Football Events Analysis

TEAM XGBOOST

Project Lead
Elijah Kingutu

Assistant Project Lead
Omar Alqaysi

Query Analyst
Victor Jokanola

ACTIVE MEMBERS

- Yaswanth Teja Yarlagadda
- Ridwan Tiamiyu
- Kingsman Kushieme
- Anne Leah
- Abdulkabir Bello
- Ifeoluwa Oduwaiye
- Julius Markwei
- Fausat Wasiu
- Grace Nzambi
- Hema Kokku

HAMOYE

AIMS & OBJECTIVES

AIMS & OBJECTIVES

The aim of the project is to develop a machine learning model to predict whether **a goal will occur or not based on several events or conditions**, such as:

- Location of the ball on the field.
- Body part used to shoot the ball.
- Assist method that resulted to a goal.
- Whether it is a Corner, Free kick or an Open play..
- Shot place

HAMOYE

OUR APPROACH

- Exploratory Data Analysis is done first to get some insights and ideas.
- Data wrangling & preparation .
- Goal prediction idea was formed.
- Model Building.

DATASET DESCRIPTION

About Dataset: The dataset provides a granular view of 9,074 games, totaling 941,009 events from the biggest 5 European football (soccer) leagues: England, Spain, Germany, Italy, France from 2011/2012 season to 2016/2017 season as of 25.01.2017.

Data Source: Link to dataset from Kaggle
[https://www.kaggle.com/datasets/secareanualin/football-events]

The dataset is organized in 3 files:

- events.csv - contains event data about each game.
- ginf.csv - contains metadata and market odds about each game.
- dictionary.txt - contains a dictionary with the textual description of each categorical variable coded with integers.

HAMOYE

DATA PREPARATION

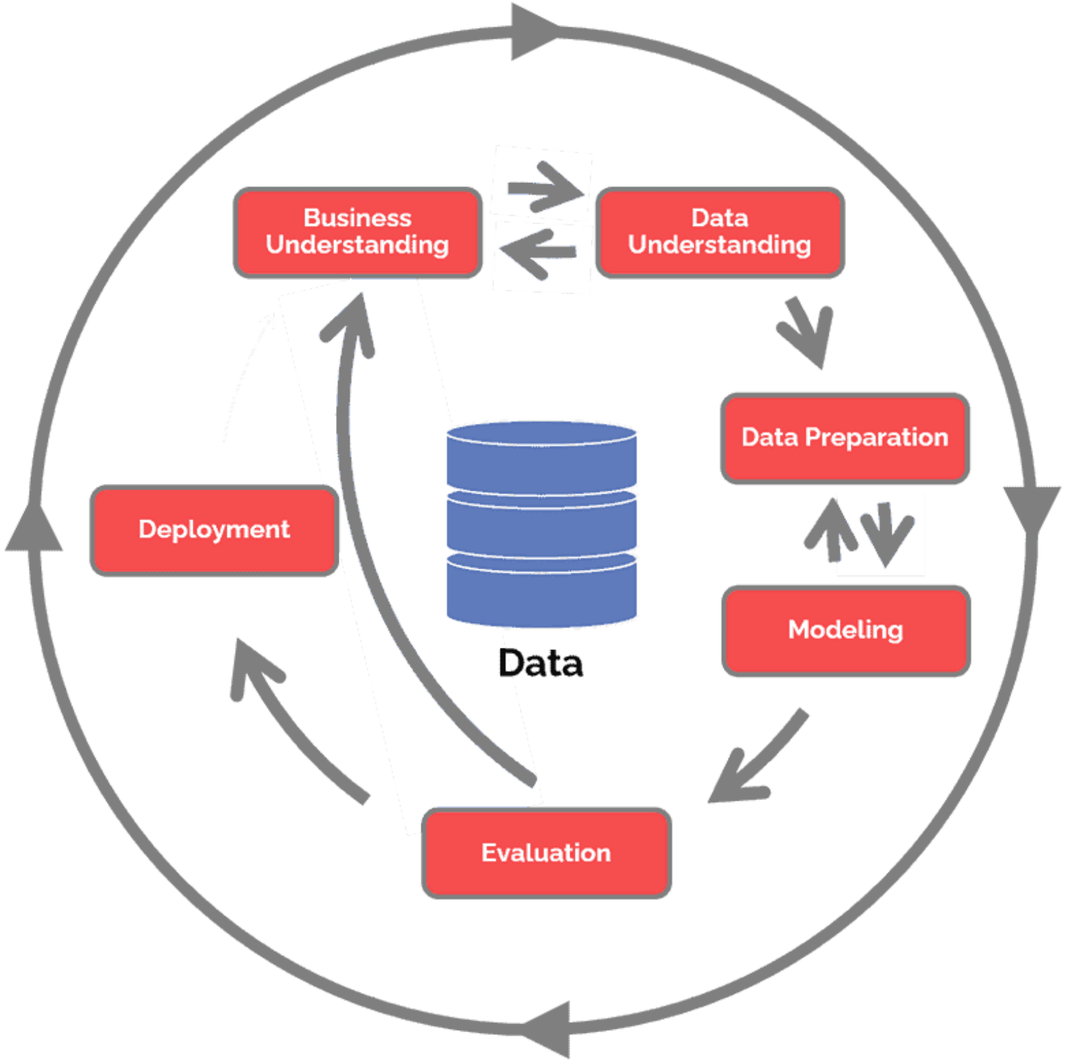
The data preparation phase includes **data cleaning**, **recording**, **selection**, and **production of training** and **testing data**. Additionally, datasets or elements may be **merged** or **aggregated** in this step.

Data Wrangling: Focuses on changing the data format by translating "raw" data into a more usable form.

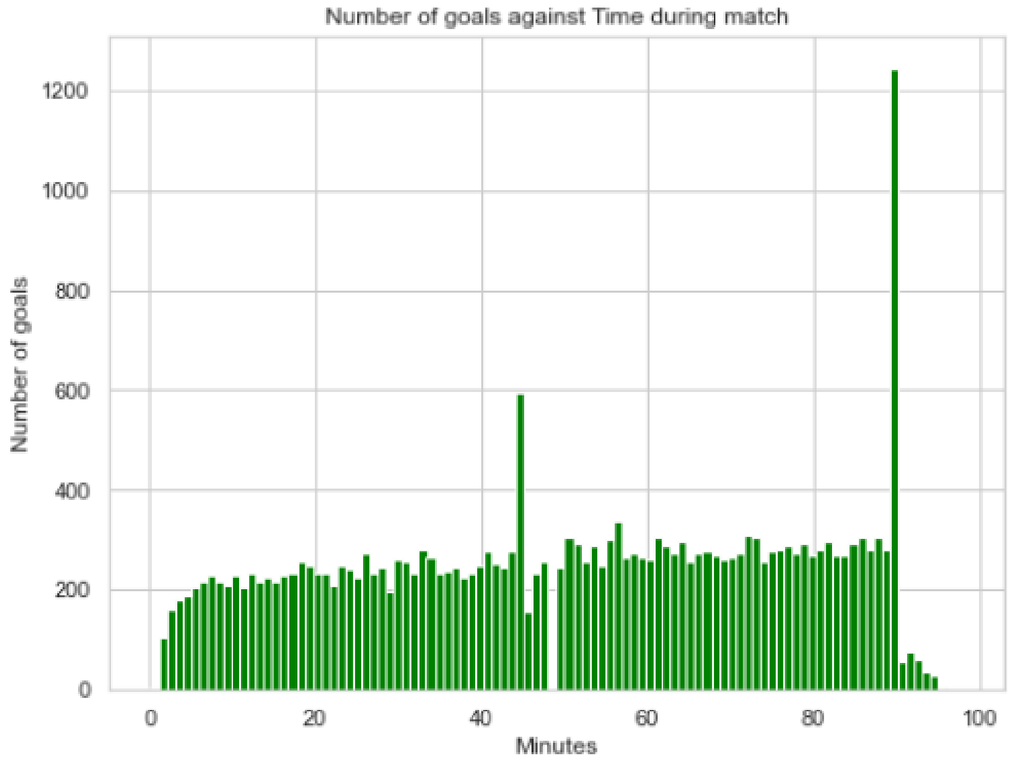
Data Processing: Converting raw data into meaningful information, and these data are machine-readable as well.

HAMOYE

DATA EXPLORATION WORKFLOW

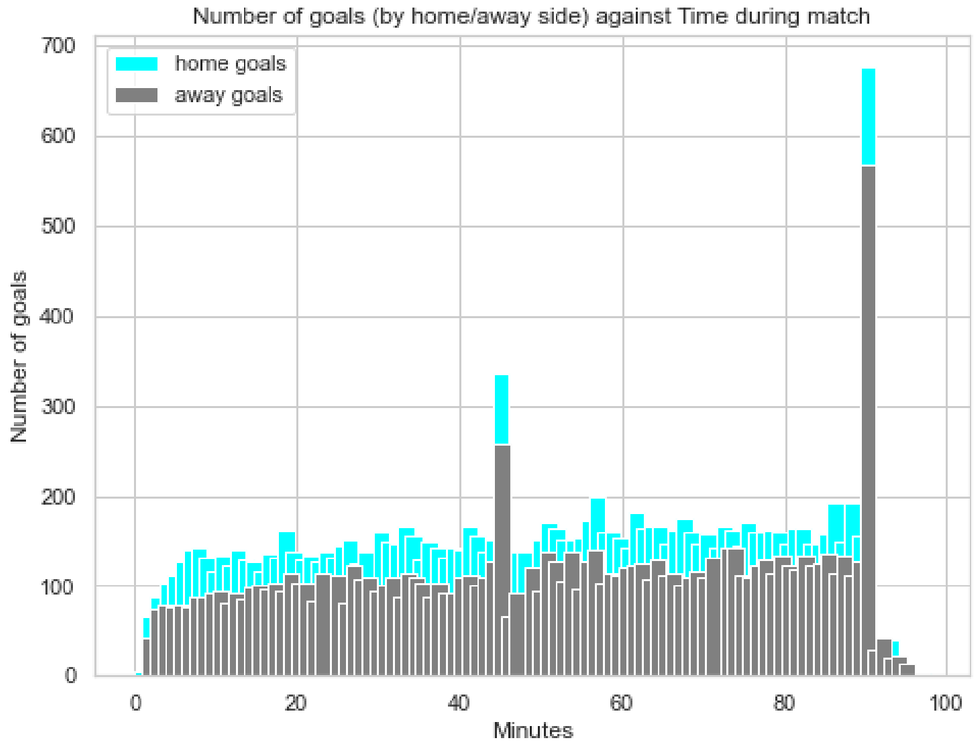


Analysis of the goals against Time



- Most goals are scored around the Half-Time (45mins + extra time) and around Full-Time (90mins + extra time)

