In [1]:
```
# Importing required Libraries
import sqlite3
import numpy as np
import pandas as pd
%matplotlib notebook
import matplotlib.pyplot as plt
import xgboost as xgb
from xgboost.sklearn import XGBRegressor
from xgboost import plot_importance

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import Imputer, StandardScaler
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import train_test_split, GridSearchCV, ShuffleSplit, RandomizedSearchCV
from sklearn.pipeline import make_pipeline

import pickle
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umat
h_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:
```
# Create your connection.
cnx = sqlite3.connect('database.sqlite')
df = pd.read_sql_query("SELECT * FROM Player_Attributes", cnx)
```

In [3]:
```
df.head()
```

Out[3]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	crossing	...	visi
0	1	218353	505942	2016-02-18 00:00:00	67.0	71.0	right	medium	medium	49.0	...	54
1	2	218353	505942	2015-11-19 00:00:00	67.0	71.0	right	medium	medium	49.0	...	54
2	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	right	medium	medium	49.0	...	54
3	4	218353	505942	2015-03-20 00:00:00	61.0	65.0	right	medium	medium	48.0	...	53
4	5	218353	505942	2007-02-22 00:00:00	61.0	65.0	right	medium	medium	48.0	...	53

5 rows × 42 columns

In [4]:
```
# Defining Target variable
target = df.pop('overall_rating')
```

In [5]:
```
df.shape
```

Out[5]:
```
(183978, 41)
```

In [6]:
```
target.head()
```

Out[6]:
```
0    67.0
1    67.0
2    62.0
3    61.0
4    61.0
Name: overall_rating, dtype: float64
```

Imputing target funtion :

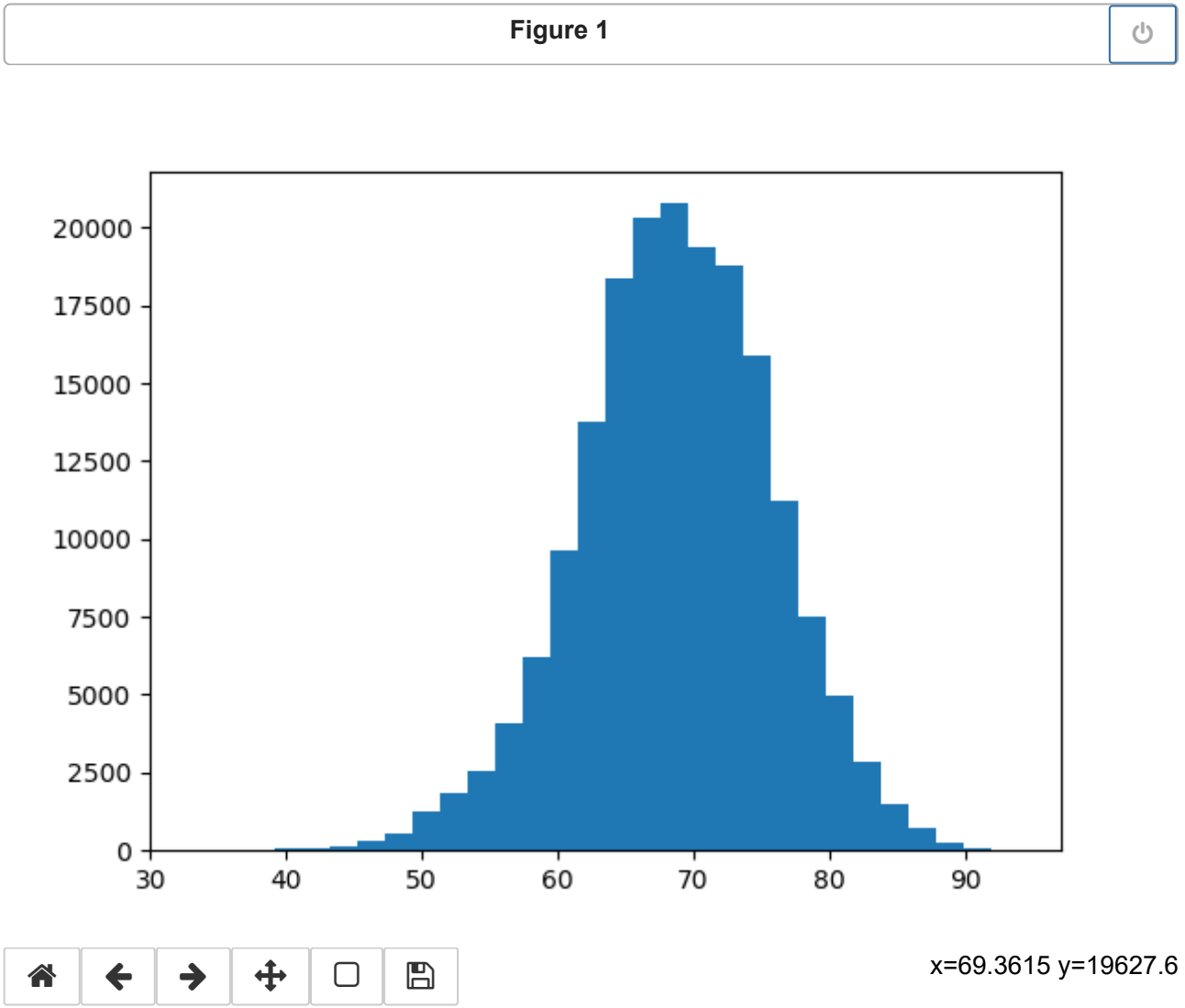
In [7]:
```
target.isnull().values.sum()
```

Out[7]:
```
836
```

In [8]:
```
target.describe()
```

Out[8]:
```
count    183142.000000
mean         68.600015
std           7.041139
min          33.000000
25%          64.000000
50%          69.000000
75%          73.000000
max          94.000000
Name: overall_rating, dtype: float64
```

In [9]:
```
# Histogram of the target variable
plt.hist(target, 30, range=(33, 94))
```




