Assignment 3

In [1]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
df=pd.read_csv('E:\AIML_Externship\Contents\data.csv')
```

In [4]:

```
na = pd.notnull(df["Position"])
```

In [5]:

```
df = df[na]
```

In [6]:

```
df.head()
```

Out[6]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	http

5 rows × 89 columns

In [7]:

```
forward = ["ST", "LW", "RW", "LF", "RF", "RS","LS", "CF"]
midfielder = ["CM","RCM","LCM", "CDM","RDM","LDM", "CAM", "LAM", "RAM", "RM", "LM"]
defender = ["CB", "RCB", "LCB", "LWB", "RWB", "LB", "RB"]
```

In [8]:

```
df.loc[df["Position"] == "GK", "Position"] = 0
df.loc[df["Position"].isin(defender), "Position"] = 1
df.loc[df["Position"].isin(midfielder), "Position"] = 2
df.loc[df["Position"].isin(forward), "Position"] = 3
```

In [9]:

```
df["Position"].value_counts()
```

Out[9]:

```
2    6838
1    5866
3    3418
0    2025
Name: Position, dtype: int64
```

In [10]:

```
df["Position"].unique()
```

Out[10]:

```
array([3, 0, 2, 1], dtype=object)
```

In [11]:

```
df = df[["Position", 'Finishing', 'HeadingAccuracy', 'ShortPassing', 'Volleys', 'Dribbling'
       'Curve', 'FKAccuracy', 'LongPassing', 'BallControl', 'Acceleration',
       'SprintSpeed', 'Agility', 'Reactions', 'Balance', 'ShotPower',
       'Jumping', 'Stamina', 'Strength', 'LongShots', 'Aggression',
       'Interceptions', 'Positioning', 'Vision', 'Penalties', 'Composure',
       'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving', 'GKHandling',
       'GKKicking', 'GKPositioning', 'GKReflexes']]
```

In [12]:

```
df.head()
```

Out[12]:

	Position	Finishing	HeadingAccuracy	ShortPassing	Volleys	Dribbling	Curve	FKAccuracy
0	3	95.0	70.0	90.0	86.0	97.0	93.0	94.0
1	3	94.0	89.0	81.0	87.0	88.0	81.0	76.0
2	3	87.0	62.0	84.0	84.0	96.0	88.0	87.0
3	0	13.0	21.0	50.0	13.0	18.0	21.0	19.0
4	2	82.0	55.0	92.0	82.0	86.0	85.0	83.0

5 rows × 34 columns

In [13]:

```
x = df.drop("Position", axis = 1)
```

In [14]:

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
```

In [15]:

```
x = pd.DataFrame(ss.fit_transform(x))
```

In [16]:

```
y = df["Position"]
```

In [17]:

```
x.head()
```

Out[17]:

	0	1	2	3	4	5	6	7	
0	2.532391	1.018293	2.130190	2.434969	2.201010	2.491028	2.925359	2.236808	2.255(
1	2.481180	2.111424	1.517765	2.491481	1.725114	1.838695	1.895584	1.584271	2.135'
2	2.122700	0.558028	1.721906	2.321945	2.148132	2.219223	2.524891	1.649524	2.195(
3	-1.666942	-1.800833	-0.591700	-1.690394	-1.976298	-1.422972	-1.365369	-0.112327	-0.981:
4	1.866643	0.155296	2.266284	2.208922	1.619359	2.056139	2.296052	2.497823	1.955:

5 rows × 33 columns

In [18]:

```
y.head()
```

Out[18]:

```
0    3
1    3
2    3
3    0
4    2
Name: Position, dtype: object
```

In [19]:

```
from keras.utils.np_utils import to_categorical
y_cat = to_categorical(y)
```

In [20]:

```
y_cat[:10]
```

Out[20]:

```
array([[0., 0., 0., 1.],
       [0., 0., 0., 1.],
       [0., 0., 0., 1.],
       [1., 0., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.]], dtype=float32)
```

In [21]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x.values, y_cat,test_size = 0.2)
```

In [22]:

```
y.shape
```

Out[22]:

```
(18147,)
```

In [23]:

```
x.shape
```

Out[23]:

```
(18147, 33)
```

In [24]:

```
import tensorflow as tf
```

In [25]:

```
import keras as ks
```

In [26]:

```
from keras.models import Sequential
from keras.layers import Dense
```

In [27]:

```
from keras import backend as K
```

In [28]:

```
K.clear_session()
model = Sequential()
model.add(Dense(60, input_shape = (33,), activation = "relu"))
model.add(Dense(15, activation = "relu"))
model.add(Dense(4, activation = "softmax"))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 60)                2040

 dense_1 (Dense)             (None, 15)                915

 dense_2 (Dense)             (None, 4)                 64

=================================================================
Total params: 3,019
Trainable params: 3,019
Non-trainable params: 0
_________________________________________________________________
```

In [29]:

```
model.compile(loss = 'binary_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
```

In [30]:

```
model.fit(x_train, y_train, verbose=1, epochs = 10)
```

```
Epoch 1/10
454/454 [==============================] - 1s 2ms/step - loss: 0.2642 - accu
racy: 0.7714
Epoch 2/10
454/454 [==============================] - 1s 2ms/step - loss: 0.1550 - accu
racy: 0.8694
Epoch 3/10
454/454 [==============================] - 1s 1ms/step - loss: 0.1475 - accu
racy: 0.8775
Epoch 4/10
454/454 [==============================] - 1s 1ms/step - loss: 0.1443 - accu
racy: 0.8775
Epoch 5/10
454/454 [==============================] - 1s 2ms/step - loss: 0.1417 - accu
racy: 0.8813
Epoch 6/10
454/454 [==============================] - 1s 2ms/step - loss: 0.1400 - accu
racy: 0.8832
Epoch 7/10
454/454 [==============================] - 1s 2ms/step - loss: 0.1382 - accu
racy: 0.8843
Epoch 8/10
454/454 [==============================] - 1s 2ms/step - loss: 0.1377 - accu
racy: 0.8833
Epoch 9/10
454/454 [==============================] - 1s 2ms/step - loss: 0.1362 - accu
racy: 0.8859
Epoch 10/10
454/454 [==============================] - 1s 2ms/step - loss: 0.1352 - accu
racy: 0.8868
```

Out[30]:

```
<keras.callbacks.History at 0x283ca42cd00>
```

In [31]:

```
y_pred = model.predict(x_test)
```

In [32]:

```
y_pred
```

Out[32]:

```
array([[8.2068048e-08, 1.4185864e-05, 9.9997199e-01, 1.3689852e-05],
       [9.5788637e-06, 6.2420772e-04, 7.8975612e-01, 2.0961004e-01],
       [7.8040241e-09, 9.9987841e-01, 1.2076200e-04, 8.4082154e-07],
       ...,
       [1.5676377e-06, 9.6960795e-01, 2.9482599e-02, 9.0785127e-04],
       [2.2979043e-07, 7.8547151e-07, 3.0361066e-04, 9.9969530e-01],
       [1.7098776e-06, 1.3583532e-05, 4.3264343e-04, 9.9955207e-01]],
      dtype=float32)
```

In [33]:

```
x_test[0]
```

Out[33]:

```
array([ 1.09847254, -0.01730402,  1.04143382,  1.36124449,  0.72044499,
        1.34944494,  1.95279377,  1.3232557 ,  0.75666966, -0.97876209,
       -0.11782076,  0.44003681,  0.90567385,  0.63921271,  0.95937002,
        0.41523981,  1.05551592,  0.37295491,  1.3438891 ,  0.98605766,
        1.02900667,  0.97480132,  1.38498068,  2.1304406 ,  1.42949429,
        0.63889499,  0.66002396,  0.34450769, -0.14786349, -0.4963977 ,
       -0.49886191, -0.37505521, -0.59653633])
```

In [34]:

```
y_test[0]
```

Out[34]:

```
array([0., 0., 1., 0.], dtype=float32)
```

In [35]:

```
predict = model.predict(x_test)
```

In [36]:

```
predict > 0.5
```

Out[36]:

```
array([[False, False,  True, False],
       [False, False,  True, False],
       [False,  True, False, False],
       ...,
       [False,  True, False, False],
       [False, False, False,  True],
       [False, False, False,  True]])
```

In [ ]:

```

```