Project Stage – 4

Surya Pratap Singh Tanwar, Yash Trivedi, Yugali Gullapalli

Merging Tables –

In order to combine tables A and B, we followed these steps –

I. applied the trained matcher from stage 3 on the entire candidate dataset obtained after blocking.
II. If the matcher predicted a '1' i.e., if two tuples were predicted to be a match, we applied a set of rules to merge the tuples.
III. The following table lists the rules used to different types of attributes –

Attribute Name	Attribute Type	Rule
full name	String	Take the longer of two names from A and B
name	String	Initially decided to take the longer of the two names from A and B, then realized that due to bad Unicode encoding, data in B was longer, hence always chose value from A
club	String	Take the longer of two club names from A and B
nation	String	Take the longer of two country names from A and B
overall	Integer	Take the ceil of the mean of the values of overall from A and B
league	String	Take the longer of two league names from A and B
age	Integer	Take the greater of the two age values from A and B
height	Integer	Take the greater of the two height values from A and B
reactions	Integer	Take the ceil of the mean of the values of reactions from A and B
positioning	Integer	Take the ceil of the mean of the values of positioning from A and B
penalties	Integer	Take the ceil of the mean of the values of penalties from A and B
long passing	Integer	Take the ceil of the mean of the values of long passing from A and B
standing tackle	Integer	Take the ceil of the mean of the values of standing tackle from A and B
ball control	Integer	Take the ceil of the mean of the values of ball control from A and B
interceptions	Integer	Take the ceil of the mean of the values of interceptions from A and B
acceleration	Integer	Take the ceil of the mean of the values of acceleration from A and B
marking	Integer	Take the ceil of the mean of the values of marking from A and B

strength	Integer	Take the ceil of the mean of the values of strength from A and B
pac	Integer	Take the ceil of the mean of the values of pac from A and B
aggression	Integer	Take the ceil of the mean of the values of aggression from A and B
agility	Integer	Take the ceil of the mean of the values of agility from A and B
volleys	Integer	Take the ceil of the mean of the values of volleys from A and B
sprint speed	Integer	Take the ceil of the mean of the values of sprint speed from A and B
phy	Integer	Take the ceil of the mean of the values of phy from A and B
pas	Integer	Take the ceil of the mean of the values of pas from A and B
sliding tackle	Integer	Take the ceil of the mean of the values of sliding tackle from A and B
long shots	Integer	Take the ceil of the mean of the values of long shots from A and B
short passing	Integer	Take the ceil of the mean of the values of short passing from A and B
shot power	Integer	Take the ceil of the mean of the values of shot power from A and B
curve	Integer	Take the ceil of the mean of the values of curve from A and B
dri	Integer	Take the ceil of the mean of the values of dri from A and B
jumping	Integer	Take the ceil of the mean of the values of jumping from A and B
vision	Integer	Take the ceil of the mean of the values of vision from A and B
crossing	Integer	Take the ceil of the mean of the values of crossing from A and B
finishing	Integer	Take the ceil of the mean of the values of finishing from A and B
balance	Integer	Take the ceil of the mean of the values of balance from A and B
heading	Integer	Take the ceil of the mean of the values of heading from A and B
defending	Integer	Take the ceil of the mean of the values of defending from A and B
sho	Integer	Take the ceil of the mean of the values of sho from A and B
free kick accuracy	Integer	Take the ceil of the mean of the values of free kick accuracy from A and B
composure	Integer	Take the ceil of the mean of the values of composure from A and B
strong foot	String	Take the value from A since it is more authoritative with respect to the strong foot attribute
dribbling	Integer	Take the ceil of the mean of the values of dribbling from A and B
stamina	Integer	Take the ceil of the mean of the values of stamina from A and B

Statistics on Table E –

Number of tuples – 3604

Schema –

Attribute Name	Attribute Type
full name	String
name	String
club	String
nation	String
overall	Integer
league	String
age	Integer
height	Integer
reactions	Integer
positioning	Integer
penalties	Integer
long passing	Integer
standing tackle	Integer
ball control	Integer
interceptions	Integer
acceleration	Integer
marking	Integer
strength	Integer
pac	Integer
aggression	Integer
agility	Integer
volleys	Integer
sprint speed	Integer
phy	Integer
pas	Integer
sliding tackle	Integer
long shots	Integer
short passing	Integer
shot power	Integer
curve	Integer
dri	Integer
jumping	Integer
vision	Integer
crossing	Integer
finishing	Integer
balance	Integer
heading	Integer
defending	Integer
sho	Integer
free kick accuracy	Integer
composure	Integer
strong foot	String
dribbling	Integer
stamina	Integer