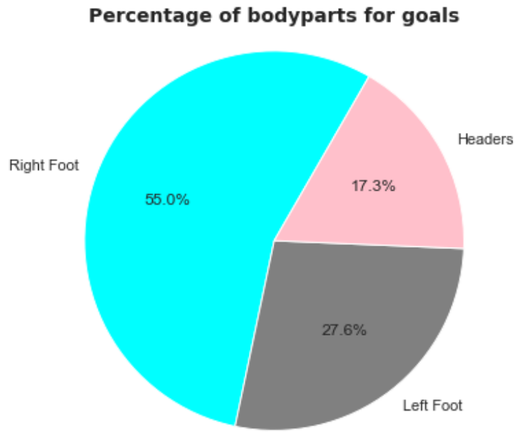
Analysis of goals for Home and Away team

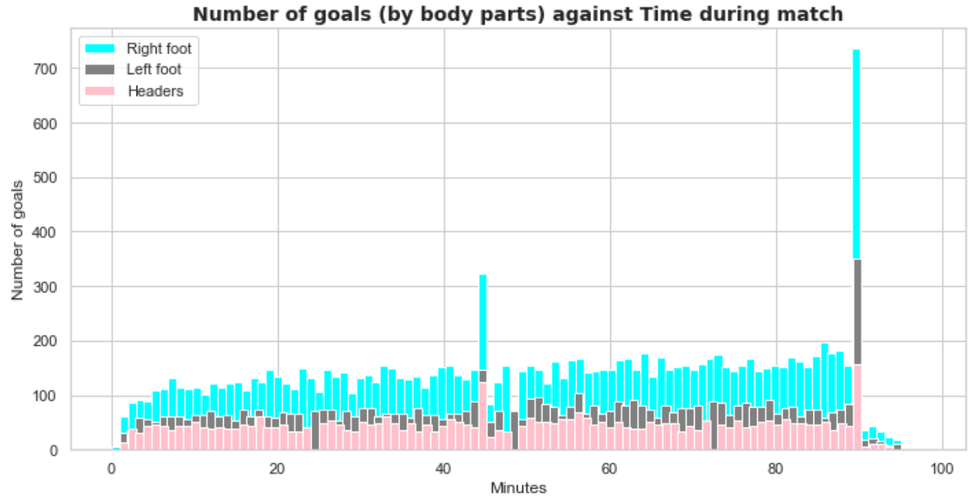


- For every minute, most of the goals scored are by the home side

This supports the general notion that the home side has a statistical advantage.

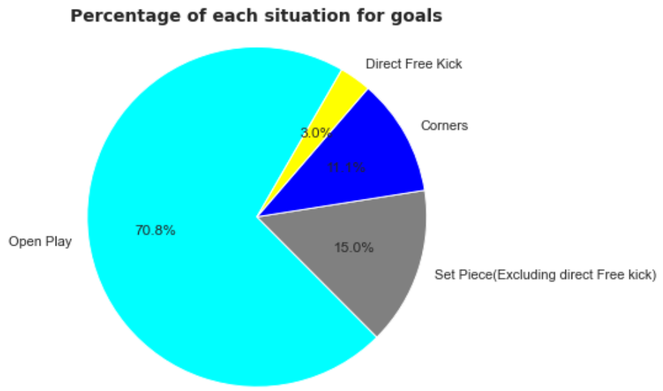
Analysis of HOW goals are scored

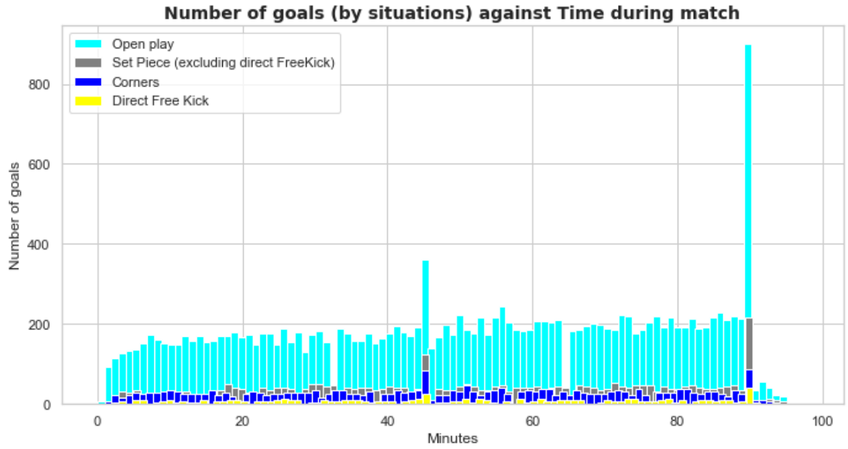




- Most of the goals (slightly more than half) scored are by Right Footed, followed by Left Footed and lastly, by Headers.
- Perhaps this might be because the majority of humans are right-footed and hence, most players are right-footed. It is also not surprising that most goals have been scored by foot not head, as after all, soccer is meant to be played by foot.

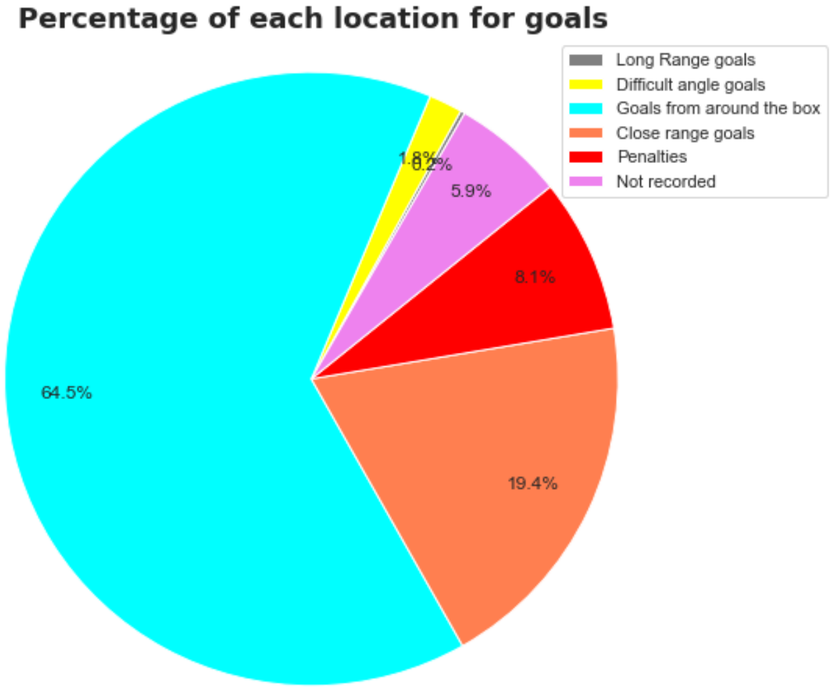
Percentage of each situation for goals





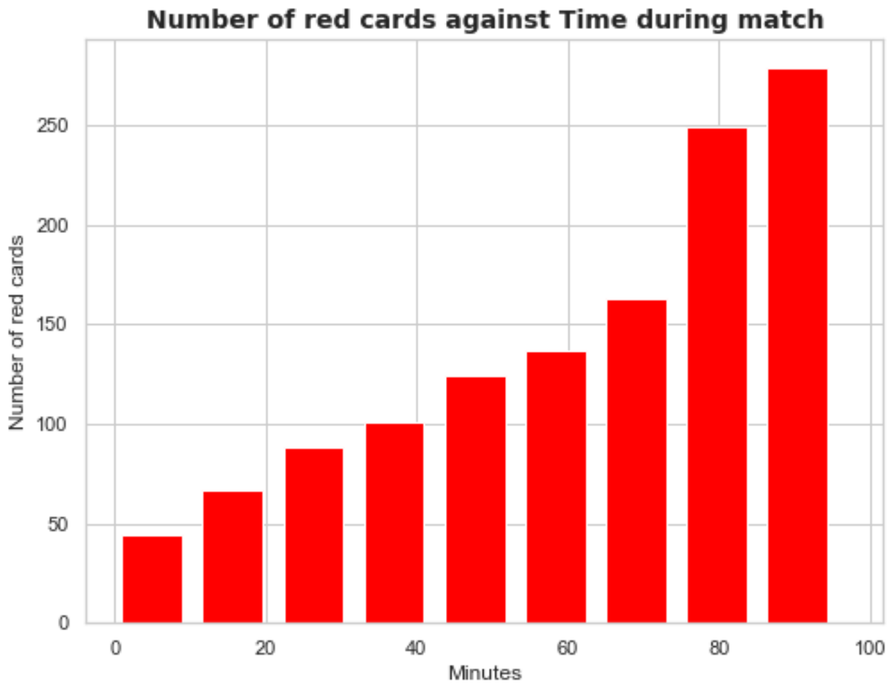
- About 70.8% of the goals scored are from open play

