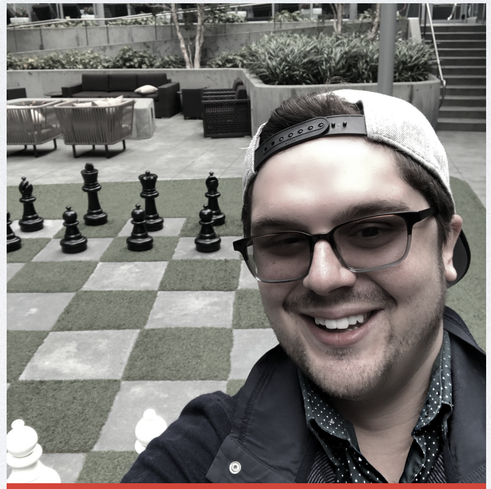
LICHESS CHESS DATASET ANALYSIS



Module 3

ds-ft-041519

Paul Woody

CLASSIFICATION: PREDICTING THE WINNER OF CHESS GAMES

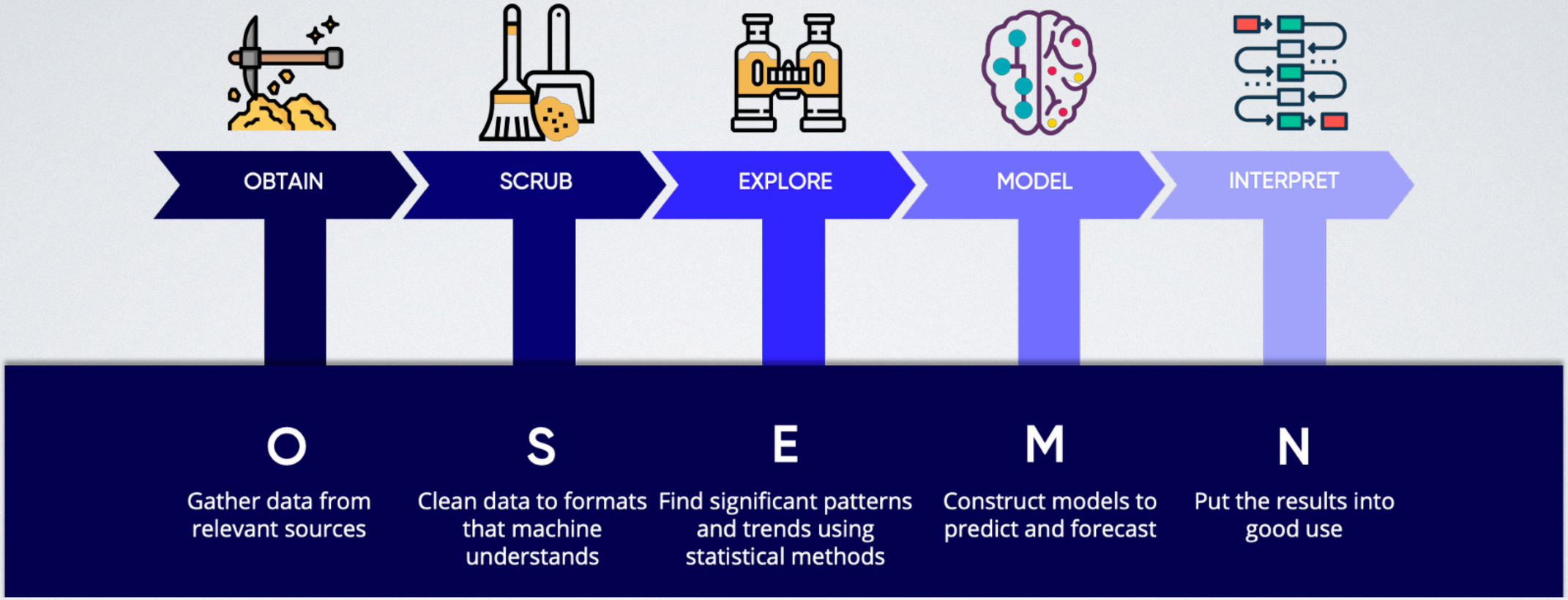
Methodology

Background

Feature Engineering

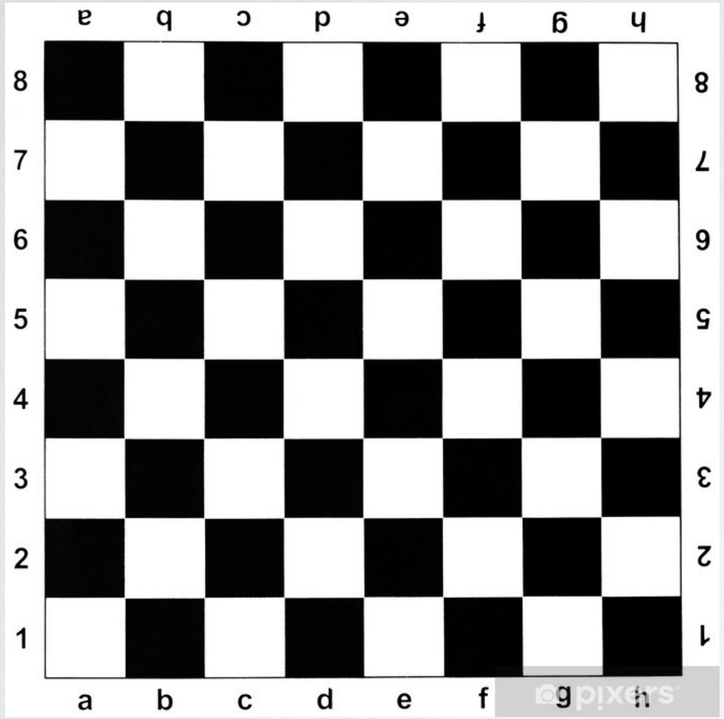
Insights

METHODOLOGY: OSEMN

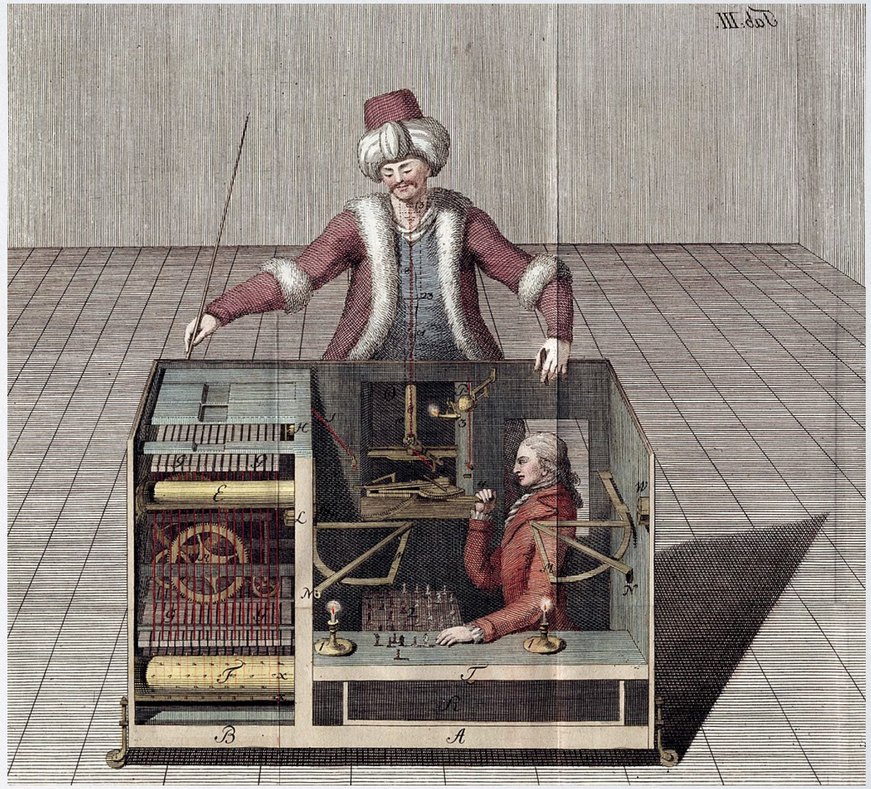


Perform exploratory data analysis to identify meaningful trends.
Use models and appropriate hypothesis tests to evaluate significance.

BACKGROUND



There are more possible positions in chess than atoms in the observable universe, as characterized by the “Shannon number” - a value which approximates chess game-tree complexity.



Interest in automation in chess has long captivated audiences, as evidenced by the debunked **Mechanical Turk** of the 18th century**.**

BACKGROUND

A chess player's **Elo score** offers an approximation of skill level and fluctuates as games between ranked players are won or lost.

