CHESS

Team 6
Yu Xiang Zhang
Karim Rhoualem
Nafisa Shamsuzzaman
Cheikh Idrissa Diagne

SUPERVISED

Classification In a new match between two players, who is most likely to win?
Regression Can a player's rating be determined from a single match record with another player?

LIST OF MOVES

01

DATA INSPECTION

Data Types, Anomalies, Features and Labels

02

DATA PREPARATION

Anomalies Handling, Feature Engineering, Train/Test Splits

03

FEATURE SELECTION

Spark Vectors

04

MODEL SELECTION

Decision Tree, Logistic Regression, Linear Regression

05

MODEL EVALUATION

Metric

06

PROJECT OUTLOOK

Improvements

DATA INSPECTION: OVERVIEW

Categorical

opening eco
opening name
moves
black id
winner
rated id
white id
victory status
increment code

Numerical

opening ply
turns
white rating
last move at
created at
black rating

Labels

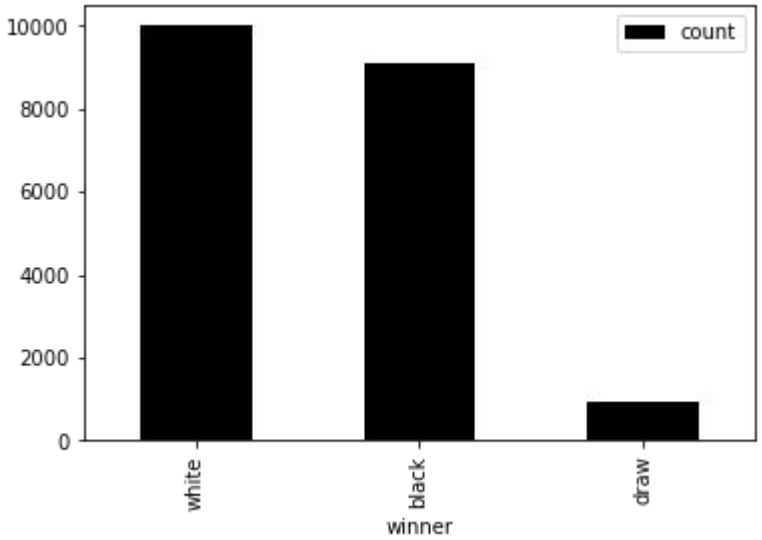
* For classification:
  winner
* For regression:
  black_rating

Anomalies

* Duplicate Rows
* No Missing Values

DATA INSPECTION: LABELS

Classification Class Distribution



Regression Label Summary

summary	black_rating	white_rating	rating_gap
count	20058.00	20058.00	20058.00
mean	1588.83	1596.63	-7.80
stddev	291.04	291.25	249.04
min	789.00	784.00	-1499.00
max	2723.00	2700.00	1605.00

DATA PREPARATION

WORD2VEC

`moves`

'e4 c6 Nc3 d5 Nf3 Bg4 h3 Bh5 exd5 cxd5 Bb5+ Nc6'
→ [0.1149, 0.0761, 0.019, ..., -0.056, 0.0682, 0.0676]

EXTRACT NEW FEATURES

`increment_code`

'30+25' → 30, 25
'10+0' → 10, 0

`opening_eco`

'B11' → 'B' → 1.0
'C60' → 'C' → 0.0
'D10' → 'D' → 3.0
'A00' → 'A' → 2.0
'E91' → 'E' → 4.0

ONE-HOT ENCODE

`victory_status`

'resign' → 0.0 → SparseVector(4, {0: 1.0}))
'outoftime' → 2.0 → SparseVector(4, {2: 1.0}))
'mate' → 1.0 → SparseVector(4, {1: 1.0}))
'draw' → 3.0 → SparseVector(4, {3: 1.0}))

ENCODE TO NUMERICAL

`winner`

'draw' → 2.0
'white' → 0.0
'black' → 1.0

`white_id, black_id`

'konst767' → 0
'ducksandcats' → 1
'everybodylovesjesus' → 2

RECAP

Features

- ❑ rated
- ❑ turns
- ❑ match_status
- ❑ clock
- ❑ increment
- ❑ white_rating
- ❑ black_rating
- ❑ open_cat
- ❑ opening_ply
- ❑ moves_vec
- ❑ winner

- ~~❑ id~~
- ~~❑ white_id~~
- ~~❑ black_id~~
- ~~❑ created_at~~
- ~~❑ last_move_at~~

CHESS

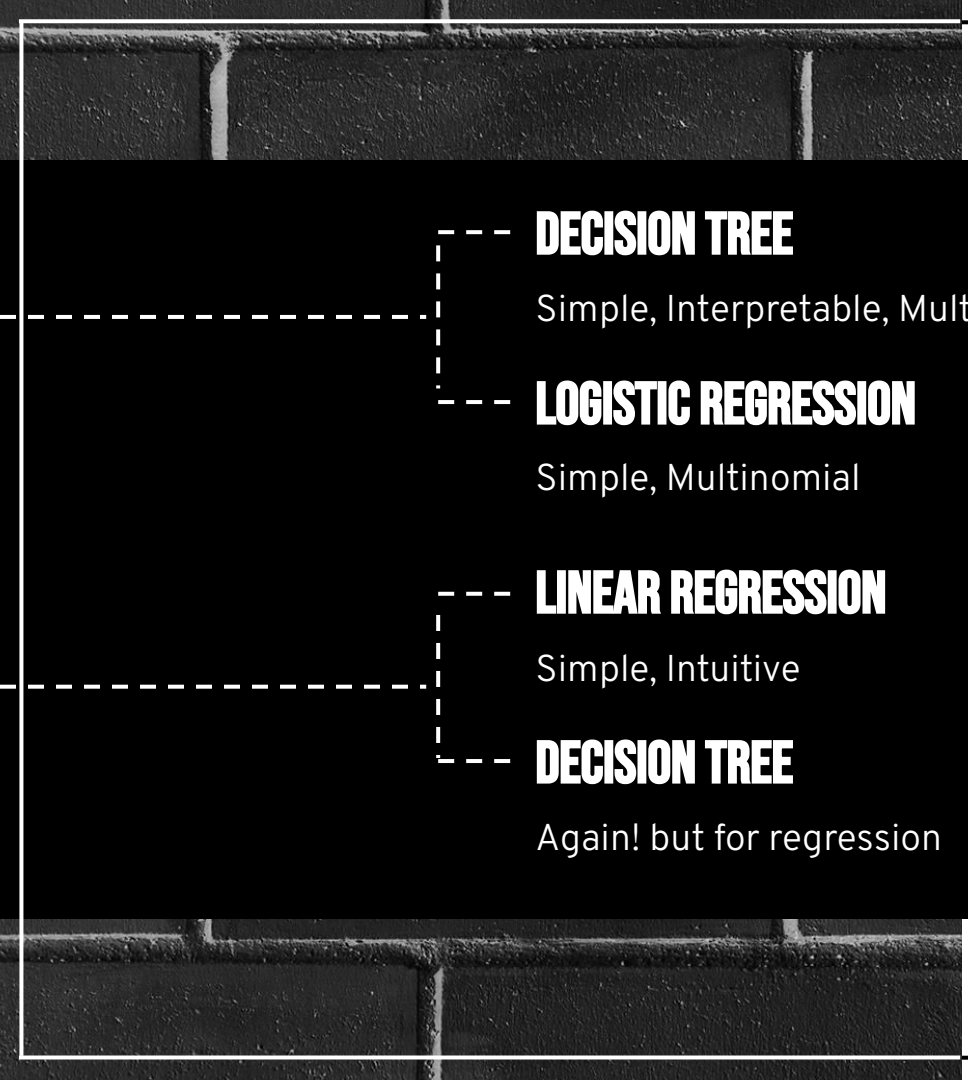
20,000+ Lichess Games, including moves, victor, rating, opening details

CLASSIFICATION

Predict winner given a match record

REGRESSION

Predict rating of one player



DECISION TREE

Simple, Interpretable, Multiclass

LOGISTIC REGRESSION

Simple, Multinomial

LINEAR REGRESSION

Simple, Intuitive

DECISION TREE

Again! but for regression

MODEL SELECTION

- DecisionTreeClassifier
- LogisticRegression
- LinearRegression
- DecisionTreeRegressor

MODEL IMPLEMENTATION

Decision Tree for Regression

```
dt = DecisionTreeRegressor(labelCol=target, featuresCol="features", seed=seed)

dtParamGrid = (ParamGridBuilder()
               .addGrid(dt.maxDepth, list(range(2, 8)))
               .addGrid(dt.maxBins, [4, 8, 16, 32, 64])
               .addGrid(dt.minInstancesPerNode, list(range(1, 4)))
               .build())

dtEvaluator = RegressionEvaluator(labelCol=target, metricName='rmse')
```

Hyperparameter Tuning (Grid Search) ~18 min

```
if not skipGridSearch:
    dtCv = CrossValidator(estimator = dt,
                          estimatorParamMaps = dtParamGrid,
                          evaluator = dtEvaluator,
                          numFolds = 5)

    dtCvModel = dtCv.fit(reg_train_df)

    # Best Parameters found
    bestParamMap = dtCvModel.bestModel.extractParamMap()
    print( ','.join(['='.join((param.name, '=', repr(value))) for param, va
```

Model Train

```
# Use Best Parameters to initialize and train a separate regressor model
dtBest = DecisionTreeRegressor(cacheNodeIds=False,checkpointInterval=10,featuresCol
# assert dtBest.extractParamMap() == bestParamMap, 'Check if the input parameters a
bestModel = dtBest.fit(reg_train_df)
```

Model Evaluation

```
trainPredAndLabels = bestModel.transform(reg_train_df).select('prediction', target)
testPredAndLabels = bestModel.transform(reg_test_df).select('prediction', target)
print('train rmse', dtEvaluator.evaluate(trainPredAndLabels))
print('test rmse', dtEvaluator.evaluate(testPredAndLabels))
```

MODEL EVALUATION

Decision Tree CLF

Train F1 0.70613

Test F1 0.67201

Logistic Regression

Train F1 0.76362

Test F1 0.74706

Decision Tree REG

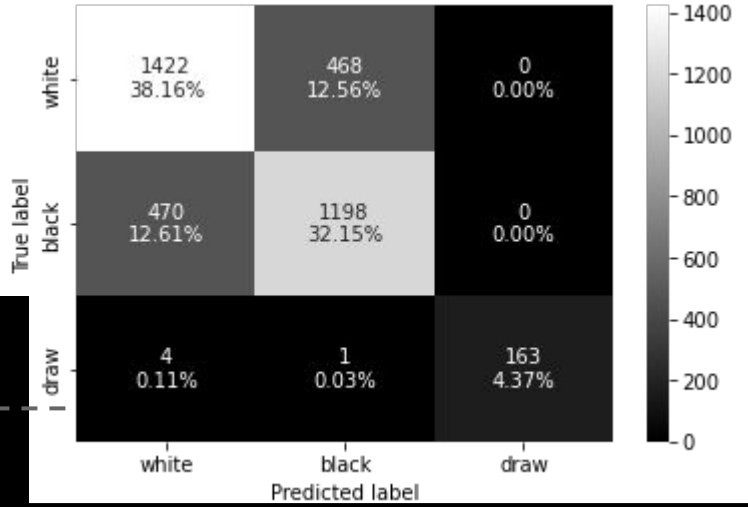
Train RMSE 259.35

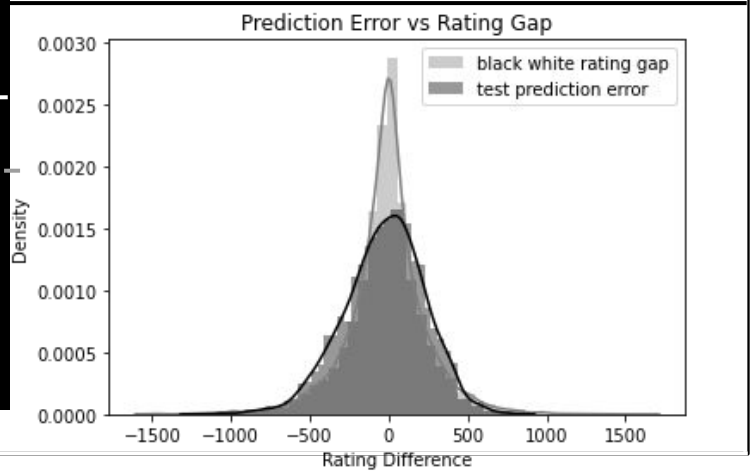
Test RMSE 275.86

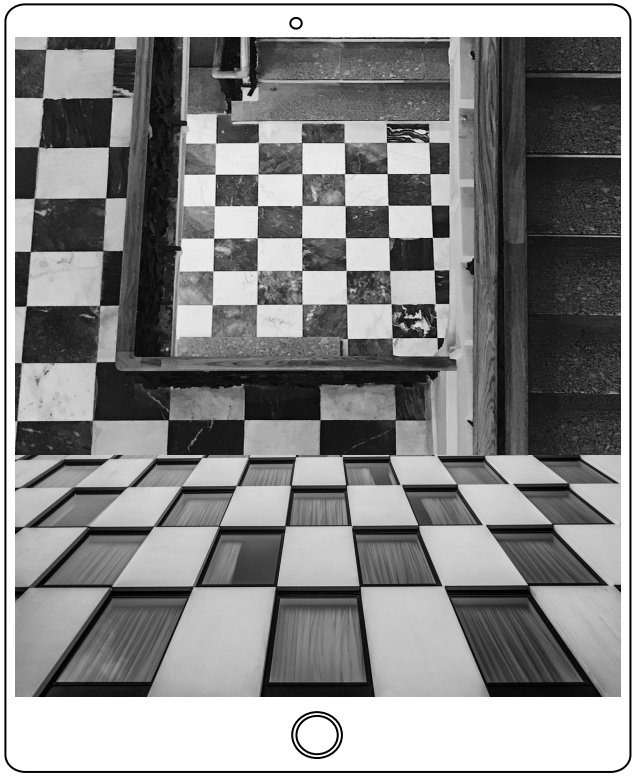
Linear Regression

Train RMSE 250.66

Test RMSE 258.24







PROJECT OUTLOOK

WAYS TO IMPROVE

- Use more robust models like Random Forest
- Leverage Lichess API and Spark ML Pipelines to build an end-to-end system
- Apply model to more data from Lichess

LEARNINGS

- Attempted Supervised Learning Problems
- Explored Spark API for ML
- Explored how to preprocess sequence data

THANKS

Team 6
Yu Xiang Zhang
Karim Rhoualem
Nafisa Shamsuzzaman
Cheikh Idrissa Diagne