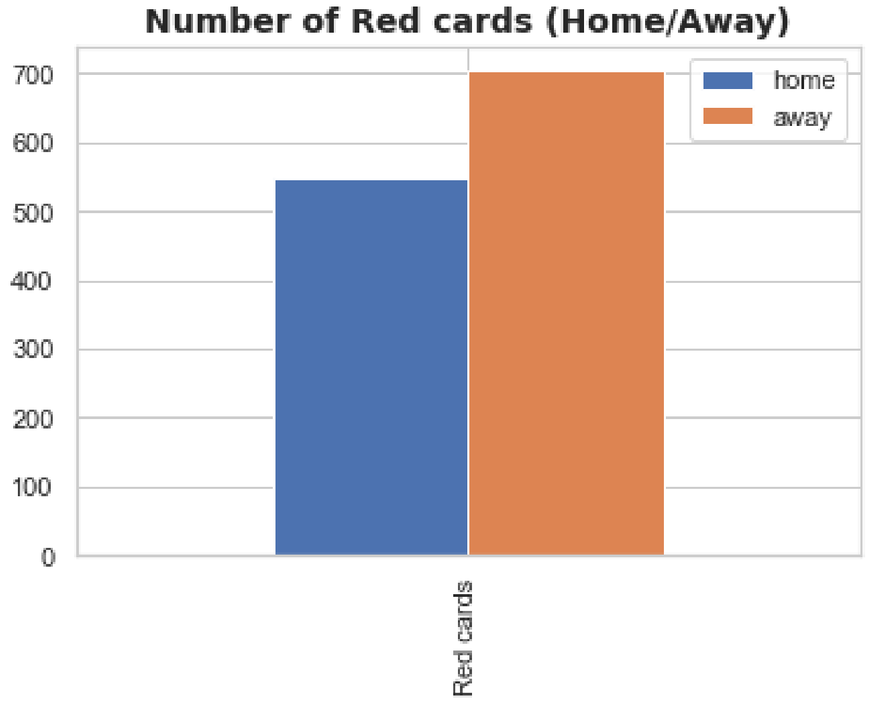
Percentage of goals by location of the shoot



- Most of the goals scored are attempted from around the box (64.5%) and close range (19.4)

Red cards Analysis





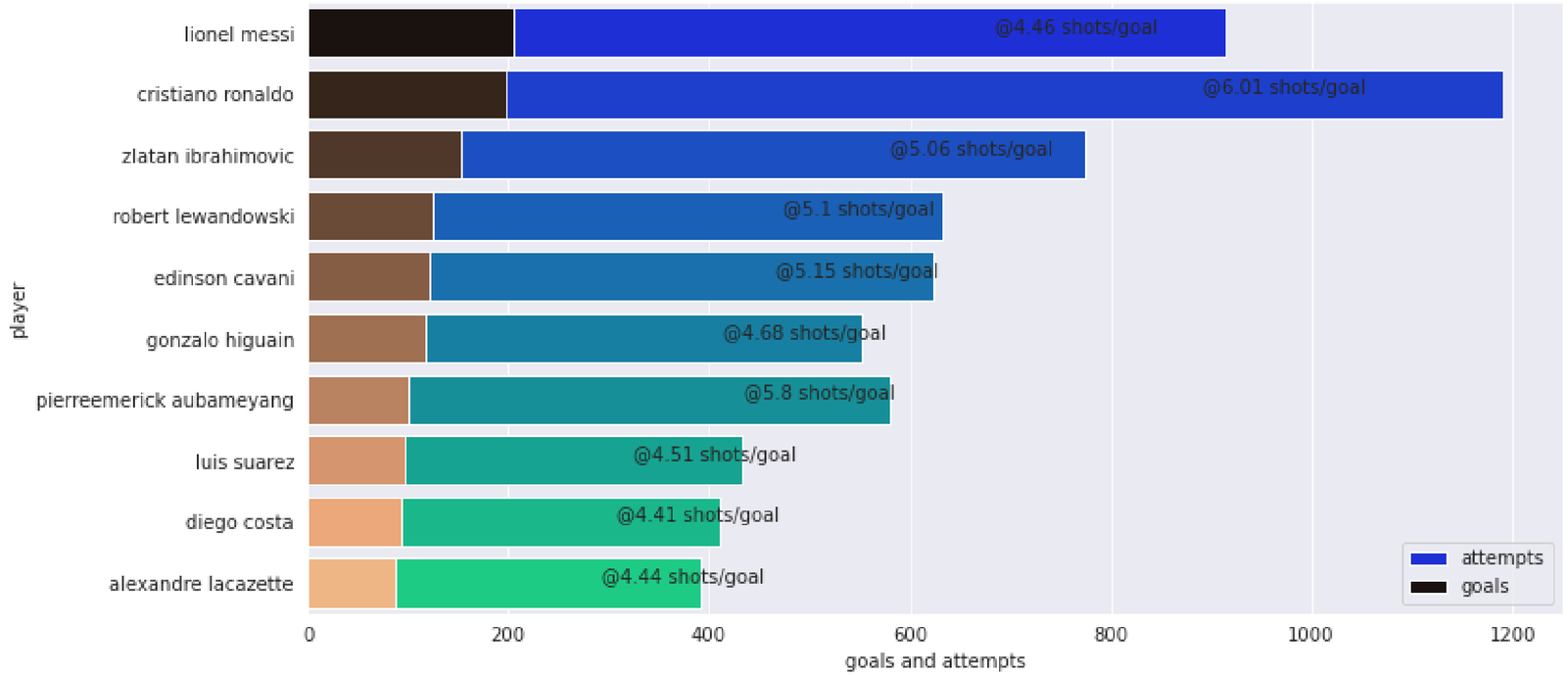
- Clear increasing pattern where the probability of observing a red card is higher as the minute increases
- Sharp increase at about 80mins (similar to yellow cards)

-this makes sense as the players will get more emotional and the atmosphere would be more tense towards the end of the game.

- Similar to yellow cards, more red cards are issued to away teams rather than home teams

Do referee really favor the Home Team or is it just a coincidence?

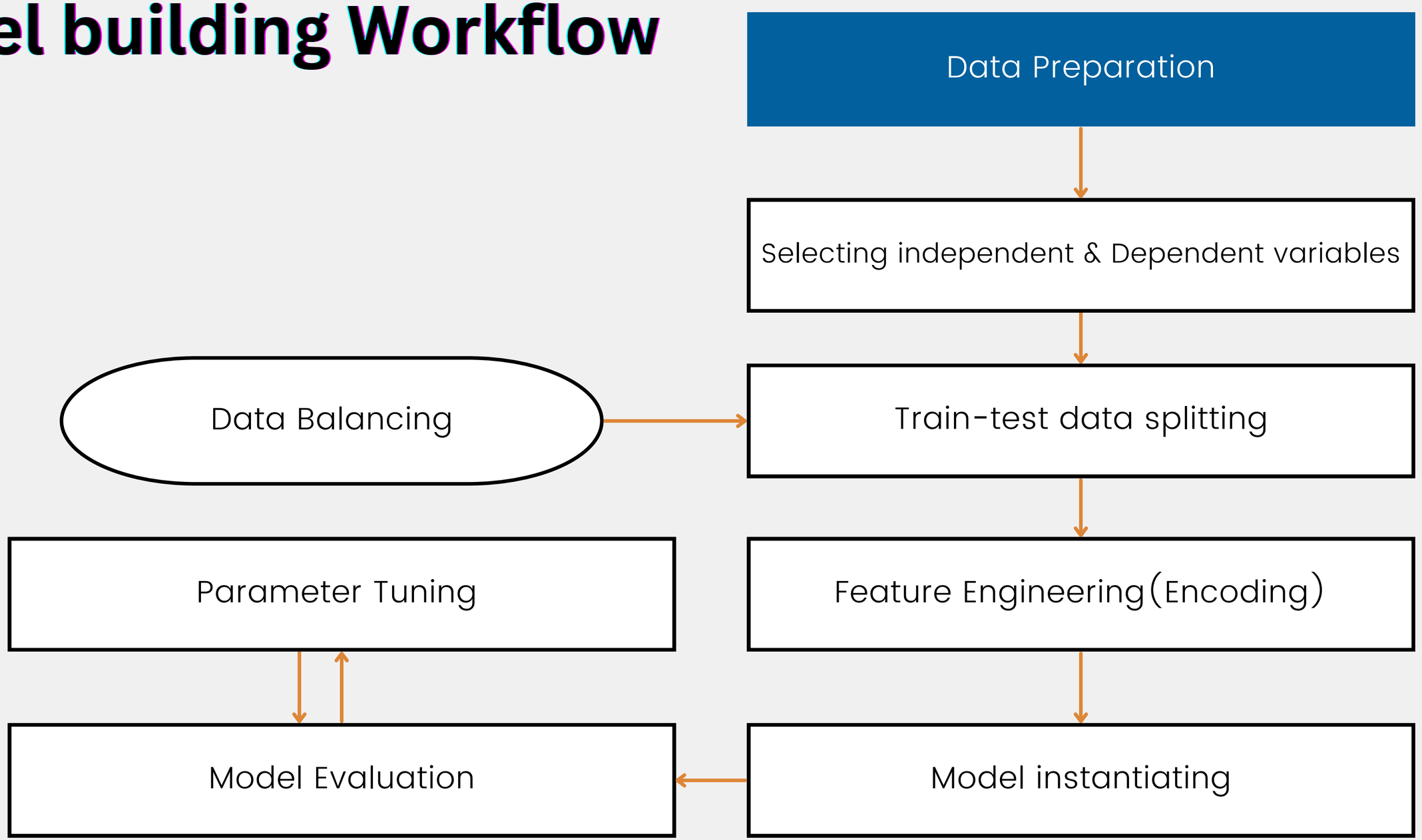
Analysis of Attempts against goals



- From the conversion rates we can see that it takes Messi more than 4 (1/0.224289) attempts on goal to score whereas, it takes Ronaldo 6 attempts to score. Of the 10, Diego Costa has the best conversion rate meaning he is deadliest attacker of the 10

MODEL TRAINING, EVALUATION & VALIDATION

Model building Workflow



Data Preparation

- Defined a new DataFrame to work on.
- Removed rows with all null values(if available).
- Make sure that no Nan values are left in our Features.

