plot the graph by using the above code. We can find this in the coding section.

By plotting the bar diagram with features on x-axis and their importance on y-axis we will get the below bar diagram showing the ranks of the features according to their importance.

Plot for the feature importance



From the above graph we conclude the important features according to their rank that is wrr_4 is the most important feature and wkr_4 is the least important feature.

Reflection:

1. I have learnt how to visualize and understand the data.
2. I have learnt that the data cleaning place a very vital role in data analytics.
3. Removing the data features which are not necessary in evaluating model is very important.
4. I got to know how to use the best technique for the data using appropriate ways
5. I got to know how to tune the parameters in order to achieve the best score.
6. On a whole I learnt how to graph a dataset and applying cleaning techniques on it and to fit the best techniques to get best score.

Improvement:

The process which i have followed can be improved to classify not only for the King-Rook vs. King Endgame. This can also be applied for the all the other endgames in chess like King-Rook vs. King-Pawn end game. This model is a part of my researching in Endgames. This application can be taken to next level with the many more applications. As we can say that there has never been an end in the machine learning there will be many more models to learn. By taking the more amount of dataset which consists the information about all the endgames in chess this can be more and more be generalized to all the endgames in the chess.