```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:746: RuntimeWarning: invalid value encountered in g
reater_equal
  keep = (tmp_a >= first_edge)
C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:747: RuntimeWarning: invalid value encountered in l
```

```
  ess_equal
    keep &= (tmp_a <= last_edge)
```

Out[9]:
```
(array([7.0000e+00, 6.0000e+00, 2.0000e+01, 6.5000e+01, 9.4000e+01,
        1.4200e+02, 2.9400e+02, 5.2600e+02, 1.2510e+03, 1.8450e+03,
        2.5780e+03, 4.0870e+03, 6.1890e+03, 9.6500e+03, 1.3745e+04,
        1.8366e+04, 2.0310e+04, 2.0773e+04, 1.9382e+04, 1.8784e+04,
        1.5915e+04, 1.1254e+04, 7.5250e+03, 4.9470e+03, 2.8290e+03,
        1.4590e+03, 7.4800e+02, 2.2800e+02, 8.4000e+01, 3.9000e+01]),
 array([33.        , 35.03333333, 37.06666667, 39.1       , 41.13333333,
        43.16666667, 45.2       , 47.23333333, 49.26666667, 51.3       ,
        53.33333333, 55.36666667, 57.4       , 59.43333333, 61.46666667,
        63.5       , 65.53333333, 67.56666667, 69.6       , 71.63333333,
        73.66666667, 75.7       , 77.73333333, 79.76666667, 81.8       ,
        83.83333333, 85.86666667, 87.9       , 89.93333333, 91.96666667,
        94.        ]),
 <a list of 30 Patch objects>)
```

In [10]:
```
# filling the missing values with mean of the target variable
y = target.fillna(target.mean())
```

In [28]:
```
# Checking for null values in target variable
y.isnull().sum()
```

Out[28]:
```
0
```

Data Exploration :

In [12]:
```
# Displaying all the columns of the data frame
df.columns
```

Out[12]:
```
Index(['id', 'player_fifa_api_id', 'player_api_id', 'date', 'potential',
       'preferred_foot', 'attacking_work_rate', 'defensive_work_rate',
       'crossing', 'finishing', 'heading_accuracy', 'short_passing', 'volleys',
       'dribbling', 'curve', 'free_kick_accuracy', 'long_passing',
       'ball_control', 'acceleration', 'sprint_speed', 'agility', 'reactions',
       'balance', 'shot_power', 'jumping', 'stamina', 'strength', 'long_shots',
       'aggression', 'interceptions', 'positioning', 'vision', 'penalties',
       'marking', 'standing_tackle', 'sliding_tackle', 'gk_diving',
       'gk_handling', 'gk_kicking', 'gk_positioning', 'gk_reflexes'],
      dtype='object')
```

In [13]:
```
# to display the column details
df. info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183978 entries, 0 to 183977
Data columns (total 41 columns):
id                    183978 non-null int64
player_fifa_api_id    183978 non-null int64
player_api_id         183978 non-null int64
date                  183978 non-null object
potential             183142 non-null float64
preferred_foot        183142 non-null object
attacking_work_rate   180748 non-null object
defensive_work_rate   183142 non-null object
crossing              183142 non-null float64
finishing             183142 non-null float64
heading_accuracy      183142 non-null float64
short_passing         183142 non-null float64
volleys               181265 non-null float64
dribbling             183142 non-null float64
curve                 181265 non-null float64
free_kick_accuracy    183142 non-null float64
long_passing          183142 non-null float64
ball_control          183142 non-null float64
acceleration          183142 non-null float64
sprint_speed          183142 non-null float64
agility               181265 non-null float64
reactions             183142 non-null float64
balance               181265 non-null float64
shot_power            183142 non-null float64
jumping               181265 non-null float64
stamina               183142 non-null float64
strength              183142 non-null float64
long_shots            183142 non-null float64
aggression            183142 non-null float64
interceptions         183142 non-null float64
positioning           183142 non-null float64
vision                181265 non-null float64
penalties             183142 non-null float64
marking               183142 non-null float64
standing_tackle       183142 non-null float64
```