BEFORE

```
df.isna().sum()
```

id_odsp	0
id_event	0
sort_order	0
time	0
text	0
event_type	0
event_type2	726716
side	0
event_team	0
opponent	0
player	61000
player2	649699
player_in	889294
player_out	889271
shot_place	713550

AFTER

```
df_cleaned.isnull().sum()
```

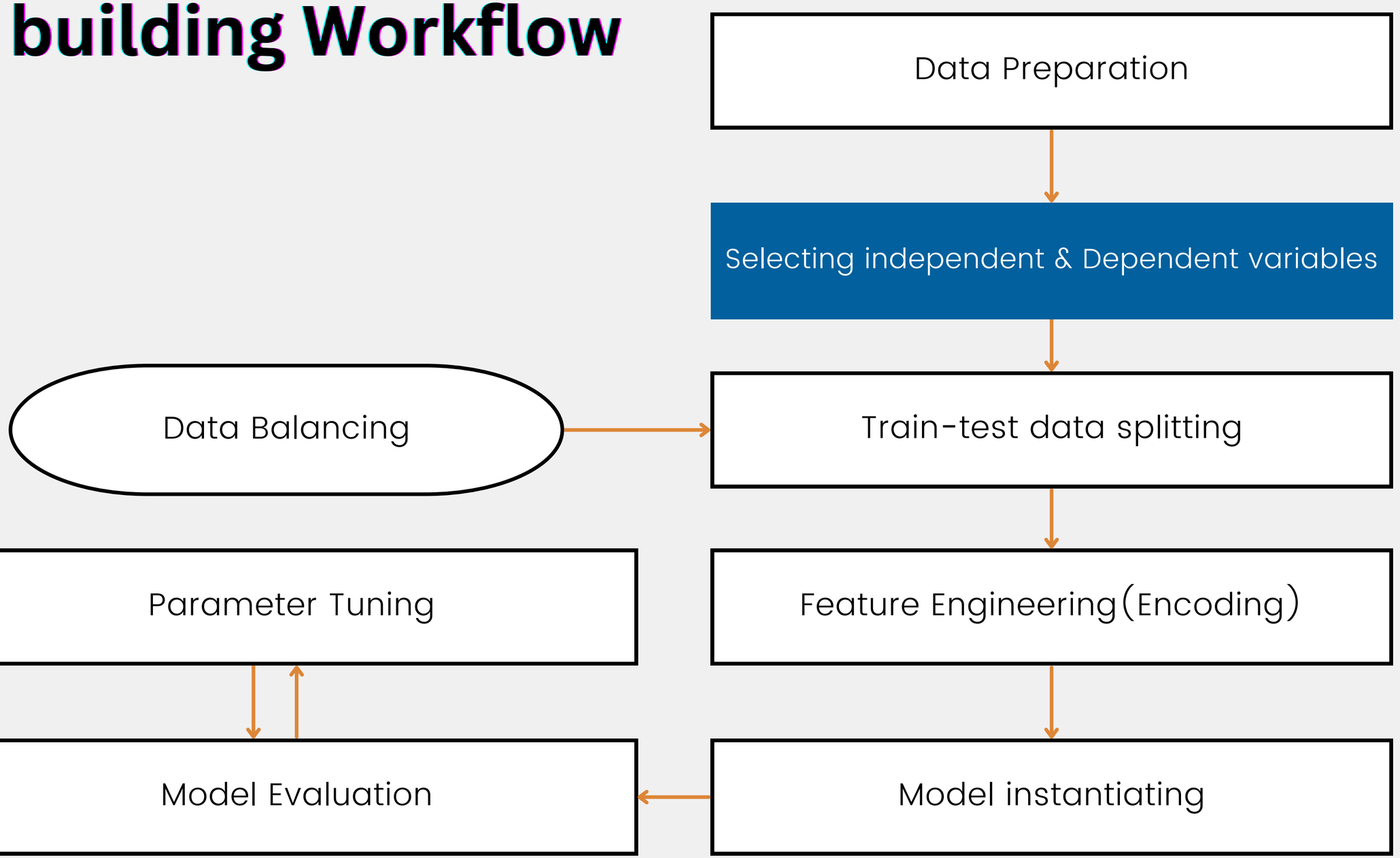
location	0
bodypart	0
assist_method	0
situation	0
time	0
player	0
side	0
shot_place	0
is_goal	0
event_type	0

dtype: int64

```
df_cleaned = df[['location', 'bodypart','assist_method','situation','time','player','side','shot_place','is_goal','event_type']].copy()
df_cleaned.dropna(how='all') # this will remove rows will all null values
df_cleaned.dropna(subset=['player', 'shot_place'], inplace=True)
```

HAMOYE

Model building Workflow



Selecting independent & Dependent variables

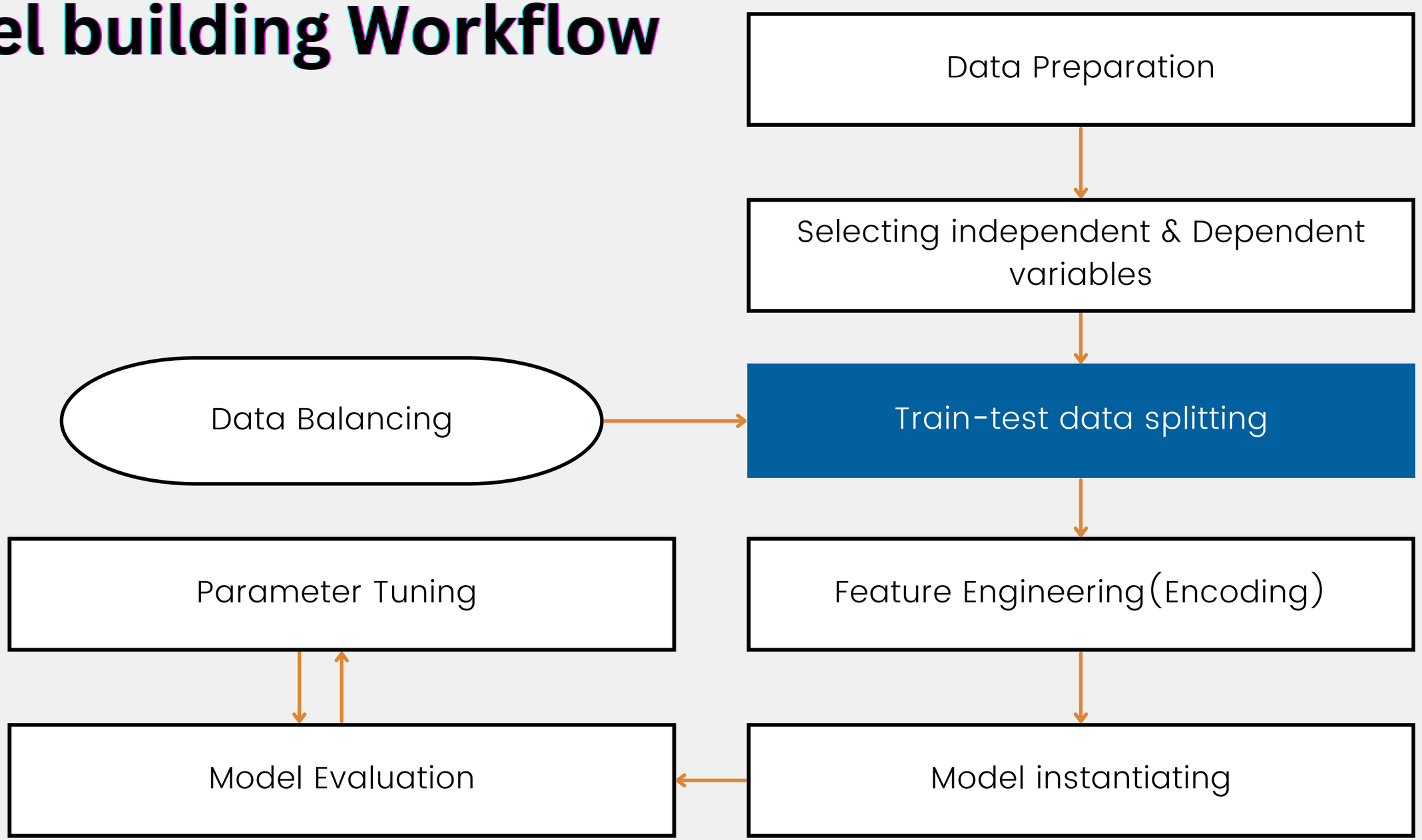
```
x = df_cleaned[['time','location', 'bodypart', 'assist_method', 'situation','side','shot_place']]

y=df_cleaned['is_goal']
```

- The Features are X
- Response feature is Y

- Time (min) when the shot occur.
- Location of the ball on the field.
- Body part used to shoot the ball.
- Assist method that resulted to a goal.
- Situation: Whether it is a Corner, Free kick or an Open play..
- Side: Home or Away.
- Shot place

