```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
data=pd.read_csv('players_20.csv')
```

```
df=data.copy()
```

```
df.shape
```

```
(18278, 104)
```

There are total 18278 rows and 104 columns in the data

Handling Duplicated Rows

```
df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
18273    False
18274    False
18275    False
18276    False
18277    False
Length: 18278, dtype: bool
```

```
dup_index=df[df.duplicated()].index
```

No Duplicate row found

General Information About Dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18278 entries, 0 to 18277
Columns: 104 entries, sofifa_id to rb
dtypes: float64(16), int64(45), object(43)
memory usage: 14.5+ MB
```

```
df.isnull().sum()
```

```
sofifa_id        0
player_url       0
short_name       0
long_name        0
age              0
                ...
lb            2036
lcb           2036
cb            2036
rcb           2036
rb            2036
Length: 104, dtype: int64
```

```
df.describe()
```

	sofifa_id	age	height_cm	weight_kg	overall	potential	value_eur	wage_eur	international_repu
count	18278.000000	18278.000000	18278.000000	18278.000000	18278.000000	18278.000000	1.827800e+04	18278.000000	18278.
mean	219738.864482	25.283291	181.362184	75.276343	66.244994	71.546887	2.484038e+06	9456.942773	1.
std	27960.200461	4.656964	6.756961	7.047744	6.949953	6.139669	5.585481e+06	21351.714095	0.
min	768.000000	16.000000	156.000000	50.000000	48.000000	49.000000	0.000000e+00	0.000000	1.
25%	204445.500000	22.000000	177.000000	70.000000	62.000000	67.000000	3.250000e+05	1000.000000	1.
50%	226165.000000	25.000000	181.000000	75.000000	66.000000	71.000000	7.000000e+05	3000.000000	1.
75%	240795.750000	29.000000	186.000000	80.000000	71.000000	75.000000	2.100000e+06	8000.000000	1.
max	252905.000000	42.000000	205.000000	110.000000	94.000000	95.000000	1.055000e+08	565000.000000	5.

8 rows × 61 columns

Extracting Columns as per their datatype

```
df_cols_all=list(df.columns)
df_cols_all
```

```
'attacking_heading_accuracy',
'attacking_short_passing',
'attacking_volleys',
'skill_dribbling',
'skill_curve',
'skill_fk_accuracy',
'skill_long_passing',
'skill_ball_control',
'movement_acceleration',
```

```
        'cb',
        'rcb',
        'rb']
```

```
df_cols_int=list(df.select_dtypes('int').columns)
df_cols_flt=list(df.select_dtypes('float').columns)
df_cols_obj=list(df.select_dtypes('object').columns)
df[df_cols_int].isnull().sum()
```

```
sofifa_id                     0
age                           0
height_cm                     0
weight_kg                     0
overall                       0
potential                     0
value_eur                     0
wage_eur                      0
international_reputation      0
weak_foot                     0
skill_moves                   0
attacking_crossing            0
attacking_finishing           0
attacking_heading_accuracy    0
attacking_short_passing       0
attacking_volleys             0
skill_dribbling               0
skill_curve                   0
skill_fk_accuracy             0
skill_long_passing            0
skill_ball_control            0
movement_acceleration         0
movement_sprint_speed         0
movement_agility              0
movement_reactions            0
movement_balance              0
power_shot_power              0
power_jumping                 0
power_stamina                 0
power_strength                0
power_long_shots              0
mentality_aggression          0
mentality_interceptions       0
mentality_positioning         0
mentality_vision              0
mentality_penalties           0
mentality_composure           0
defending_marking             0
defending_standing_tackle     0
defending_sliding_tackle      0
goalkeeping_diving            0
goalkeeping_handling          0
goalkeeping_kicking           0
goalkeeping_positioning       0
goalkeeping_reflexes          0
dtype: int64
```

There are no integer coulmns with null value

```
df[df_cols_flt].isnull().sum()
```

```
release_clause_eur       1298
team_jersey_number        240
contract_valid_until      240
nation_jersey_number    17152
pace                     2036
shooting                 2036
passing                  2036
dribbling                2036
defending                2036
physic                   2036
gk_diving               16242
gk_handling             16242
gk_kicking              16242
gk_reflexes             16242
gk_speed                16242
gk_positioning          16242
dtype: int64
```

There are null values in float columns

```
df[df_cols_obj].isnull().sum()
```

```
player_url             0
short_name             0
long_name              0
dob                    0
nationality            0
club                   0
player_positions       0
preferred_foot         0
work_rate              0
body_type              0
real_face              0
player_tags        16779
team_position        240
loaned_from        17230
joined              1288
nation_position    17152
player_traits      10712
ls                  2036
st                  2036
rs                  2036
lw                  2036
lf                  2036
cf                  2036
rf                  2036
rw                  2036
lam                 2036
cam                 2036
ram                 2036
lm                  2036
lcm                 2036
cm                  2036
rcm                 2036
rm                  2036
lwb                 2036
ldm                 2036
cdm                 2036
rdm                 2036
rwb                 2036
lb                  2036
lcb                 2036
cb                  2036
rcb                 2036
rb                  2036
dtype: int64
```

There are null values in object columns

EDA

Univarient Analysis

```
!pip install sweetviz
import sweetviz as sv
```

```
Collecting sweetviz
  Downloading sweetviz-2.3.1-py3-none-any.whl (15.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.1/15.1 MB 34.3 MB/s eta 0:00:00
Requirement already satisfied: pandas!=1.0.0,!=1.0.1,!=1.0.2,>=0.25.3 in /usr/local/lib/python3.10/dist-packages (from sweetviz) (1.5.3
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from sweetviz) (1.25.2)
Requirement already satisfied: matplotlib>=3.1.3 in /usr/local/lib/python3.10/dist-packages (from sweetviz) (3.7.1)
Requirement already satisfied: tqdm>=4.43.0 in /usr/local/lib/python3.10/dist-packages (from sweetviz) (4.66.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from sweetviz) (1.11.4)
Requirement already satisfied: jinja2>=2.11.1 in /usr/local/lib/python3.10/dist-packages (from sweetviz) (3.1.3)
Requirement already satisfied: importlib-resources>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from sweetviz) (6.4.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2>=2.11.1->sweetviz) (2.1.5)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (4.50.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1.3->sweetviz) (2.8.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.0.0,!=1.0.1,!=1.0.2,>=0.25.3->sw
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.1.3->sweet
```

```
Installing collected packages: sweetviz
Successfully installed sweetviz-2.3.1
```

```
# df_univar_rpt=sv.analyze(df)
```

```
# df_univar_rpt.show_html()
```

Based On The Data We can say that most of the integer columns are normally distributed

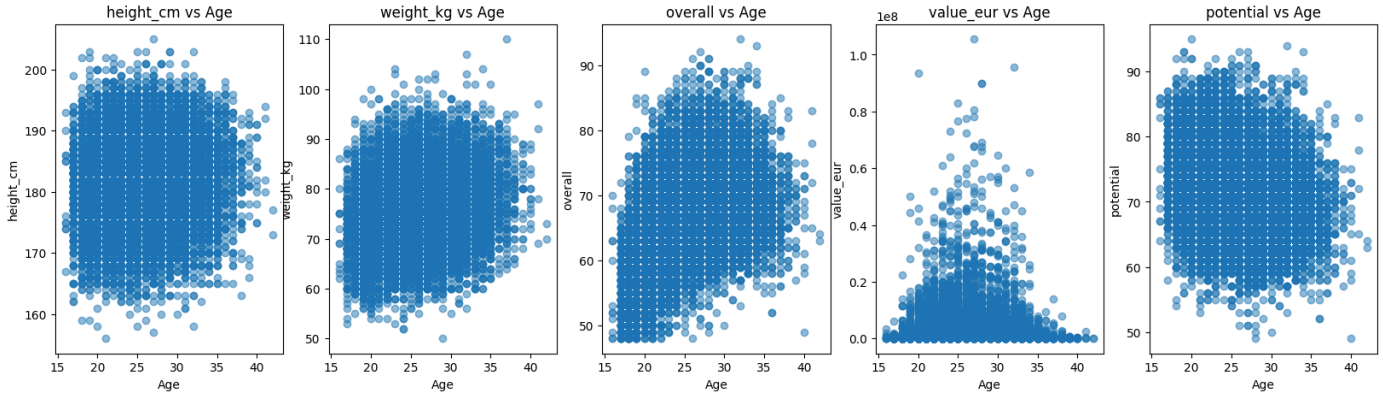
Bi-Var Analysis

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

```
df.head(1)
```

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club	overall
0	158023	https://sofifa.com/player/158023/lionel-messi/...	L. Messi	Lionel Andrés Messi Cuccittini	32	1987-06-24	170	72	Argentina	FC Barcelona	94

```
fig,axs=plt.subplots(1,5, figsize=(20, 5))
for i, column in enumerate(['height_cm', 'weight_kg', 'overall', 'value_eur', 'potential']):
    axs[i].scatter(df['age'], df[column], alpha=0.5)
    axs[i].set_title(f'{column} vs Age')
    axs[i].set_xlabel('Age')
    axs[i].set_ylabel(column)
```



-We can see that overall increases as age increases but only upto age of 35

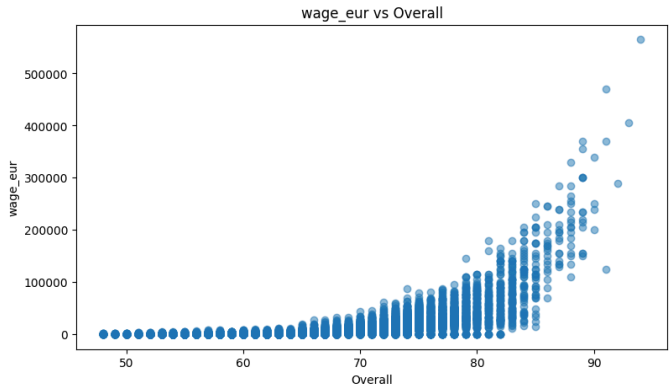
-We can see that value increases between age of 25 to 35

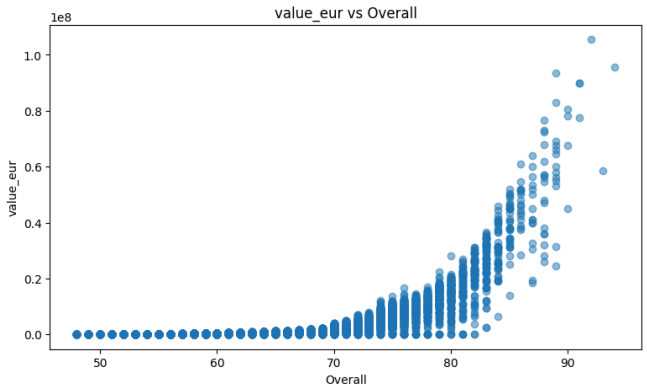
```
df_cols_all
```

```
 'power_shot_power',
 'power_jumping',
 'power_stamina',
 'power_strength',
 'power_long_shots',
 'mentality_aggression',
 'mentality_interceptions',
 'mentality_positioning',
 'mentality_vision',
 'mentality_penalties',
 'mentality_composure',
 'defending_marking',
 'defending_standing_tackle',
 'defending_sliding_tackle',
 'goalkeeping_diving',
 'goalkeeping_handling',
 'goalkeeping_kicking',
 'goalkeeping_positioning',
 'goalkeeping_reflexes',
 'ls',
 'st',
 'rs',
 'lw',
 'lf',
 'cf',
 'rf',
 'rw',
 'lam',
 'cam',
 'ram',
 'lm',
 'lcm',
 'cm',
 'rcm',
 'rm',
 'lwb',
 'ldm',
 'cdm',
 'rdm',
 'rwb',
 'lb',
 'lcb',
 'cb',
 'rcb',
 'rb']
```

Money with Overall

```
fig,axs=plt.subplots(1,2, figsize=(20, 5))
for i, column in enumerate(["wage_eur","value_eur"]):
    axs[i].scatter(df['overall'], df[column], alpha=0.5)
    axs[i].set_title(f'{column} vs Overall')
    axs[i].set_xlabel('Overall')
    axs[i].set_ylabel(column)
```



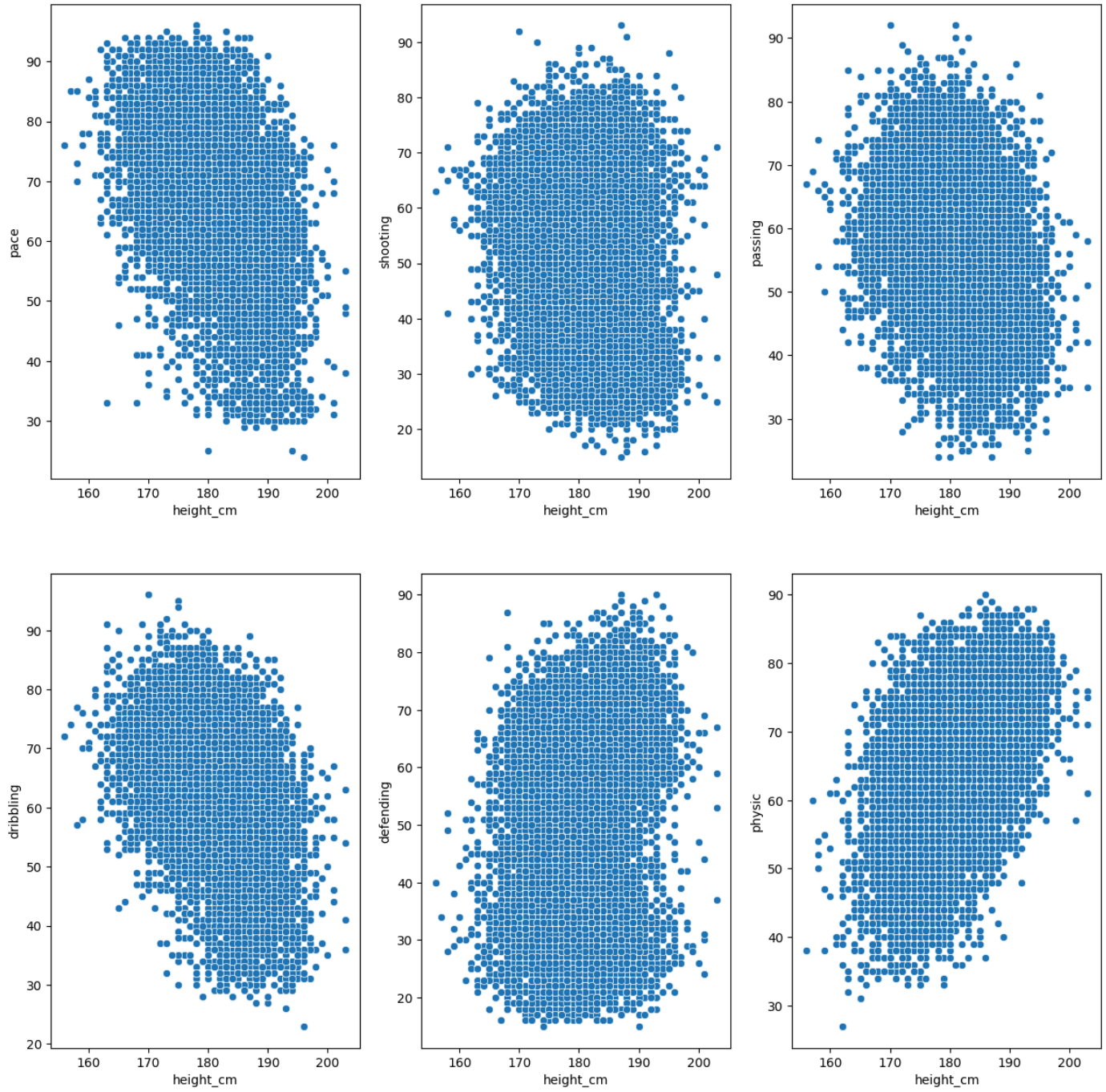


As Overall Increases Wages and value increases

Height With Few Skills

```
import seaborn as sns
plt.figure(figsize=(15,15))
```

```
plt.figure(figsize=(15,15))
plt_num = 1

fifa20_bivar_cols = ["pace", "shooting", "passing", "dribbling", "defending", "physic"]

for column in fifa20_bivar_cols:
    if plt_num <= 6:
        plt.subplot(2,3,plt_num)
        sns.scatterplot(x="height_cm", y=column, data=df)
    plt_num += 1
plt.show()
```



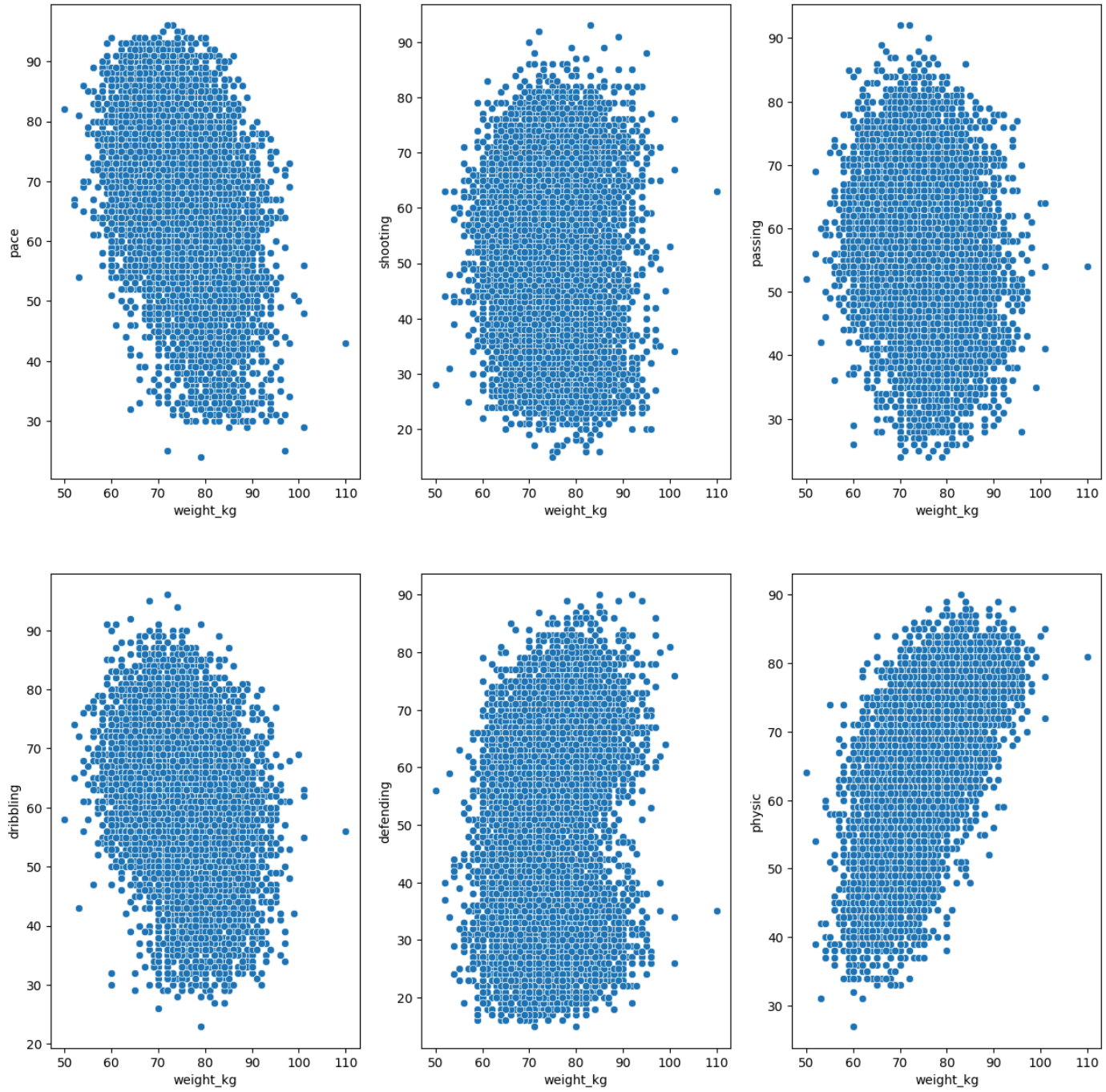
1) As the height of the player increases,

a) their pace, passing and dribbling skills get decreased(beyond 190 cm)

b) shooting skill is on high for very few players; otherwise it follows a decline as the hieght increases

2) Players with height between 170 to 200 cm tend to have a good physic ranking and defending skill

```
plt.figure(figsize=(15,15))
plt_num = 1

fifa20_bivar_cols = ["pace", "shooting", "passing", "dribbling", "defending", "physic"]

for column in fifa20_bivar_cols:
    if plt_num <= 6:
        plt.subplot(2,3,plt_num)
        sns.scatterplot(x='weight_kg', y=column, data=df)
    plt_num += 1
plt.show()
```



Weight almost effects every skills where weight increases every skill except physic decreases

```
df['player_positions']
```

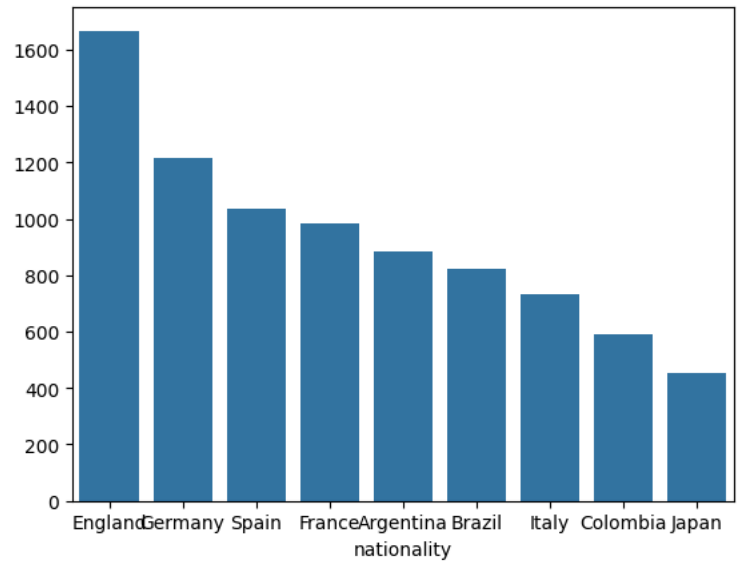
```
18221             CB
18222          RM, LM
18223             GK
18224          LB, LM
18225             GK
18226         CM, CAM
18227             CM
18228             GK
18229             GK
18230             LB
18231             GK
18232      RM, ST, CM
18233          LB, RB
18234             RB
18235          RM, LM
18236             RM
18237             LB
18238          CB, LB
18239             GK
18240            CDM
18241         CDM, CB
18242         CB, CDM
18243             ST
18244             CM
18245             CM
18246             CM
18247             CM
18248          LW, LM
18249             ST
18250             ST
18251             GK
18252             CM
18253             CB
18254             ST
18255             CM
18256          CB, RB
18257             LW
18258             CM
18259             GK
18260         CDM, CM
18261             ST
18262          CB, RB
18263             RB
18264             CM
18265         CDM, RB
18266            CAM
18267             CM
18268      LM, RM, CM
18269             CB
18270             CM
18271             CM
18272             GK
18273             CB
18274             CB
18275             CM
18276             CM
18277             CM
Name: player_positions, dtype: object
```

Task 3

3.1Rank Top 10 Countries with most players

```
rank=df.groupby('nationality').size().sort_values(ascending=False).head(9)
sns.barplot(rank)
```

```
<Axes: xlabel='nationality'>
```

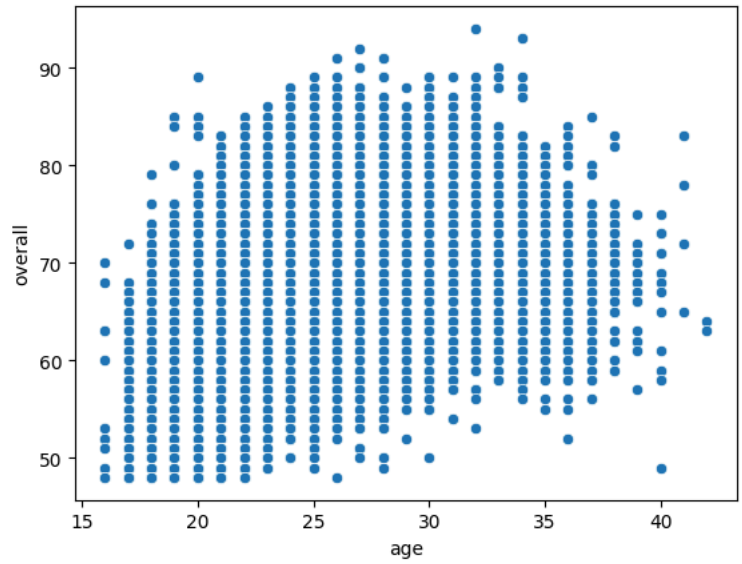


England Is The country with most players

3.2 overall rating vs age

```
sns.scatterplot(x='age',y='overall',data=df)
```

```
<Axes: xlabel='age', ylabel='overall'>
```



Based On the data we can say that players stop improving around 40 years of age

```
df['player_positions']
```

```
18236              RM
18237              LB
18238          CB, LB
18239              GK
18240             CDM
18241         CDM, CB
18242         CB, CDM
18243              ST
18244              CM
18245              CM
18246              CM
18247              CM
18248          LW, LM
18249              ST
18250              ST
18251              GK
18252              CM
18253              CB
18254              ST
18255              CM
18256          CB, RB
18257              LW
18258              CM
18259              GK
18260         CDM, CM
18261              ST
18262          CB, RB
18263              RB
18264              CM
18265         CDM, RB
18266             CAM
18267              CM
18268      LM, RM, CM
18269              CB
18270              CM
18271              CM
18272              GK
18273              CB
18274              CB
18275              CM
18276              CM
18277              CM
Name: player_positions, dtype: object
```

3.3)Which Type of Offensive players tends to paid more LW,RW,ST?