```
sliding_tackle         181265 non-null float64
gk_diving              183142 non-null float64
gk_handling            183142 non-null float64
gk_kicking             183142 non-null float64
gk_positioning         183142 non-null float64
gk_reflexes            183142 non-null float64
dtypes: float64(34), int64(3), object(4)
memory usage: 57.5+ MB
```

we can see only four features have the type 'object'. here the feature named 'date' has no significance in this problem so can ignore it and perform one hot encoding on the rest of 3 features.

In [50]:
```
# one hot encoding for dummy column/categorical columns
df_dummy = pd.get_dummies(df, columns=['preferred_foot', 'attacking_work_rate', 'defensive_work_rate'])
df_dummy.head()
```

Out[50]:

	id	player_fifa_api_id	player_api_id	date	potential	crossing	finishing	heading_accuracy	short_passing	volleys	...	defensive_work_rate_9
0	1	218353	505942	2016-02-18 00:00:00	71.0	49.0	44.0	71.0	61.0	44.0	...	0
1	2	218353	505942	2015-11-19 00:00:00	71.0	49.0	44.0	71.0	61.0	44.0	...	0
2	3	218353	505942	2015-09-21 00:00:00	66.0	49.0	44.0	71.0	61.0	44.0	...	0
3	4	218353	505942	2015-03-20 00:00:00	65.0	48.0	43.0	70.0	60.0	43.0	...	0
4	5	218353	505942	2007-02-22 00:00:00	65.0	48.0	43.0	70.0	60.0	43.0	...	0

5 rows × 67 columns

In [51]:
```
df_dummy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183978 entries, 0 to 183977
Data columns (total 67 columns):
id                          183978 non-null int64
player_fifa_api_id          183978 non-null int64
player_api_id               183978 non-null int64
date                        183978 non-null object
potential                   183142 non-null float64
crossing                    183142 non-null float64
finishing                   183142 non-null float64
heading_accuracy            183142 non-null float64
short_passing               183142 non-null float64
volleys                     181265 non-null float64
dribbling                   183142 non-null float64
curve                       181265 non-null float64
free_kick_accuracy          183142 non-null float64
long_passing                183142 non-null float64
ball_control                183142 non-null float64
acceleration                183142 non-null float64
sprint_speed                183142 non-null float64
agility                     181265 non-null float64
reactions                   183142 non-null float64
balance                     181265 non-null float64
shot_power                  183142 non-null float64
jumping                     181265 non-null float64
stamina                     183142 non-null float64
strength                    183142 non-null float64
long_shots                  183142 non-null float64
aggression                  183142 non-null float64
interceptions               183142 non-null float64
positioning                 183142 non-null float64
vision                      181265 non-null float64
penalties                   183142 non-null float64
marking                     183142 non-null float64
standing_tackle             183142 non-null float64
sliding_tackle              181265 non-null float64
gk_diving                   183142 non-null float64
gk_handling                 183142 non-null float64
gk_kicking                  183142 non-null float64
```