Model building Workflow



Train-test-split

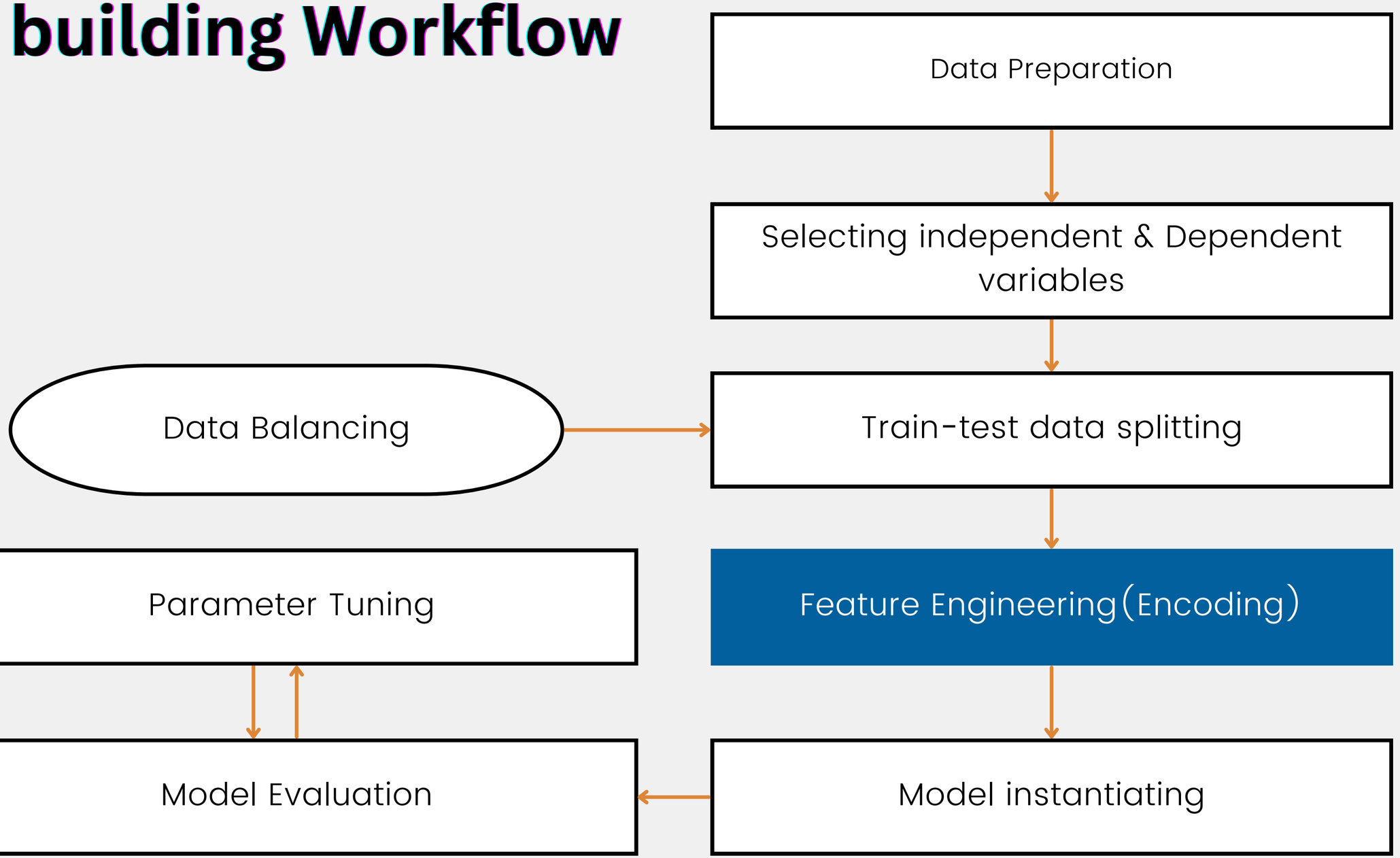
```
from sklearn.model_selection import train_test_split,
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=1)
```

- From Scikit-learn library, train_test_split was imported
- 30% of the data was selected for testing.
- random_state = 1 for reproducibility of the split

HAMOYE

Model building Workflow



Feature Engineering(Encoding)

20 rows | 20 rows × 6 columns | CSV

	locations	bodyparts	assist_methods	situations	sides	shot_places
0	Attacking half	right foot	Pass	Open play	Home	Bit too high
1	Defensive half	left foot	Cross	Set piece	Away	Blocked
2	Centre of the box	head	Headed pass	Corner	NaN	Bottom left corner
3	Left wing	NaN	Through ball	Free kick	NaN	Bottom right corner
4	Right wing	NaN	NaN	NaN	NaN	Centre of the goal
5	Difficult angle and long range	NaN	NaN	NaN	NaN	High and wide
6	Difficult angle on the left	NaN	NaN	NaN	NaN	Hits the bar
7	Difficult angle on the right	NaN	NaN	NaN	NaN	Misses to the left
8	Left side of the box	NaN	NaN	NaN	NaN	Misses to the right
9	Left side of the six yard box	NaN	NaN	NaN	NaN	Too high
10	Right side of the box	NaN	NaN	NaN	NaN	Top centre of the goal
11	Right side of the six yard box	NaN	NaN	NaN	NaN	Top left corner
12	Very close range	NaN	NaN	NaN	NaN	Top right corner
13	Penalty spot	NaN	NaN	NaN	NaN	NaN
14	Outside the box	NaN	NaN	NaN	NaN	NaN
15	Long range	NaN	NaN	NaN	NaN	NaN
16	More than 35 yards	NaN	NaN	NaN	NaN	NaN
17	More than 40 yards	NaN	NaN	NaN	NaN	

HAMOYE

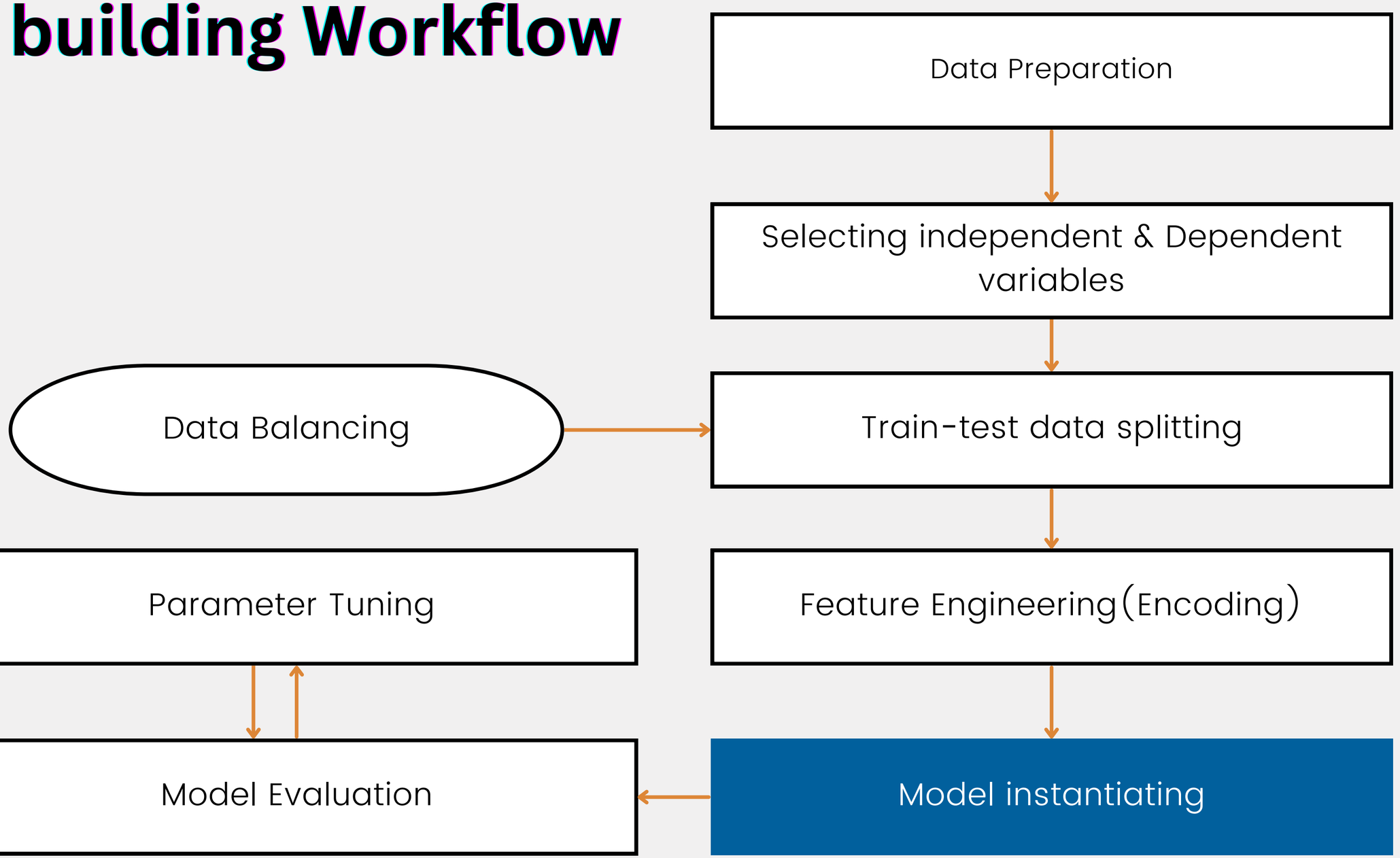
Feature Engineering(Encoding)

- One-hot encoding : not alot of classes that would make dimensionality curse.
- get_dummies() was used from pandas library.

```
x_train = pd.get_dummies(x_train, columns=[
                  'location', 'bodypart', 'assist_method', 'situation','side','shot_place'], drop_first=True)
x_test = pd.get_dummies(x_test, columns=[
                  'location', 'bodypart', 'assist_method', 'situation','side','shot_place'], drop_first=True)

y=df_cleaned['is_goal']
```