```
df_st=df[df['player_positions'].str.contains('ST')]
df_rw=df[df['player_positions'].str.contains('RW')]
df_lw=df[df['player_positions'].str.contains('LW')]
df_combined=pd.concat([df_rw,df_lw,df_st],ignore_index=True)
```

```
df_combined.shape
```

```
(5810, 104)
```

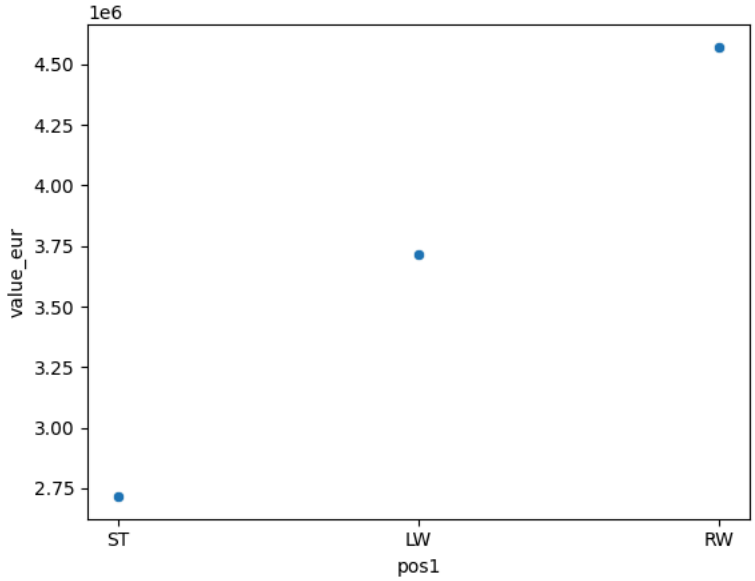
```
def pos(position):
    pla_pos_len = len(position)
    if 2 <= pla_pos_len <= 3:
        pos1 = position
        pos2 = pos3 = ''
    elif 4 <= pla_pos_len <= 8:
        pos1, pos2 = position.split(',')
        pos3 = ''
    else:
        pos1, pos2, pos3 = position.split(',')
    return pos1.strip(), pos2.strip(), pos3.strip()

for i in range(0, len(df_combined['player_positions'])):
    pos1, pos2, pos3 = pos(df_combined.loc[i, 'player_positions'])
    df_combined.loc[i, 'pos1'] = pos1
    df_combined.loc[i, 'pos2'] = pos2
    df_combined.loc[i, 'pos3'] = pos3
```

```
df_combined[['player_positions','pos1','pos2','pos3']].head()
```

	player_positions	pos1	pos2	pos3
0	RW, CF, ST	RW	CF	ST
1	RW, ST	RW	ST	
2	ST, RW	ST	RW	
3	CAM, RW	CAM	RW	
4	RW, LW	RW	LW	

```
# Assuming df_combined is your DataFrame
# First, filter the DataFrame based on the specified conditions
filtered_df = df_combined[df_combined['pos1'].isin(['LW', 'RW', 'ST'])]

# Then, group by 'pos1' and calculate the mean of 'value_eur'
mean_values = filtered_df.groupby('pos1')['value_eur'].mean()

# Finally, sort the mean values in ascending order
sorted_mean_values = mean_values.sort_values(ascending=True)

# Print or further process sorted_mean_values
print(sorted_mean_values)
sns.scatterplot(data=sorted_mean_values)
```

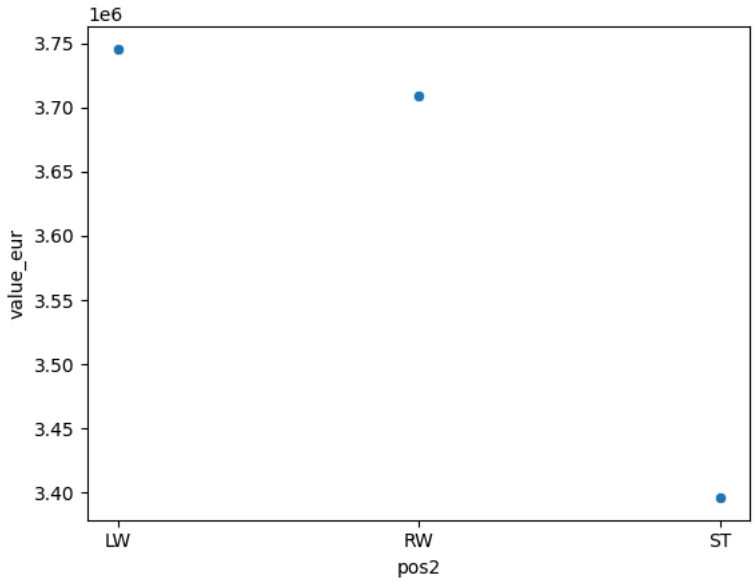
```
pos1
ST    2.716590e+06
LW    3.714095e+06
RW    4.571033e+06
Name: value_eur, dtype: float64
<Axes: xlabel='pos1', ylabel='value_eur'>
```



For Position_1 RW is highest paid followed by LW and then ST

```
filtered_df=df_combined[df_combined['pos2'].isin(['ST','LW','RW'])]
sorted_mean_values=filtered_df.groupby('pos2')['value_eur'].mean().sort_values(ascending=False)
sns.scatterplot(data=sorted_mean_values)
sorted_mean_values
```

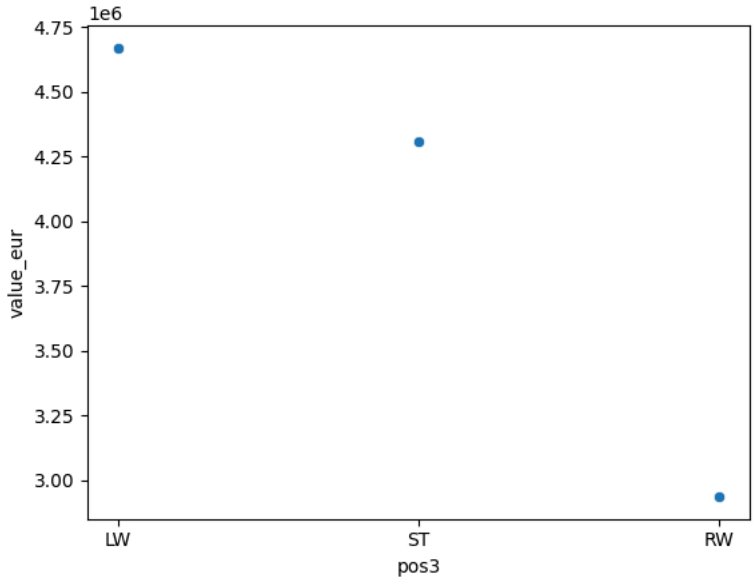
```
pos2
LW    3.745440e+06
RW    3.708816e+06
ST    3.396417e+06
Name: value_eur, dtype: float64
```



For Pos2 Lw is highest paid followed by RW & ST

```
filtered_df=df_combined[df_combined['pos3'].isin(['ST','LW','RW'])]
sorted_mean_values=filtered_df.groupby('pos3')['value_eur'].mean().sort_values(ascending=False)
sns.scatterplot(data=sorted_mean_values)
sorted_mean_values
```

```
pos3
LW    4.668730e+06
ST    4.308399e+06
RW    2.938004e+06
Name: value_eur, dtype: float64
```



For Pos3 Lw is highest paid Followed by ST and then RW

pos3

LW 4.668730e+06

ST 4.308399e+06

RW 2.938004e+06

pos2

LW 3.745440e+06

RW 3.708816e+06

ST 3.396417e+06

pos1

ST 2.716590e+06

LW 3.714095e+06

RW 4.571033e+06

LW:4.04

ST:3.47

RW:3.73

LW is highest paid offensive position

DATA PREPROCESSING

Columns to be deleted

```
fifa20_cols_del = ['sofifa_id', 'player_url', 'short_name', 'long_name', 'dob', 'nationality', 'club','overall',
                       'potential', 'value_eur', 'wage_eur', 'preferred_foot','international_reputation', 'weak_foot',
                       'skill_moves', 'body_type', 'real_face', 'release_clause_eur', 'player_tags', 'team_position',
                       'team_jersey_number', 'loaned_from', 'joined', 'contract_valid_until', 'nation_position',
                       'nation_jersey_number', 'player_traits', 'goalkeeping_diving', 'goalkeeping_handling',
                       'goalkeeping_kicking', 'goalkeeping_positioning', 'goalkeeping_reflexes']

updated_df = df.drop(columns=fifa20_cols_del, inplace=False)


updated_df.shape
```

```
(18278, 72)
```

1) Few columns should be deleted from the input dataset. These columns are NOT the actual skills of the players; but depict few attributes. These are not required for the ML model development. Columns that are to be deleted(for each player) and their deletion reason are given below: 1) sofifa_id - Unique FIFA ID 2) player_url - Unique URL 3) short_name - Short name 3) long_name - Long name 5) dob - Date of birth 6) nationality - Nationality 7) club - Football club to which each player belongs to 8) overall - Overall ranking 9) potential - Potential ranking that each player can reach 10) value_eur - Player's value in euro 11) wage_eur - Player's monthly wage in euro 12) preferred_foot - Player's preferred foot while playing 13) international_reputation - reputation rated between 1-99 based on the player's skillset 14) weak_foot - Represents how well a player can use his weak foot 15) skill_moves - Cumulative ranking of various skills the player exhibit on the ground 16) body_type - Player's body type - normal, lean or stocky 17) real_face - whether real face present in fifa simulation game or not 18) release_clause_eur - Amount to be paid in euro if a player is to be released from the current club 19) player_tags - Tags assigned for each player portraying his skills and the positions he can play 20) team_position - On-field position in which the player playes for his club 21) team_jersey_number - Player's jersey number in his club 22) loaned_from - Club from which the player has been loaned from 23) joined - Player's joining date in the current club 24) contract_valid_until - Year till which the player's contract is valid with the current club 25) nation_position - On-field position in which the player playes for his national team 26) nation_jersey_number - Player's jersey number in his national team 27) player_traits - details the characteristics that a player possesses

```
Features given below are deleted as they are applicable only for goal keepers(gk).
28) goalkeeping_diving
29) goalkeeping_handling
30) goalkeeping_kicking
31) goalkeeping_positioning
32) goalkeeping_reflexes

They have values for non-goal keepers too.  Also, we have features gk_diving, gk_handling, gk_kicking, gk_reflexes,
```

gk_speed and gk_positioning that has values for goal keepers alone. Hence, we can consider goal keeping features starting with gk_* and delete the features starting with goalkeeping_*.

```
updated_df.info()
```

```
 16  gk_positioning              2036 non-null   float64
 17  attacking_crossing          18278 non-null  int64
 18  attacking_finishing         18278 non-null  int64
 19  attacking_heading_accuracy  18278 non-null  int64
 20  attacking_short_passing     18278 non-null  int64
 21  attacking_volleys           18278 non-null  int64
 22  skill_dribbling             18278 non-null  int64
 23  skill_curve                 18278 non-null  int64
 24  skill_fk_accuracy           18278 non-null  int64
 25  skill_long_passing          18278 non-null  int64
 26  skill_ball_control          18278 non-null  int64
 27  movement_acceleration       18278 non-null  int64
 28  movement_sprint_speed       18278 non-null  int64
 29  movement_agility            18278 non-null  int64
 30  movement_reactions          18278 non-null  int64
 31  movement_balance            18278 non-null  int64
 32  power_shot_power            18278 non-null  int64
 33  power_jumping               18278 non-null  int64
 34  power_stamina               18278 non-null  int64
 35  power_strength              18278 non-null  int64
 36  power_long_shots            18278 non-null  int64
 37  mentality_aggression        18278 non-null  int64
 38  mentality_interceptions     18278 non-null  int64
 39  mentality_positioning       18278 non-null  int64
 40  mentality_vision            18278 non-null  int64
 41  mentality_penalties         18278 non-null  int64
 42  mentality_composure         18278 non-null  int64
 43  defending_marking           18278 non-null  int64
 44  defending_standing_tackle   18278 non-null  int64
 45  defending_sliding_tackle    18278 non-null  int64
 46  ls                          16242 non-null  object
 47  st                          16242 non-null  object
 48  rs                          16242 non-null  object
 49  lw                          16242 non-null  object
 50  lf                          16242 non-null  object
 51  cf                          16242 non-null  object
 52  rf                          16242 non-null  object
 53  rw                          16242 non-null  object
 54  lam                         16242 non-null  object
 55  cam                         16242 non-null  object
 56  ram                         16242 non-null  object
 57  lm                          16242 non-null  object
 58  lcm                         16242 non-null  object
 59  cm                          16242 non-null  object
 60  rcm                         16242 non-null  object
 61  rm                          16242 non-null  object
 62  lwb                         16242 non-null  object
 63  ldm                         16242 non-null  object
 64  cdm                         16242 non-null  object
 65  rdm                         16242 non-null  object
 66  rwb                         16242 non-null  object
 67  lb                          16242 non-null  object
 68  lcb                         16242 non-null  object
 69  cb                          16242 non-null  object
 70  rcb                         16242 non-null  object
 71  rb                          16242 non-null  object
dtypes: float64(12), int64(32), object(28)
memory usage: 10.0+ MB
```

```
# Selecting columns by data type
float_columns = updated_df.select_dtypes(include=['float64']).columns.tolist()
int_columns = updated_df.select_dtypes(include=['int64']).columns.tolist()
obj_columns=updated_df.select_dtypes(include=['object']).columns.tolist()
```

```
float_columns
int_columns
obj_columns
```

```
['player_positions',
 'work_rate',
 'ls',
 'st',
 'rs',
 'lw',
```

```
        'lf',
        'cf',
        'rf',
        'rw',
        'lam',
        'cam',
        'ram',
        'lm',
        'lcm',
        'cm',
        'rcm',
        'rm',
        'lwb',
        'ldm',
        'cdm',
        'rdm',
        'rwb',
        'lb',
        'lcb',
        'cb',
        'rcb',
        'rb']
```

Handling Missing Values For Object Datatype

```
updated_df[obj_columns].isnull().sum()
```

```
player_positions       0
work_rate              0
ls                  2036
st                  2036
rs                  2036
lw                  2036
lf                  2036
cf                  2036
rf                  2036
rw                  2036
lam                 2036
cam                 2036
ram                 2036
lm                  2036
lcm                 2036
cm                  2036
rcm                 2036
rm                  2036
lwb                 2036
ldm                 2036
cdm                 2036
rdm                 2036
rwb                 2036
lb                  2036
lcb                 2036
cb                  2036
rcb                 2036
rb                  2036
dtype: int64
```

Null Value Columns(Object datatype)

```
words_list = [
    'ls', 'st', 'rs', 'lw', 'lf', 'cf', 'rf', 'rw', 'lam', 'cam', 'ram',
    'lm', 'lcm', 'cm', 'rcm', 'rm', 'lwb', 'ldm', 'cdm', 'rdm', 'rwb',
    'lb', 'lcb', 'cb', 'rcb', 'rb'
]
updated_df[words_list].head()
```

	ls	st	rs	lw	lf	cf	rf	rw	lam	cam	ram	lm	lcm	cm	rcm	rm	lwb	ldm	cdm	rdm	rwb	lb	
0	89+2	89+2	89+2	93+2	93+2	93+2	93+2	93+2	93+2	93+2	93+2	92+2	87+2	87+2	87+2	92+2	68+2	66+2	66+2	66+2	68+2	63+2	5
1	91+3	91+3	91+3	89+3	90+3	90+3	90+3	89+3	88+3	88+3	88+3	88+3	81+3	81+3	81+3	88+3	65+3	61+3	61+3	61+3	65+3	61+3	5
2	84+3	84+3	84+3	90+3	89+3	89+3	89+3	90+3	90+3	90+3	90+3	89+3	82+3	82+3	82+3	89+3	66+3	61+3	61+3	61+3	66+3	61+3	4
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	83+3	83+3	83+3	89+3	88+3	88+3	88+3	89+3	89+3	89+3	89+3	89+3	83+3	83+3	83+3	89+3	66+3	63+3	63+3	63+3	66+3	61+3	4

```
tempdf=updated_df
```

Converting Object DataType to Integer and Filling Nan Values with 0

```
for column in tempdf[words_list]:
    tempdf[column] = [int(value.split('+')[0]) + int(value.split('+')[1]) if pd.notnull(value) else 0 for value in tempdf[column]]
```

```
updated_df=tempdf
```

Encoding Player_position,work_rate columns('Categorical Columns')

```
updated_df[['player_positions','work_rate']].head()
```

	player_positions	work_rate
0	RW, CF, ST	Medium/Low
1	ST, LW	High/Low
2	LW, CAM	High/Medium
3	GK	Medium/Medium
4	LW, CF	High/Medium

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

```
# Encode 'player_position' column

updated_df['player_positions'] = label_encoder.fit_transform(updated_df['player_positions'])

# Encode 'work_rate' column
updated_df['work_rate'] = label_encoder.fit_transform(updated_df['work_rate'])
```

Handling Missing Values For Float Datatype

```
updated_df[float_columns].isnull().sum()
```

```
pace              2036
shooting          2036
passing           2036
dribbling         2036
defending         2036
physic            2036
gk_diving        16242
gk_handling      16242
gk_kicking       16242
gk_reflexes      16242
gk_speed         16242
gk_positioning   16242
dtype: int64
```

Filled Null values with 0

```
updated_df[float_columns]=updated_df[float_columns].fillna(0)


updated_df.info()
```

```
 16  gk_positioning              18278 non-null  float64
 17  attacking_crossing          18278 non-null  int64
 18  attacking_finishing         18278 non-null  int64
 19  attacking_heading_accuracy  18278 non-null  int64
 20  attacking_short_passing     18278 non-null  int64
 21  attacking_volleys           18278 non-null  int64
 22  skill_dribbling             18278 non-null  int64
 23  skill_curve                 18278 non-null  int64
 24  skill_fk_accuracy           18278 non-null  int64
 25  skill_long_passing          18278 non-null  int64
 26  skill_ball_control          18278 non-null  int64
 27  movement_acceleration       18278 non-null  int64
 28  movement_sprint_speed       18278 non-null  int64
 29  movement_agility            18278 non-null  int64
 30  movement_reactions          18278 non-null  int64
 31  movement_balance            18278 non-null  int64
 32  power_shot_power            18278 non-null  int64
 33  power_jumping               18278 non-null  int64
 34  power_stamina               18278 non-null  int64
 35  power_strength              18278 non-null  int64
 36  power_long_shots            18278 non-null  int64
 37  mentality_aggression        18278 non-null  int64
 38  mentality_interceptions     18278 non-null  int64
 39  mentality_positioning       18278 non-null  int64
 40  mentality_vision            18278 non-null  int64
 41  mentality_penalties         18278 non-null  int64
 42  mentality_composure         18278 non-null  int64
 43  defending_marking           18278 non-null  int64
 44  defending_standing_tackle   18278 non-null  int64
 45  defending_sliding_tackle    18278 non-null  int64
 46  ls                          18278 non-null  int64
 47  st                          18278 non-null  int64
 48  rs                          18278 non-null  int64
 49  lw                          18278 non-null  int64
 50  lf                          18278 non-null  int64
 51  cf                          18278 non-null  int64
 52  rf                          18278 non-null  int64
 53  rw                          18278 non-null  int64
 54  lam                         18278 non-null  int64
 55  cam                         18278 non-null  int64
 56  ram                         18278 non-null  int64
 57  lm                          18278 non-null  int64
 58  lcm                         18278 non-null  int64
 59  cm                          18278 non-null  int64
 60  rcm                         18278 non-null  int64
 61  rm                          18278 non-null  int64
 62  lwb                         18278 non-null  int64
 63  ldm                         18278 non-null  int64
 64  cdm                         18278 non-null  int64
 65  rdm                         18278 non-null  int64
 66  rwb                         18278 non-null  int64
 67  lb                          18278 non-null  int64
 68  lcb                         18278 non-null  int64
 69  cb                          18278 non-null  int64
 70  rcb                         18278 non-null  int64
 71  rb                          18278 non-null  int64
dtypes: float64(12), int64(60)
memory usage: 10.0 MB
```

Filled all The missing values

Standardize The value To bring them on single scale

```
from sklearn.preprocessing import StandardScaler
std_scl=StandardScaler()
fifa20_upd_scl=std_scl.fit_transform(updated_df)
fifa20_upd_scl=pd.DataFrame(fifa20_upd_scl,columns=updated_df.columns)


fifa20_upd_scl.to_csv('fifa20_scaled.csv', index=False)
fifa20_upd_scl.shape
```

```
(18278, 72)
```

Feature Selection

Deleting the duplicate columns

```
#-----------------------------------FUNCTION TO REMOVE DUPLICATE COLUMNS-------------------------------------
def get_duplicate_columns(df):

    duplicate_columns = {}
    seen_columns = {}

    for column in df.columns:
        current_column = df[column]

        # Convert column data to bytes
        try:
            current_column_hash = current_column.values.tobytes()
        except AttributeError:
            current_column_hash = current_column.to_string().encode()

        if current_column_hash in seen_columns:
            if seen_columns[current_column_hash] in duplicate_columns:
                duplicate_columns[seen_columns[current_column_hash]].append(column)
            else:
                duplicate_columns[seen_columns[current_column_hash]] = [column]
        else:
            seen_columns[current_column_hash] = column

    return duplicate_columns


#Getting Duplicate Columns
duplicate_col=get_duplicate_columns(fifa20_upd_scl)
duplicate_col
#note :-here keys are columns and values are duplictae columns
```

```
{'ls': ['st', 'rs'],
 'lf': ['cf', 'rf'],
 'lw': ['rw'],
 'lam': ['cam', 'ram'],
 'lcm': ['cm', 'rcm'],
 'lm': ['rm'],
 'ldm': ['cdm', 'rdm'],
 'lwb': ['rwb'],
 'lcb': ['cb', 'rcb'],
 'lb': ['rb']}
```

```
fifa20_upd_scl[['st', 'rs',
 'cf', 'rf',
 'rw',
  'cam', 'ram',
  'cm', 'rcm',
  'rm',
  'cdm', 'rdm',
  'rwb',
  'cb', 'rcb',
  'rb']].head()
  #Checking columns are duplicate or not
```

	st	rs	cf	rf	rw	cam	ram	cm	rcm	rm	cdm	rdm	rwb	
0	1.823542	1.823542	1.926804	1.926804	1.905263	1.911920	1.911920	1.712003	1.712003	1.835064	0.758242	0.758242	0.837146	(
1	1.968237	1.968237	1.832760	1.832760	1.764831	1.724170	1.724170	1.470541	1.470541	1.694257	0.566184	0.566184	0.739820	(
2	1.630615	1.630615	1.785738	1.785738	1.811641	1.818045	1.818045	1.518833	1.518833	1.741193	0.566184	0.566184	0.788483	-(
3	-2.565541	-2.565541	-2.540294	-2.540294	-2.541747	-2.547158	-2.547158	-2.586022	-2.586022	-2.576875	-2.506745	-2.506745	-2.569266	-2
4	1.582383	1.582383	1.738716	1.738716	1.764831	1.771107	1.771107	1.567125	1.567125	1.741193	0.662213	0.662213	0.788483	(

```
#Dropping Duplicate columns
for one_list in duplicate_col.values():
    fifa20_upd_scl.drop(columns=one_list,inplace=True)
```

```
fifa20_upd_scl.shape
```

```
(18278, 56)
```

Filter Method

VARIANCE THRESHOLD

```
#------------------1)VARIANCE THRESHOLD-----------------------------------
#Importing Variance Threshold And creating object with desired threshold
from sklearn.feature_selection import VarianceThreshold
sel=VarianceThreshold(threshold=0.1)
```

```
#Fitting variance threshold on x_train and removing those cols that have variance less than 0.05
sel.fit(fifa20_upd_scl)
```

```
VarianceThreshold
VarianceThreshold(threshold=0.1)
```

```
#checking how many fetures have variance more than 0.05
sum(sel.get_support())
```

```
56
```

NO Feature have variance below 0.1

So We do not drop any column

Correlation based Filter Selection

```
corr_matrix=fifa20_upd_scl.corr()
corr_matrix
```

	age	height_cm	weight_kg	player_positions	work_rate	pace	shooting	passing	dribbling	defend
age	1.000000	0.081391	0.237169	-0.031916	-0.093286	-0.149646	0.079445	0.079801	-0.001230	0.111
height_cm	0.081391	1.000000	0.768816	-0.131949	0.185027	-0.505643	-0.403903	-0.450774	-0.497105	-0.133
weight_kg	0.237169	0.768816	1.000000	-0.084519	0.129082	-0.455300	-0.315243	-0.373093	-0.420207	-0.107
player_positions	-0.031916	-0.131949	-0.084519	1.000000	-0.182683	0.233084	0.332701	0.055350	0.193918	-0.356
work_rate	-0.093286	0.185027	0.129082	-0.182683	1.000000	-0.347574	-0.364284	-0.332218	-0.351667	-0.176
pace	-0.149646	-0.505643	-0.455300	0.233084	-0.347574	1.000000	0.795246	0.847413	0.910737	0.553
shooting	0.079445	-0.403903	-0.315243	0.332701	-0.364284	0.795246	1.000000	0.880694	0.911618	0.387
passing	0.079801	-0.450774	-0.373093	0.055350	-0.332218	0.847413	0.880694	1.000000	0.963203	0.686
dribbling	-0.001230	-0.497105	-0.420207	0.193918	-0.351667	0.910737	0.911618	0.963203	1.000000	0.597
defending	0.111014	-0.133681	-0.107868	-0.356437	-0.176379	0.553123	0.387169	0.686713	0.597507	1.000
physic	0.096377	-0.156552	-0.092064	0.003510	-0.299504	0.779378	0.711828	0.826584	0.809736	0.816
gk_diving	0.094771	0.371909	0.342477	-0.068987	0.269350	-0.887481	-0.773387	-0.871296	-0.890026	-0.717
gk_handling	0.095913	0.372211	0.342873	-0.069019	0.269473	-0.887886	-0.773740	-0.871694	-0.890432	-0.718
gk_kicking	0.094069	0.370081	0.341016	-0.068958	0.269235	-0.887102	-0.773057	-0.870924	-0.889646	-0.717
gk_reflexes	0.094970	0.371596	0.342246	-0.068945	0.269186	-0.886940	-0.772916	-0.870765	-0.889484	-0.717
gk_speed	0.108867	0.354082	0.328340	-0.066625	0.260127	-0.857094	-0.746907	-0.841464	-0.859552	-0.693
gk_positioning	0.103689	0.371929	0.344271	-0.068843	0.268787	-0.885626	-0.771771	-0.869476	-0.888166	-0.716
attacking_crossing	0.128189	-0.498103	-0.401643	0.147927	-0.336692	0.756305	0.767710	0.892675	0.856284	0.542
attacking_finishing	0.078531	-0.376091	-0.291504	0.444910	-0.346363	0.680065	0.953057	0.742992	0.798130	0.162
attacking_heading_accuracy	0.149946	0.010296	0.039679	0.056639	-0.251892	0.609688	0.630890	0.682152	0.669375	0.711
attacking_short_passing	0.135681	-0.371445	-0.291090	-0.023007	-0.300166	0.726822	0.799736	0.941380	0.880944	0.674
attacking_volleys	0.143063	-0.351059	-0.260903	0.347775	-0.336635	0.658112	0.910841	0.765281	0.791787	0.245
skill_dribbling	0.016735	-0.500107	-0.415861	0.259826	-0.356890	0.849083	0.904264	0.919376	0.970215	0.475
skill_curve	0.141891	-0.447578	-0.349230	0.190647	-0.338128	0.685495	0.839540	0.849219	0.826614	0.396
skill_fk_accuracy	0.190419	-0.410071	-0.310277	0.073908	-0.284529	0.592361	0.782000	0.800528	0.746816	0.393
skill_long_passing	0.184008	-0.335343	-0.259906	-0.187131	-0.242842	0.576393	0.658597	0.856821	0.746289	0.667
skill_ball_control	0.088359	-0.425013	-0.340154	0.162757	-0.343363	0.823992	0.897209	0.949290	0.963660	0.584
movement_acceleration	-0.153145	-0.558354	-0.493125	0.311720	-0.339764	0.878554	0.651919	0.658846	0.746121	0.295
movement_sprint_speed	-0.144310	-0.474736	-0.420978	0.320136	-0.337211	0.885558	0.643829	0.646537	0.731508	0.314
movement_agility	-0.023131	-0.633770	-0.545514	0.256751	-0.319113	0.756965	0.683156	0.688548	0.759055	0.262
movement_reactions	0.463331	-0.019698	0.090986	0.011948	-0.219703	0.153815	0.362521	0.388676	0.337892	0.295
movement_balance	-0.089609	-0.789806	-0.669125	0.160343	-0.250882	0.674105	0.575030	0.617959	0.674610	0.262
power_shot_power	0.261916	-0.174280	-0.055855	0.256679	-0.292187	0.368967	0.745463	0.562741	0.546944	0.127
power_jumping	0.185964	-0.041149	0.031246	0.010581	-0.137918	0.219011	0.153209	0.182530	0.189169	0.312
power_stamina	0.115859	-0.291791	-0.219505	0.040645	-0.361512	0.729760	0.649166	0.762037	0.747752	0.684
power_strength	0.341852	0.535400	0.614334	-0.075363	-0.065008	-0.020856	0.074265	0.085934	0.028898	0.340
power_long_shots	0.161616	-0.384213	-0.279097	0.272527	-0.339731	0.667834	0.937791	0.819263	0.822910	0.331
mentality_aggression	0.260420	-0.045307	0.030643	-0.179081	-0.246028	0.468413	0.442742	0.617042	0.548029	0.800
mentality_interceptions	0.196346	-0.053148	-0.028345	-0.453601	-0.118921	0.348399	0.198128	0.524597	0.407478	0.935
mentality_positioning	0.086503	-0.445302	-0.355329	0.386043	-0.378064	0.771647	0.927899	0.844804	0.885098	0.340
mentality_vision	0.195372	-0.371546	-0.280242	0.123156	-0.273979	0.466636	0.711914	0.710605	0.661330	0.233
mentality_penalties	0.141375	-0.335786	-0.246309	0.326579	-0.308561	0.648863	0.879155	0.751371	0.773835	0.281
mentality_composure	0.370149	-0.159296	-0.054167	0.027255	-0.291546	0.442573	0.631673	0.669378	0.623339	0.460
defending_marking	0.157007	-0.077038	-0.049683	-0.418749	-0.135447	0.407975	0.247479	0.561961	0.457968	0.947
defending_standing_tackle	0.116857	-0.068334	-0.052770	-0.464229	-0.112857	0.385884	0.201804	0.538894	0.427713	0.960
defending_sliding_tackle	0.101534	-0.076953	-0.064492	-0.469558	-0.100995	0.369953	0.163033	0.508803	0.399491	0.944

ls	0.047672	-0.396519	-0.319913	0.234039	-0.365330	0.891362	0.955351	0.948434	0.973151	0.590
lw	0.018116	-0.477735	-0.400388	0.212484	-0.364195	0.916207	0.936031	0.969852	0.993354	0.590
lf	0.031265	-0.449754	-0.372693	0.216909	-0.363718	0.901143	0.949147	0.965788	0.988771	0.586
lam	0.033339	-0.459943	-0.383848	0.174640	-0.355525	0.896520	0.934680	0.978804	0.990732	0.613
lm	0.022428	-0.468094	-0.393100	0.172073	-0.360426	0.917017	0.916540	0.978803	0.992226	0.635
lcm	0.068836	-0.412047	-0.340513	0.050179	-0.336176	0.863827	0.876910	0.987949	0.967736	0.739
lwb	0.060500	-0.355628	-0.298386	-0.088269	-0.303106	0.833571	0.719012	0.923186	0.885179	0.893
ldm	0.096439	-0.287180	-0.232918	-0.159508	-0.276361	0.766016	0.680997	0.899800	0.841918	0.926
lb	0.062642	-0.318489	-0.267643	-0.129978	-0.286281	0.805133	0.671379	0.892036	0.847436	0.924
lcb	0.102474	-0.186565	-0.143984	-0.213972	-0.240479	0.696822	0.570531	0.807428	0.746049	0.969