Performance in a game depends on a number of factors such as the first move played and number of opponent pieces captured from the board.

FEATURE ENGINEERING

Without an understanding of Chess, we'd still be able to fit the data above into a model and do an 'okay' job with the features provided by LiChess.

However, applying what we know about Chess, we're able to manipulate the existing features of the dataset to produce new input variables that will hopefully improve the predictive accuracy of our model(s). This process, known as **Feature Engineering** is crucial to the implementation and improvement of classification models.

```
# Feature Importance
features = pd.DataFrame(columns = ['feature',
'importance'])
features['feature']    = X.columns
features['importance'] = rf.feature_importances_
features.sort_values('importance', ascending=False)
[0:15]
```

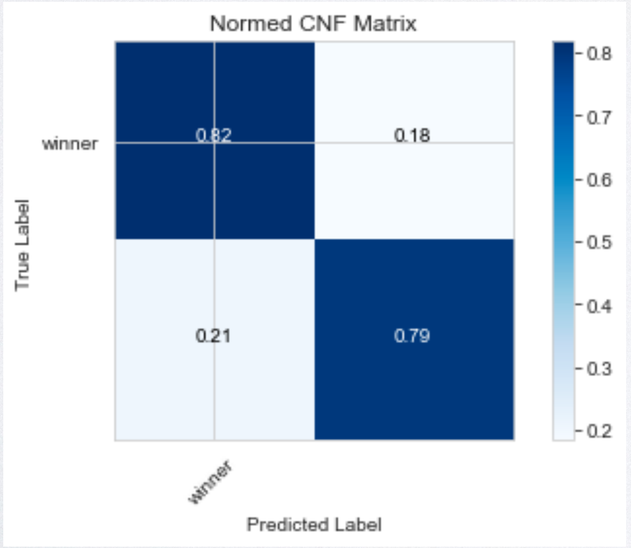
Without After applying our knowledge of Chess to create new features, we found that of the top 15 'best' predictors, **12/15 were created as a result of feature engineering.**

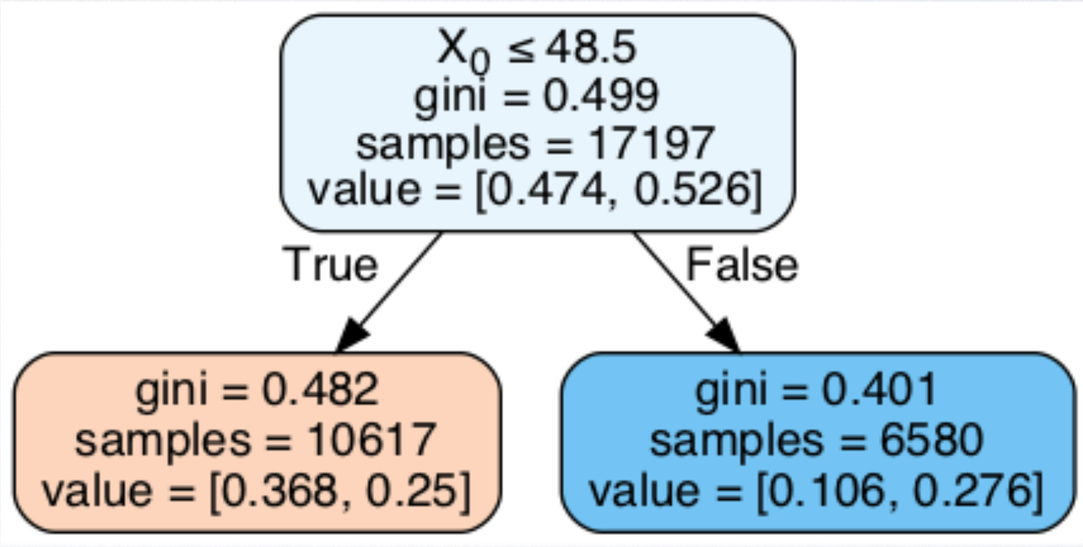
INSIGHTS

Model Performance:

```
Logistic Regression mean score:  0.8248381410406866
Random Forest mean score:  0.8219059307271458
Bagged Tree mean score:  0.805736082639723 1
Adaboost Tree mean score:  0.8204943718042659
```

Adaboosting: A Closer Look at Model Performance





THANK YOU FOR YOUR TIME