Model building Workflow



Model instantiating

1. Logistic Regression

```
lr= LogisticRegression(n_jobs=-1)
lr.fit(x_train, y_train)
y_pred_lr = lr.predict(x_test)
```

```
LogisticRegression(n_jobs=-1) scores are:
Accuracy: 0.935
AUC: 0.777
f1: 0.642
And the confusion matrix is:
 [[39924  1022]
 [ 1916  2629]]

LogisticRegression(n_jobs=-1) predicted 39924 True positives (actual goals) and 2629 True negatives (not goals).
And a total of 2938 False positives & negatives.
It means 93.5% of times the model predicted a shot result correctly.
------------------------------
```

Model instantiating

2. Decision-Tree classifier

```
dt = DecisionTreeClassifier()
%time dt.fit(x_train, y_train)
y_pred_dt = dt.predict(x_test)
```

```
DecisionTreeClassifier() scores are:
Accuracy: 0.923
AUC: 0.74
f1: 0.569
And the confusion matrix is:
 [[39650  1296]
 [ 2220  2325]]

DecisionTreeClassifier() predicted 39650 True positives (actual goals) and 2325 True negatives (not goals).
And a total of 3516 False positives & negatives.
It means 92.3% of times the model predicted a shot result correctly.
------------------------------
```

Model instantiating

3. Random Forest

```
rf = RandomForestClassifier(n_jobs=-1, random_state=1)
%time  rf.fit(x_train, y_train)
y_pred_rf = rf.predict(x_test)
```

```
RandomForestClassifier(n_jobs=-1, random_state=1) scores are:
Accuracy: 0.924
AUC: 0.758
f1: 0.591
And the confusion matrix is:
 [[39532  1414]
 [ 2043  2502]]

RandomForestClassifier(n_jobs=-1, random_state=1) predicted 39532 True positives (actual goals) and 2502 True negatives (not goals).
And a total of 3457 False positives & negatives.
It means 92.4% of times the model predicted a shot result correctly.
------------------------------
```

Model instantiating

4. Gradient Boosting

```
gbc = GradientBoostingClassifier(n_estimators=200, max_depth=6, random_state=1)
%time gbc.fit(x_train, y_train)
y_pred_gbc = gbc.predict(x_test)
```

```
Accuracy: 0.936
AUC: 0.785
f1: 0.652
And the confusion matrix is:
 [[39895  1051]
 [ 1838  2707]]

GradientBoostingClassifier(max_depth=6, n_estimators=200, random_state=1) predicted 39895 True positives (actual goals) and 2707 True
 negatives (not goals).
And a total of 2889 False positives & negatives.
It means 93.6% of times the model predicted a shot result correctly.
------------------------------
```

Model instantiating

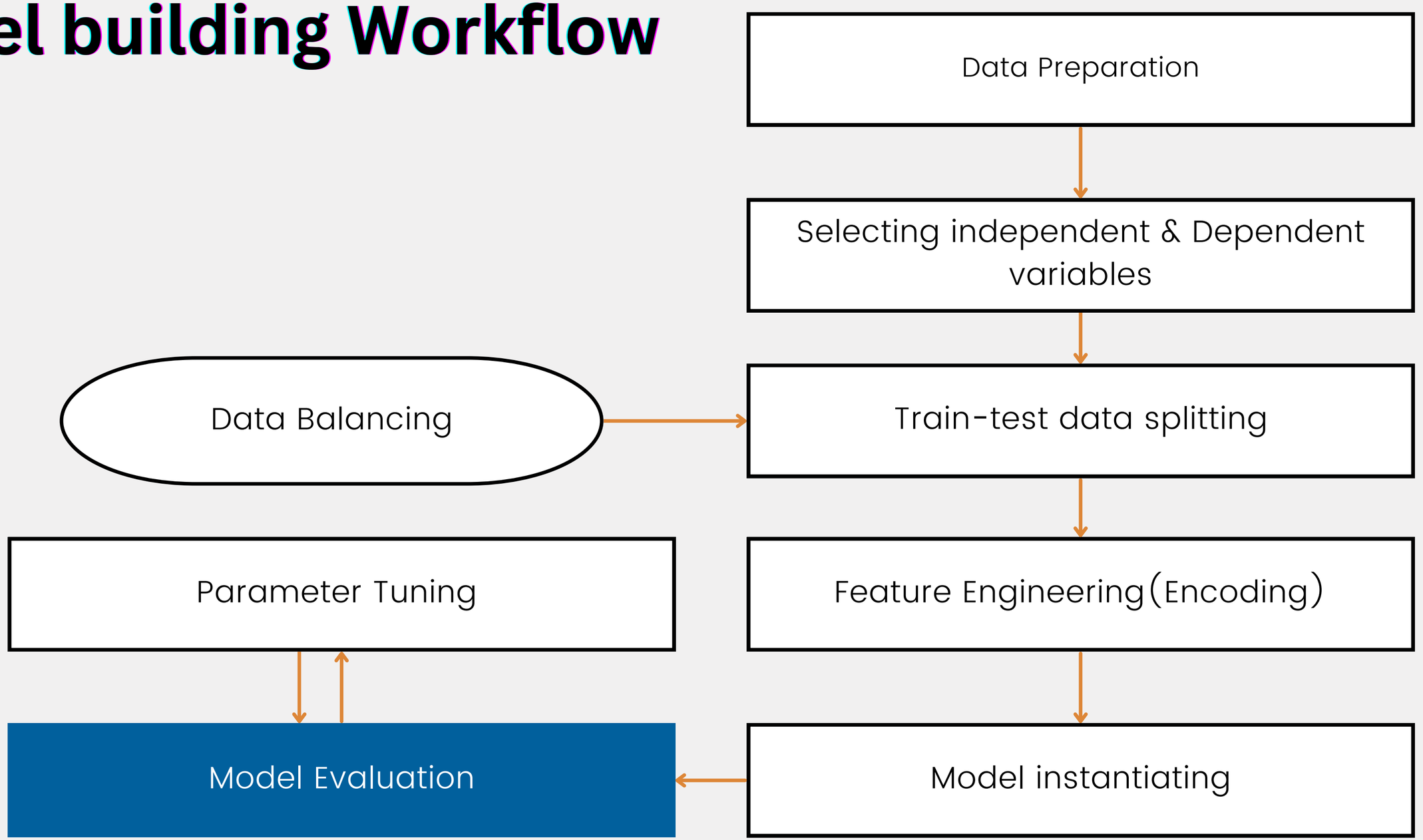
Model scores comparison:

	Model	Accuracy	f1 score	AUC score
3	GradientBoostingClassifier(max_depth=6, n_esti...	93.65	0.652	0.785
0	LogisticRegression(n_jobs=-1)	93.54	0.642	0.777
2	RandomForestClassifier(n_jobs=-1, random_state=1)	92.40	0.591	0.758
1	DecisionTreeClassifier()	92.27	0.569	0.740

- Gradiant Boosting performed the best!

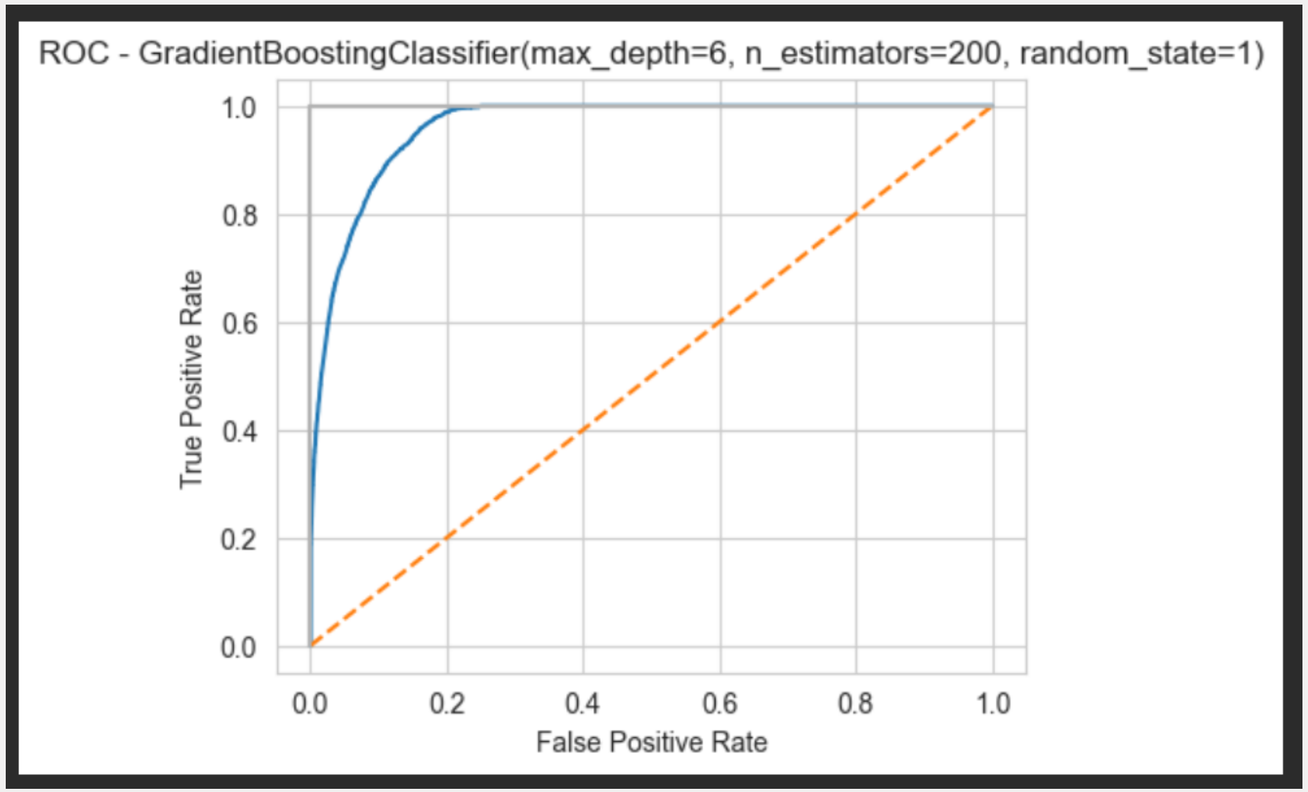
HAMOYE

Model building Workflow



Model Evaluating

ROC Curve:

