Methodology: IPL FP and Captaincy Predictor

Team: TensileTensors

1. **Data Pre-processing**
    a. Reduction
        i. Removed rows with NaN values for the important attributes, match_id and prev_Total_FP as we can't use such rows.
        ii. Removed the irrelevant columns: Starting_11 (all same values), match_name (home team and away team sufficed), bowling_innings (redundant), prev_Overs_Bowled (no. of overs bowled sufficed) for predicting FP and just the Starting_11 for predicting captaincy; prev_overs, prev_conceded, prev_fours, prev_sixes, prev_runs were removed due to other features engineered to replace them.
    b. Cleaning
        i. NaN values were filled via extrapolation and aggregation.
    c. Transformation
        i. Textual labels were encoded using the LabelEncoder() function from sci-kit learn.
        ii. Strike rate, number of balls bowled and boundaries, three new features, were engineered from other features.
    d. Resampling
        i. SMOTE was used to resample "captain" and "vice-captain" to increase the number of data points for better training.
2. **Training**
   Statistical models and deep models were used for the prediction.
    a. Autogluon (AutoML)
       Autogluon was used to train several modes to predict captaincy, the best possible model being a weighted ensemble of XGBoost with bagging, with 0.5 weight, and a neural network with bagging, with a weight of 0.5.
       However, its accuracy was not great, so we trained some models ourselves.
    b. Custom Models
       Decision Classifier, stacking using XGBoost & Isolated Forest, XGBoost & RandomForest, Decision Classifier & RandomForest, and bagging with RandomForest and CatBoost were done to predict captaincy. The Decision Classifier turned out to be the best model.
       Linear regression, polynomial regression, and deep neural networks were used to predict FP. FFNN with dropout and Adam optimiser performed well enough for us to use it finally (Adadelta optimiser was also considered, but it performed worse)