CRISP-DM analysis of second-half goal scoring in football

Machine Learning Assignment

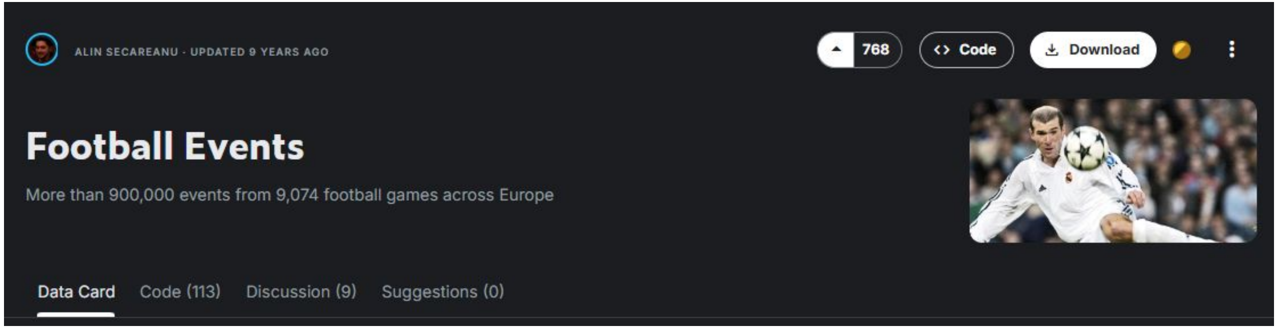
Gabriele Santi -
A.Y 2025-2026

Business Understanding

- Football matches are highly unpredictable
- Goal: binary classification
- Target: home team scores at least one goal in second half
- Only first-half data used

Dataset & Data Understanding

- Public dataset from Kaggle
- Two main files: events.csv, ginf.csv
- Event-level data for football matches
- Large number of raw events



Data Preparation

- Filter only first-half events
- Feature selection (remove irrelevant columns)
- Feature engineering: relevant event
    - shots on/off target
    - fouls, corners, free kicks
- Aggregation at match level

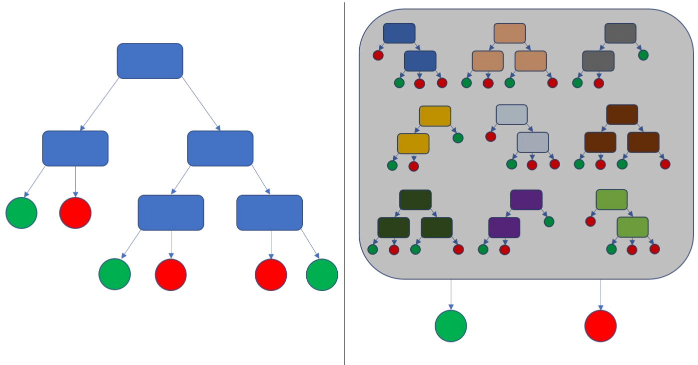
```
def create_dataset():

    # Reduce events_df dataset, filtering only first-half events
    first_half_events = events_df[events_df['time'] <= 45].copy()

    #Discarding useless features
    first_half_events = first_half_events.drop(columns=['sort_order' , 'time', 'text' , 'event

    #Extracting only matches with events
    matches_with_events = first_half_events['id_odsp'].unique()
    ginf_filtered = ginf_df[ginf_df['id_odsp'].isin(matches_with_events)].copy()

    # Creating new features from dictionary.txt
    first_half_events['shot_on_target'] = (
      (first_half_events['event_type'] == 1) &
      (first_half_events['shot_outcome'] == 1)
    ).astype(int)

    first_half_events['shot_off_target'] = (
      (first_half_events['event_type'] == 1) &
      (first_half_events['shot_outcome'] == 2)
    ).astype(int)

    first_half_events['corner'] = (first_half_events['event_type'] == 2).astype(int)
    first_half_events['free_kick'] = (first_half_events['event_type'] == 8).astype(int)
    first_half_events['offside'] = (first_half_events['event_type'] == 9).astype(int)
    first_half_events['foul'] = (first_half_events['event_type'] == 3).astype(int)
    first_half_events['yellow_card'] = (first_half_events['event_type'] == 4).astype(int)
    first_half_events['goal'] = first_half_events['is_goal'].astype(int)

    # Aggregation : calculating how many occurences of every event
    matches_features = (first_half_events.groupby(['id_odsp', 'side'] , as_index=False).agg({
            'shot_on_target': 'sum',
            'shot_off_target': 'sum',
            'corner': 'sum',
```

Modeling

- Target: home_scored_second_half
- Binary variable (0 / 1)
- 80% training – 20% test
- Fixed random seed for reproducibility

Models and approaches used :

- Decision Tree
- Random Forest
- Auto ML



Decision Tree

Random Forest

Decision Tree

- Simple and interpretable model
- Used as a baseline
- Hyperparameter tuning with RandomizedSearchCV
- 5-fold cross-validation

Hyperparameter considered :

- max_depth
- min_samples_split
- min_samples_leaf
- criterion
- max_features

Random Forest

- Ensemble of multiple Decision Trees
- Reduces overfitting compared to a single tree
- Better stability and generalization
- Hyperparameter tuning with RandomizedSearchCV

Hyperparameter considered :

- n_estimators
- max_depth
- min_samples_split
- min_samples_leaf
- max_features

Auto ML

Automatic model selection and tuning

Implemented using FLAML

Same training and test split

Limited time budget

Key aspect :

- No manual hyperparameter tuning
- Focus on fast and efficient models
- Used as a comparison baseline

Evaluation

Metrics used :

- Accuracy
- Precision
- Recall
- Confusion Matrix

	Model	Accuracy	Precision	Recall
0	Decision Tree	0.566942	0.571250	0.901381
1	Random Forest	0.582920	0.589923	0.831361
2	AutoML	0.576860	0.579870	0.880671

Results and Conclusion

- Similar performance across models
- Random Forest slightly higher accuracy
- Decision Tree higher recall
- AutoML competitive with manual models
- Football is hard to predict