```
#Storing columns in columns variable
columns=corr_matrix.columns
# Get the column names of the DataFrame
columns = corr_matrix.columns

# Create an empty list to keep track of columns to drop
columns_to_drop = []

# Loop over the columns
for i in range(len(columns)):
    for j in range(i + 1, len(columns)):
        # Access the cell of the DataFrame
        if corr_matrix.loc[columns[i], columns[j]] > 0.95:
            columns_to_drop.append(columns[j])

print(columns_to_drop)
```

```
['attacking_finishing', 'ls', 'dribbling', 'lw', 'lf', 'lam', 'lm', 'lcm', 'skill_dribbling', 'skill_ball_control', 'ls', 'lw', 'lf', '
```

```
columns_to_drop = set(columns_to_drop)


len(columns_to_drop)
```

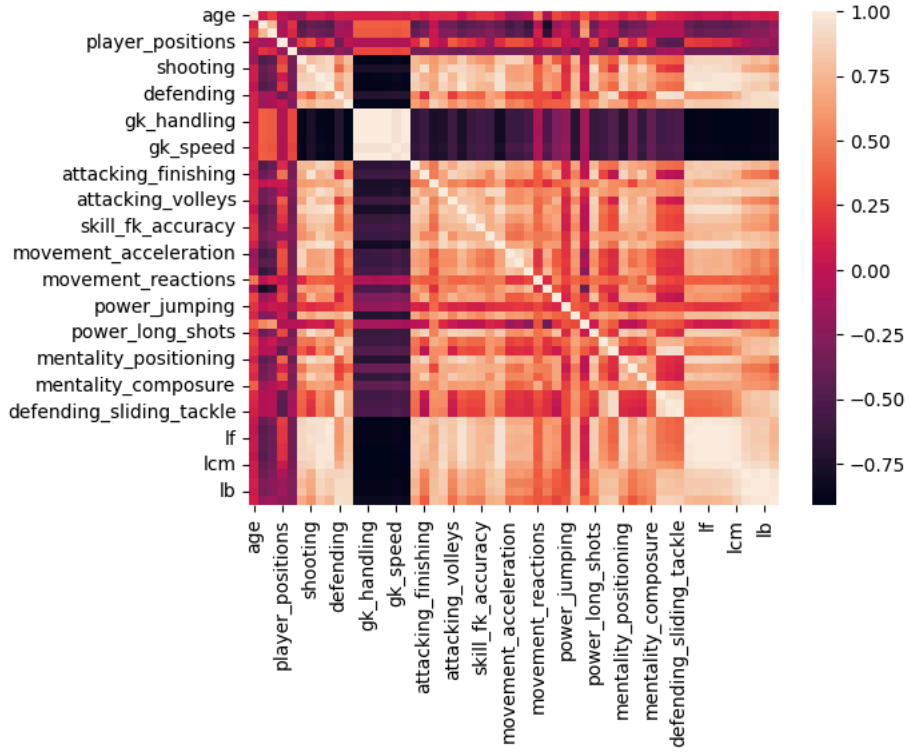
```
21
```

```
columns_to_drop
```

```
{'attacking_finishing',
 'defending_sliding_tackle',
 'defending_standing_tackle',
 'dribbling',
 'gk_handling',
 'gk_kicking',
 'gk_positioning',
 'gk_reflexes',
 'gk_speed',
 'lam',
 'lb',
 'lcb',
 'lcm',
 'ldm',
 'lf',
 'lm',
 'ls',
 'lw',
 'lwb',
 'skill_ball_control',
 'skill_dribbling'}
```

```
sns.heatmap(corr_matrix)
```

```
<Axes: >
```

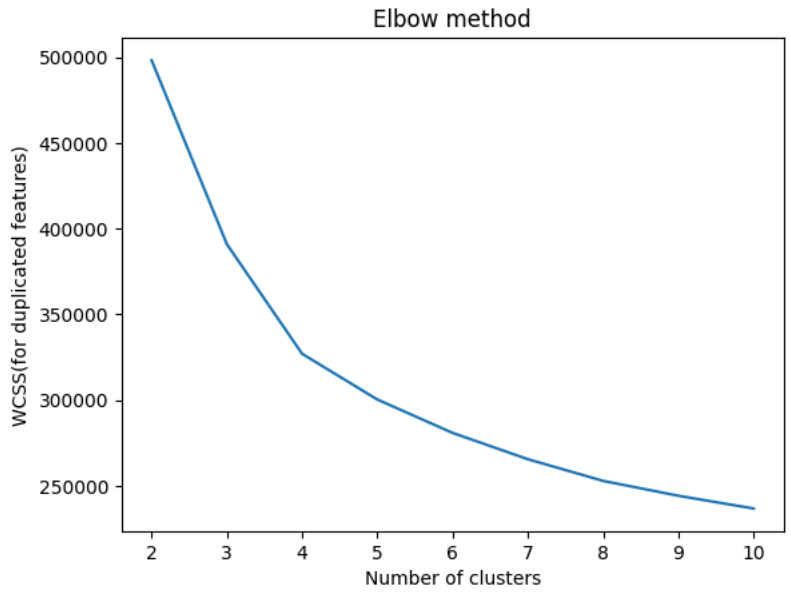


There Are more than 21 columns having correlation more than 0.95

Hence dropping them will lead to data loss hence keeping them

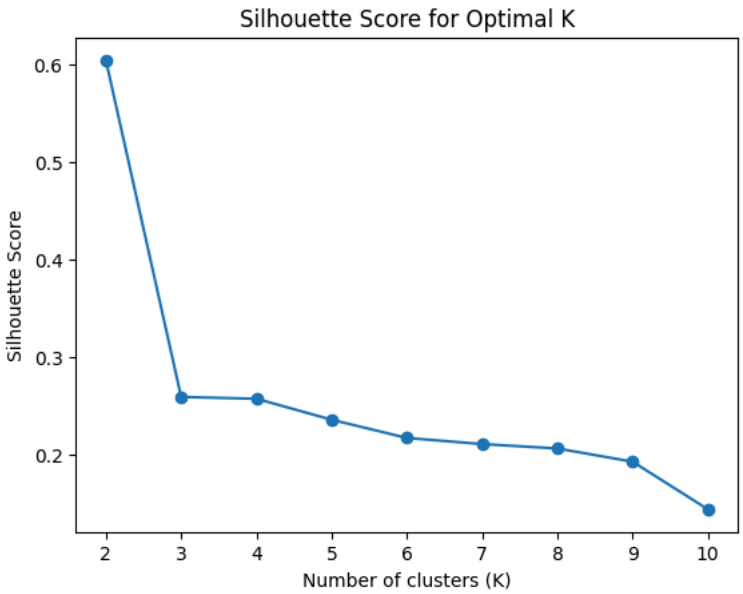
MODEL CREATION

```
from sklearn.cluster import KMeans

wcss_= []
for cluster in range(2,11):
    kme_fifa20 = KMeans(n_clusters=cluster, random_state=9)
    kme_fifa20.fit(fifa20_upd_scl)
    wcss_.append(kme_fifa20.inertia_)

plt.plot(range(2,11), wcss_)
plt.title("Elbow method")
plt.xlabel("Number of clusters")
plt.ylabel("WCSS(for duplicated features)")
plt.show()
```



Start coding or generate with AI.

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

def find_best_k_silhouette(df, max_k):
    """
    Find the best K value using silhouette score.
    """
    silhouette_scores = []

    for k in range(2, max_k + 1):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(df)
        labels = kmeans.labels_
        silhouette_avg = silhouette_score(df, labels)
        silhouette_scores.append(silhouette_avg)

    # Plot the silhouette scores against K values
    plt.plot(range(2, max_k + 1), silhouette_scores, marker='o')
    plt.xlabel('Number of clusters (K)')
    plt.ylabel('Silhouette Score')
    plt.title('Silhouette Score for Optimal K')
    plt.show()

    # Find the best K value using the highest silhouette score
    best_k = np.argmax(silhouette_scores) + 2  # Add 2 because K starts from 2
    return best_k

best_k_silhouette = find_best_k_silhouette(fifa20_upd_scl, max_k=10)
print("Best K value using silhouette score:", best_k_silhouette)
```



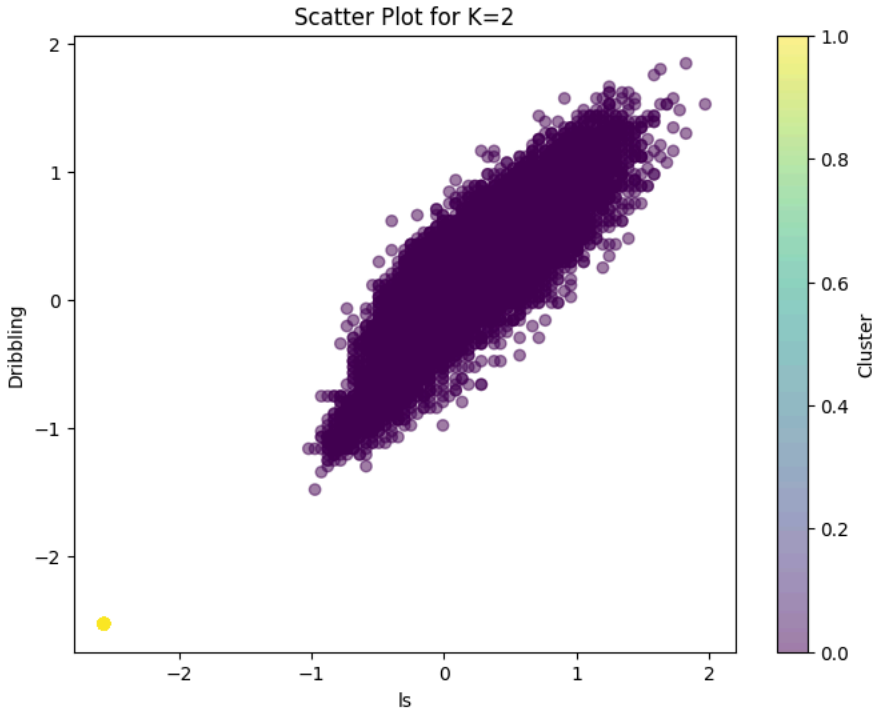

```
Best K value using silhouette score: 2
```

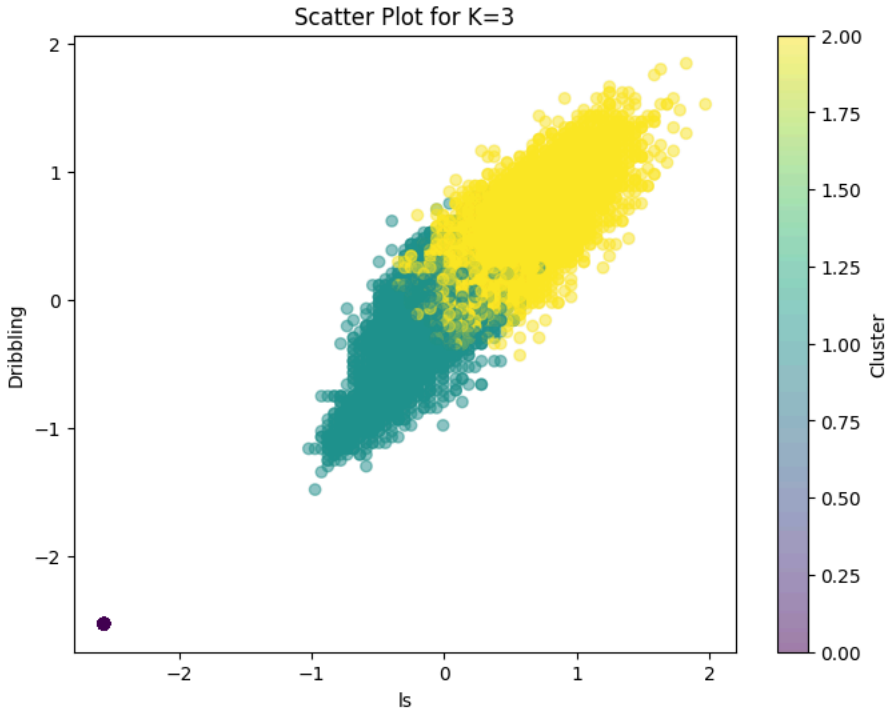
CREATING KMEANS FOR CLUSTER 2 TO 5

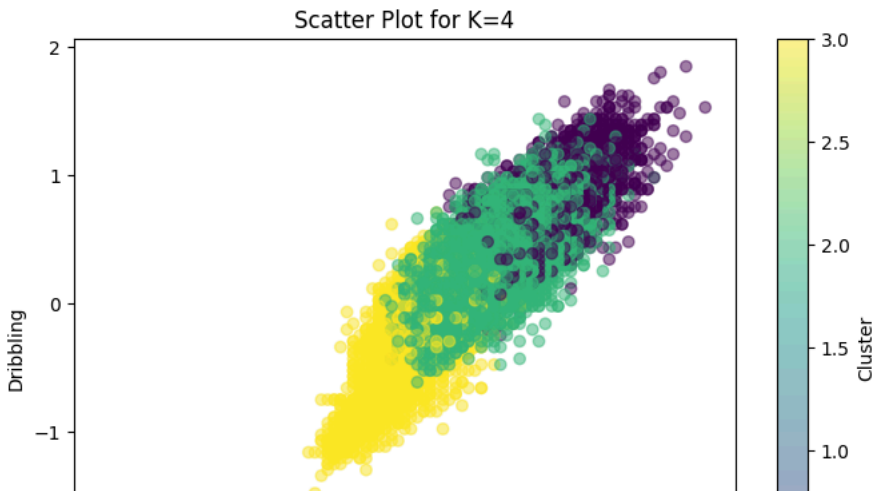
```
from sklearn.cluster import KMeans
#For Cluster 2
kmeans_clus2=KMeans(n_clusters=2, random_state=42)
kmeans_clus2.fit(fifa20_upd_scl)
kmeans_clus2_lables= kmeans_clus2.labels_

#For Cluster 3
kmeans_clus3=KMeans(n_clusters=3, random_state=42)
kmeans_clus3.fit(fifa20_upd_scl)
kmeans_clus3_lables= kmeans_clus3.labels_

#For Cluster 4
kmeans_clus4=KMeans(n_clusters=4, random_state=42)
kmeans_clus4.fit(fifa20_upd_scl)
kmeans_clus4_lables= kmeans_clus4.labels_

#For Cluster 5
kmeans_clus5=KMeans(n_clusters=5, random_state=42)
kmeans_clus5.fit(fifa20_upd_scl)
kmeans_clus5_lables= kmeans_clus5.labels_
```

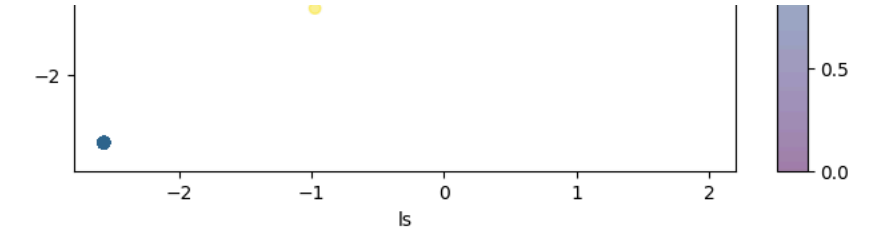
Striker VS Dribbling

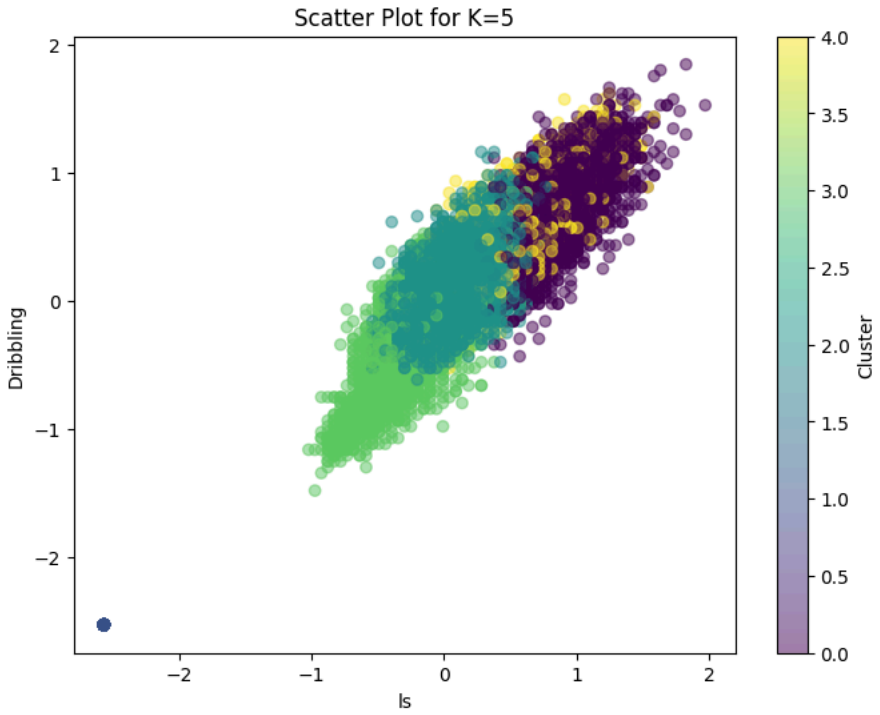
```
import matplotlib.pyplot as plt

def plot_clusters(df, labels, k):
    """
    Plot scatter plot for each cluster.
    """
    plt.figure(figsize=(8, 6))
    plt.scatter(df['ls'], df['dribbling'], c=labels, cmap='viridis', alpha=0.5)
    plt.title(f'Scatter Plot for K={k}')
    plt.xlabel('ls')
    plt.ylabel('Dribbling')
    plt.colorbar(label='Cluster')
    plt.show()


plot_clusters(fifa20_upd_scl, kmeans_clus2_lables, k=2)
plot_clusters(fifa20_upd_scl, kmeans_clus3_lables, k=3)
plot_clusters(fifa20_upd_scl, kmeans_clus4_lables, k=4)
plot_clusters(fifa20_upd_scl, kmeans_clus5_lables, k=5)
```


Scatter Plot for K=2
Dribbling
ls
Cluster


Scatter Plot for K=3
Dribbling
ls
Cluster

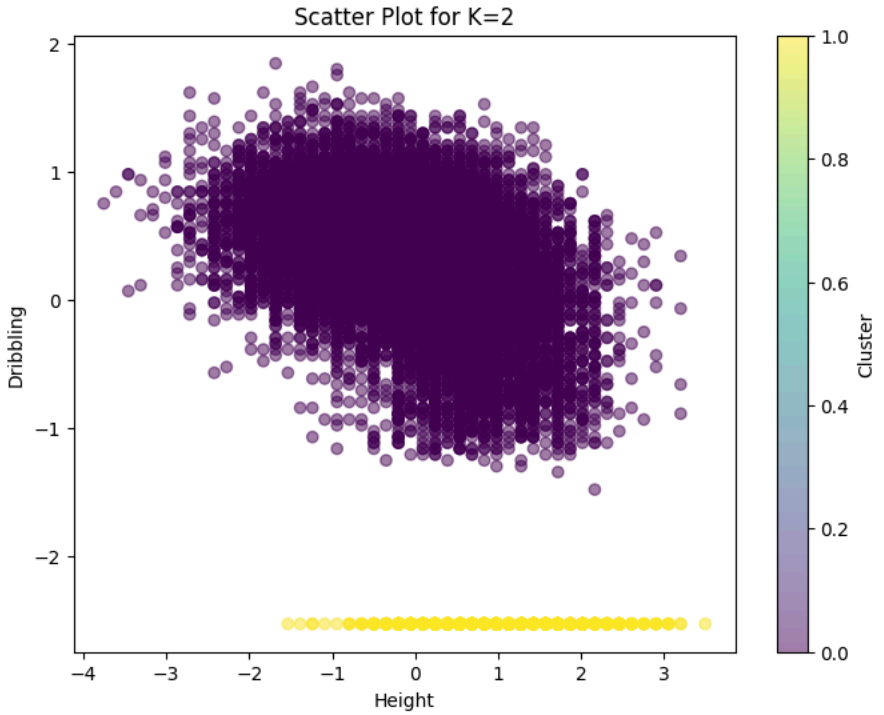

Scatter Plot for K=4
Dribbling
Cluster

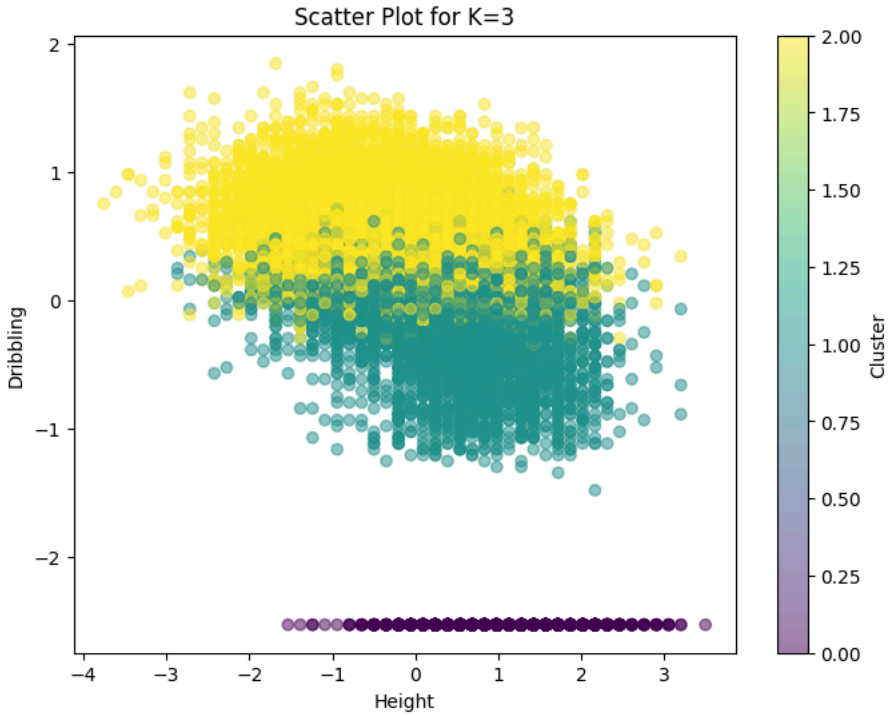


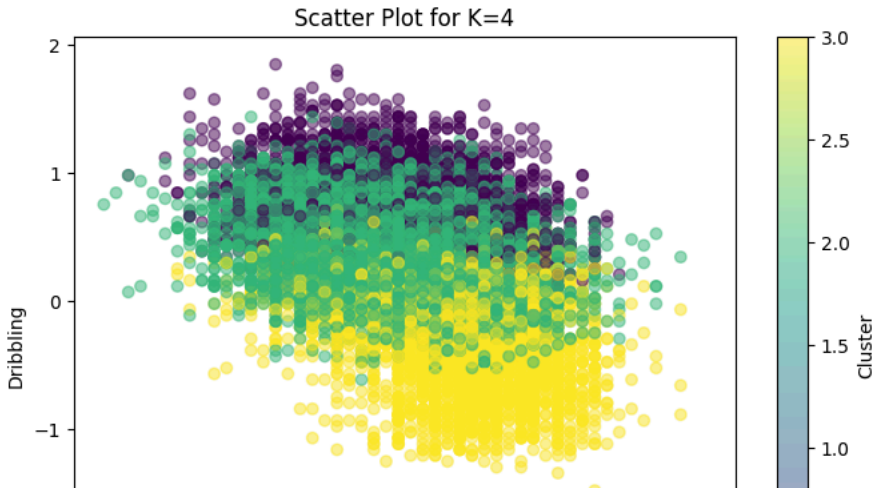


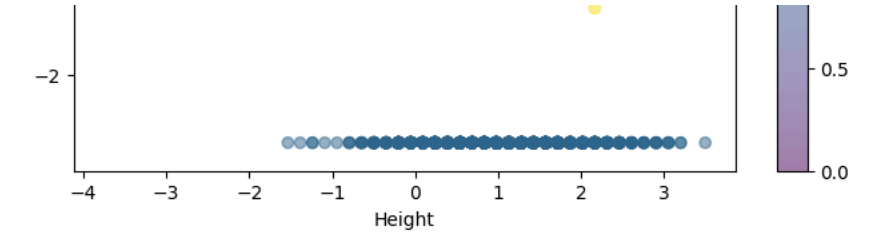
Height Vs Dribbling

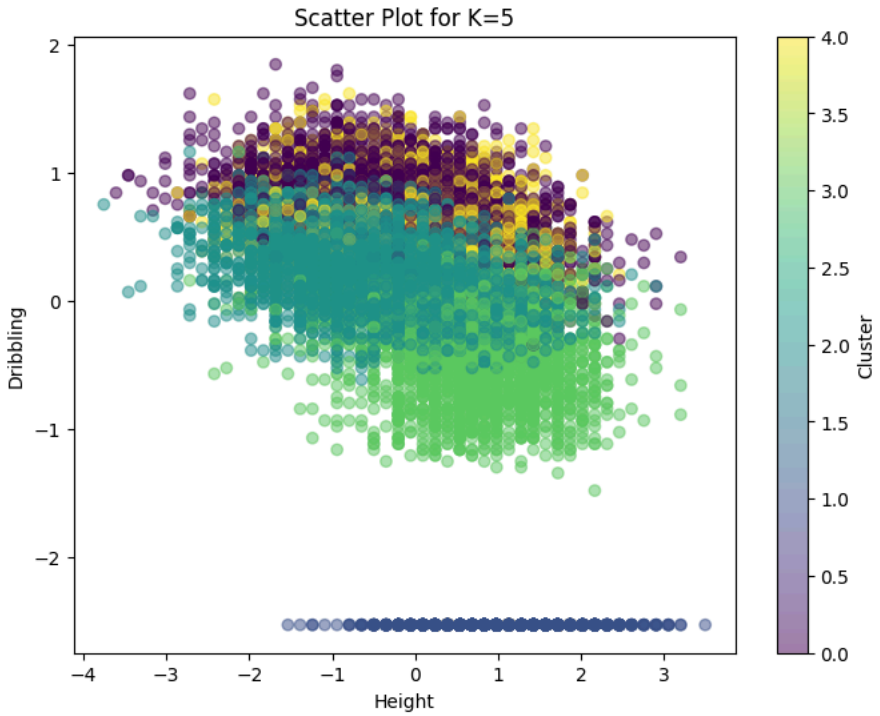
```
def plot_clusters(df, labels, k):
    """
    Plot scatter plot for each cluster.
    """
    plt.figure(figsize=(8, 6))
    plt.scatter(df['height_cm'], df['dribbling'], c=labels, cmap='viridis', alpha=0.5)
    plt.title(f'Scatter Plot for K={k}')
    plt.xlabel('Height')
    plt.ylabel('Dribbling')
    plt.colorbar(label='Cluster')
    plt.show()


plot_clusters(fifa20_upd_scl, kmeans_clus2_lables, k=2)
plot_clusters(fifa20_upd_scl, kmeans_clus3_lables, k=3)
plot_clusters(fifa20_upd_scl, kmeans_clus4_lables, k=4)
plot_clusters(fifa20_upd_scl, kmeans_clus5_lables, k=5)
```


Scatter Plot for K=2
Dribbling
Height
Cluster


Scatter Plot for K=3
Dribbling
Height
Cluster


Scatter Plot for K=4
Dribbling
Cluster



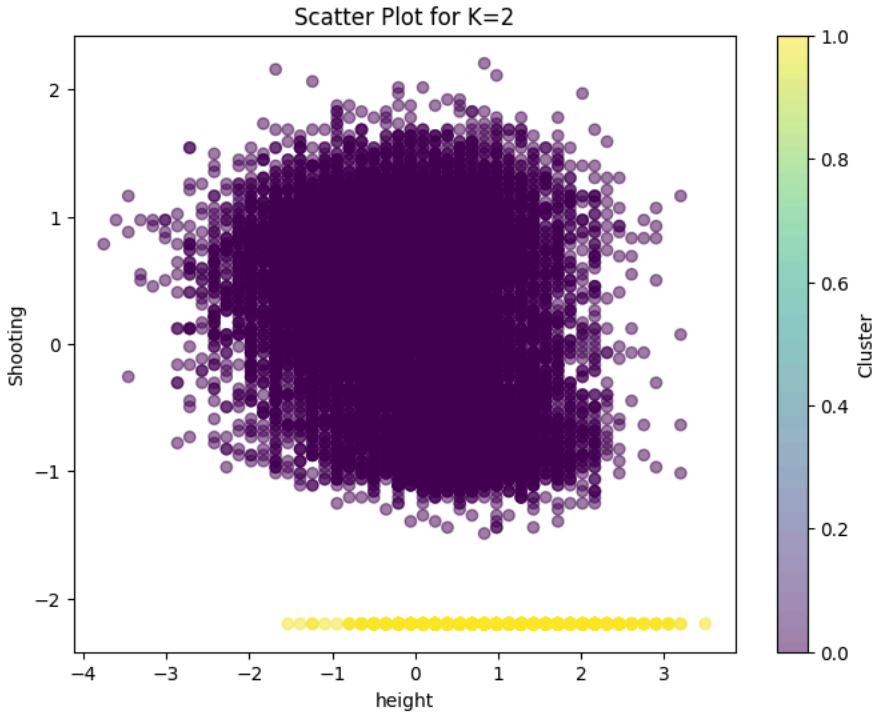


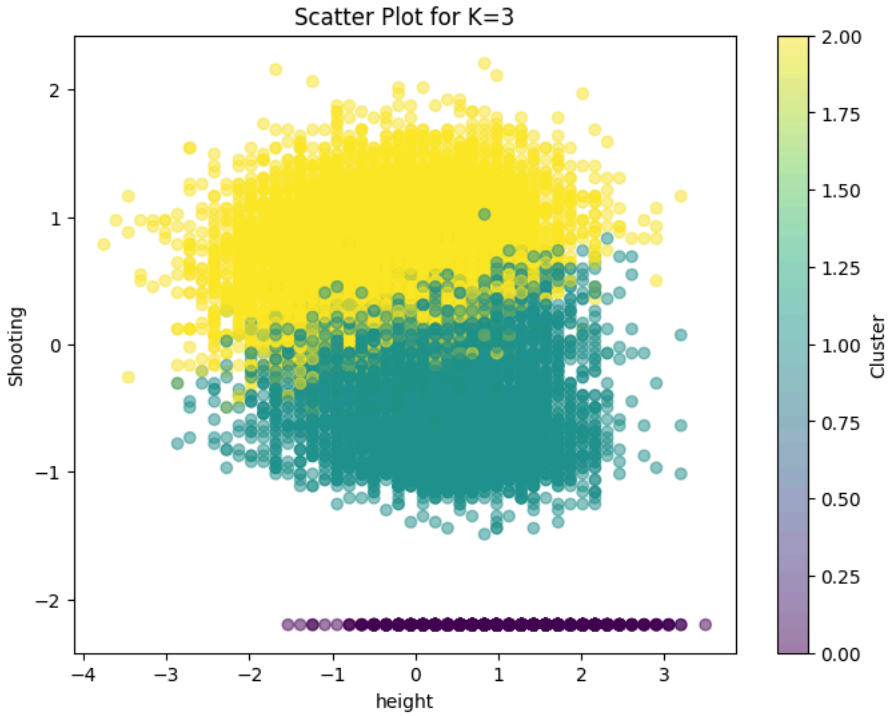

```
fifa20_upd_scl.head()
```

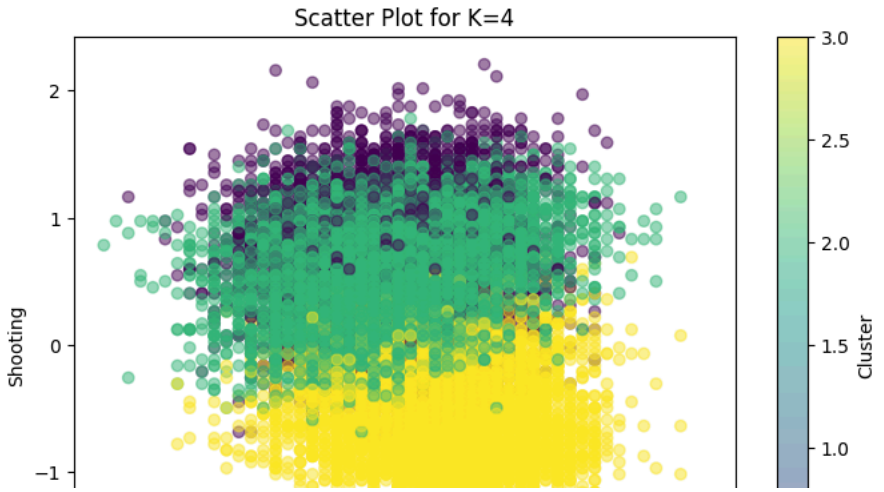
	age	height_cm	weight_kg	player_positions	work_rate	pace	shooting	passing	dribbling	defending	physic	gk_diving
0	1.442333	-1.681598	-0.464891	1.433033	0.415506	1.127098	2.156700	2.006332	1.843560	-0.303796	0.372952	-0.351302
1	1.871809	0.834394	1.095935	1.822451	-1.691770	1.253076	2.204072	1.518666	1.524399	-0.482213	0.908933	-0.351302
2	0.368643	-0.941600	-1.032464	0.408248	-1.340558	1.295069	1.825099	1.762499	1.797965	-0.616026	0.015632	-0.351302
3	0.153905	0.982394	1.663508	-0.196375	0.766719	-2.526264	-2.201490	-2.480193	-2.533508	-2.043363	-2.574941	3.842656
4	0.583381	-0.941600	-0.181104	0.438991	-1.340558	1.295069	1.730355	1.713733	1.752371	-0.482213	0.372952	-0.351302

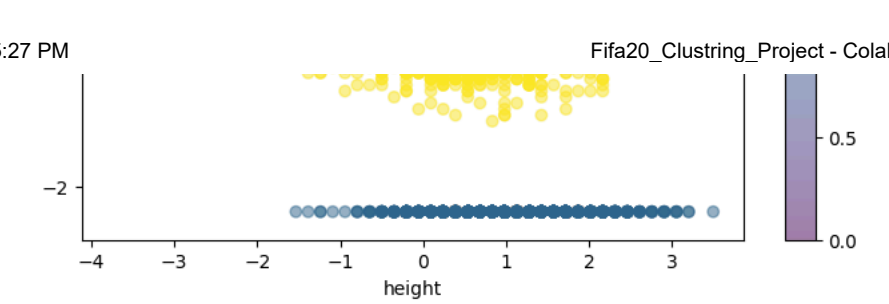
Height VS Shooting

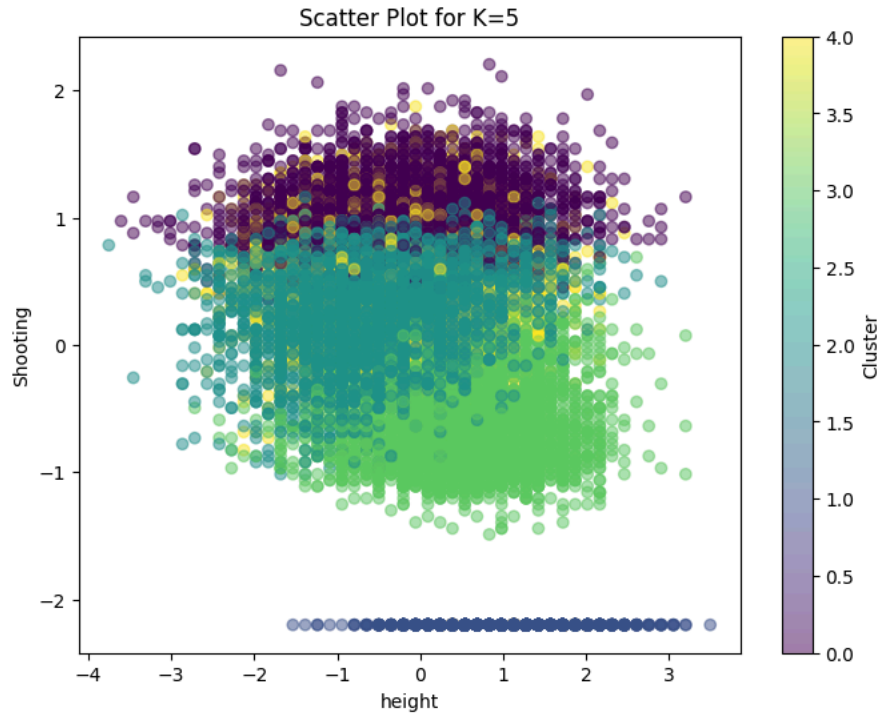
```
def plot_clusters(df, labels, k):
    """
    Plot scatter plot for each cluster.
    """
    plt.figure(figsize=(8, 6))
    plt.scatter(df['height_cm'], df['shooting'], c=labels, cmap='viridis', alpha=0.5)
    plt.title(f'Scatter Plot for K={k}')
    plt.xlabel('height')
    plt.ylabel('Shooting')
    plt.colorbar(label='Cluster')
    plt.show()


plot_clusters(fifa20_upd_scl, kmeans_clus2_lables, k=2)
plot_clusters(fifa20_upd_scl, kmeans_clus3_lables, k=3)
plot_clusters(fifa20_upd_scl, kmeans_clus4_lables, k=4)
plot_clusters(fifa20_upd_scl, kmeans_clus5_lables, k=5)
```


Scatter Plot for K=2
Shooting
height
Cluster


Scatter Plot for K=3
Shooting
height
Cluster

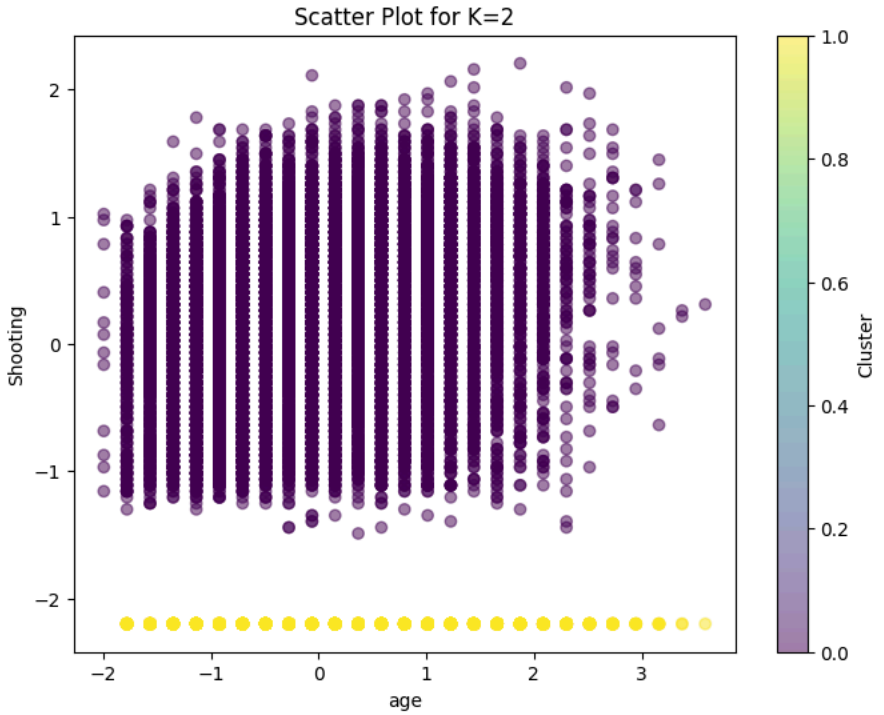

Scatter Plot for K=4
Shooting
Cluster

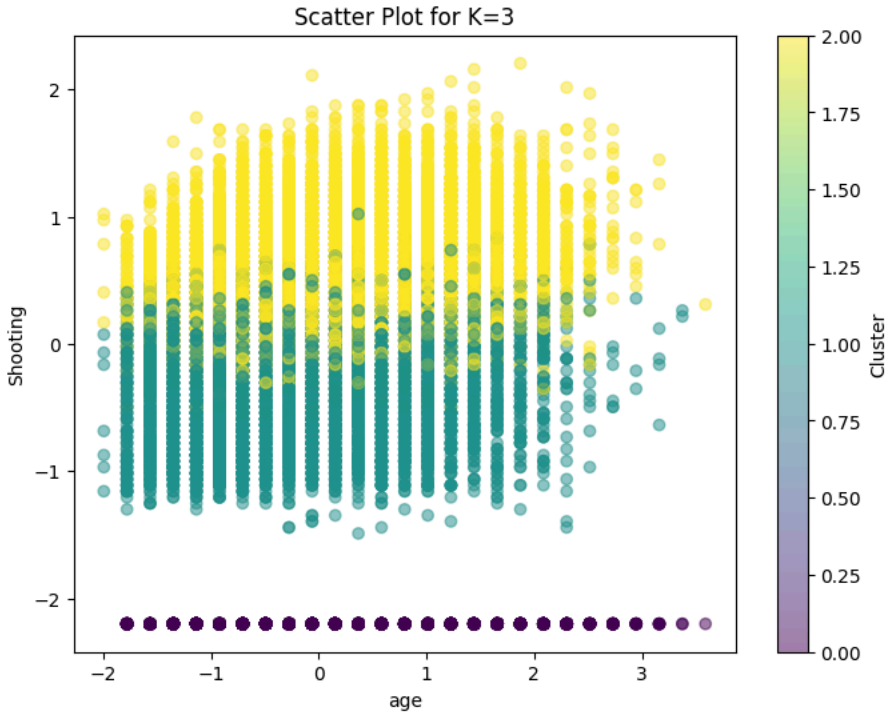


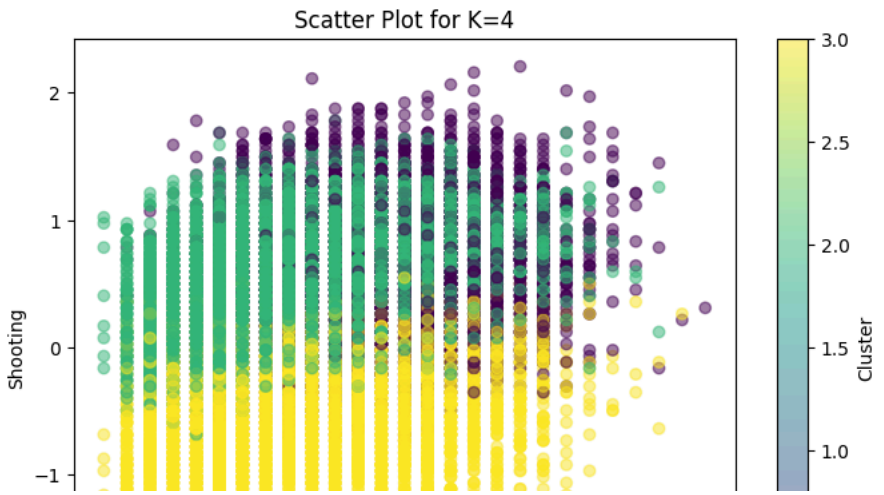


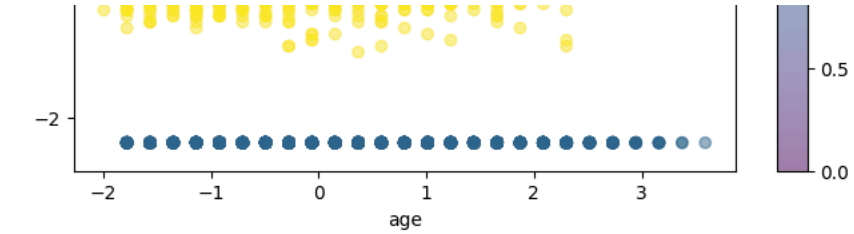
Age VS Shooting

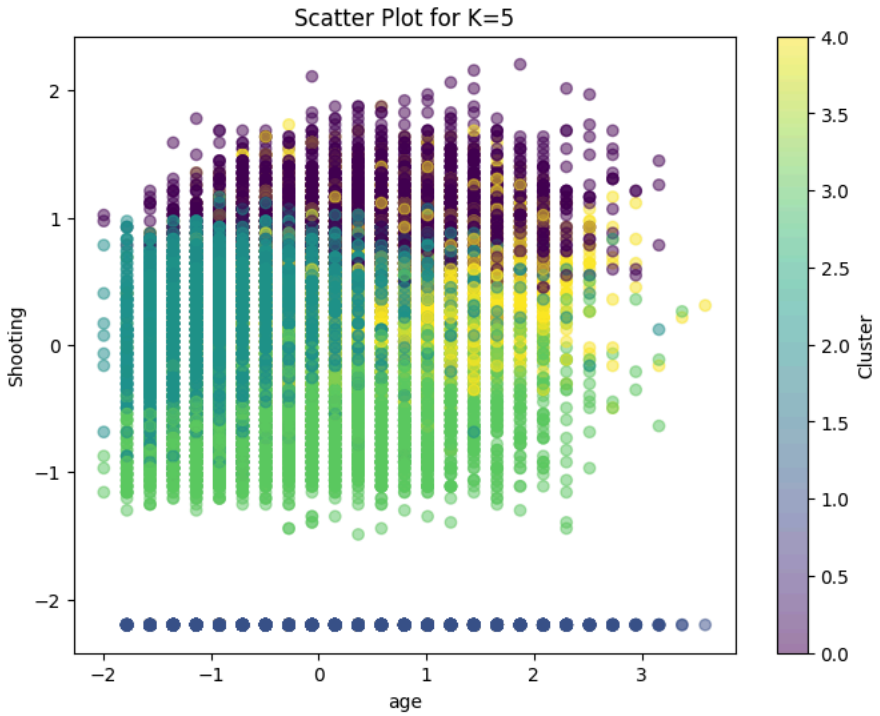
```
def plot_clusters(df, labels, k):
    """
    Plot scatter plot for each cluster.
    """
    plt.figure(figsize=(8, 6))
    plt.scatter(df['age'], df['shooting'], c=labels, cmap='viridis', alpha=0.5)
    plt.title(f'Scatter Plot for K={k}')
    plt.xlabel('age')
    plt.ylabel('Shooting')
    plt.colorbar(label='Cluster')
    plt.show()


plot_clusters(fifa20_upd_scl, kmeans_clus2_lables, k=2)
plot_clusters(fifa20_upd_scl, kmeans_clus3_lables, k=3)
plot_clusters(fifa20_upd_scl, kmeans_clus4_lables, k=4)
plot_clusters(fifa20_upd_scl, kmeans_clus5_lables, k=5)
```


Scatter Plot for K=2
Shooting
age
Cluster


Scatter Plot for K=3
Shooting
age
Cluster


Scatter Plot for K=4
Shooting
Cluster



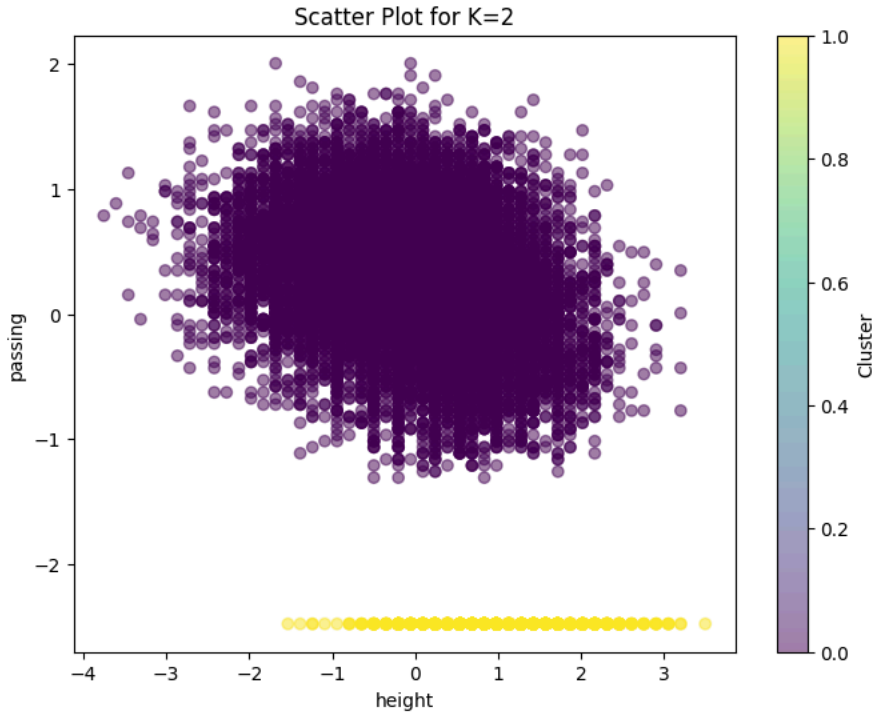


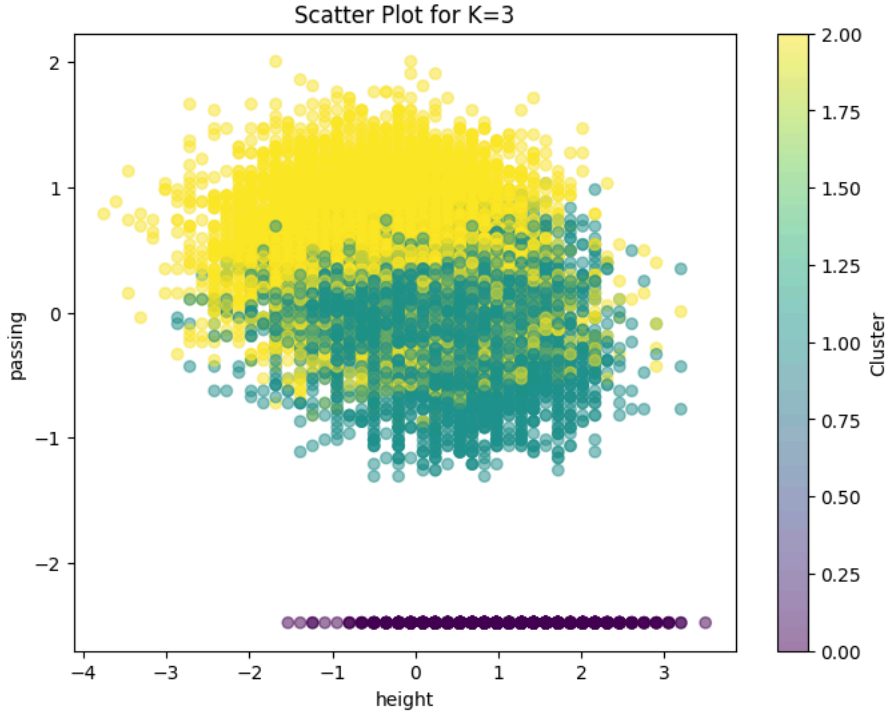
Height Vs Passing

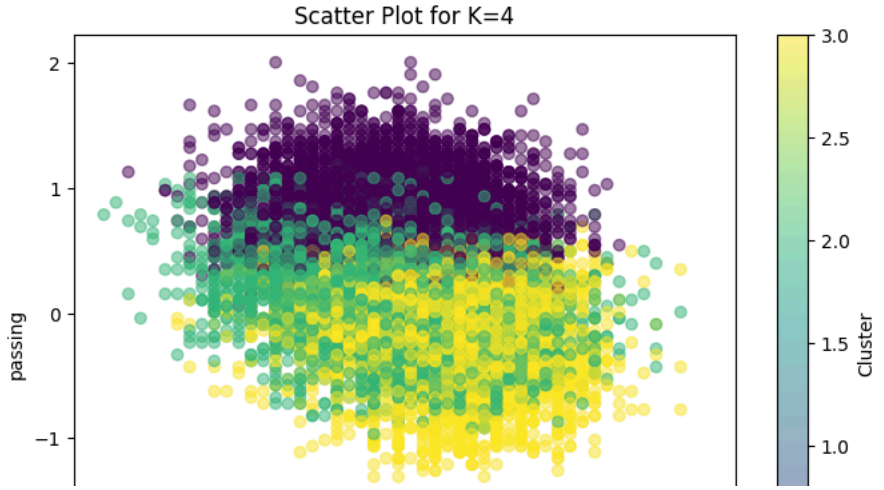
```
fifa20_upd_scl.head(2)
```

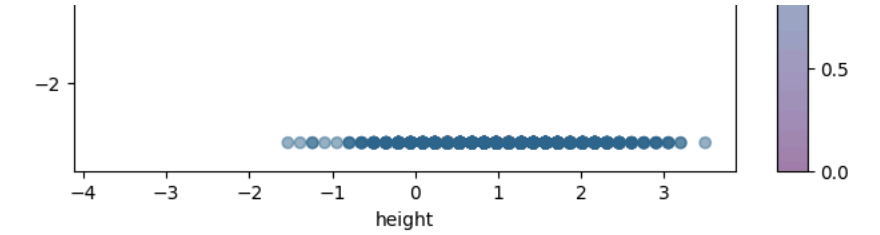
gk_kicking	gk_reflexes	gk_speed	gk_positioning	attacking_crossing	attacking_finishing	attacking_heading_accuracy	attacking_short_p
-0.351152	-0.351088	-0.339273	-0.350568	2.089048	2.521670	1.020116	2.
-0.351152	-0.351088	-0.339273	-0.350568	1.870766	2.470634	2.110319	1.

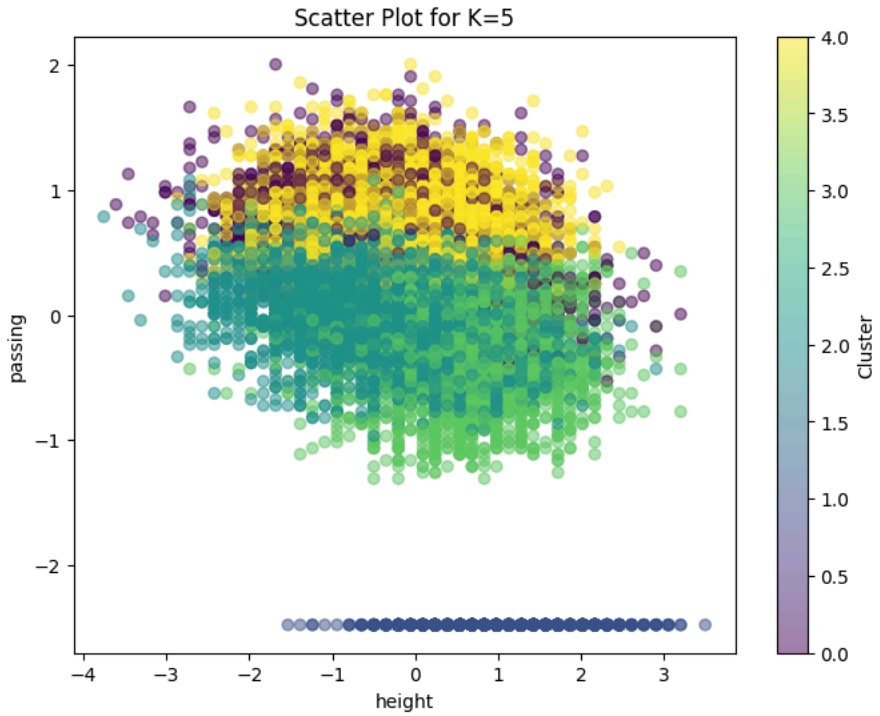
```
def plot_clusters(df, labels, k):
    """
    Plot scatter plot for each cluster.
    """
    plt.figure(figsize=(8, 6))
    plt.scatter(df['height_cm'], df['passing'], c=labels, cmap='viridis', alpha=0.5)
    plt.title(f'Scatter Plot for K={k}')
    plt.xlabel('height')
    plt.ylabel('passing')
    plt.colorbar(label='Cluster')
    plt.show()


plot_clusters(fifa20_upd_scl, kmeans_clus2_lables, k=2)
plot_clusters(fifa20_upd_scl, kmeans_clus3_lables, k=3)
plot_clusters(fifa20_upd_scl, kmeans_clus4_lables, k=4)
plot_clusters(fifa20_upd_scl, kmeans_clus5_lables, k=5)
```


Scatter Plot for K=2
passing
height
Cluster


Scatter Plot for K=3
passing
height
Cluster


Scatter Plot for K=4
passing
Cluster





PCA

```
n_components = 10
pca = PCA(n_components=n_components)
fifa20_pca = pca.fit_transform(fifa20_upd_scl)  # Transform data using PCA

# Create DataFrame from transformed data
columns = [f'PC{i}' for i in range(1, n_components + 1)]
fifa20_pca_df = pd.DataFrame(data=fifa20_pca, columns=columns)

# Variance explained by each principal component
explained_variance_ratio = pca.explained_variance_ratio_

# Total variance explained
total_variance_explained = explained_variance_ratio.sum()

# Print variance information
print("Variance explained by each principal component:")
for i, var in enumerate(explained_variance_ratio, 1):
    print(f"Principal Component {i}: {var:.2f}")
print(f"Total variance explained: {total_variance_explained:.2f}")
```

```
Variance explained by each principal component:
Principal Component 1: 0.60
Principal Component 2: 0.13
Principal Component 3: 0.07
Principal Component 4: 0.04
Principal Component 5: 0.03
Principal Component 6: 0.02
Principal Component 7: 0.01
Principal Component 8: 0.01
Principal Component 9: 0.01
Principal Component 10: 0.01
Total variance explained: 0.93
```

Total Variance Explained by the pca 1 to 10 is 93%

PCA KMEANS

```
fifa20_pca_df.head()
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
0	-9.927773	-5.830742	4.261144	-2.204207	-0.121338	0.740087	-1.585360	0.722704	0.713019	0.949570
1	-9.169349	-4.630591	6.140112	1.150944	-2.645055	0.038218	-0.606948	-0.501130	0.347484	0.864706
2	-9.026783	-6.179698	2.964572	-2.234732	-0.117984	-0.870910	-0.461850	-0.280842	0.795823	1.598283
3	14.197567	-1.227334	5.141114	-3.507878	-3.314819	-0.958135	-2.371893	-0.967118	0.901303	0.817336
4	-8.907958	-5.502158	3.217509	-1.768408	-0.336655	-1.152592	-0.566927	0.095858	0.378072	1.794139

Next steps: Generate code with `fifa20_pca_df` View recommended plots

```
# For dataframe "# For fifa_data_pca"

# For 2 clusters
kmea_clu_pca_clu2 = KMeans(n_clusters=2, random_state=9)
kmea_clu_pca_clu2.fit(fifa20_pca_df)
print("WCSS for pca with 2 clusters:", kmea_clu_pca_clu2.inertia_)

# For 3 clusters
kmea_clu_pca_clu3 = KMeans(n_clusters=3, random_state=9)
kmea_clu_pca_clu3.fit(fifa20_pca_df)
print("WCSS for pca with 3 clusters:", kmea_clu_pca_clu3.inertia_)

# For 4 clusters
kmea_clu_pca_clu4 = KMeans(n_clusters=4, random_state=9)
kmea_clu_pca_clu4.fit(fifa20_pca_df)
print("WCSS for pca with 4 clusters:", kmea_clu_pca_clu4.inertia_)

# For 5 clusters
kmea_clu_pca_clu5 = KMeans(n_clusters=5, random_state=9)
kmea_clu_pca_clu5.fit(fifa20_pca_df)
print("WCSS for pca with 5 clusters:", kmea_clu_pca_clu5.inertia_)
```

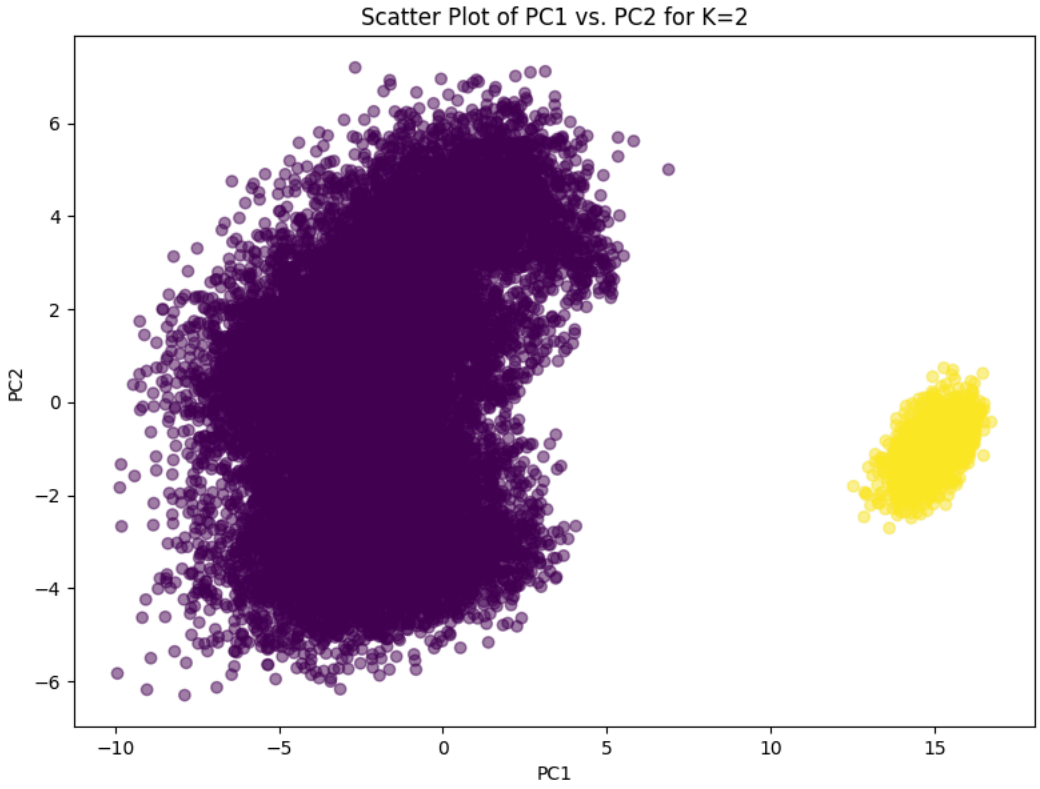
```
WCSS for pca with 2 clusters: 425270.32024429075
WCSS for pca with 3 clusters: 317982.94704059744
WCSS for pca with 4 clusters: 254120.3098089064
WCSS for pca with 5 clusters: 227697.5279327882
```

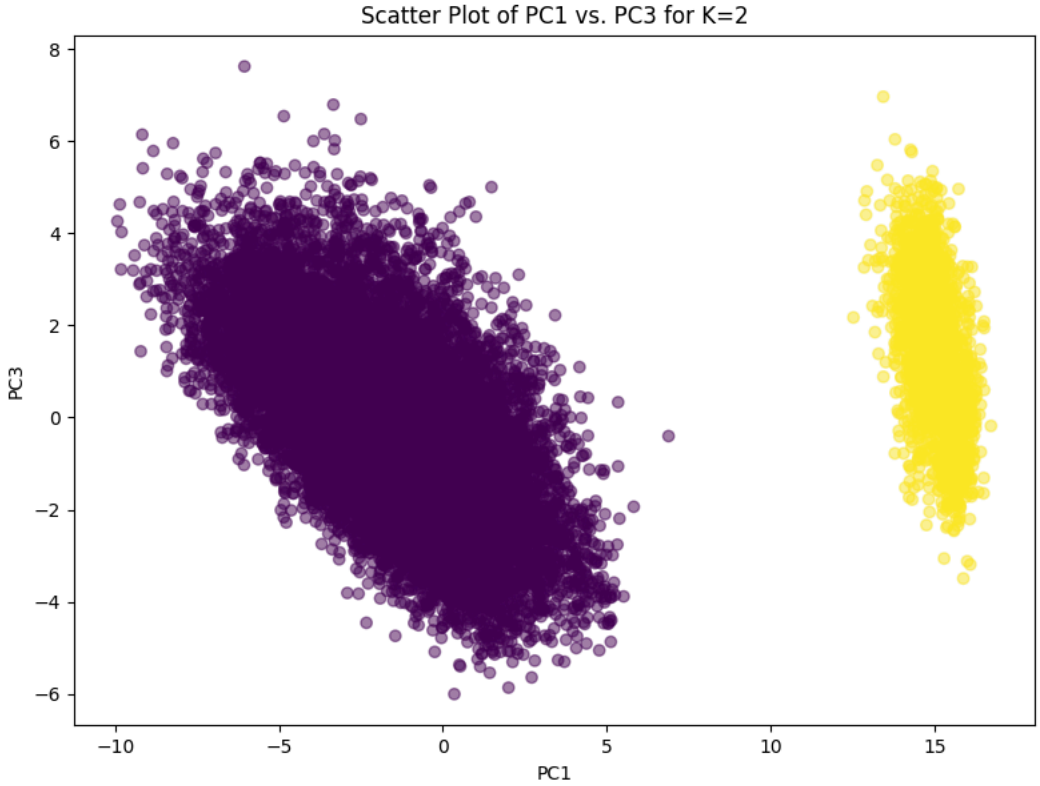
Extracting Labels

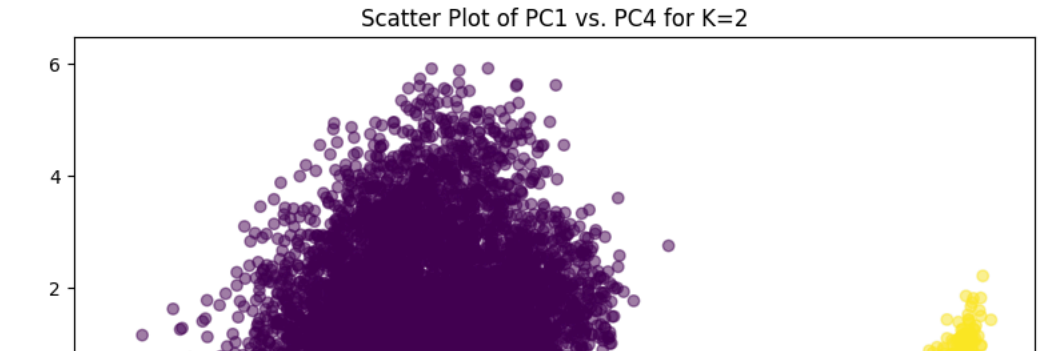
```
# For dataframe "fifa_data_pca

kmea_clu_pca_clu2_labels = kmea_clu_pca_clu2.labels_
kmea_clu_pca_clu3_labels = kmea_clu_pca_clu3.labels_
kmea_clu_pca_clu4_labels = kmea_clu_pca_clu4.labels_
kmea_clu_pca_clu5_labels = kmea_clu_pca_clu5.labels_
```

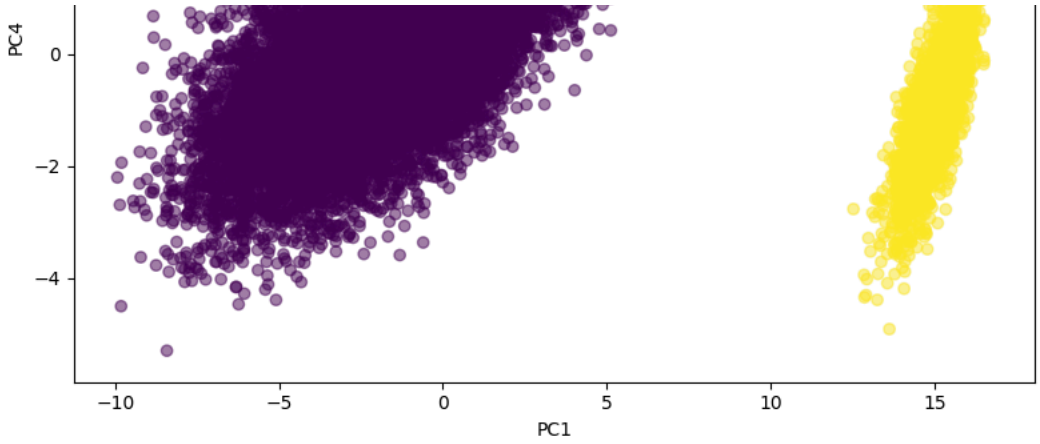
PCA1 VS ALL PCA

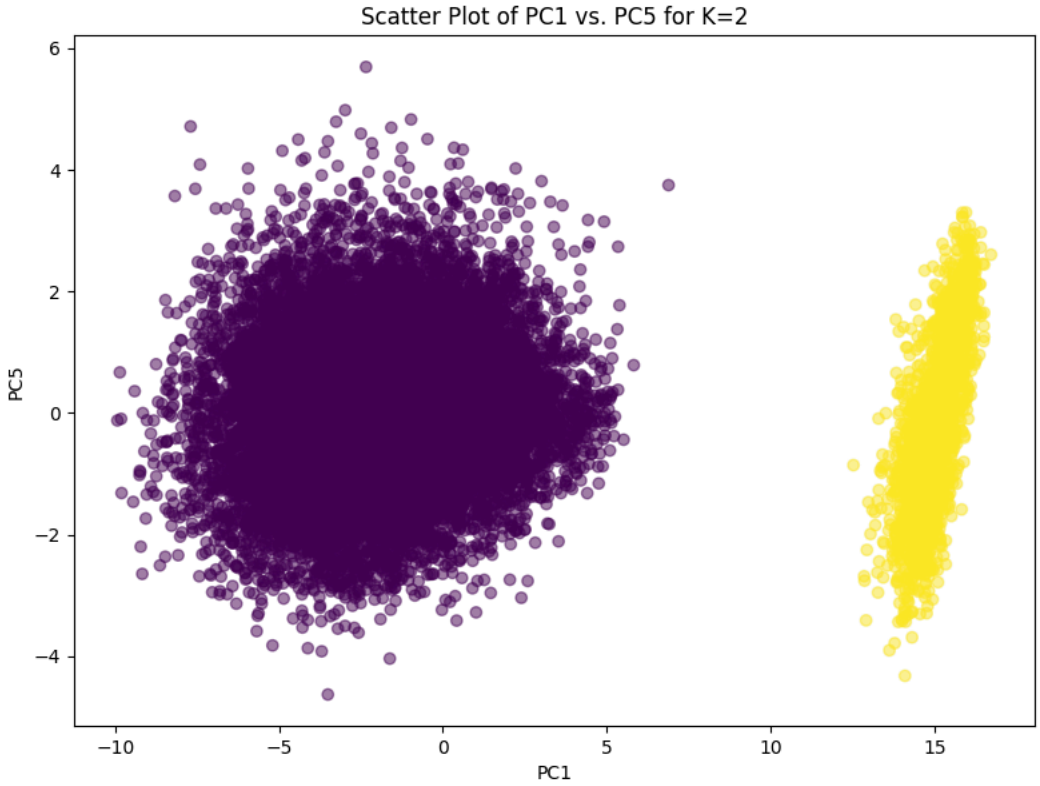
```
import matplotlib.pyplot as plt

def scatter_pca_vs_all(pca_df, labels, k):
    """
    Create scatter plots of PCA1 vs. all other principal components for each K value.
    """
    num_components = pca_df.shape[1]  # Number of principal components
    fig, axes = plt.subplots(num_components - 1, 1, figsize=(8, 6 * (num_components - 1)))

    for i in range(1, num_components):
        ax = axes[i - 1] if num_components > 2 else axes
        ax.scatter(pca_df['PC1'], pca_df[f'PC{i + 1}'], c=labels, cmap='viridis', alpha=0.5)
        ax.set_xlabel('PC1')
        ax.set_ylabel(f'PC{i + 1}')
        ax.set_title(f'Scatter Plot of PC1 vs. PC{i + 1} for K={k}')

    plt.tight_layout()
    plt.show()

# Assuming 'fifa20_pca_df', 'kmea_clu_pca_clu2_labels', 'kmea_clu_pca_clu3_labels', 'kmea_clu_pca_clu4_labels', 'kmea_clu_pca_clu5_labels' a

# For 2 clusters
scatter_pca_vs_all(fifa20_pca_df, kmea_clu_pca_clu2_labels, k=2)

# For 3 clusters
scatter_pca_vs_all(fifa20_pca_df, kmea_clu_pca_clu3_labels, k=3)

# For 4 clusters
scatter_pca_vs_all(fifa20_pca_df, kmea_clu_pca_clu4_labels, k=4)

# For 5 clusters
scatter_pca_vs_all(fifa20_pca_df, kmea_clu_pca_clu5_labels, k=5)
```

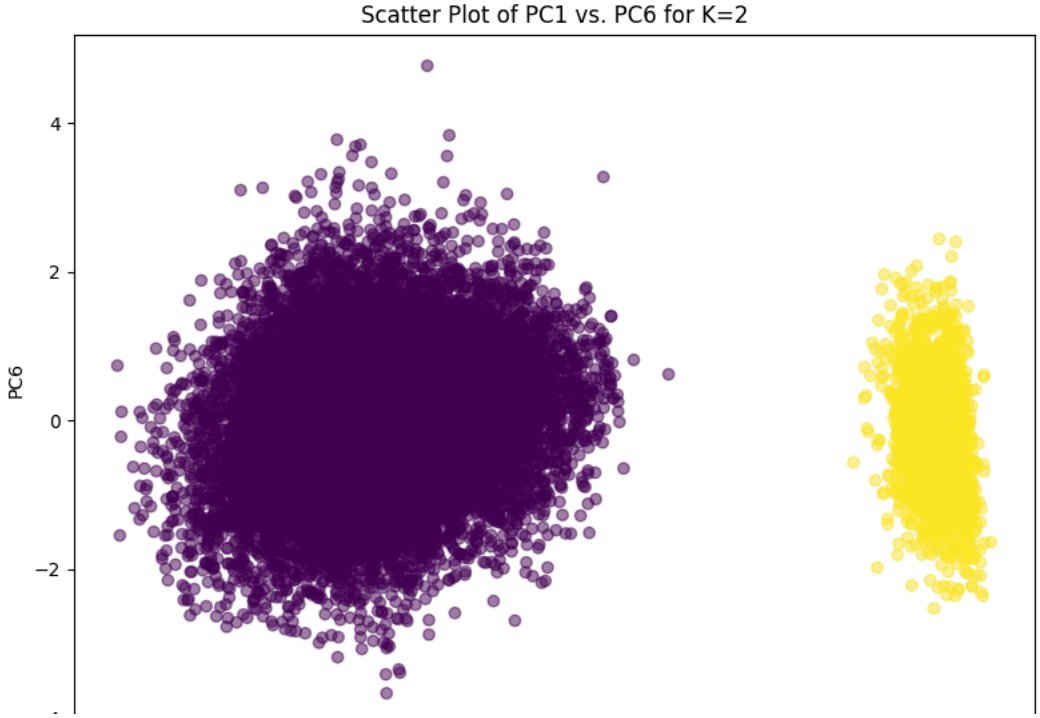

Scatter Plot of PC1 vs. PC2 for K=2
PC2
PC1
6
4
2
0
−2
−4
−6
−10
−5
0
5
10
15

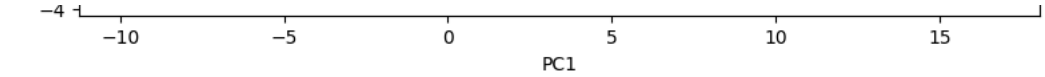

Scatter Plot of PC1 vs. PC3 for K=2
PC3
PC1
8
6
4
2
0
−2
−4
−6
−10
−5
0
5
10
15

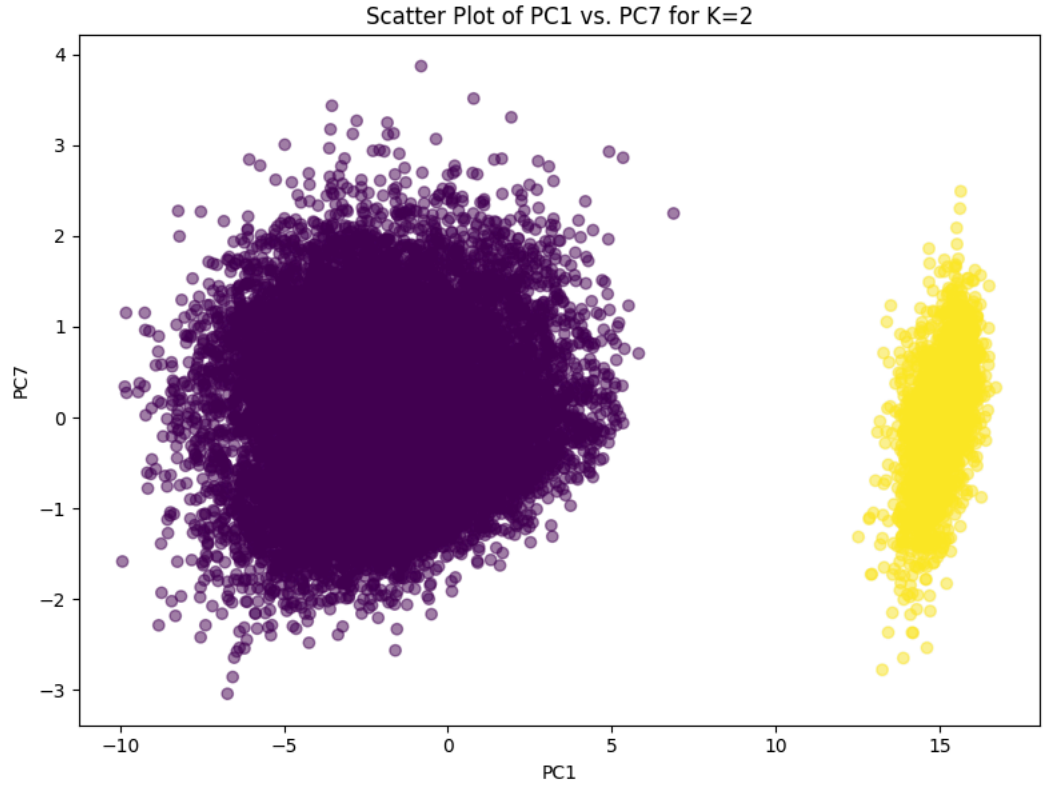

Scatter Plot of PC1 vs. PC4 for K=2
6
4
2

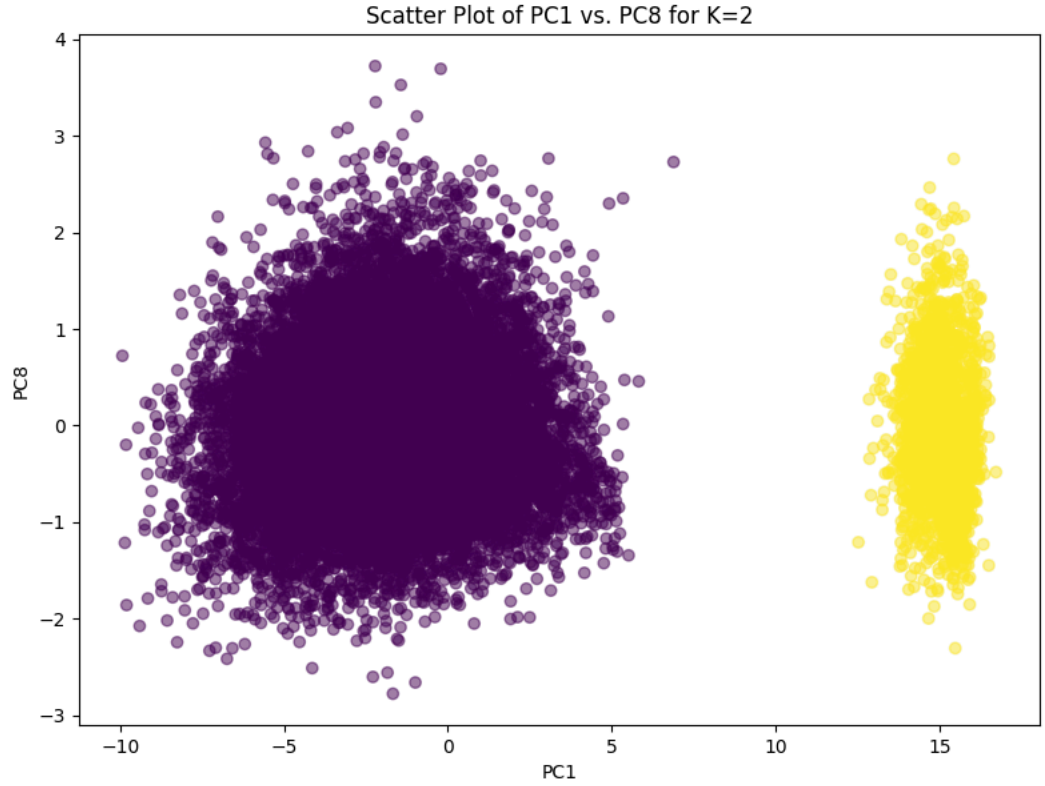

PC4
PC1

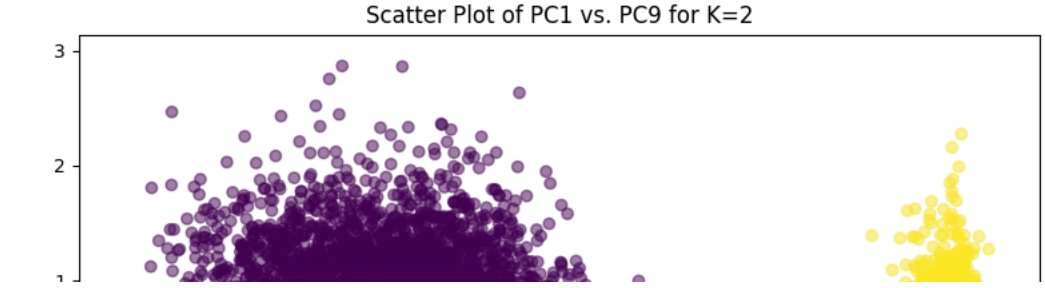

Scatter Plot of PC1 vs. PC5 for K=2
PC5
PC1

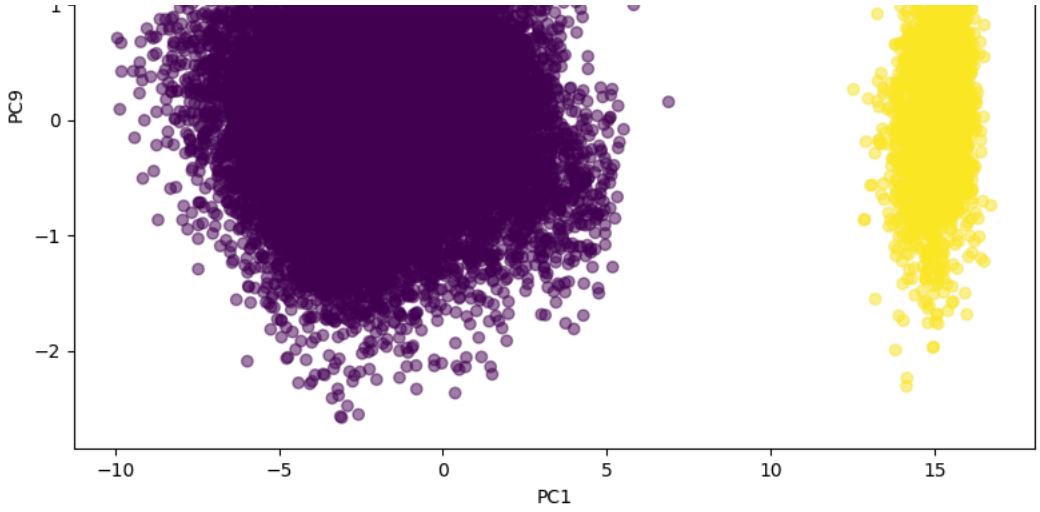

Scatter Plot of PC1 vs. PC6 for K=2
PC6

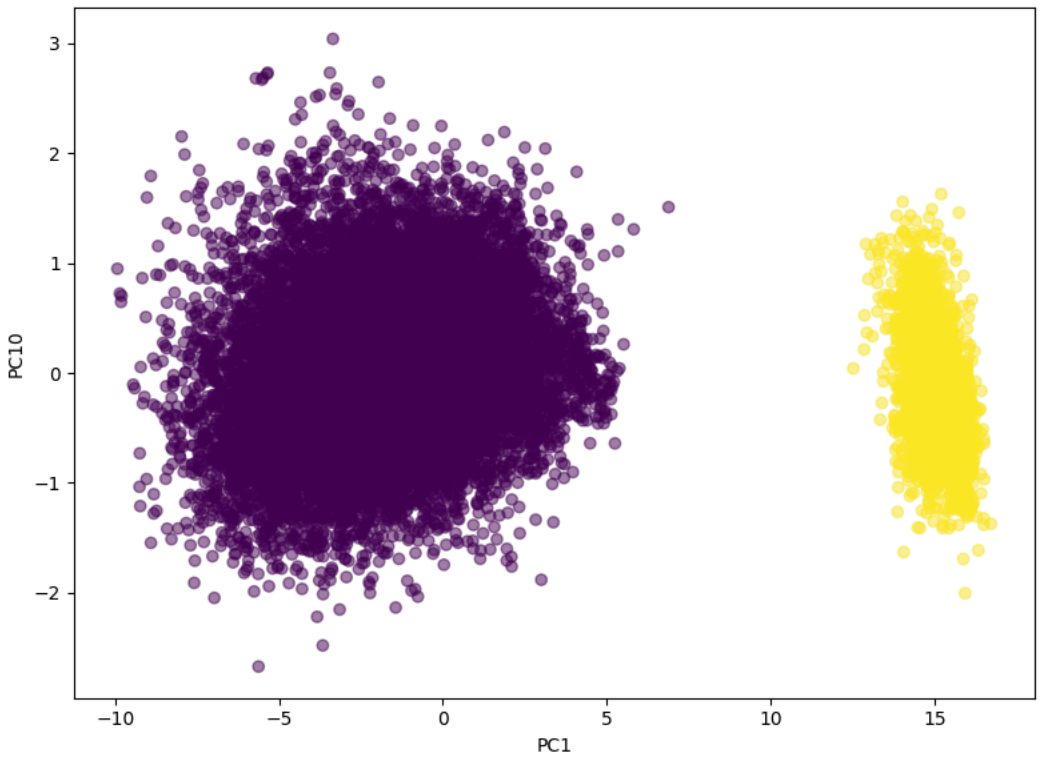

−4
−10
−5
0
5
10
15
PC1

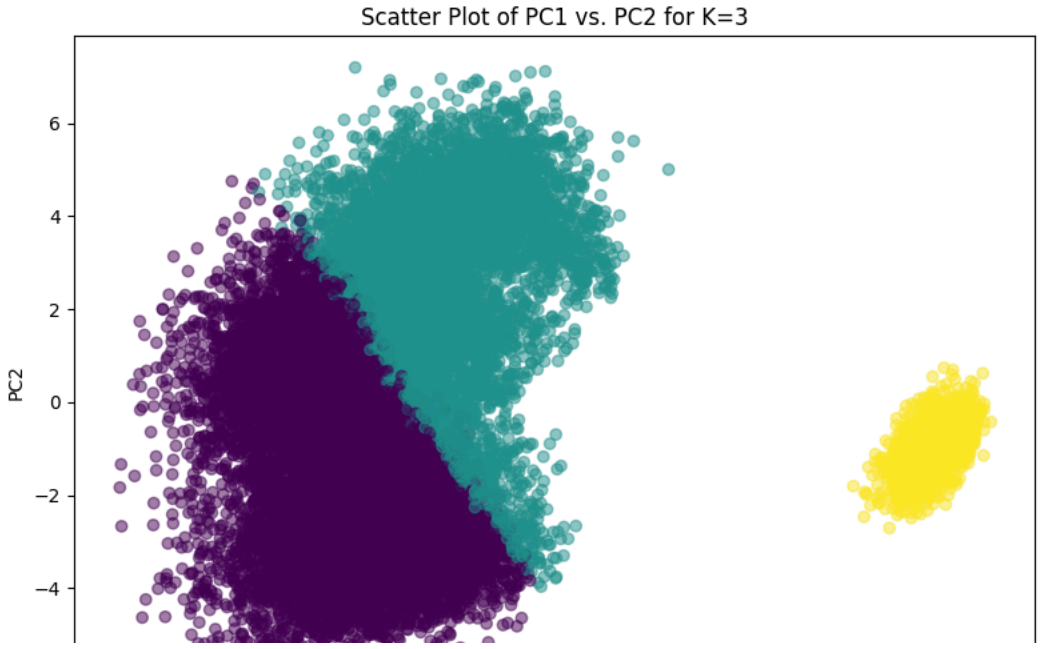

Scatter Plot of PC1 vs. PC7 for K=2
4
3
2
1
0
−1
−2
−3
PC7
−10
−5
0
5
10
15
PC1


Scatter Plot of PC1 vs. PC8 for K=2
4
3
2
1
0
−1
−2
−3
PC8
−10
−5
0
5
10
15
PC1

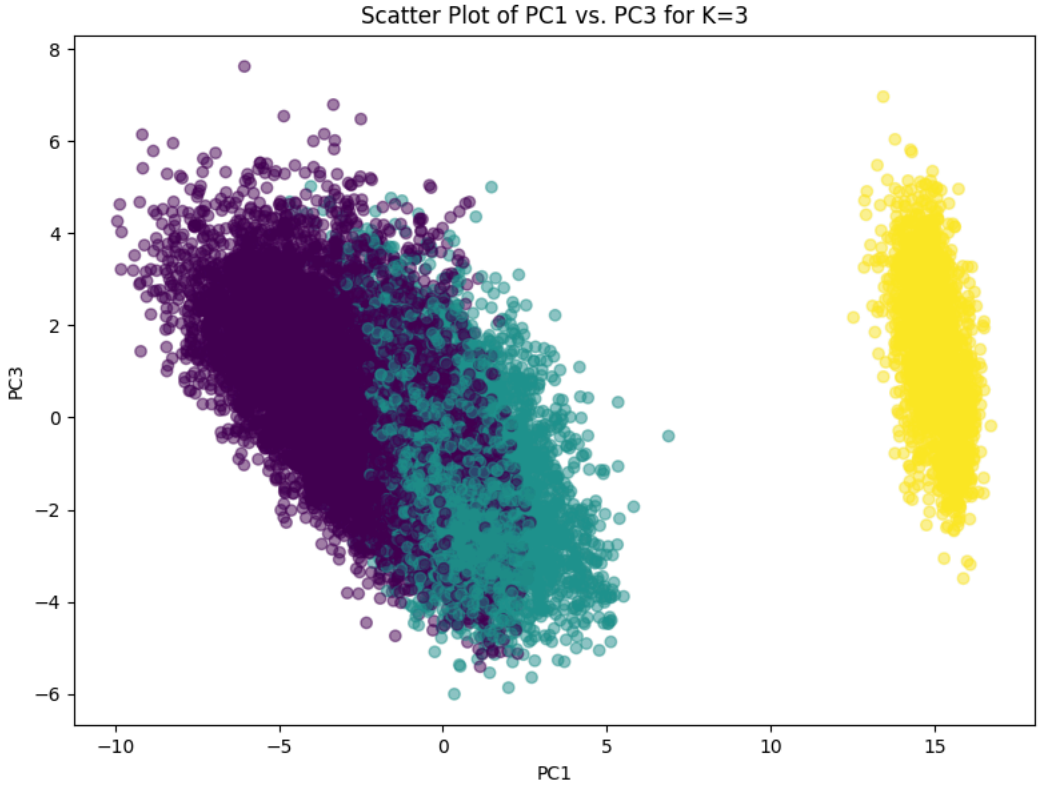

Scatter Plot of PC1 vs. PC9 for K=2
3
2
1

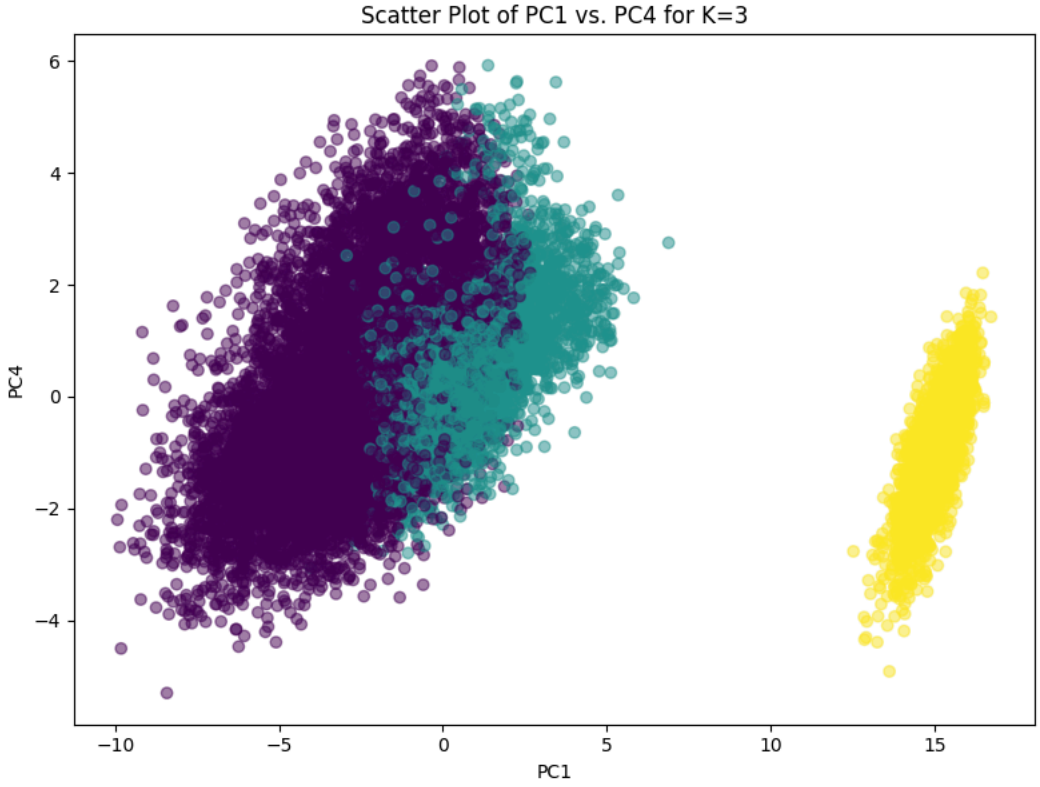

PC9
PC1
−10
−5
0
5
10
15
−2
−1
0
1

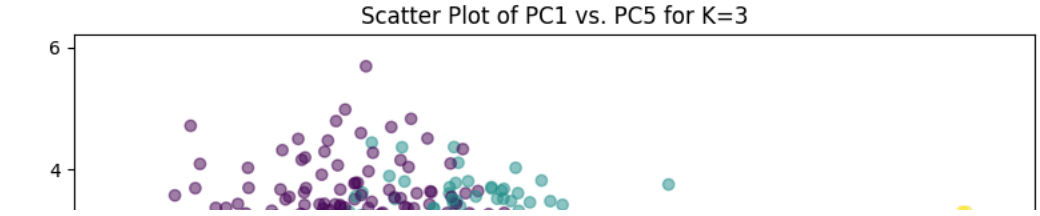

Scatter Plot of PC1 vs. PC10 for K=2
PC10
PC1
−10
−5
0
5
10
15
−2
−1
0
1
2
3

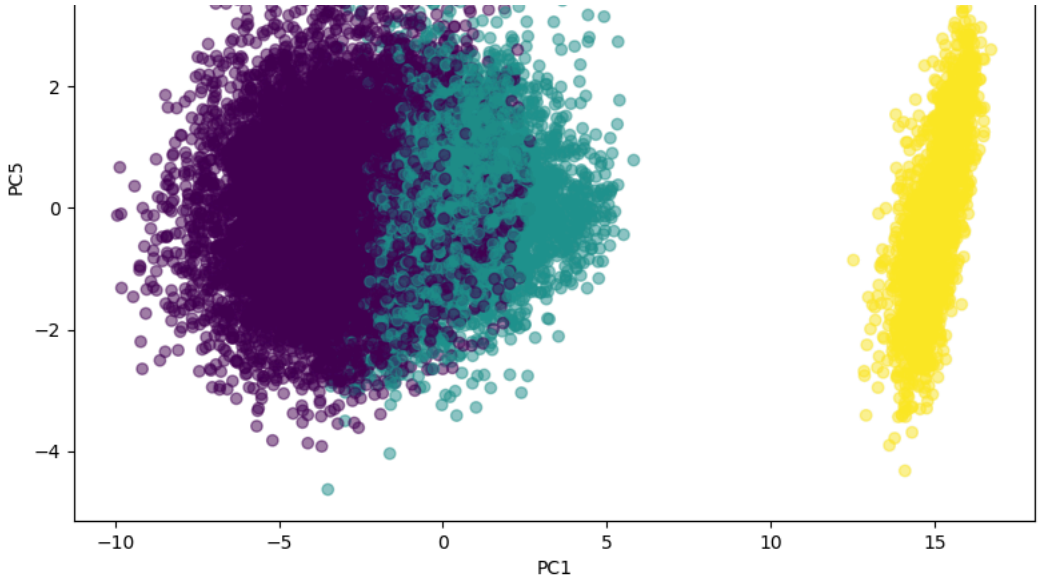

Scatter Plot of PC1 vs. PC2 for K=3
PC2
−4
−2
0
2
4
6

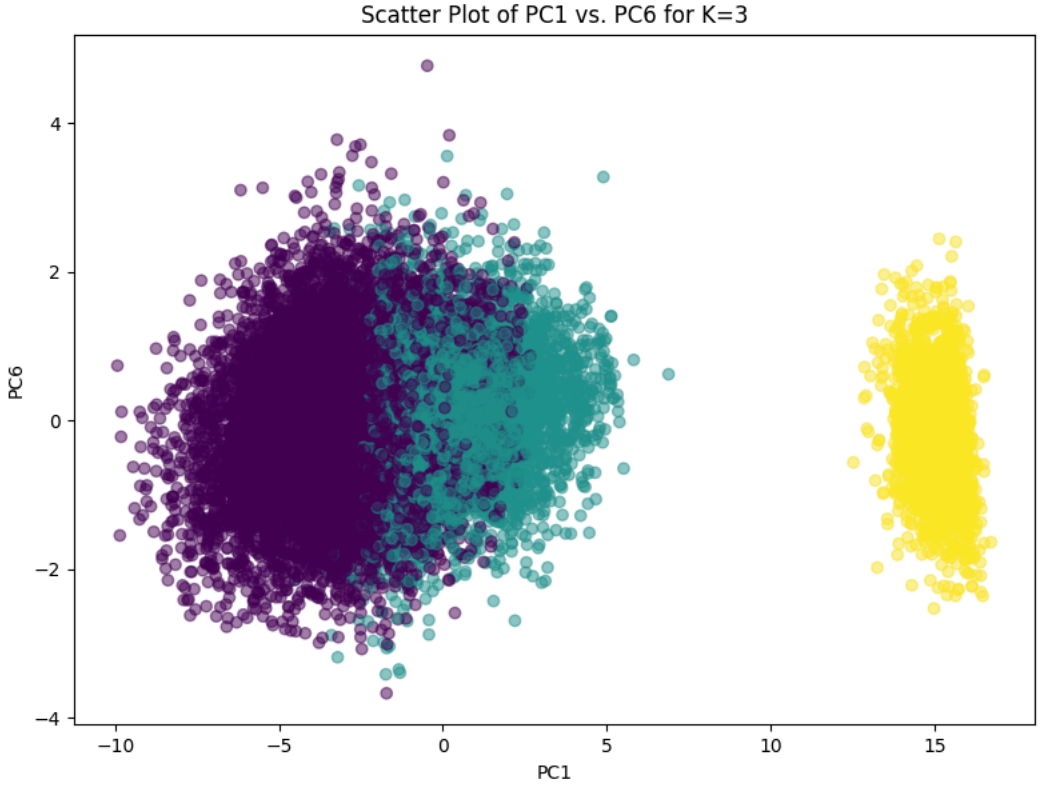

−6
−10
−5
0
5
10
15
PC1

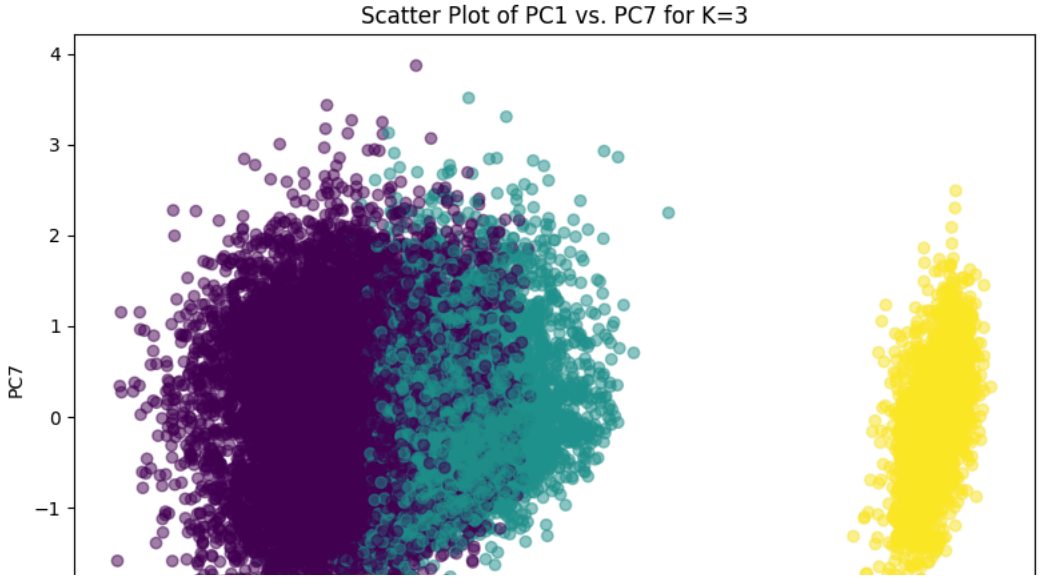

Scatter Plot of PC1 vs. PC3 for K=3
8
6
4
2
0
−2
−4
−6
PC3
−10
−5
0
5
10
15
PC1

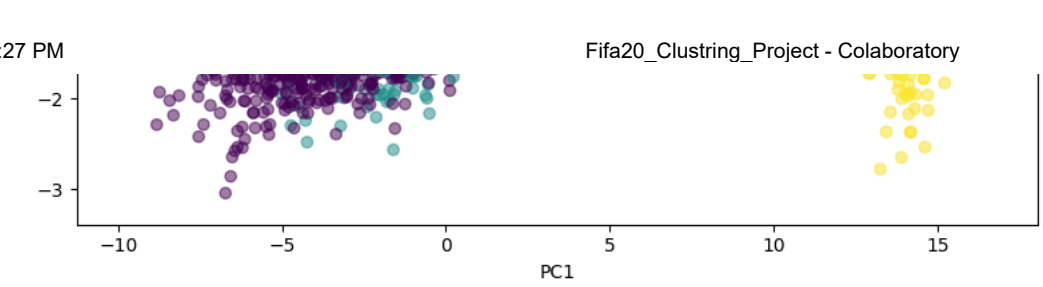

Scatter Plot of PC1 vs. PC4 for K=3
6
4
2
0
−2
−4
PC4
−10
−5
0
5
10
15
PC1

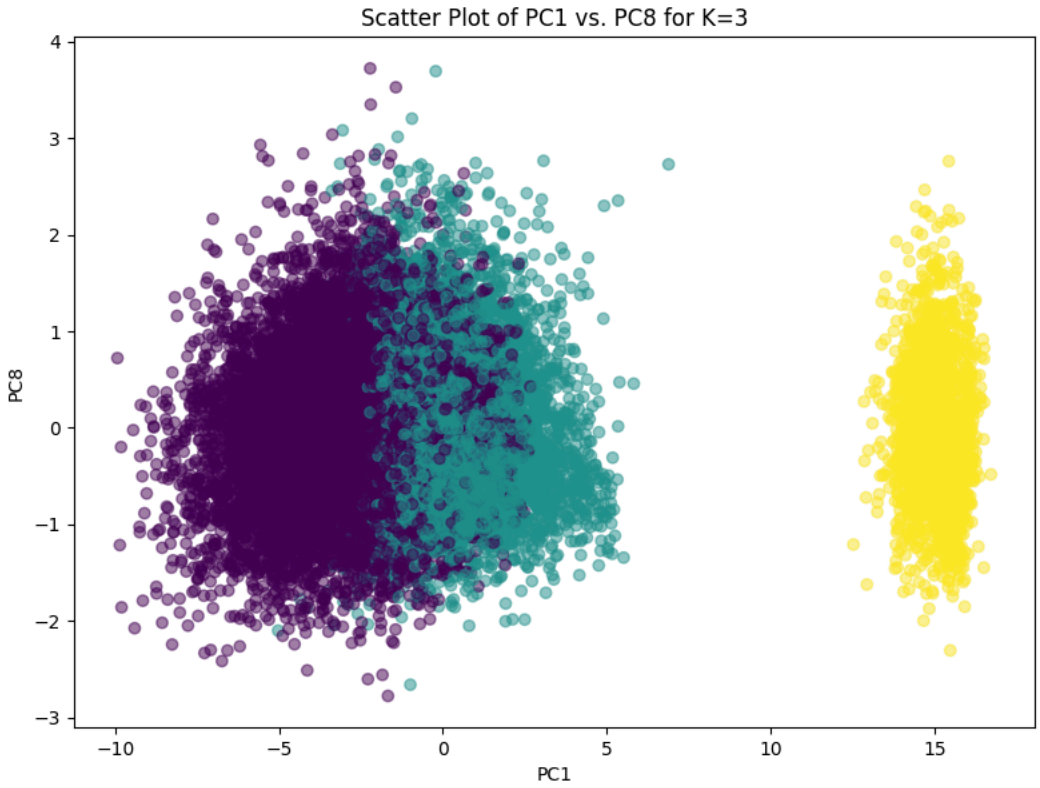

Scatter Plot of PC1 vs. PC5 for K=3
6
4

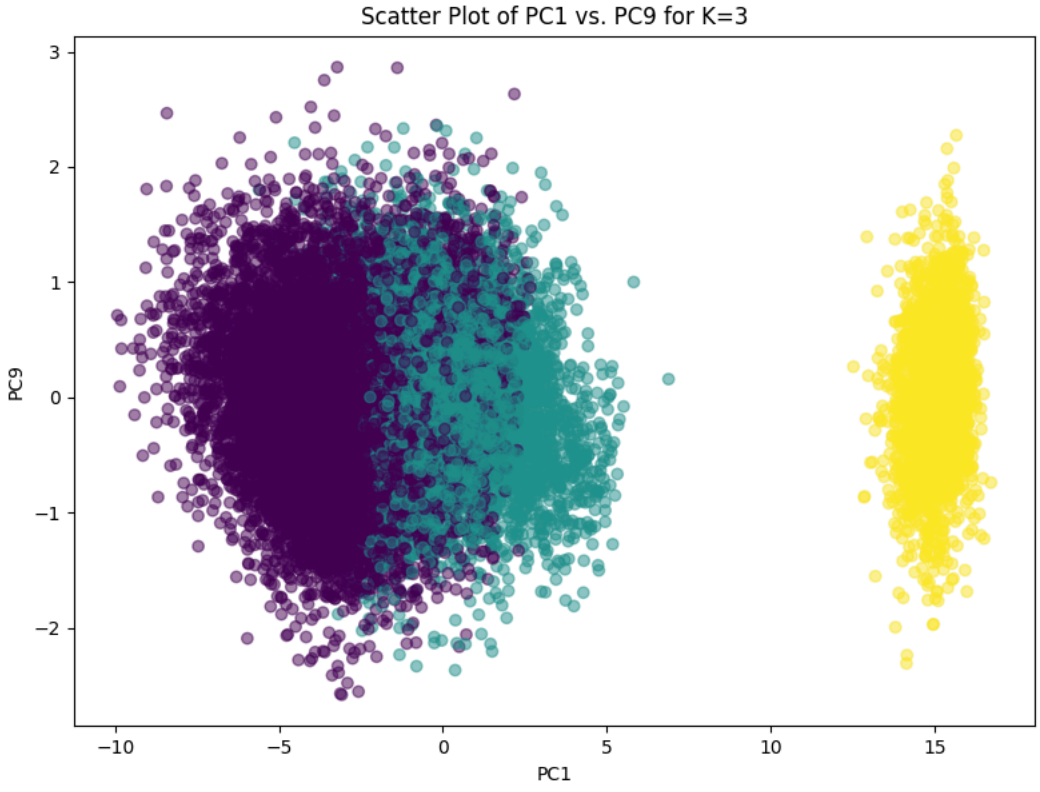

PC5
PC1
2
0
−2
−4
−10
−5
0
5
10
15

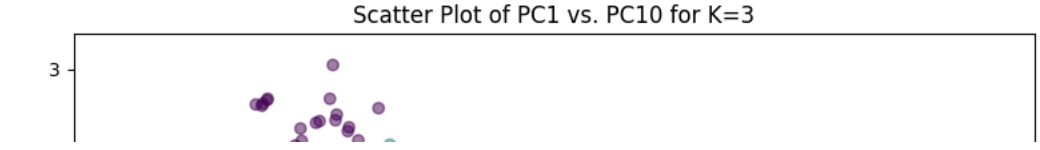

Scatter Plot of PC1 vs. PC6 for K=3
PC6
PC1
4
2
0
−2
−4
−10
−5
0
5
10
15

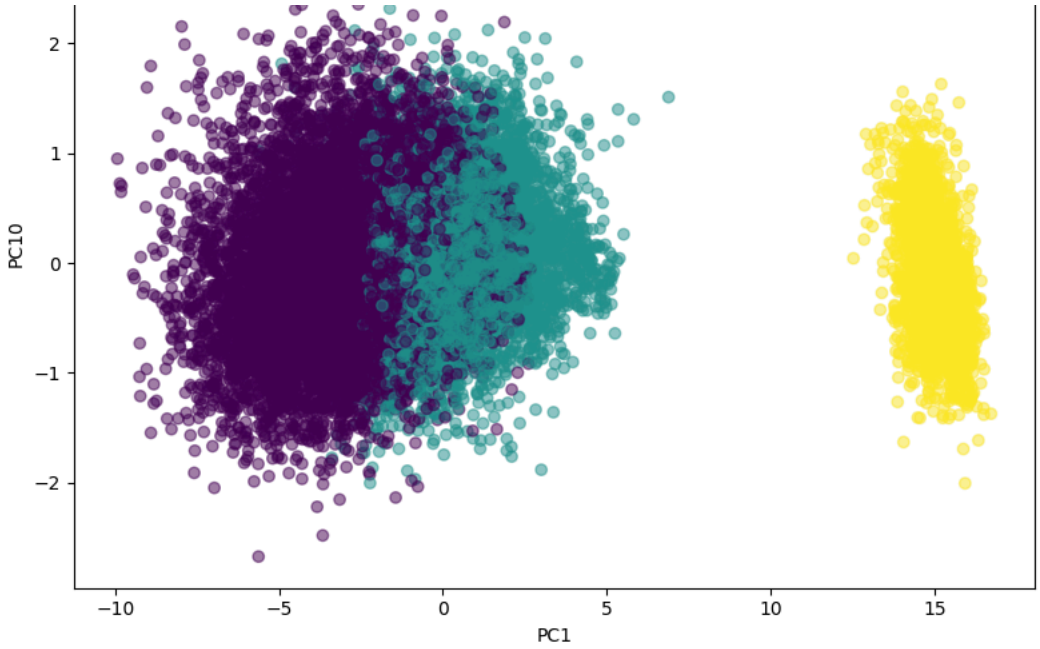

Scatter Plot of PC1 vs. PC7 for K=3
PC7
4
3
2
1
0
−1

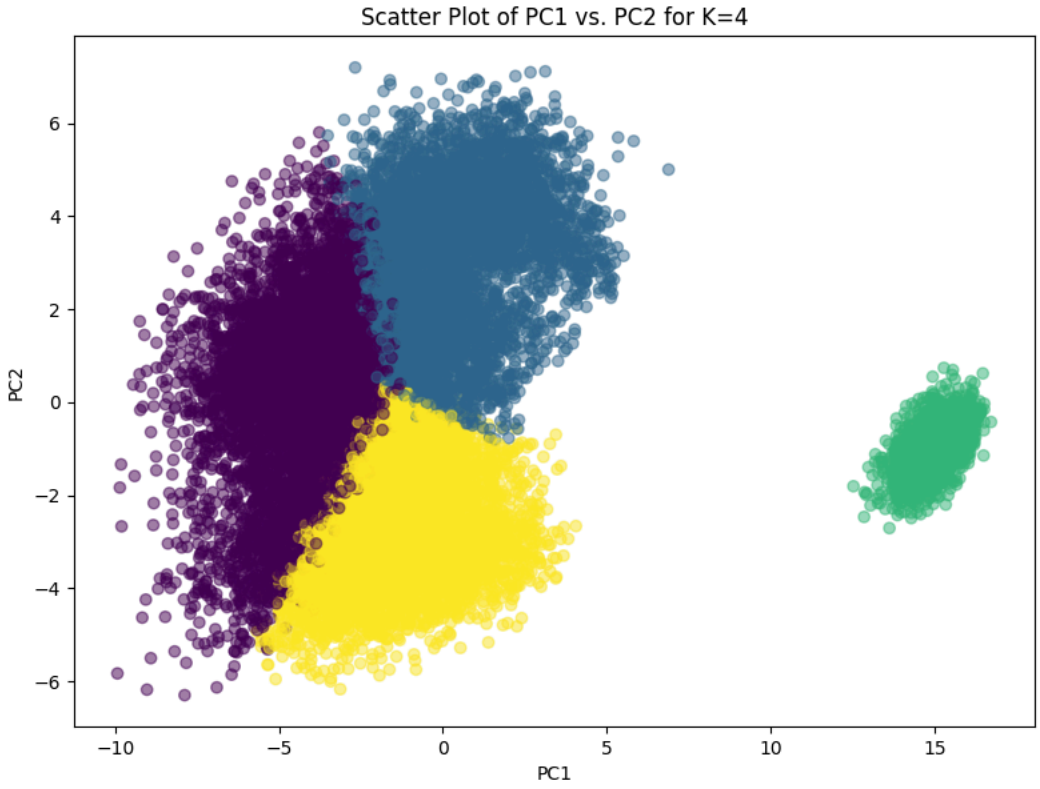

−2
−3
−10
−5
0
5
10
15
PC1

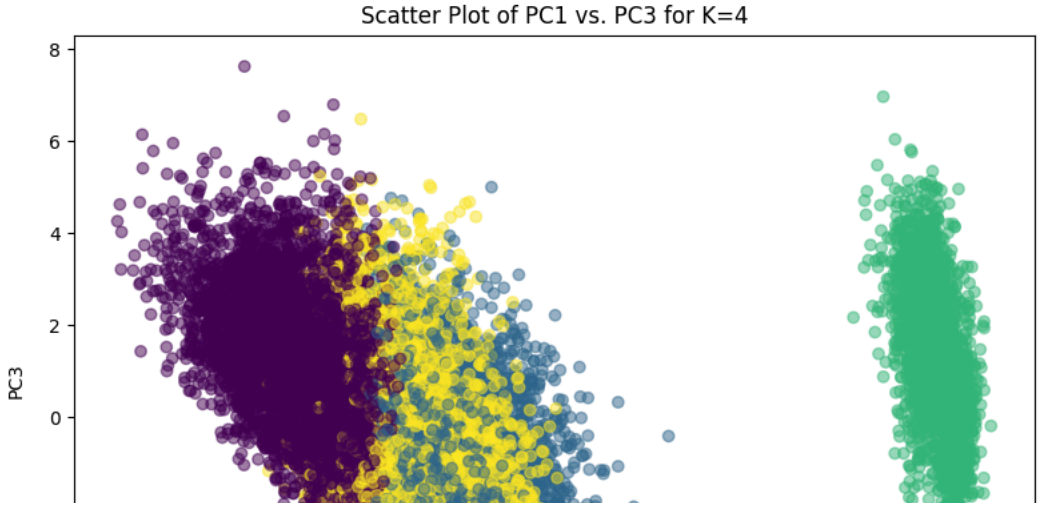

Scatter Plot of PC1 vs. PC8 for K=3
4
3
2
1
0
−1
−2
−3
PC8
−10
−5
0
5
10
15
PC1

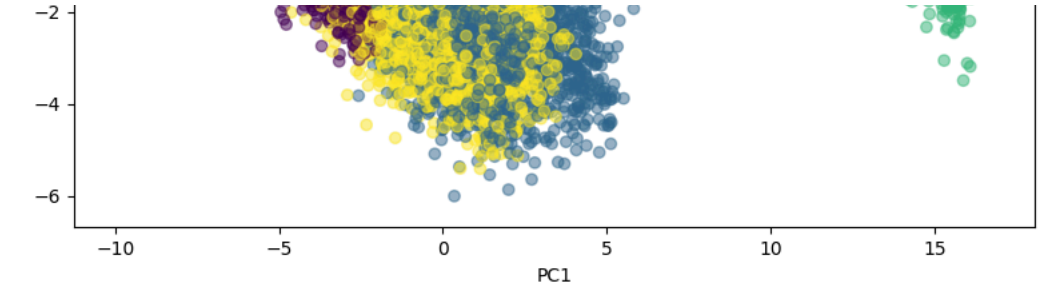

Scatter Plot of PC1 vs. PC9 for K=3
3
2
1
0
−1
−2
PC9
−10
−5
0
5
10
15
PC1

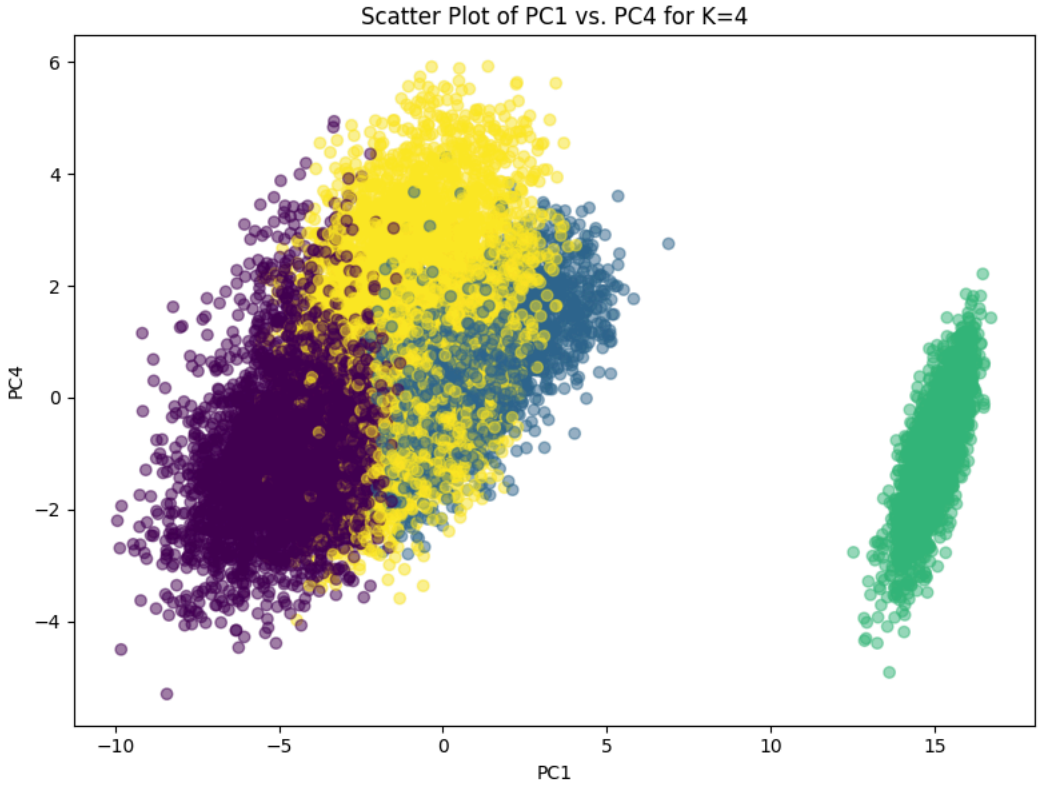

Scatter Plot of PC1 vs. PC10 for K=3
3

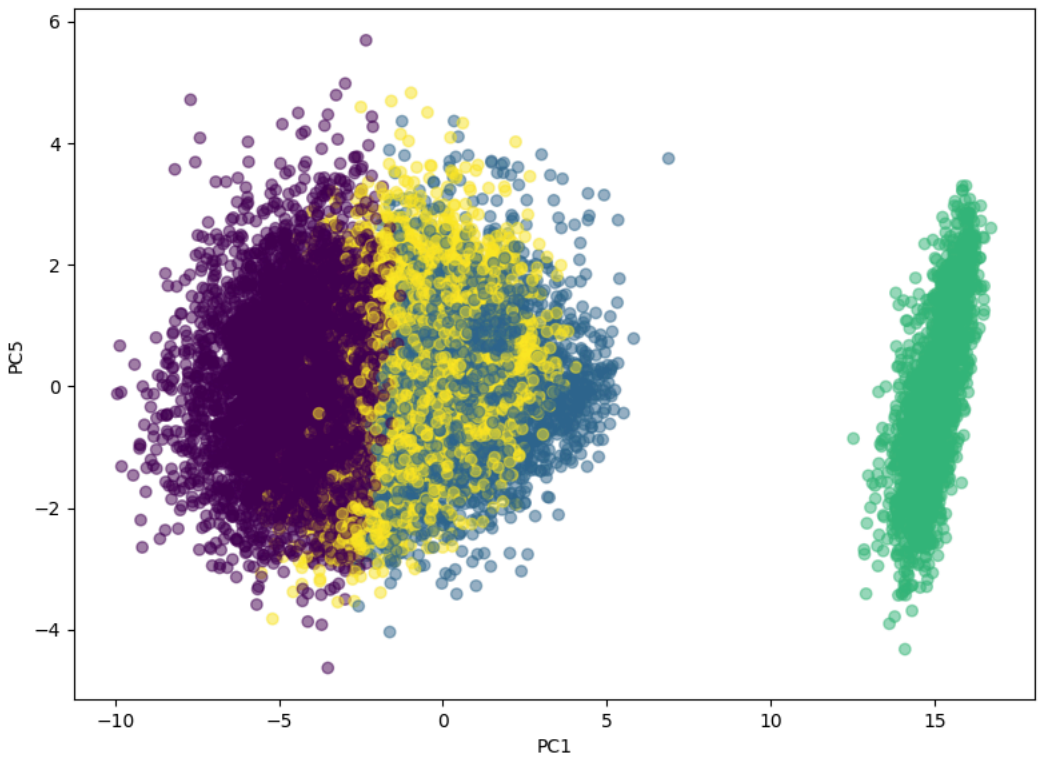

PC10
PC1
2
1
0
−1
−2
−10
−5
0
5
10
15

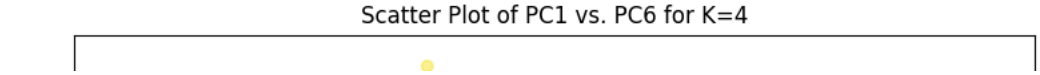

Scatter Plot of PC1 vs. PC2 for K=4
PC2
PC1
6
4
2
0
−2
−4
−6
−10
−5
0
5
10
15

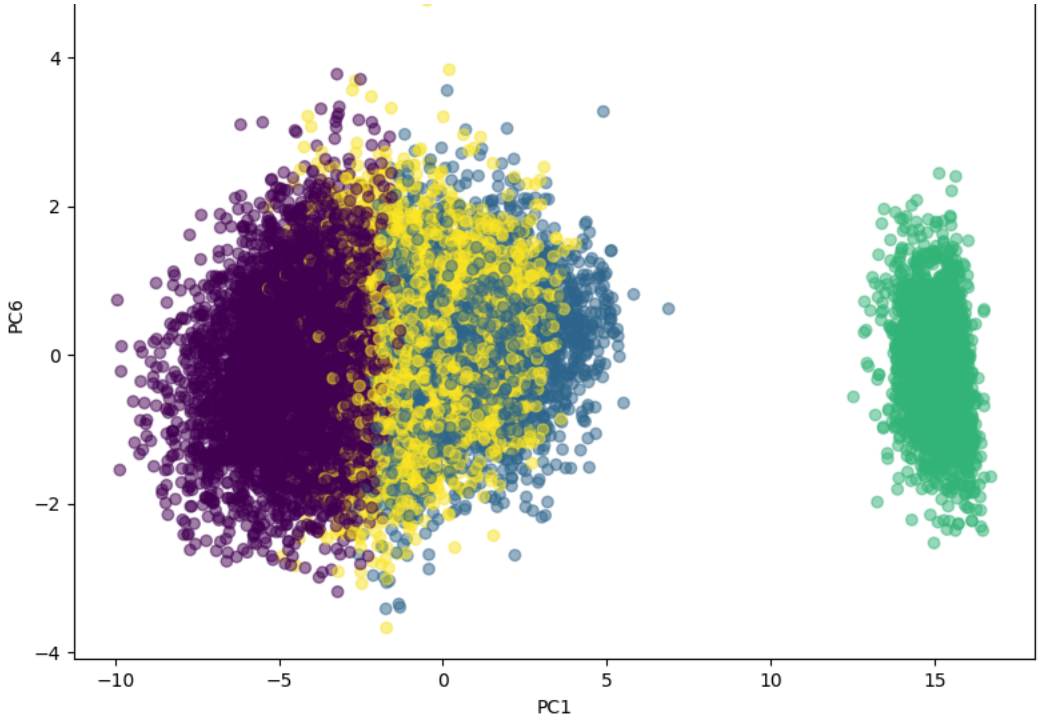

Scatter Plot of PC1 vs. PC3 for K=4
PC3
8
6
4
2
0


−2
−4
−6
−10
−5
0
5
10
15
PC1

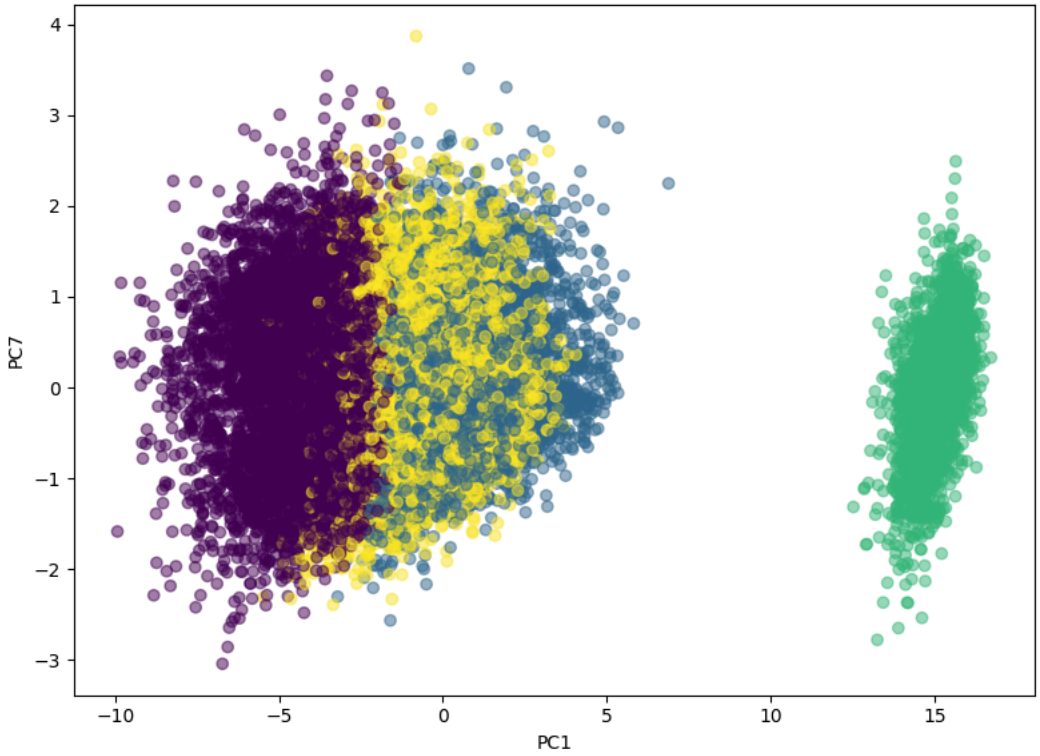

Scatter Plot of PC1 vs. PC4 for K=4
6
4
2
0
−2
−4
PC4
−10
−5
0
5
10
15
PC1


Scatter Plot of PC1 vs. PC5 for K=4
6
4
2
0
−2
−4
PC5
−10
−5
0
5
10
15
PC1

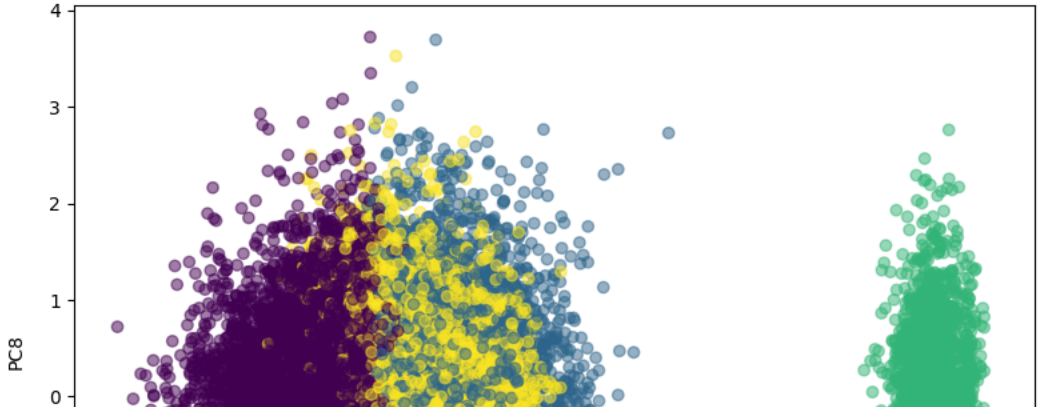

Scatter Plot of PC1 vs. PC6 for K=4

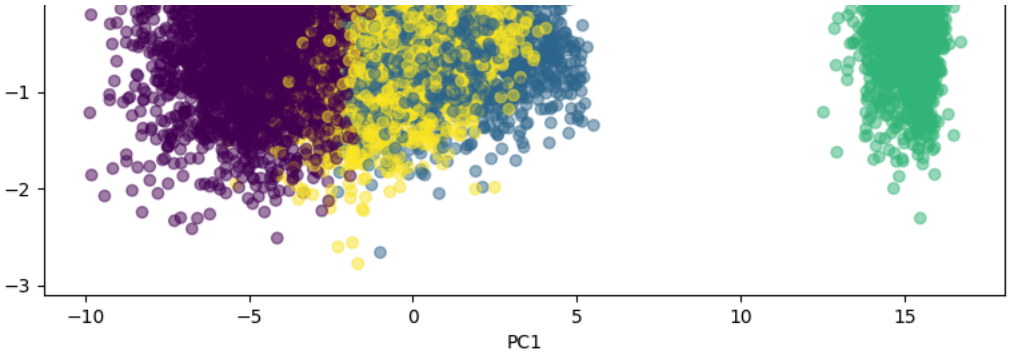


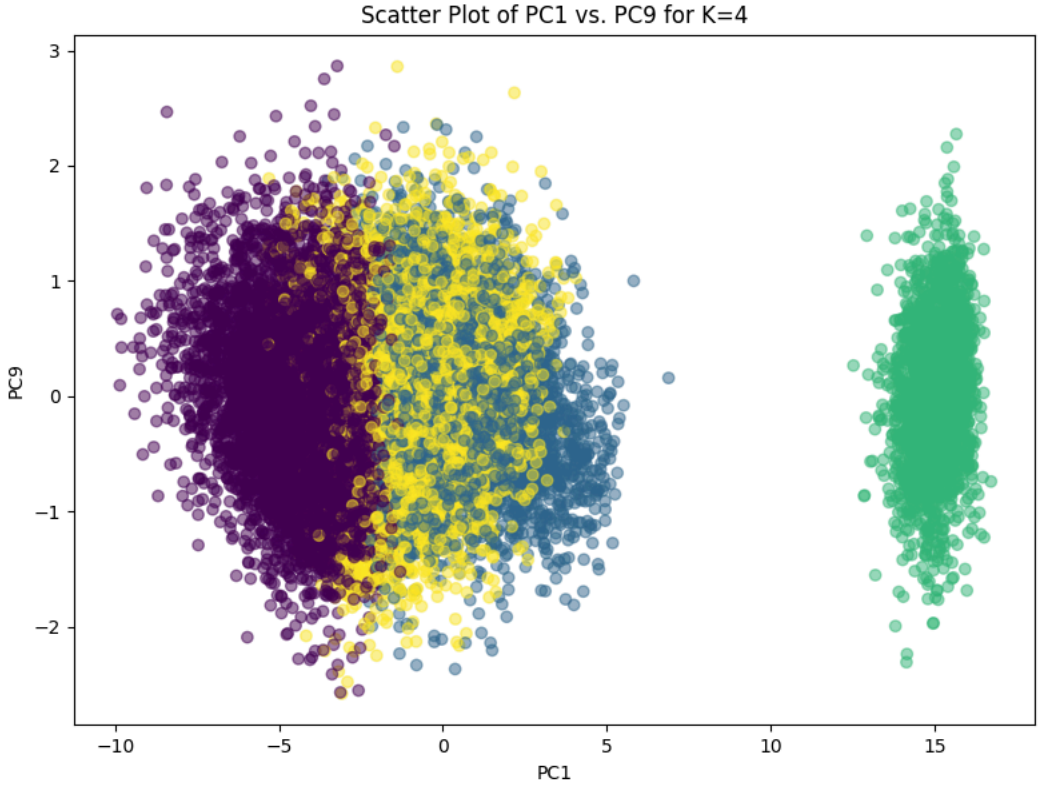
Scatter Plot of PC1 vs. PC7 for K=4

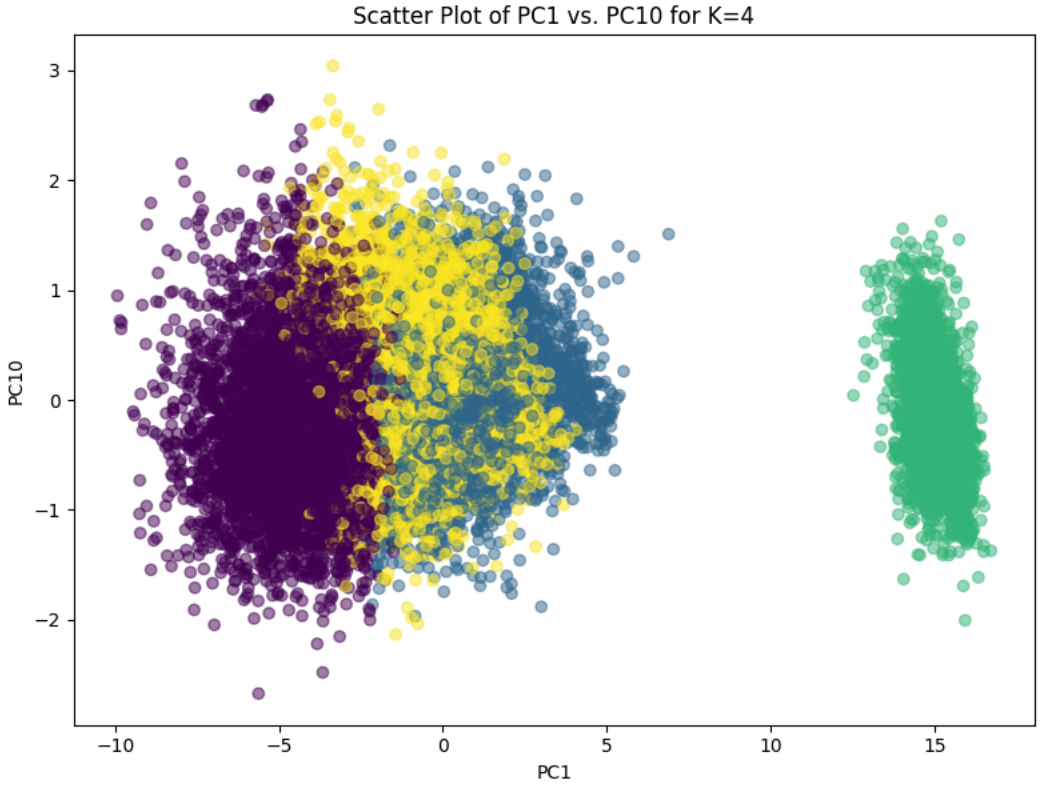


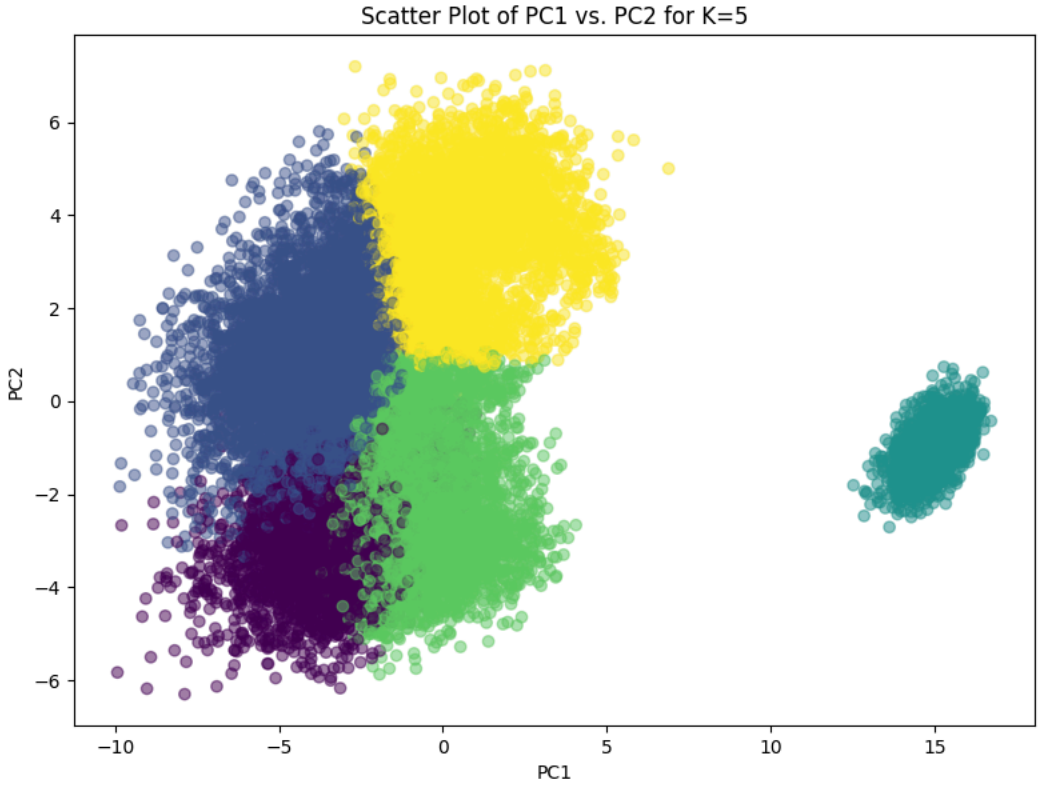
Scatter Plot of PC1 vs. PC8 for K=4

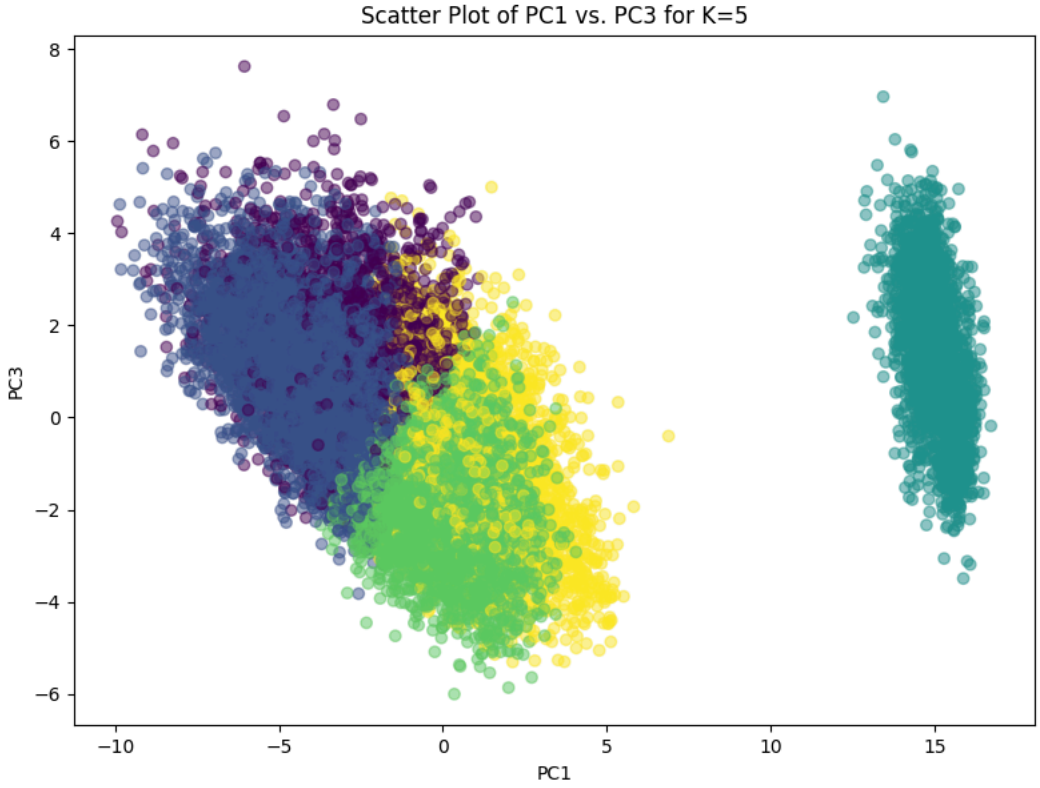


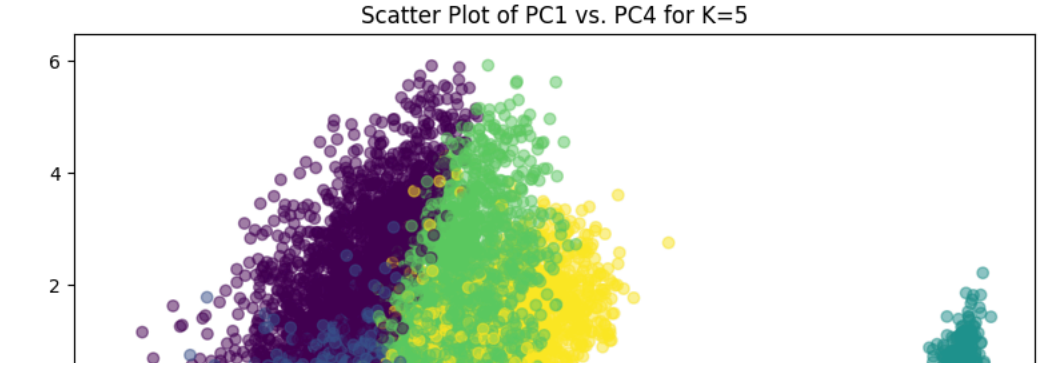

−1
−2
−3
−10
−5
0
5
10
15
PC1

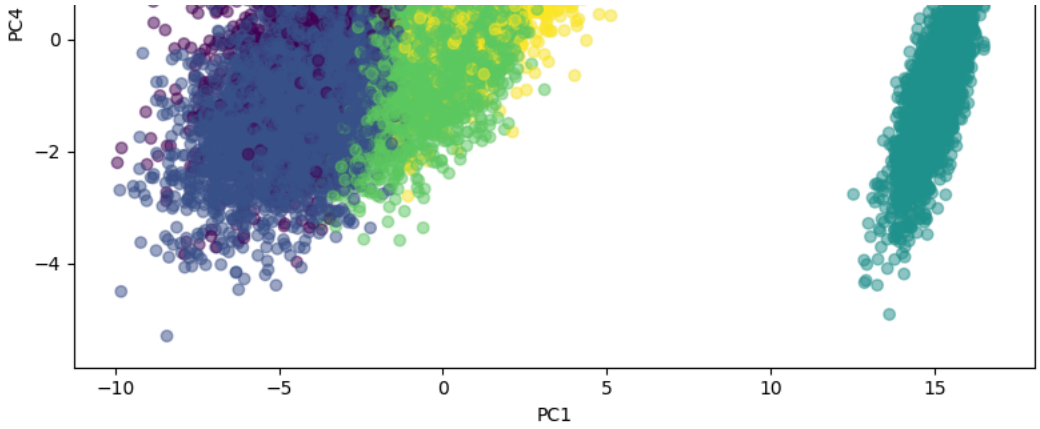

Scatter Plot of PC1 vs. PC9 for K=4
3
2
1
0
−1
−2
PC9
−10
−5
0
5
10
15
PC1

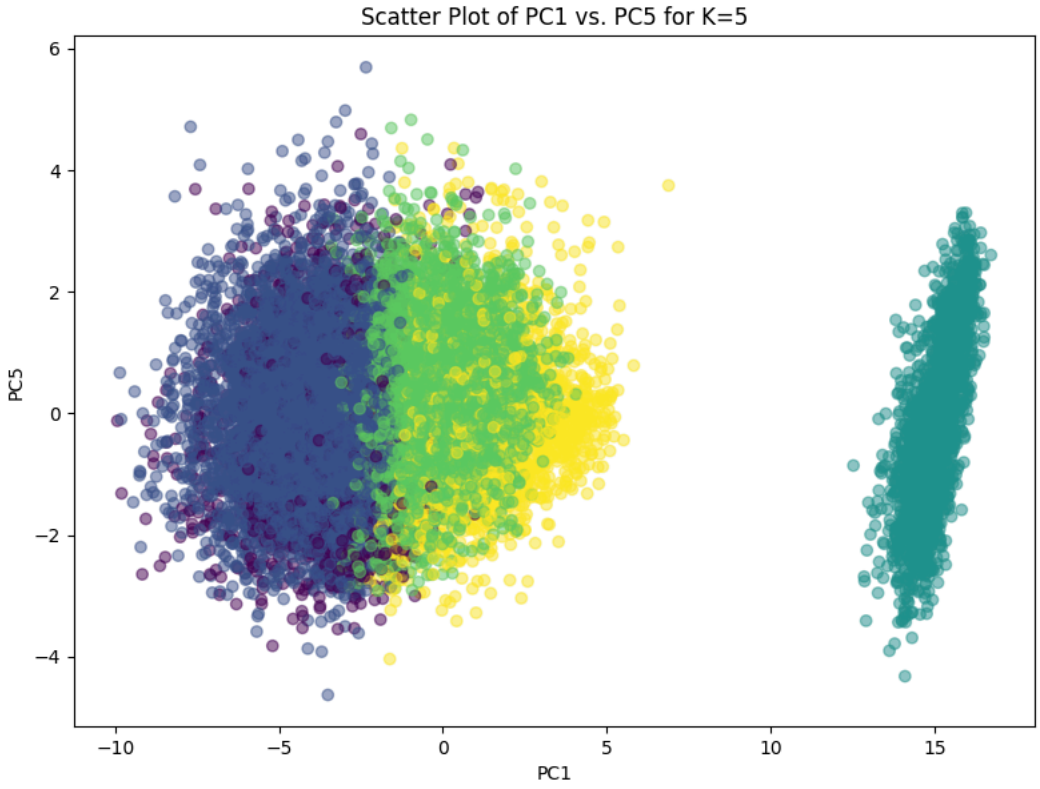

Scatter Plot of PC1 vs. PC10 for K=4
3
2
1
0
−1
−2
PC10
−10
−5
0
5
10
15
PC1

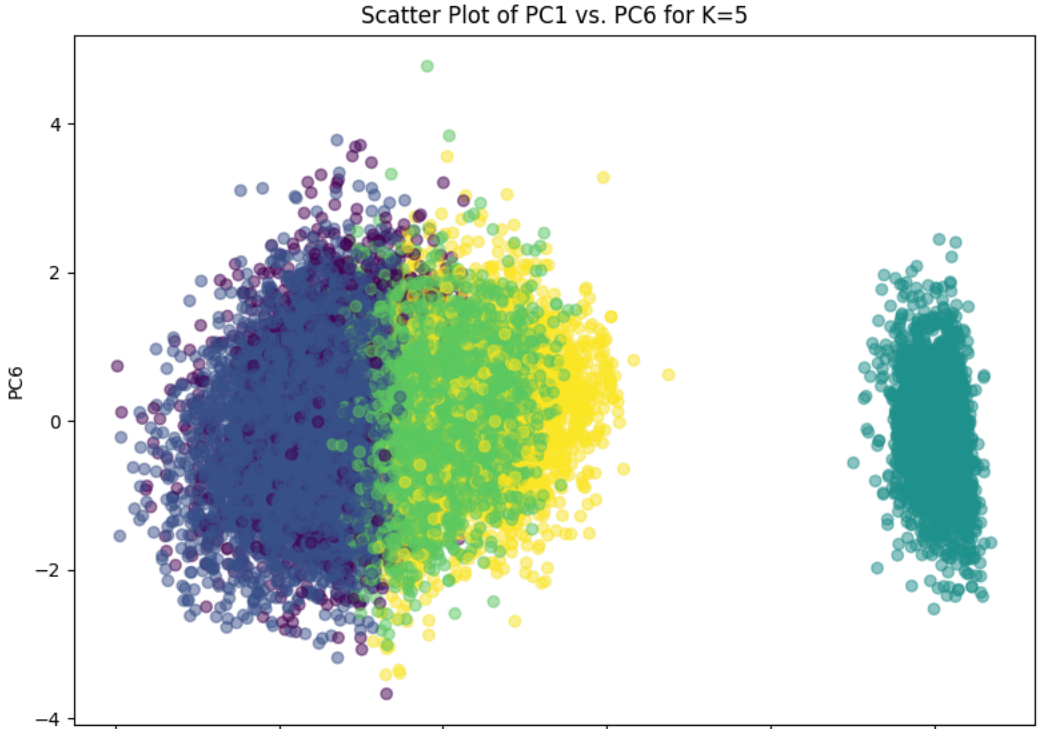

Scatter Plot of PC1 vs. PC2 for K=5
PC2
PC1
6
4
2
0
−2
−4
−6
−10
−5
0
5
10
15


Scatter Plot of PC1 vs. PC3 for K=5
PC3
PC1
8
6
4
2
0
−2
−4
−6
−10
−5
0
5
10
15

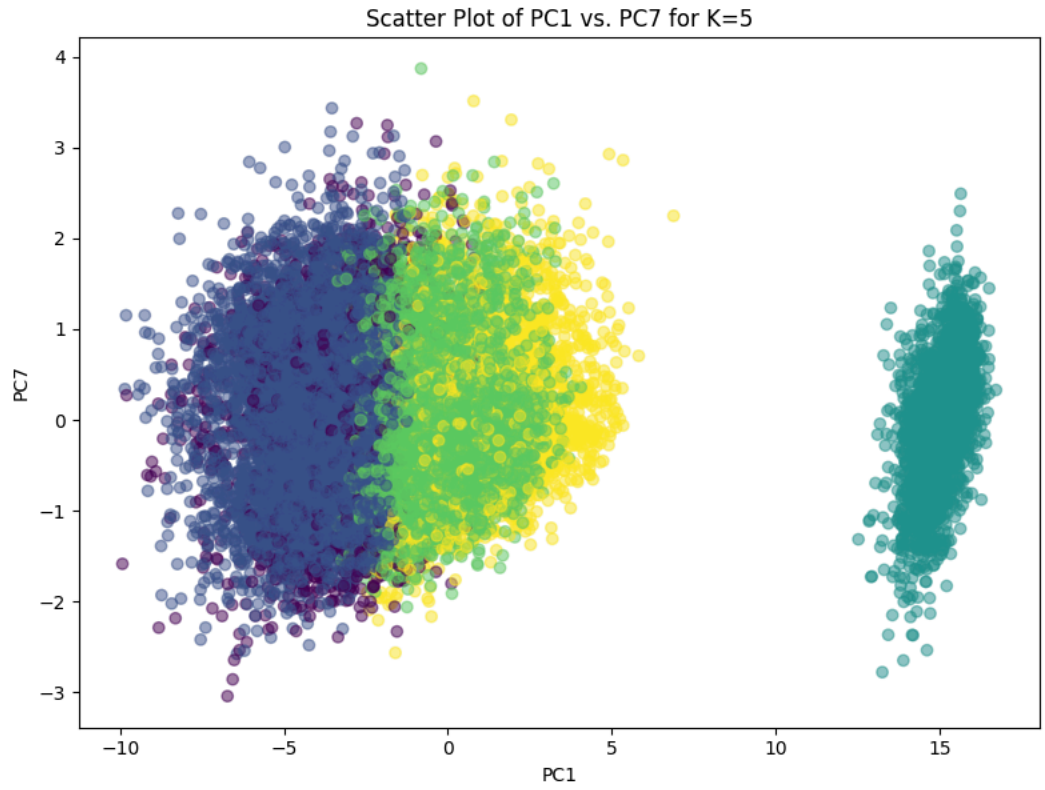

Scatter Plot of PC1 vs. PC4 for K=5
6
4
2

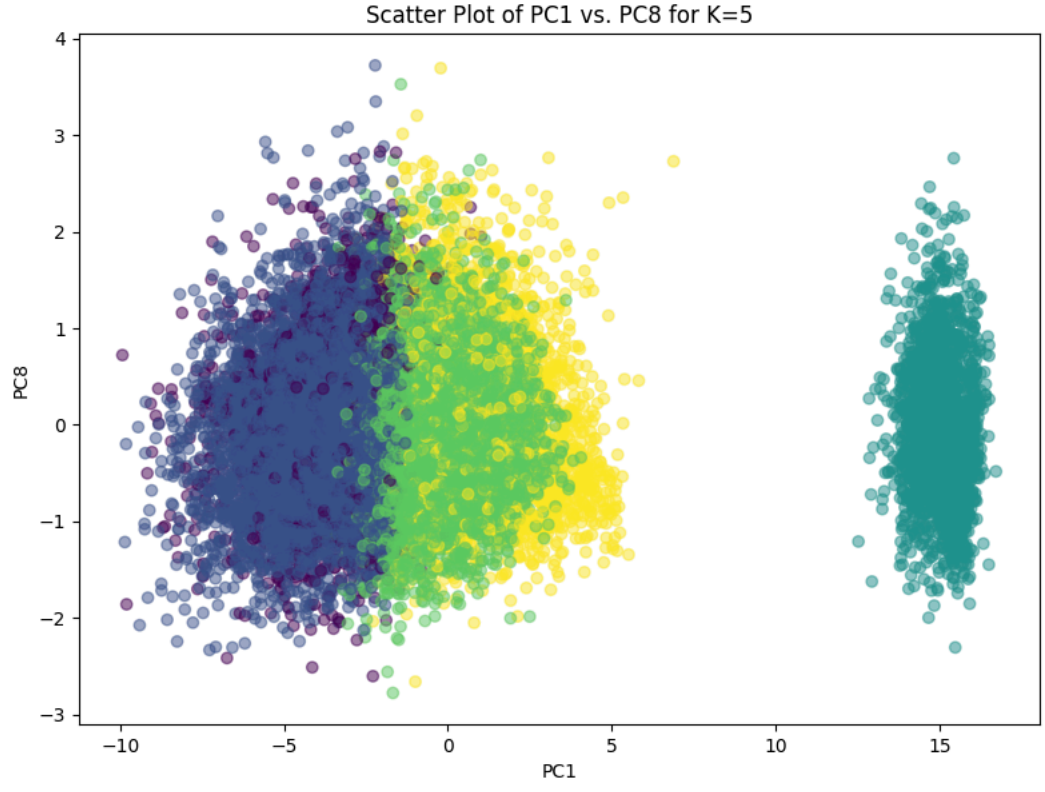

PC4
PC1

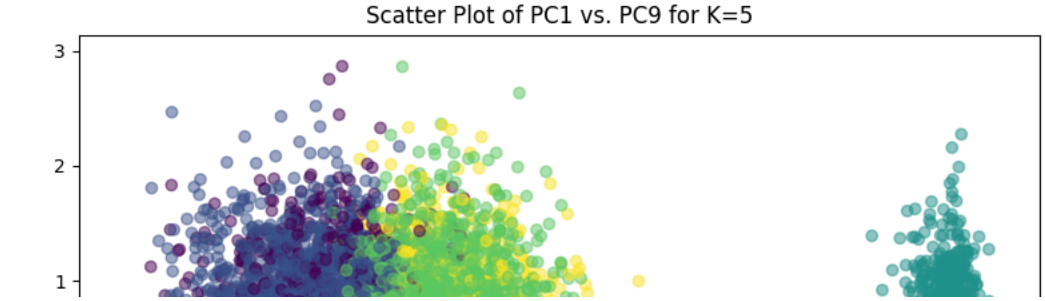

Scatter Plot of PC1 vs. PC5 for K=5
PC5
PC1

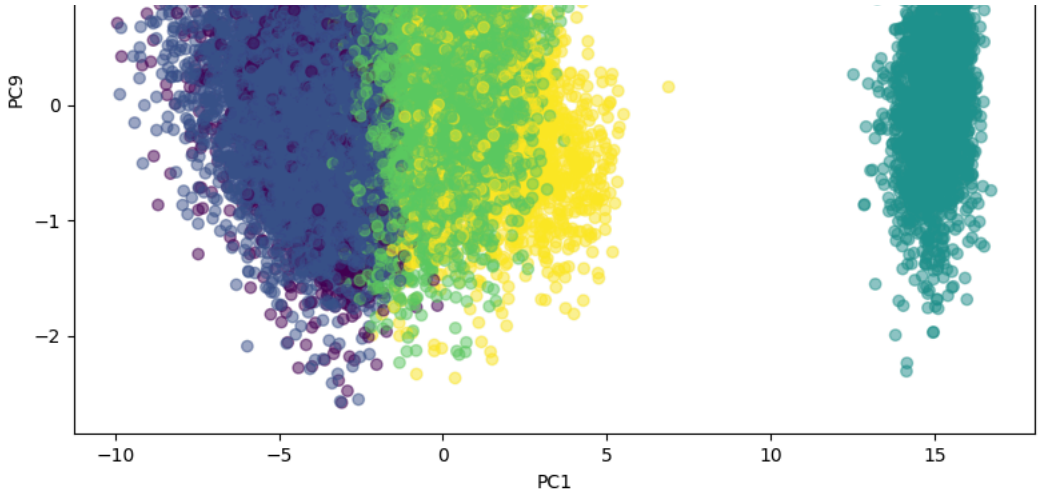

Scatter Plot of PC1 vs. PC6 for K=5
PC6

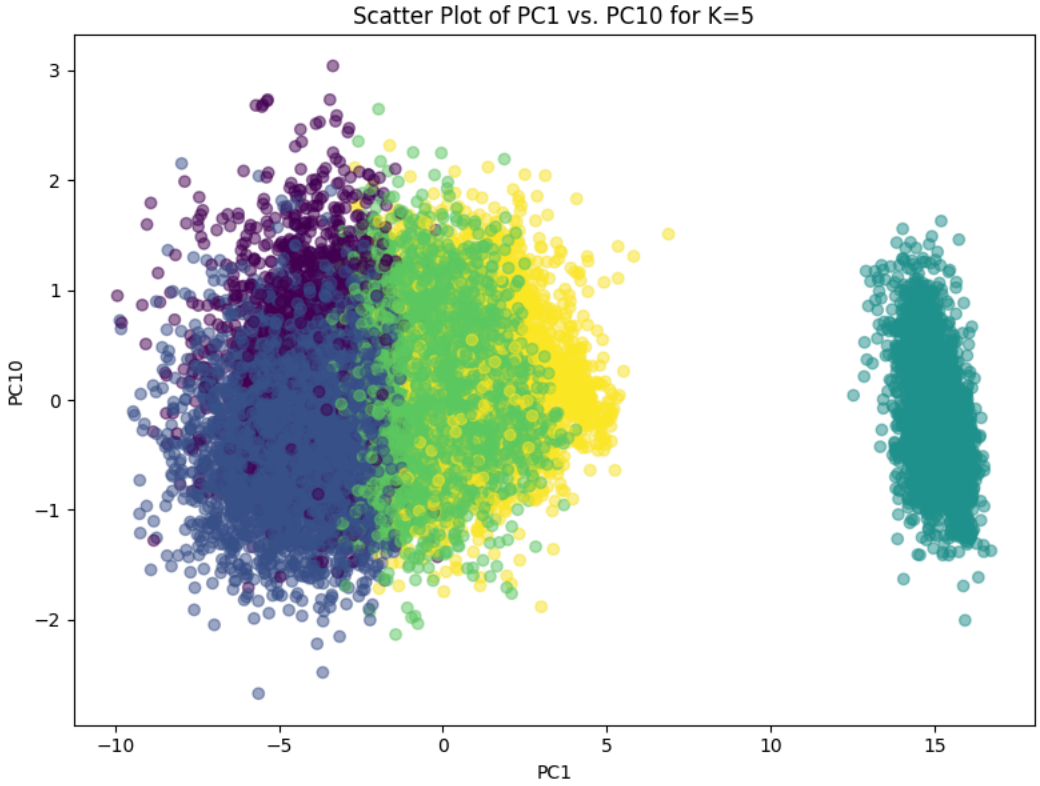

−10
−5
0
5
10
15
PC1


Scatter Plot of PC1 vs. PC7 for K=5
PC7
PC1
4
3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15


Scatter Plot of PC1 vs. PC8 for K=5
PC8
PC1
4
3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15

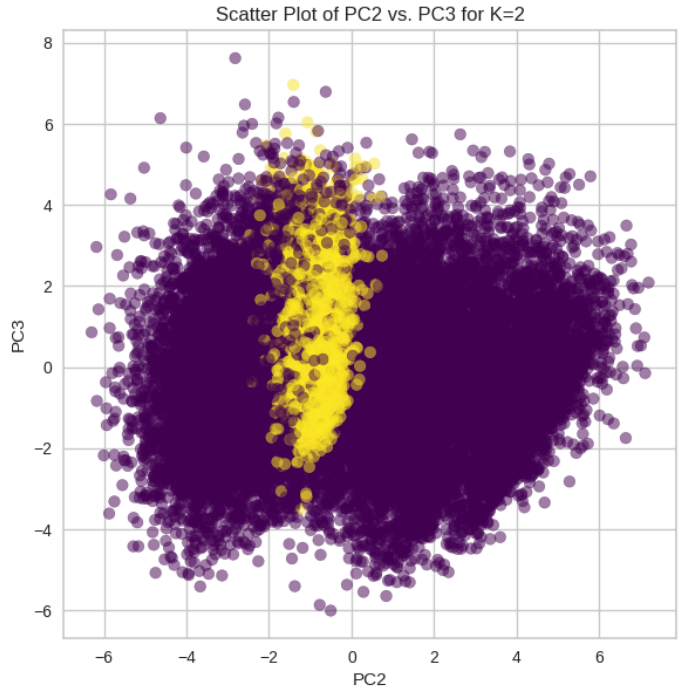

Scatter Plot of PC1 vs. PC9 for K=5
3
2
1

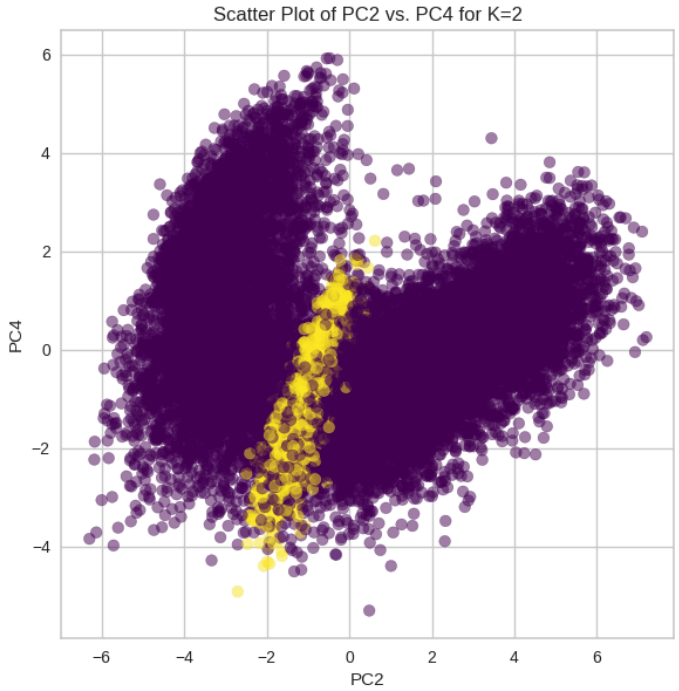


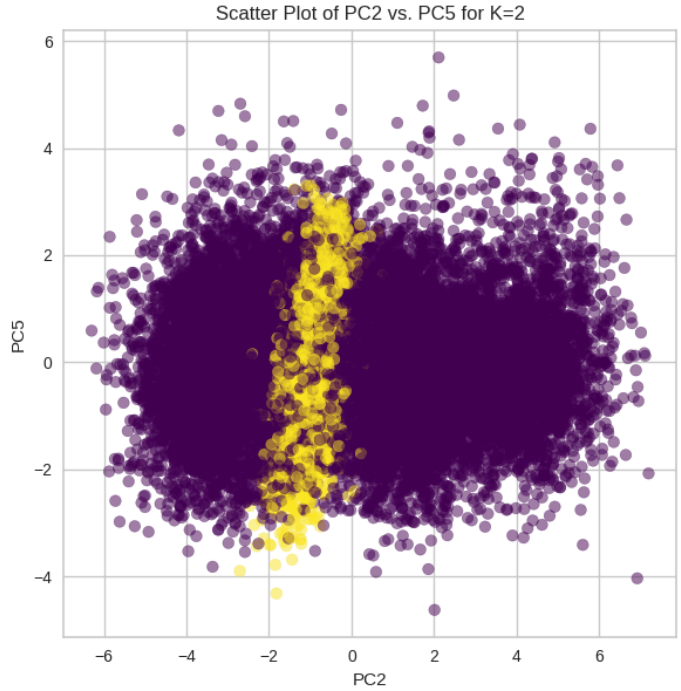


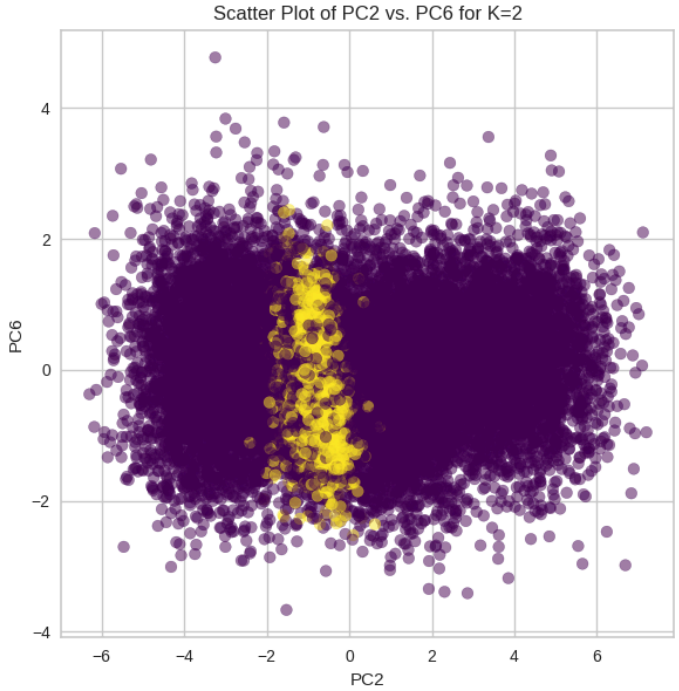
PCA 2 VS All PCA

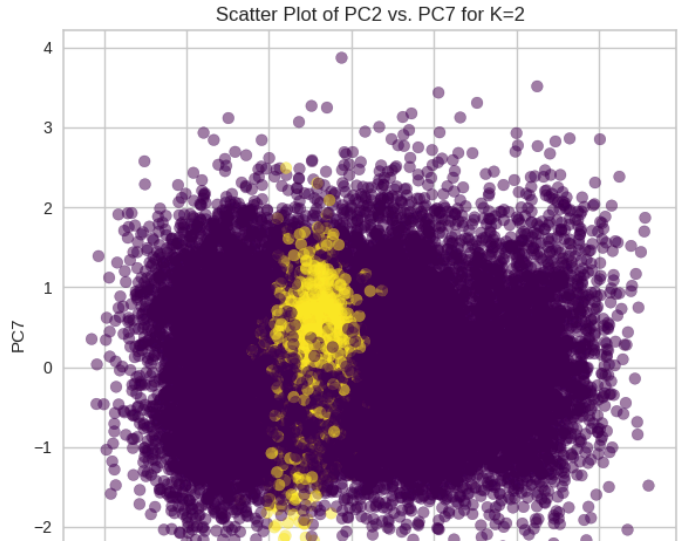
```
import matplotlib.pyplot as plt