```
gk_positioning                183142 non-null float64
gk_reflexes                   183142 non-null float64
preferred_foot_left           183978 non-null uint8
preferred_foot_right          183978 non-null uint8
attacking_work_rate_None      183978 non-null uint8
attacking_work_rate_high      183978 non-null uint8
attacking_work_rate_le        183978 non-null uint8
attacking_work_rate_low       183978 non-null uint8
attacking_work_rate_medium    183978 non-null uint8
attacking_work_rate_norm      183978 non-null uint8
attacking_work_rate_stoc      183978 non-null uint8
attacking_work_rate_y         183978 non-null uint8
defensive_work_rate_0         183978 non-null uint8
defensive_work_rate_1         183978 non-null uint8
defensive_work_rate_2         183978 non-null uint8
defensive_work_rate_3         183978 non-null uint8
defensive_work_rate_4         183978 non-null uint8
defensive_work_rate_5         183978 non-null uint8
defensive_work_rate_6         183978 non-null uint8
defensive_work_rate_7         183978 non-null uint8
defensive_work_rate_8         183978 non-null uint8
defensive_work_rate_9         183978 non-null uint8
defensive_work_rate__0        183978 non-null uint8
defensive_work_rate_ean       183978 non-null uint8
defensive_work_rate_es        183978 non-null uint8
defensive_work_rate_high      183978 non-null uint8
defensive_work_rate_low       183978 non-null uint8
defensive_work_rate_medium    183978 non-null uint8
defensive_work_rate_o         183978 non-null uint8
defensive_work_rate_ormal     183978 non-null uint8
defensive_work_rate_tocky     183978 non-null uint8
dtypes: float64(34), int64(3), object(1), uint8(29)
memory usage: 58.4+ MB
```

In [52]:
```
#Making the X variable as independent variable list
X = df_dummy.drop(['id', 'date'], axis=1)
```

Feature selection :

In [16]:
```
# Splitting the data into Train and Test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [17]:
```
#imputing null value of each column with the mean of that column
imput = Imputer()
X_train = imput.fit_transform(X_train)
X_test = imput.fit_transform(X_test)
```

In [18]:
```
#finding feature_importance for feature selection. from it we'll be able to decide threshold value
model = XGBRegressor()
model.fit(X_train, y_train)
print(model.feature_importances_)
```

```
[0.01714286 0.02       0.10714286 0.02       0.04714286 0.03571429
 0.03285714 0.         0.04       0.         0.         0.00714286
 0.04714286 0.01       0.01571429 0.00142857 0.18571429 0.
 0.01428571 0.00571429 0.00857143 0.03142857 0.00571429 0.00714286
 0.01857143 0.01857143 0.00571429 0.         0.04285714 0.04571429
 0.01       0.04571429 0.02571429 0.02428572 0.06       0.04285714
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.        ]
```

In [19]:
```
selection = SelectFromModel(model, threshold=0.01, prefit=True)

select_X_train = selection.transform(X_train)
select_X_test = selection.transform(X_test)
```

In [20]:
```
select_X_train.shape
```

Out[20]:
```
(137983, 24)
```

Training data using Linear Regression

In [44]:
```
pline = make_pipeline(StandardScaler(),LinearRegression())  #preprocessing(standard scalling), estimator(linear regression

cross_val = ShuffleSplit(random_state=0)   #defining type of cross_validation(shuffle spliting)

param_grid = {'linearregression__n_jobs': [50]}     #parameters for model tunning

grid = GridSearchCV(pline, param_grid=param_grid, cv=cross_val)
```

In [45]:
```
grid.fit(select_X_train, y_train)          #training data set
```

Out[45]:
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=0, test_size='default',
       train_size=None),
       error_score='raise',
       estimator=Pipeline(memory=None,
     steps=[('standardscaler', StandardScaler(copy=True, with_mean=True, with_std=True)), ('linearregression', LinearRe
gression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False))]),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'linearregression__n_jobs': [50]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [46]:
```
grid.best_params_
```

Out[46]:
```
{'linearregression__n_jobs': 50}
```

In [47]:
```
lin_reg = pickle.dumps(grid)
```

In [48]:
```
lin_reg = pickle.loads(lin_reg)
```

In [49]:
```
print("""Linear Regressor accuracy is {lin}""".format(lin=lin_reg.score(select_X_test, y_test)))
```

```
Linear Regressor accuracy is 0.8547439877398585
```

In [ ]: