Predicting a Player's Position Using Skills-Based Attributes (FIFA19)

Samaye Lohan
Brown University – Data Science Initiative
December 8, 2021.
GitHub Repository Link: https://github.com/SamayeLohan/Data1030-Project

Objective

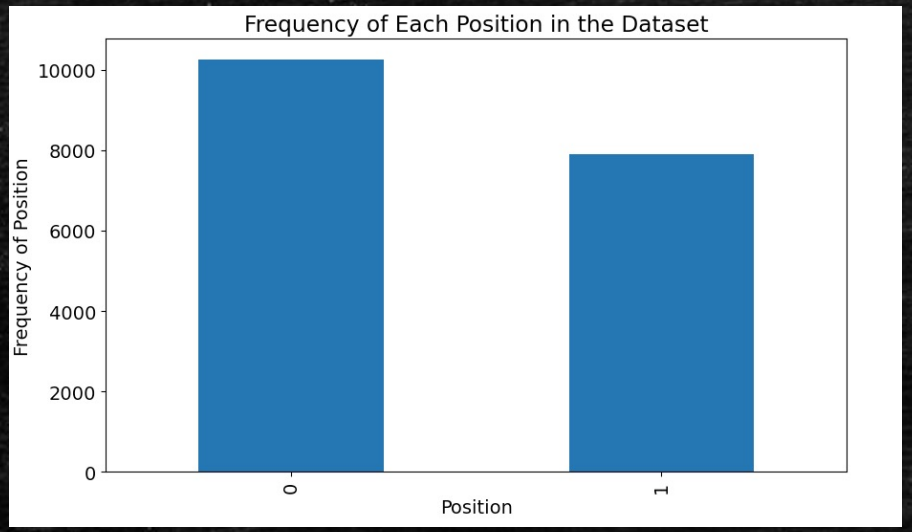
- Predict a player's position by evaluating relevant skills-based metrics
- ***Importance***: Leverage the flex ability by assigning players to various positions during a 90-minute game (*optimize strategy and output*)
- ***Binary Classification*** ~ Predicting a discrete class label (*Position ~ Target Variable is a category)*
- ***Dataset:*** https://www.kaggle.com/karangadiya/fifa19 (Kaggle - FIFA19)

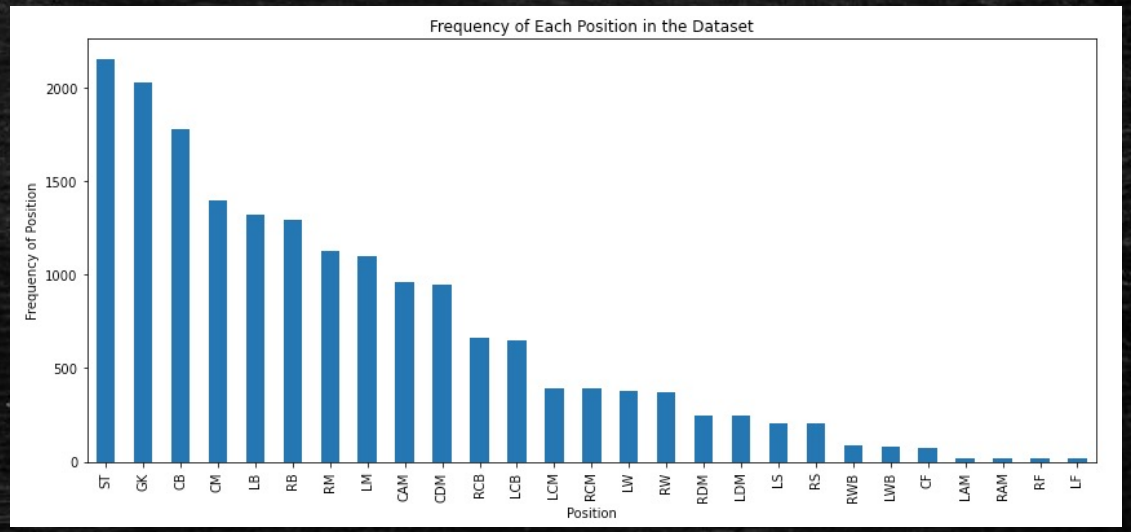
Dataset Recap

- Class Distribution – 27 distinct positions divided into two classes:

```
offense = ["ST", "LW", "RW", "LF", "RF", "RS","LS", "CF",
           "CM","RCM","LCM", "CAM", "LAM", "RAM", "RM", "LM"]
defense = ["CB", "RCB", "LCB", "LWB", "CDM","RDM","LDM", "RWB", "LB", "RB", "GK"]
```

- Class Frequency Measure: 0 -> offense, 1 – defense

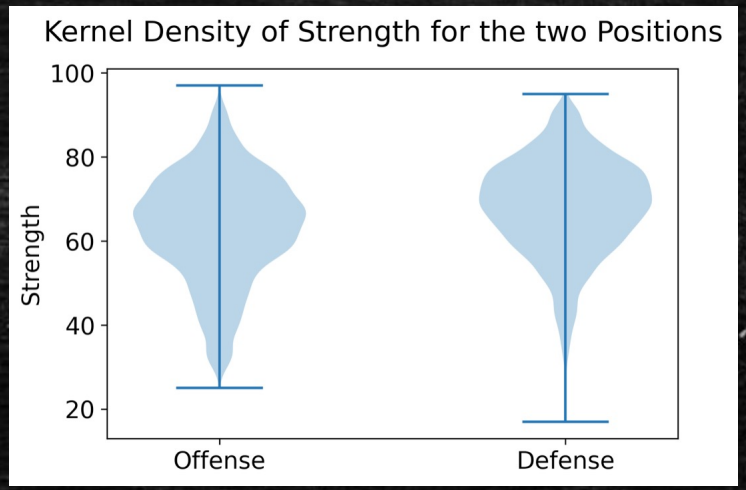




EDA Recap

- ***Columns***: Only took features that were relevant in predicting a player's position – 52 features reduced from the original dataset.
- ***Rows:*** Contains 60 fewer rows because all *NA* values in our *Position (*Target Variable) column were dropped
- The Kernel Density of Strength proved to be the most similar amongst the two classes

Dataset	Rows	Columns
Original	18, 207	87
Preprocessed	18, 147	35



Cross Validation + ML Algorithms

- ***Train/Test:*** 80/20
  - train_test_split()
    - 14, 518 Training Points
    - 3, 629 Testing Points
    - Why? – Optimize all the feature attributes by allocating a decent training percentage to them
  - KFold()
    - Allow variation in the validation set to help the model generalize
    - Ensure each model gets the same splits

- Preprocess
  - OneHotEncoding
    - *Preferred Foot*
    - Unordered Categorical Data
  - MinMaxScaler
    - *Remaining 33 Features*
    - Continuous Features
    - Range: 0 - 100

Cross Validation + ML Algorithms

- ***ML Algorithms***
  - Random Forest
    - *n_estimators, max_features, criterion*
  - Logistic Regression
    - *solver, max_iter, C*
  - Support Vector Machine
    - *gamma, kernel, C*
  - Decision Trees
    - criterion, max_features, max_depth

Parameters

```
param_grid = {
    'ml_algo__n_estimators': [100, 200, 500, 1000],
    'ml_algo__max_features': ['auto', 'sqrt', 'log2'],
    'ml_algo__criterion': ['gini', 'entropy']
}
```

```
param_grid = {
    'ml_algo__solver': ['saga', 'newton-cg', 'lbfgs', 'liblinear'],
    'ml_algo__max_iter': [1000],
    'ml_algo__C': [1e6, 1e5, 1e4, 1e3, 1e2]
}
```

```
param_grid = {
    "ml_algo__kernel":['poly', 'rbf', 'sigmoid'],
    'ml_algo__C':[10, 1, 0.1, 00.1],
    'ml_algo__gamma': [1,0.1,0.01,0.001]
}
```

```
param_grid = {
    "ml_algo__criterion":['gini', 'entropy'],
    'ml_algo__max_features':['sqrt', 'log2'],
    'ml_algo__max_depth': [5, 10, 15]
}
```

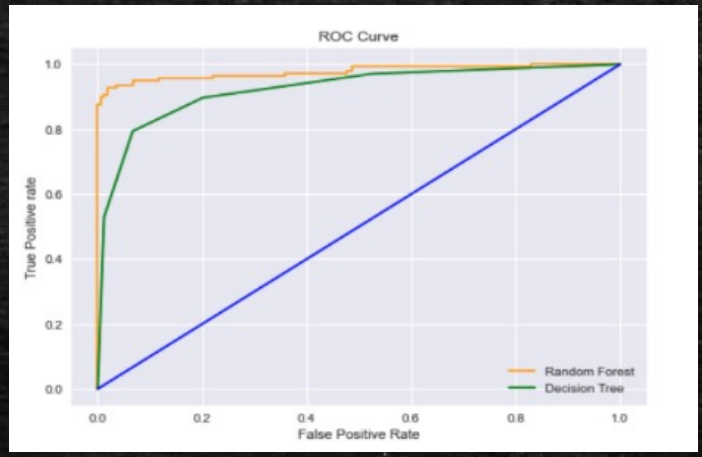
- CV Pipeline
  - GridSearchCV
    - *Scoring: ROC AUC*
    - Kfold()
    - Pipe: Preprocessor, ML_algo
  - Fitting the model
    - *.fit()*
  - Best Parameters & Scores
    - *.best_params_*
    - *.best_scores_*

Results: Model Performances & ROC Curve

- Model Performances

	Model	ROC AUC Score
1	Random Forest	0.965550
0	Support Vector Machines	0.953004
2	Logistic Regression	0.946450
3	Decision Tree	0.857801

- ROC Curve for RF and DT Models

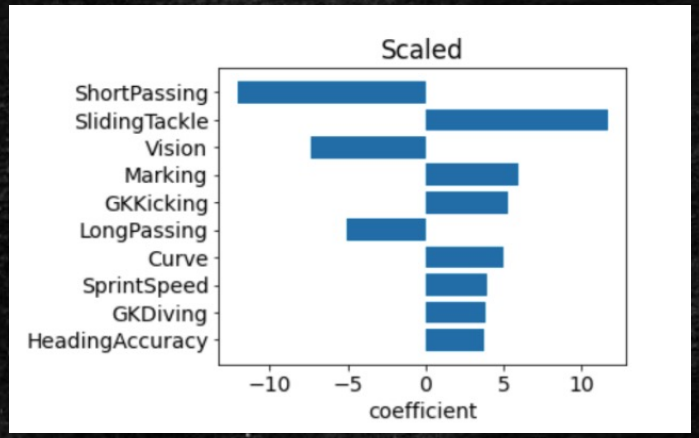


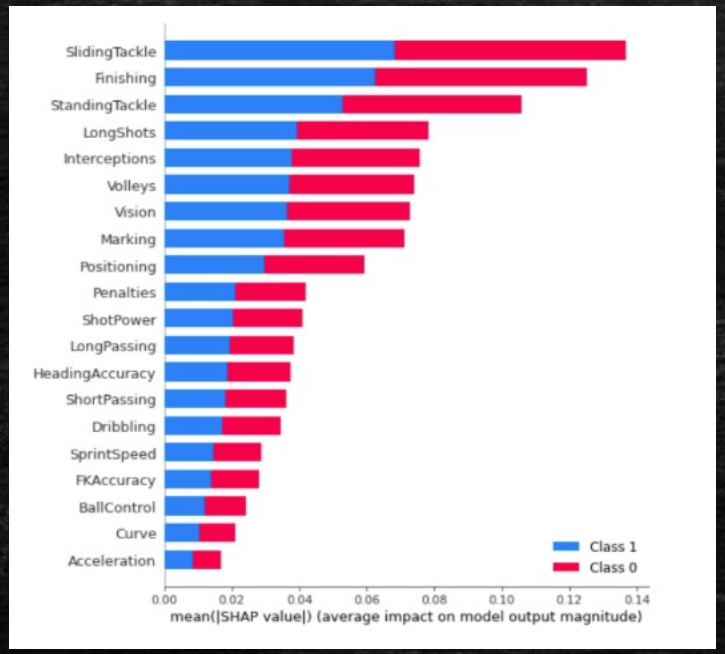
- ***Baseline Model:***
  - Average ROC AUC score: 0.72
  - Standard Deviation: 0.06
  - Standards Below Baseline: 42

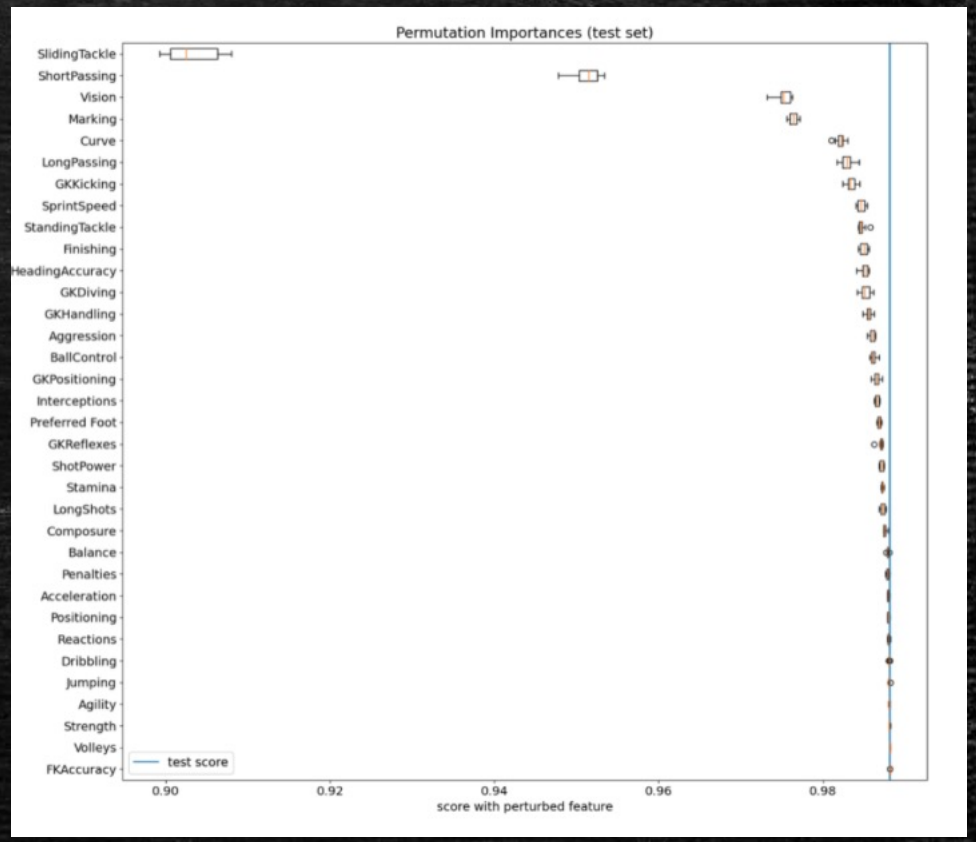
- ***Trained Models:***
  - Average ROC AUC score: 0.93
  - Standard Deviation: 0.02
  - Standards Above Baseline: 11

Results: Global Feature Importance for Random Forest

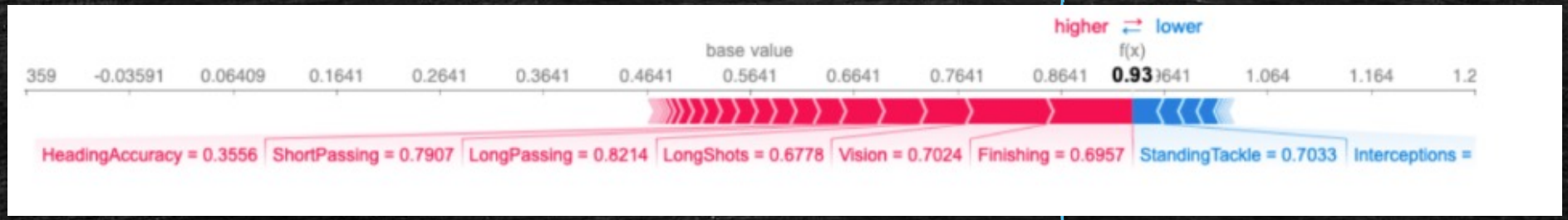
- Most Important features:
  - Sliding Tackle, Finishing, Vision
    - Finishing, Vision (flex) -> Offense (0)
    - Sliding Tackle, Vision (flex) -> Defense (1)







Results: Local Importance for Random Forest



- Most Important Features
  - Class 0: *Finishing*
  - Class 1: *Sliding Tackle*
- Least Important Features
  - Class 0: *Heading Accuracy*
  - Class 1: *Interceptions*

Outlook

- Assign higher importance:
  - *Sprint Speed*
  - *Aggression*
- Change Scoring Method:
  - *F1 Score*
  - *Increase the sensitivity of class 0 (defense)*
  - *Reduces favoring the FN over FP*
- Multiclass Problem
  - 4 classes
    - Forward
    - Midfielder
    - Defender
    - Goalkeeper
- Collect Additional Real-World Data
  - Utilize prehistoric data
    - Feature ratings constantly change
    - Addition of new players
    - Addition of new features

Questions?