def scatter_pca2_vs_all(pca_df, labels, k):
    """
    Create scatter plots of PCA2 vs. all other principal components for each K value.
    """
    num_components = pca_df.shape[1]  # Number of principal components
    num_rows = (num_components - 2) // 2 + 1  # Number of rows required for subplots
    fig, axes = plt.subplots(num_rows, 2, figsize=(12, 6 * num_rows))

    for i in range(2, num_components):
        row_idx = (i - 2) // 2
        col_idx = (i - 2) % 2
        ax = axes[row_idx, col_idx] if num_rows > 1 else axes[col_idx]
        ax.scatter(pca_df['PC2'], pca_df[f'PC{i + 1}'], c=labels, cmap='viridis', alpha=0.5)
        ax.set_xlabel('PC2')
        ax.set_ylabel(f'PC{i + 1}')
        ax.set_title(f'Scatter Plot of PC2 vs. PC{i + 1} for K={k}')

    # Hide empty subplots if num_components is odd
    if num_components % 2 != 0:
        axes[-1, -1].axis('off')

    plt.tight_layout()
    plt.show()

# Assuming 'fifa20_pca_df', 'kmea_clu_pca_clu2_labels', 'kmea_clu_pca_clu3_labels', 'kmea_clu_pca_clu4_labels', 'kmea_clu_pca_clu5_labels' a

# For 2 clusters
scatter_pca2_vs_all(fifa20_pca_df, kmea_clu_pca_clu2_labels, k=2)

# For 3 clusters
scatter_pca2_vs_all(fifa20_pca_df, kmea_clu_pca_clu3_labels, k=3)

# For 4 clusters
scatter_pca2_vs_all(fifa20_pca_df, kmea_clu_pca_clu4_labels, k=4)

# For 5 clusters
scatter_pca2_vs_all(fifa20_pca_df, kmea_clu_pca_clu5_labels, k=5)
```

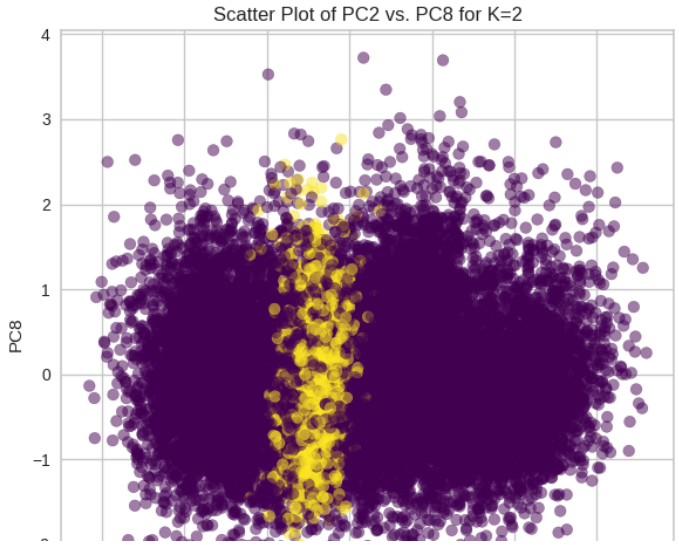

Scatter Plot of PC2 vs. PC3 for K=2
PC3
PC2


Scatter Plot of PC2 vs. PC4 for K=2
PC4
PC2


Scatter Plot of PC2 vs. PC5 for K=2
PC5
PC2


Scatter Plot of PC2 vs. PC6 for K=2
PC6
PC2


Scatter Plot of PC2 vs. PC7 for K=2
PC7


Scatter Plot of PC2 vs. PC8 for K=2
PC8