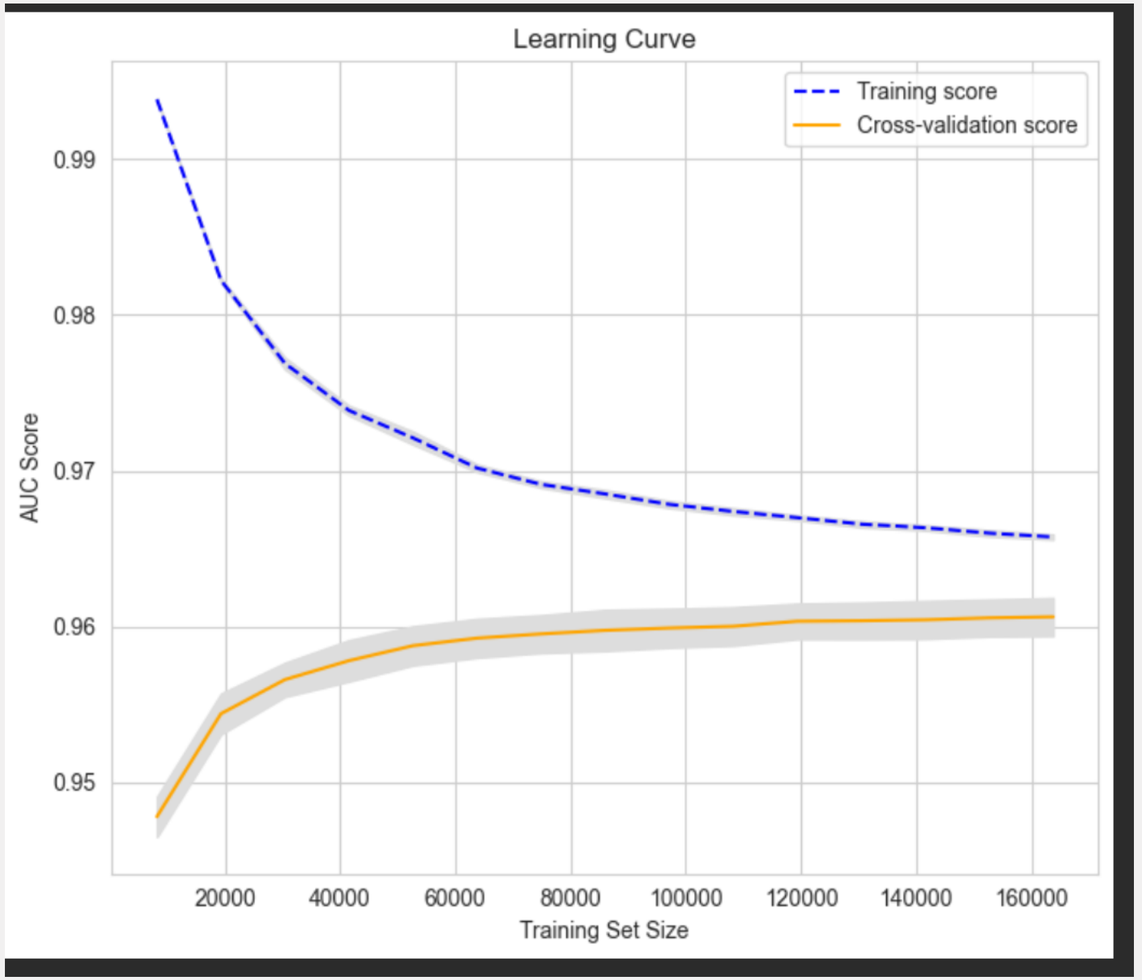


- We can observe that it has a very good ROC score (High true positive, Low False Positive)

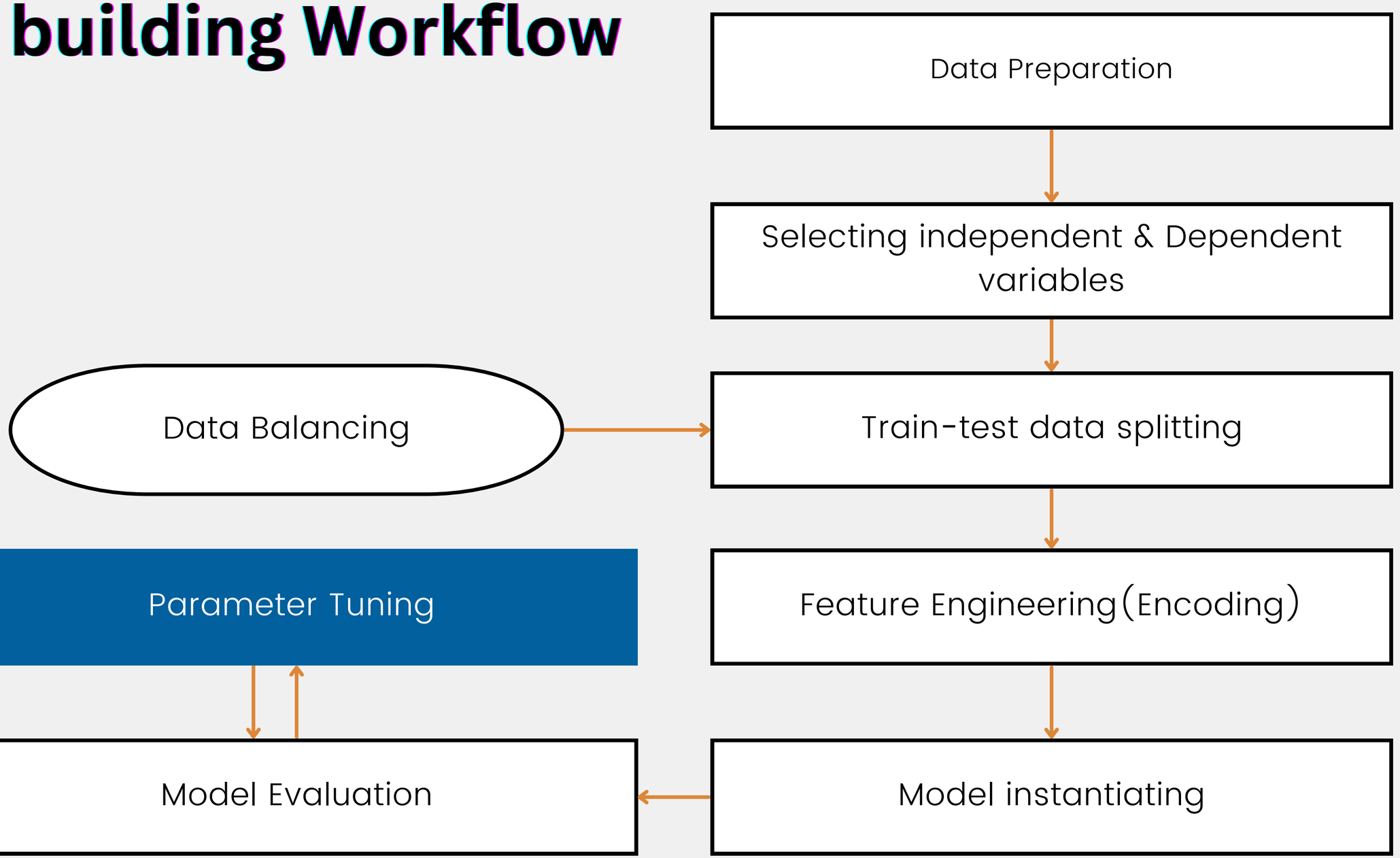
Model Evaluating

Learning curve:

- The model accuracy steadies around 0.96 with training set size ~ 80000



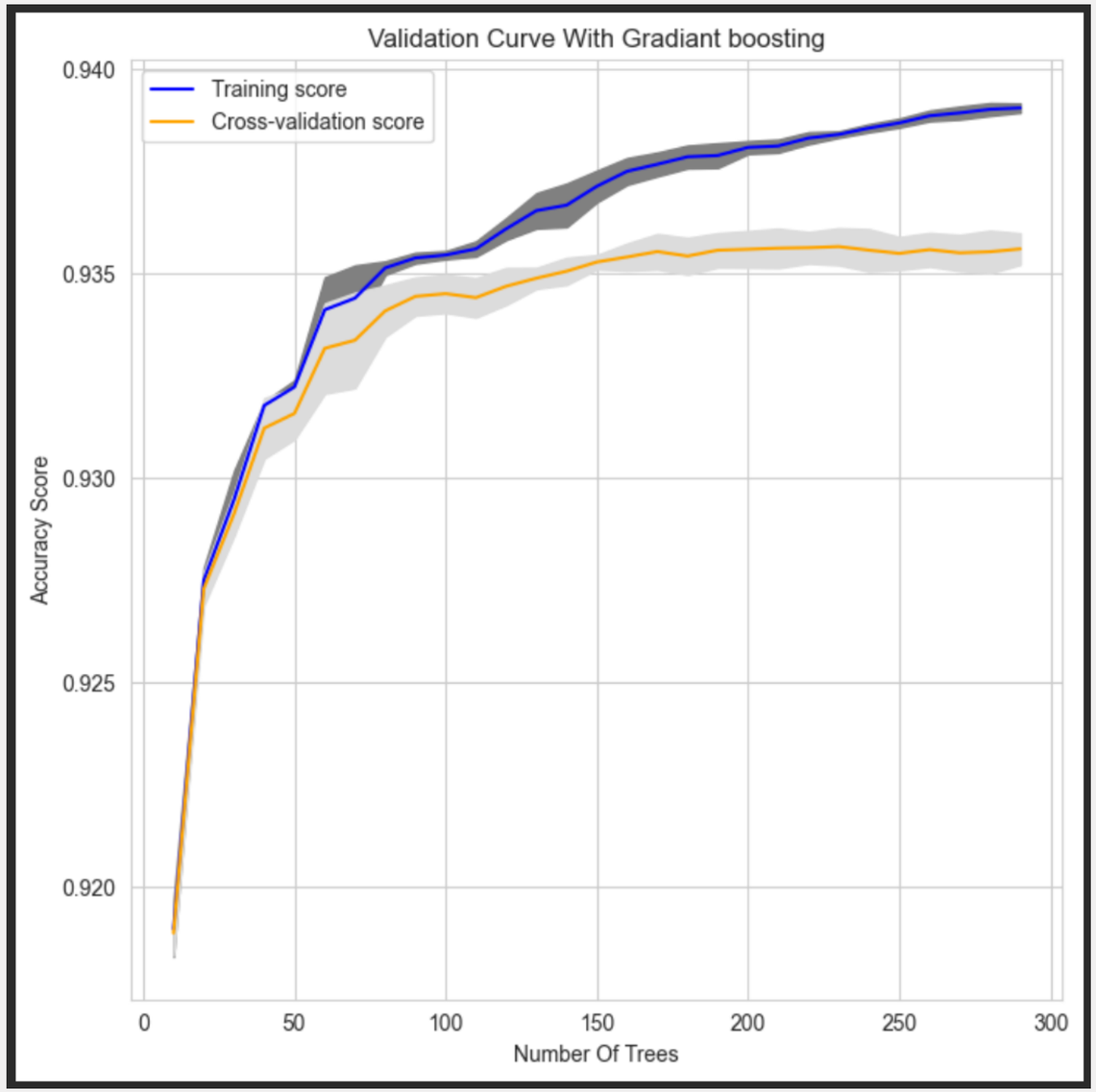
Model building Workflow



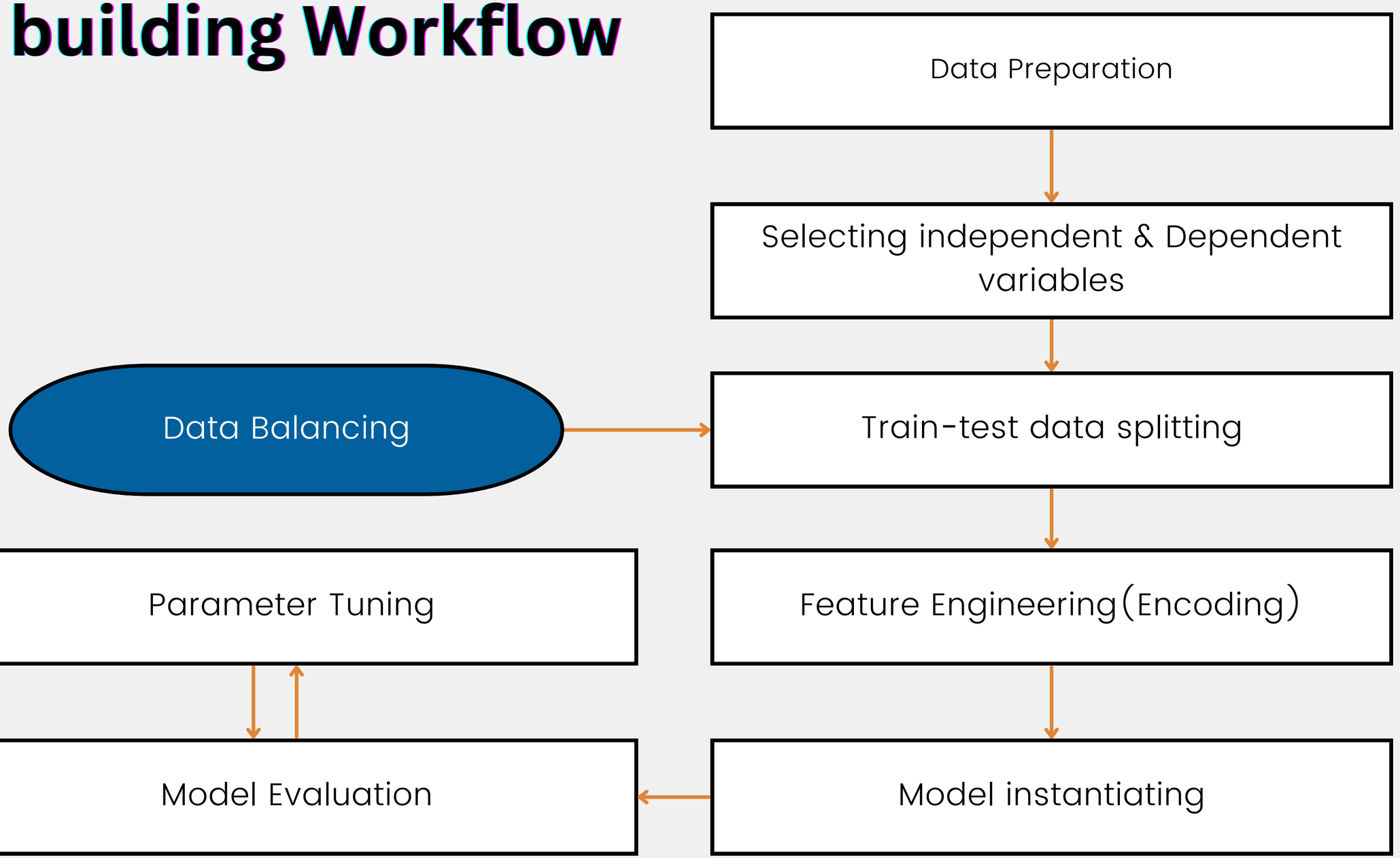
Parameter Tuning

- We tried to find the best n_estimators(Number of trees) For Gradient Boosting classifier.

- From the plot we observed that a value of 200 trees is the best.



Model building Workflow



Data Balancing

- Checking for data unbalance

```
scored = df_cleaned[df_cleaned['is_goal'] == 1]['is_goal'].sum()
all_shots = df_cleaned.shape[0]
percent = round(scored/all_shots * 100, 2)
print('The total shots of the dataset were {} . While goals were {}'.format(
    all_shots, scored))
print('And that makes {}% of the shots where goals.'.format(percent))
```

```
The total shots of the dataset were 227451 . While goals were 22770
And that makes 10.01% of the shots where goals.
```

  - Only 10 % of shots were goals. which is unbalanced.

Data Balancing

- Making the dataset a bit balanced by sampling 50000 from non-goal values and concatinating it with goals.

```
scored = new_train[new_train['is_goal'] == 1]['is_goal'].sum()
all_shots = new_train.shape[0]
percent = round(scored/all_shots * 100, 2)
print('The total shots of the dataset were {} . While goals were {}'.format(
    all_shots, scored))
print('And that makes {}% of the shots where goals.'.format(percent))
```

```
 The total shots of the dataset were 72770 . While goals were 22770
 And that makes 31.29% of the shots where goals.


now its better... lets see the result..
```

- Now 31% of shots were goals. which is better than before...Let's see now how it will perform.

HAMOYE

Balanced Model scores

4 rows | 4 rows × 4 columns

	Model	Accuracy	f1 score	AUC score
3	GradientBoostingClassifier(max_depth=5, n_esti...	88.97	0.827	0.879
0	LogisticRegression(n_jobs=-1)	88.48	0.820	0.874
2	RandomForestClassifier(n_jobs=-1, random_state=1)	86.48	0.785	0.846
1	DecisionTreeClassifier(random_state=1)	85.67	0.765	0.828

Unbalanced Model scores

4 rows | 4 rows × 4 columns

	Model	Accuracy	f1 score	AUC score
3	GradientBoostingClassifier(max_depth=5, n_esti...	93.67	0.651	0.783
0	LogisticRegression(n_jobs=-1)	93.54	0.642	0.777
2	RandomForestClassifier(n_jobs=-1, random_state=1)	92.40	0.591	0.758
1	DecisionTreeClassifier(random_state=1)	92.27	0.568	0.739

- Quite good improvement indeed!! **12%** increase in AUC score.

HAMOYE

Feature Importance

- shot place got the highest feature importance..
- Location is second most importance variable.

Weight	Feature
0.1770 ± 0.1037	shot_place_4.0
0.1675 ± 0.1146	shot_place_3.0
0.1006 ± 0.0746	location_15.0
0.0658 ± 0.1047	shot_place_5.0
0.0649 ± 0.0558	shot_place_12.0
0.0620 ± 0.0636	situation_2.0
0.0615 ± 0.0675	shot_place_13.0
0.0591 ± 0.0845	situation_3.0
0.0549 ± 0.0763	location_13.0
0.0344 ± 0.0687	location_19.0
0.0331 ± 0.0717	location_9.0
0.0232 ± 0.0544	location_11.0
0.0206 ± 0.3711	time
0.0148 ± 0.0657	location_14.0
0.0080 ± 0.0763	bodypart_3.0
0.0077 ± 0.0667	assist_method_4
0.0063 ± 0.0513	location_8.0
0.0050 ± 0.0368	location_7.0
0.0043 ± 0.0354	location_16.0
0.0039 ± 0.0420	location_6.0
… *17 more* …	

RESULTS

- Gradient Boosting were selected as it made a highly accurate (88%) model for goal prediction.
- Balancing the dataset resulted in 12% increase in accuracy.
- Shot place has the highest effect on goal prediction.

HAMOYE

RECOMMENDATION & CONCLUSION

- Next, adding Player or the team as a feature. Before adding them a performance index or a weighting factor should be calculated first.
- Try using other models or apply hyperparameter tuning to all models as it may change the accuracy for some models and for some not.
- Adding the goal keeper (his performance)as a feature.

HAMOYE

Thank you.