Examples –


```
{
   "acceleration": 89,
   "age": 33,
   "aggression": 63,
   "agility": 89,
   "balance": 63,
   "ball control": 93,
   "club": "Real Madrid Cf",
   "composure": 95,
   "crossing": 85,
   "curve": 81,
   "defending": 33,
   "dri": 90,
   "dribbling": 91,
   "finishing": 94,
   "free kick accuracy": 76,
   "full name": "C. Ronaldo Dos Santos Aveiro",
   "heading": 88,
   "height": 185,
   "interceptions": 29,
   "jumping": 95,
   "league": "Laliga Santander",
   "long passing": 77,
   "long shots": 92,
   "marking": 22,
   "name": "Cristiano Ronaldo",
   "nation": "Portugal",
   "overall": 94,
   "pac": 90,
   "pas": 82,
   "penalties": 85,
   "phy": 80,
   "positioning": 95,
   "reactions": 96,
   "sho": 93,
   "short passing": 83,
   "shot power": 94,
   "sliding tackle": 23,
   "sprint speed": 91,
   "stamina": 92,
   "standing tackle": 31,
   "strength": 80,
   "strong foot": "Right",
   "vision": 85,
   "volleys": 88
 }
```

```
{
  "acceleration": 93,
  "age": 27,
  "aggression": 54,
  "agility": 93,
  "balance": 91,
  "ball control": 92,
  "club": "Chelsea",
  "composure": 87,
  "crossing": 80,
  "curve": 82,
  "defending": 32,
  "dri": 92,
  "dribbling": 93,
  "finishing": 83,
  "free kick accuracy": 79,
  "full name": "Eden Hazard",
  "heading": 57,
  "height": 173,
  "interceptions": 41,
  "jumping": 59,
  "league": "Premier League",
  "long passing": 81,
  "long shots": 82,
  "marking": 25,
  "name": "Hazard",
  "nation": "Belgium",
  "overall": 90,
  "pac": 90,
  "pas": 84,
  "penalties": 86,
  "phy": 66,
  "positioning": 85,
  "reactions": 85,
  "sho": 82,
  "short passing": 86,
  "shot power": 79,
  "sliding tackle": 22,
  "sprint speed": 87,
  "stamina": 79,
  "standing tackle": 27,
  "strength": 65,
  "strong foot": "Right",
  "vision": 86,
  "volleys": 79
 }
```

```
{
  "acceleration": 77,
  "age": 26,
  "aggression": 68,
  "agility": 81,
  "balance": 76,
  "ball control": 89,
  "club": "Manchester City",
  "composure": 84,
  "crossing": 92,
  "curve": 84,
  "defending": 48,
  "dri": 87,
  "dribbling": 87,
  "finishing": 83,
  "free kick accuracy": 84,
  "full name": "Kevin De Bruyne",
  "heading": 56,
  "height": 181,
  "interceptions": 59,
  "jumping": 65,
  "league": "Premier League",
  "long passing": 85,
  "long shots": 87,
  "marking": 32,
  "name": "De Bruyne",
  "nation": "Belgium",
  "overall": 91,
  "pac": 76,
  "pas": 90,
  "penalties": 77,
  "phy": 76,
  "positioning": 85,
  "reactions": 90,
  "sho": 85,
  "short passing": 92,
  "shot power": 86,
  "sliding tackle": 42,
  "sprint speed": 76,
  "stamina": 88,
  "standing tackle": 54,
  "strength": 73,
  "strong foot": "Right",
  "vision": 92,
  "volleys": 82
 }
```

```
{
  "acceleration": 93,
  "age": 30,
  "aggression": 60,
  "agility": 95,
  "balance": 93,
  "ball control": 88,
  "club": "Napoli",
  "composure": 81,
  "crossing": 78,
  "curve": 83,
  "defending": 36,
  "dri": 90,
  "dribbling": 90,
  "finishing": 86,
  "free kick accuracy": 78,
  "full name": "Dries Mertens",
  "heading": 35,
  "height": 171,
  "interceptions": 36,
  "jumping": 61,
  "league": "Calcio A",
  "long passing": 75,
  "long shots": 81,
  "marking": 30,
  "name": "Mertens",
  "nation": "Belgium",
  "overall": 87,
  "pac": 90,
  "pas": 81,
  "penalties": 78,
  "phy": 56,
  "positioning": 86,
  "reactions": 85,
  "sho": 83,
  "short passing": 83,
  "shot power": 80,
  "sliding tackle": 40,
  "sprint speed": 87,
  "stamina": 76,
  "standing tackle": 40,
  "strength": 42,
  "strong foot": "Right",
  "vision": 84,
  "volleys": 70
 }
```

Python Script –

```
import pandas as pd
import math

count = 0
matches = list()
for i in range(len(predictions)):
   if predictions.iloc[i]['predicted'] == 1.0:
      count += 1
      #print A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['full name'] + " === " +
B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['full_name']

      match = dict()

      # New full name
      if len(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['full name']) >
len(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['full_name']):
         match['full name'] = str(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['full name']).title()
      else:
         match['full name'] = str(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['full_name']).title()

      # New name - names in B contained non-alphabetical characters due to translation from Unicode.
      match['name'] = str(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['name']).title()

      # New club
      if len(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['club']) >
len(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['club']):
         match['club'] = str(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['club']).title()
      else:
         match['club'] = str(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['club']).title()

      # New nation
      if len(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['nation']) >
len(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['nation']):
         match['nation'] = str(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['nation']).title()
      else:
         match['nation'] = str(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['nation']).title()

      # New overall
      match['overall'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['overall']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['overall'])) / 2)

      # New league
      if len(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['league']) >
len(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['league']):
         match['league'] = str(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['league']).title()
      else:
         match['league'] = str(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['league']).title()

      # New age
```

```
    match['age'] = int(max(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['age']),
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['age']))

    # New height
    match['height'] = int(max(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['height']),
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['height']))

    # New reactions
    match['reactions'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['reactions']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['reactions'])) / 2)

    # New positioning
    match['positioning'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['positioning']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['positioning'])) / 2)

    # New penalties
    match['penalties'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['penalties']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['penalties'])) / 2)

    # New long passing
    match['long passing'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['long passing']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['long passing'])) / 2)

    # New standing tackle
    match['standing tackle'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['standing tackle']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['standing tackle'])) / 2)

    # New ball control
    match['ball control'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['ball control']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['ball control'])) / 2)

    # New interceptions
    match['interceptions'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['interceptions']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['interceptions'])) / 2)

    # New acceleration
    match['acceleration'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['acceleration']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['acceleration'])) / 2)

    # New marking
    match['marking'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['marking']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['marking'])) / 2)

    # New strength
    match['strength'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['strength']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['strength'])) / 2)

    # New pac
    match['pac'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['pac']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['pac'])) / 2)

    # New aggression
```

```
    match['aggression'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['aggression']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['aggression'])) / 2)

    # New agility
    match['agility'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['agility']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['agility'])) / 2)

    # New volleys
    match['volleys'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['volleys']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['volleys'])) / 2)

    # New sprint speed
    match['sprint speed'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['sprint speed']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['sprint speed'])) / 2)

    # New phy
    match['phy'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['phy']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['phy'])) / 2)

    # New pas
    match['pas'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['pas']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['pas'])) / 2)

    # New sliding tackle
    match['sliding tackle'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['sliding tackle']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['sliding tackle'])) / 2)

    # New long shots
    match['long shots'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['long_shots']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['long shots'])) / 2)

    # New short passing
    match['short passing'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['short passing']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['short passing'])) / 2)

    # New shot power
    match['shot power'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['shot power']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['shot power'])) / 2)

    # New curve
    match['curve'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['curve']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['curve'])) / 2)

    # New dri
    match['dri'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['dri']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['dri'])) / 2)

    # New jumping
    match['jumping'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['jumping']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['jumping'])) / 2)

    # New vision
```

```
        match['vision'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['vision']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['vision'])) / 2)

        # New crossing
        match['crossing'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['crossing']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['crossing'])) / 2)

        # New finishing
        match['finishing'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['finishing']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['finishing'])) / 2)

        # New balance
        match['balance'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['balance']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['balance'])) / 2)

        # New heading
        match['heading'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['heading accuracy']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['heading'])) / 2)

        # New defending
        match['defending'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['defending']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['def'])) / 2)

        # New sho
        match['sho'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['sho']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['sho'])) / 2)

        # New free kick accuracy
        match['free kick accuracy'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['free kick accuracy']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['free kick'])) / 2)

        # New composure
        match['composure'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['composure']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['composure'])) / 2)

        # New strong foot
        match['strong foot'] = str(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['foot']).title()

        # New dribbling
        match['dribbling'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['dribbling']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['dribbling'])) / 2)

        # New stamina
        match['stamina'] = int(math.ceil(int(A.iloc[(int(predictions.iloc[i]['ltable_ID']))]['stamina']) +
int(B.iloc[(int(predictions.iloc[i]['rtable_ID']))]['stamina'])) / 2)

        matches.append(match)

print count

E = pd.DataFrame(matches)
E
```

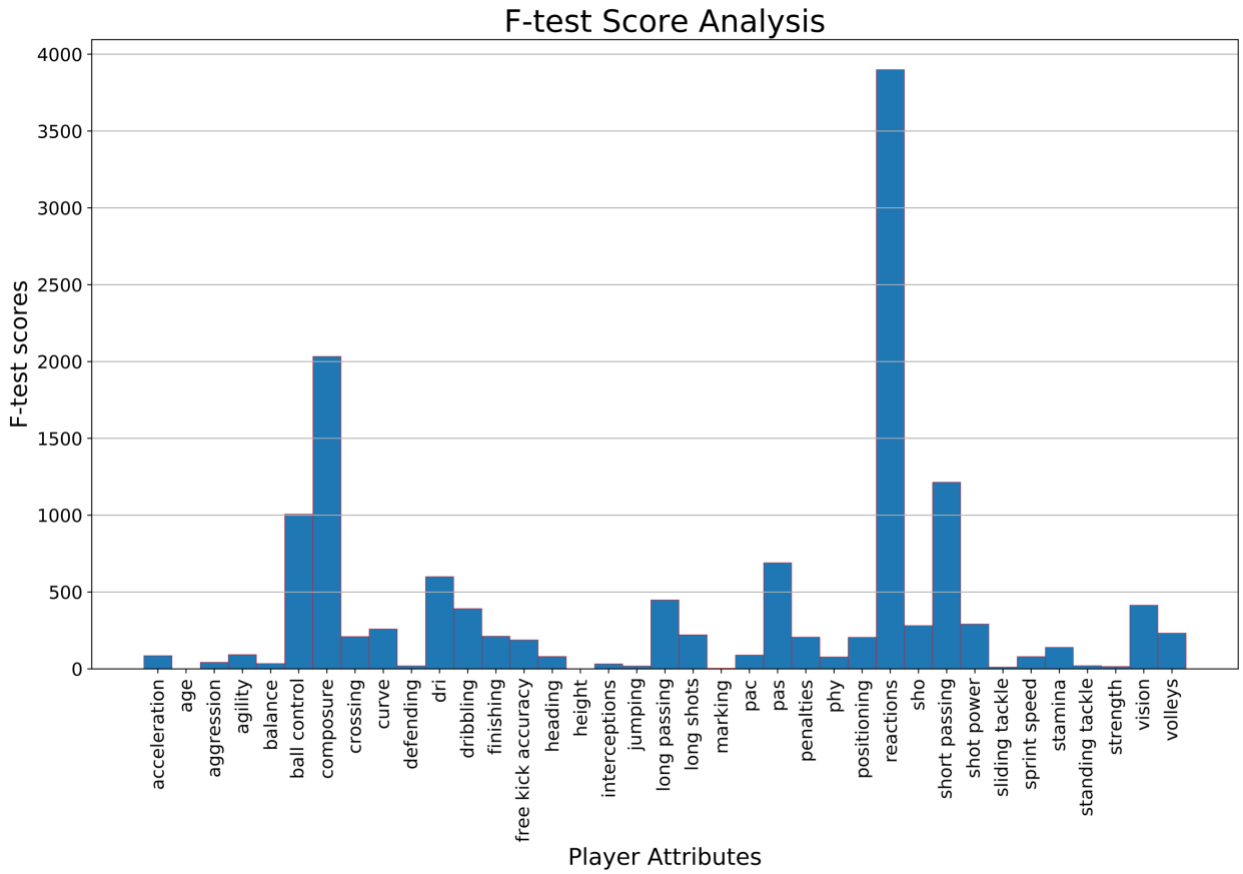
Data Analysis – Correlation Analysis

We wanted to investigate which of the different attributes were most closely related to the 'overall' attribute of each player. The 'overall' attribute is an integral numerical rating of the players on a scale of 0-100 (100 being the highest), which signifies how good the player is.

We removed attributes like the full name, name, club name, league name and nationality which obviously have no bearing on how the player is.

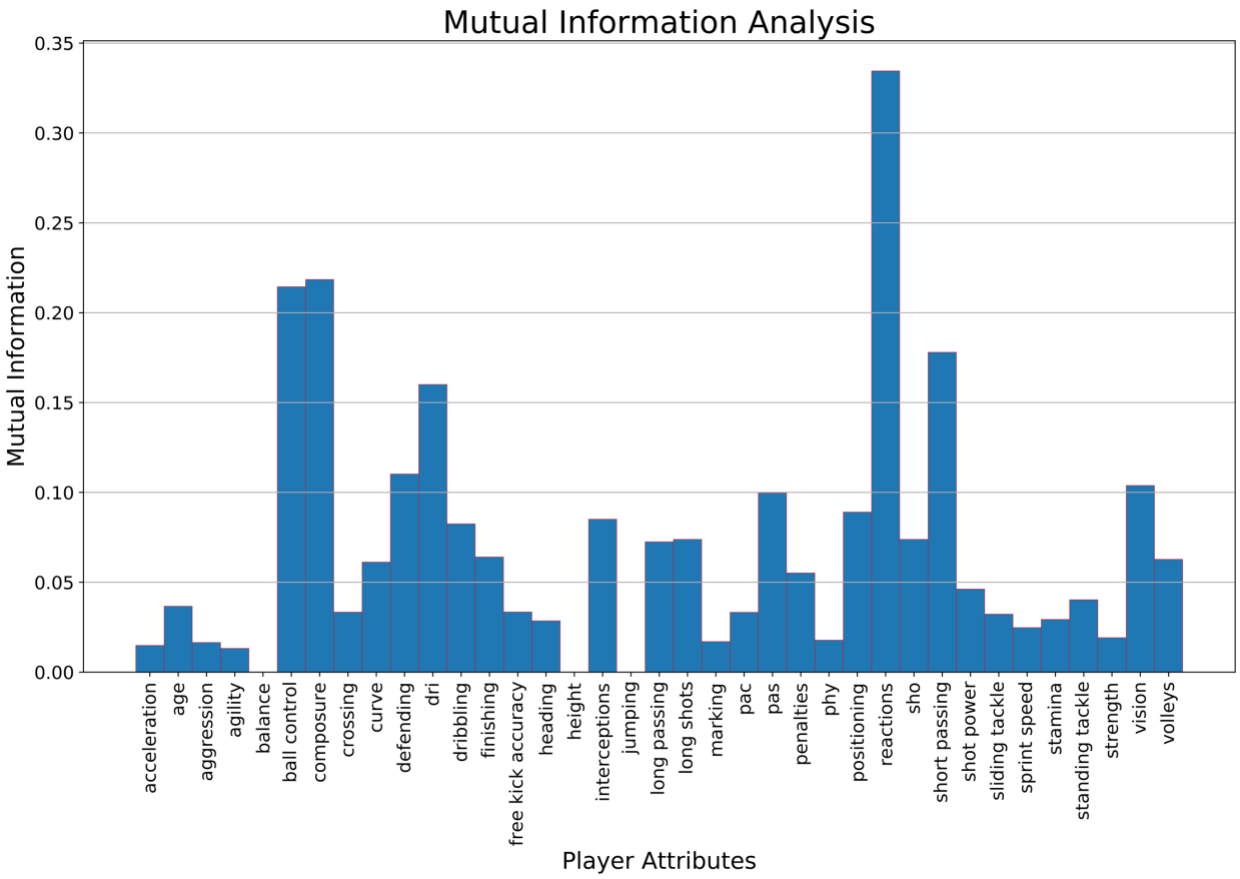
Following this, we performed various feature selection and correlation analysis tests to determine the attributes contribute the most to a player's overall.

F-test Analysis –



As we see from the graph, the attributes – reactions, composure and short passing have the highest contributions while age, height, strength appear to have the lowest significance.

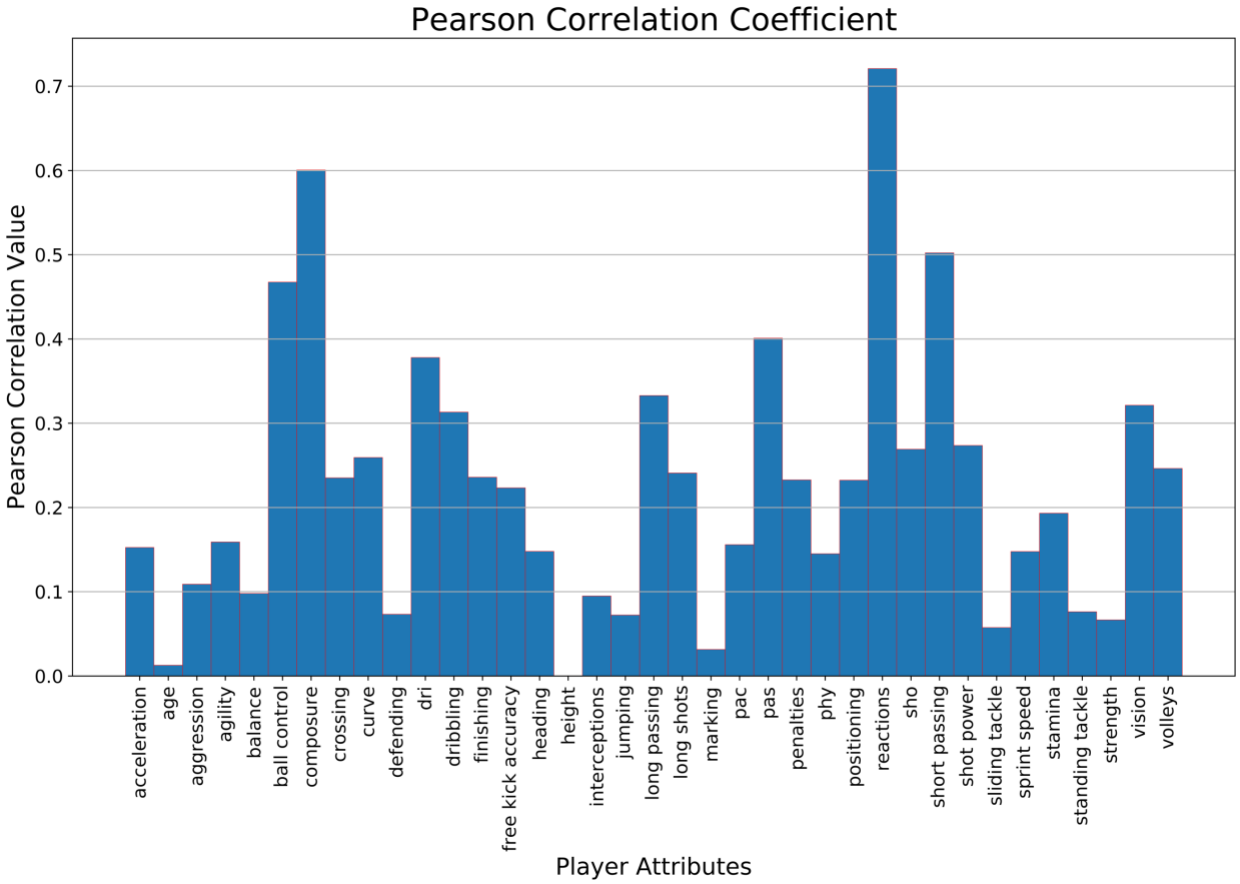
Mutual Information Analysis –



As we see from the plot, the attributes – reactions, composure and ball control - have the highest contributions while balance, height, jumping appear to have the lowest significance.

In contrary to the F-test, age seems to have some significance in the case of mutual information analysis.

Pearson Correlation –

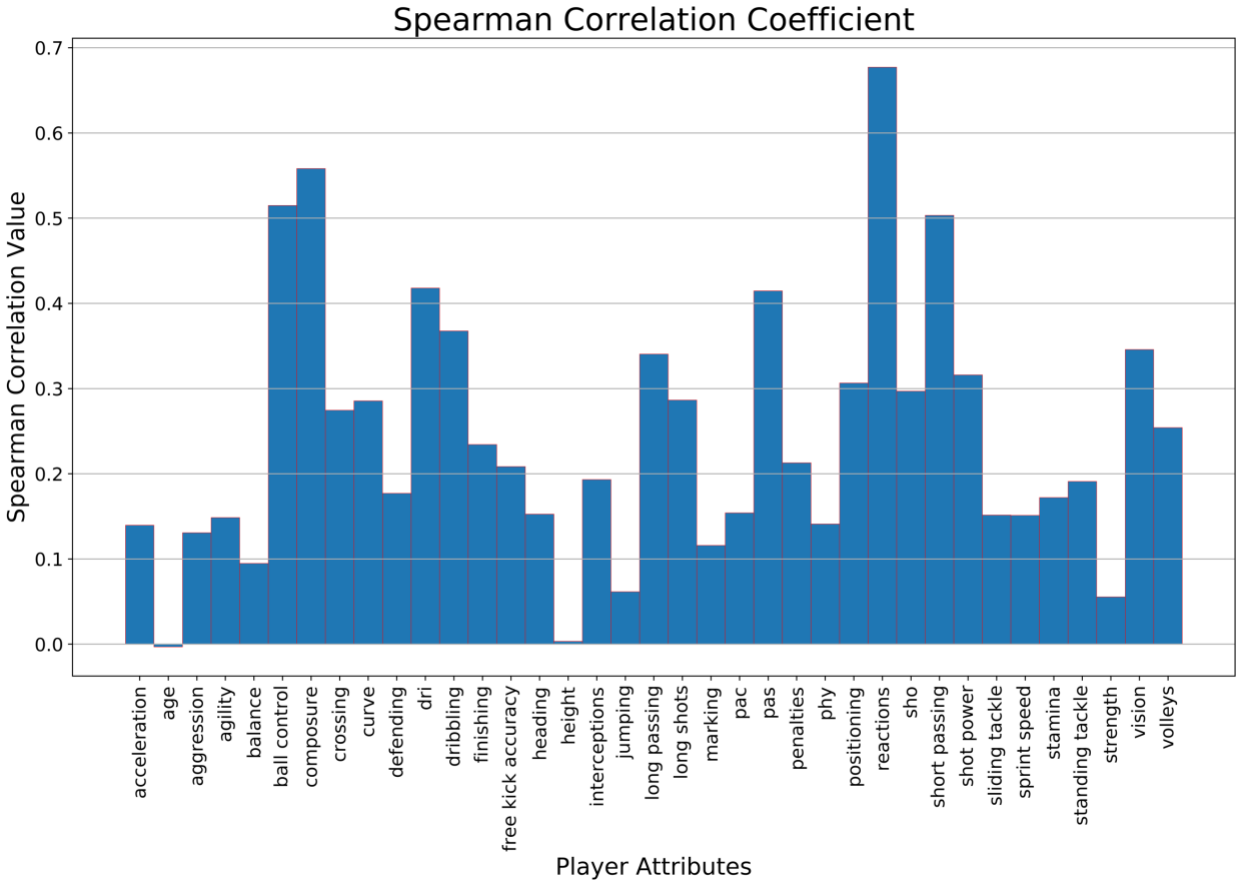


As we see from the plot, the attributes – reactions, composure, ball control and short passing - have the highest contributions while age, height, marking appear to have the lowest significance.

In contrary to the F-test, defending and strength seem to have some significance in the case of mutual information analysis.

In contrary to mutual information, jumping seems to have some significance in the case of mutual information analysis.

Spearman Correlation –



As we see from the plot, the attributes – reactions, composure, ball control and short passing - have the highest contributions while age and height appear to have the lowest significance.

From this analysis, we learnt that attributes like age and height have very little significance on the player's 'overall' which is consistent with our intuition. We also infer that reactions, composure, ball control and short passing are the most important characteristics a football player should possess.

If given more time, we can try to predict the 'overall' attribute using the correlations that we discovered using the correlation analysis techniques. Since it is an integral value and not a continuous value but has 100 distinct values, we can't formulate it as a multiclass classification value and hence we plan to formulate it as a regression problem and will consider a threshold of distance between the original value and the predicted value as a true positive.