DATA MINING

(COCSC16)

PRACTICAL FILE

NAME : Sugam Kumar Rohilla

ROLL NO : 2019UCO1541

COE-1

List of Experiments

EXPERIMENT -1

AIM:

Visualize your dataset in Weka and learn about Jitter.

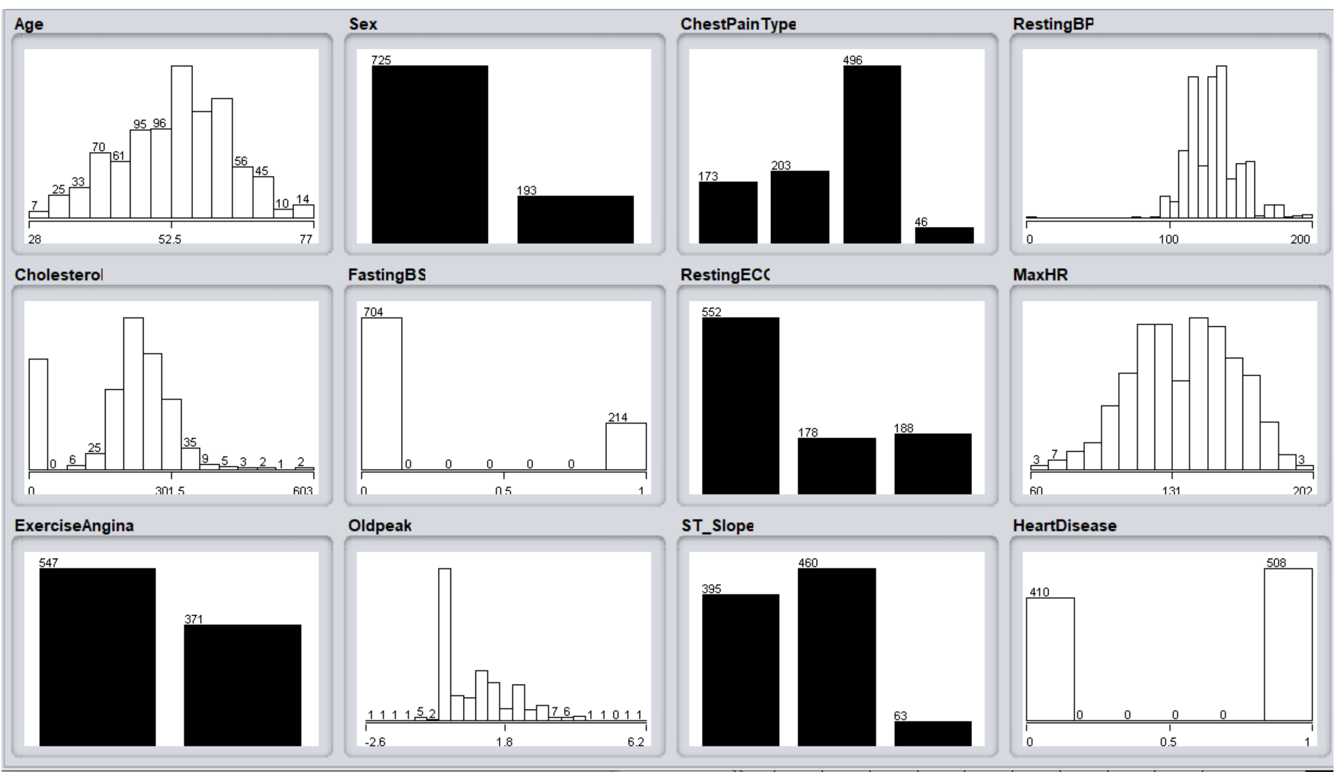
DATASET:

Heart Disease Prediction

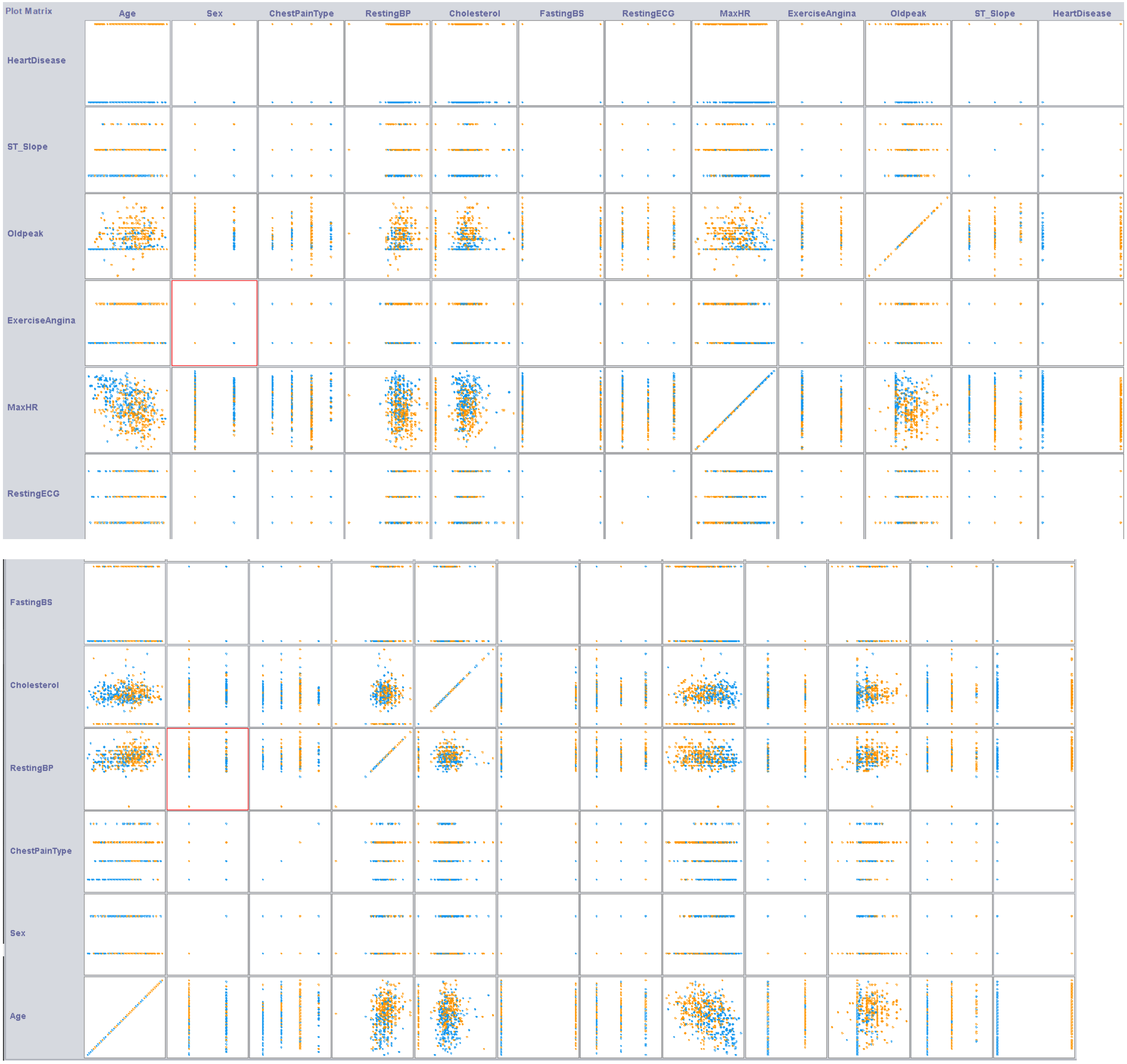
RESULTS:

Loading the dataset Heart Disease Prediction in Weka and Visualizing in the pre-processing step:

Dataset:

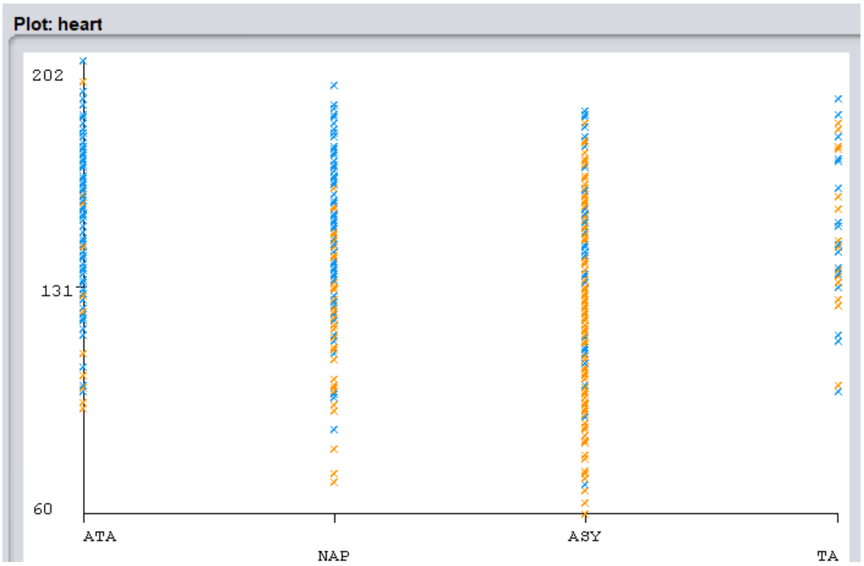


Visualizing the relation between attributes:

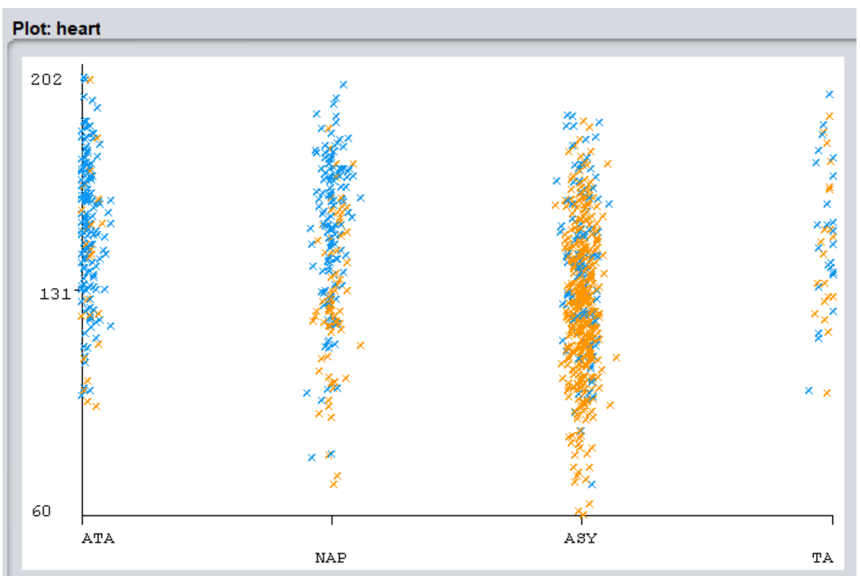


Understanding Jitter:

In a plot between MaxHR and ChestPainType, with no jitter:



With considerable Jitter:



Jitter is a function which adds artificial random noise to the coordinates of the potted points in order to spread the data out so that overlapping points become more visible thus helping in the understanding of dataset.

EXPERIMENT -2

AIM:

Calculate co-relation among attributes

DATASET:

Heart Disease Prediction

RESULTS:

Calculating Person’s coefficient between attribute:

```
Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 12 HeartDisease):
        Correlation Ranking Filter
Ranked attributes:
 0.5538   11 ST_Slope
 0.4943    9 ExerciseAngina
 0.4048    3 ChestPainType
 0.404    10 Oldpeak
 0.4004    8 MaxHR
 0.3054    2 Sex
 0.282     1 Age
 0.2673    6 FastingBS
 0.2327    5 Cholesterol
 0.1076    4 RestingBP
 0.0771    7 RestingECG

Selected attributes: 11,9,3,10,8,2,1,6,5,4,7 : 11
```

By finding the Pearson’s correlation and evaluating the worth of an attribute, we can eliminate redundant attributes thus preventing curse of dimensionality.

Since RestingECG and RestingBP are ranked least, removing these attribute from consideration

EXPEIMENT 3

AIM:

Calculate Information Gain of different attributes

DATASET:

Heart Disease Detection

RESULTS:

Calculating Information Gained by each attribute for prediction heart disease.

```
Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 10 HeartDisease):
        Information Gain Ranking Filter

Ranked attributes:
 0.2993   9 ST_Slope
 0.225    3 ChestPainType
 0.19     7 ExerciseAngina
 0.1614   8 Oldpeak
 0.1275   6 MaxHR
 0.0834   4 Cholesterol
 0.0696   1 Age
 0.0685   2 Sex
 0.0549   5 FastingBS

Selected attributes: 9,3,7,8,6,4,1,2,5 : 9
```

CONCLUSION:

ST_Slope continues to provide maximum information for predicting heart disease. While FastingBS, Sex, and Age are low indicators as per information gain, PCA and correlation analysis show a decent score hence are considered as features for prediction of heart disease.

EXPERIMENT 4:

AIM:

Perform PCA on a dataset

DATASET:

Heart Disease Detection

RESULTS:

Performing Principal Component Analysis and Ranking on the basis of variance:

```
Ranked attributes:
 0.7367   1 -0.421ST_Slope=Up+0.361ChestPainType=ASY+0.359ST_Slope=Flat+0.356ExerciseAngina=Y-0.324MaxHR...
 0.6348   2 0.48 ChestPainType=NAP-0.46Cholesterol+0.401FastingBS-0.263ChestPainType=ASY-0.255ChestPainType=ATA...
 0.5348   3 0.54 ChestPainType=NAP+0.464ST_Slope=Flat-0.363ST_Slope=Down-0.345ChestPainType=ASY-0.284ST_Slope=Up...
 0.4414   4 0.586ST_Slope=Down+0.497Oldpeak+0.367Cholesterol+0.244ChestPainType=TA-0.219ST_Slope=Flat...
 0.3612   5 0.796ChestPainType=TA-0.401ChestPainType=NAP+0.243FastingBS-0.19ExerciseAngina=Y+0.187ST_Slope=Flat...
 0.2906   6 -0.557Age-0.502Sex=F-0.441ChestPainType=ATA+0.323MaxHR+0.229ChestPainType=TA...
 0.2263   7 -0.579ChestPainType=ATA+0.438Sex=F+0.384ChestPainType=ASY+0.299ST_Slope=Up+0.263Age...
 0.1649   8 0.557FastingBS+0.462Sex=F+0.405MaxHR-0.368Age-0.215ChestPainType=TA...
 0.112    9 -0.562Cholesterol-0.515FastingBS+0.4  Sex=F+0.259ST_Slope=Down-0.257Age...
 0.0711  10 -0.588ExerciseAngina=Y-0.404Oldpeak+0.392Cholesterol+0.365ST_Slope=Down-0.295ST_Slope=Up...
 0.0312  11 -0.532ExerciseAngina=Y+0.402MaxHR+0.388Oldpeak+0.372Age-0.286Cholesterol...

Selected attributes: 1,2,3,4,5,6,7,8,9,10,11 : 11
```

By performing a PCA, a set of combinations of attributes are obtained along with the respective variance in data that they represent. By analysing, we can see ST_slope an ChestPainType capturing maximum amount of variance in data.

EXPERIMENT 5:

AIM:

Training a Decision Tree

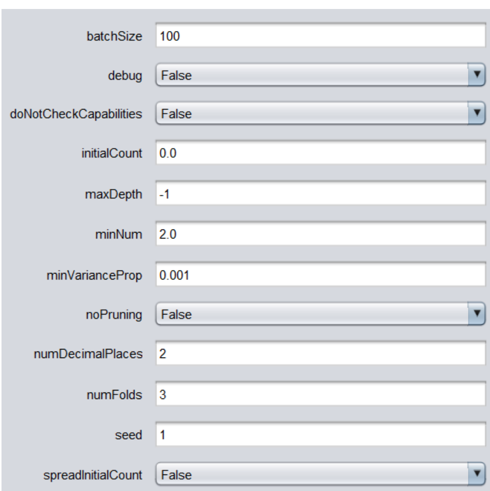
DATASET:

Heart Disease Detection

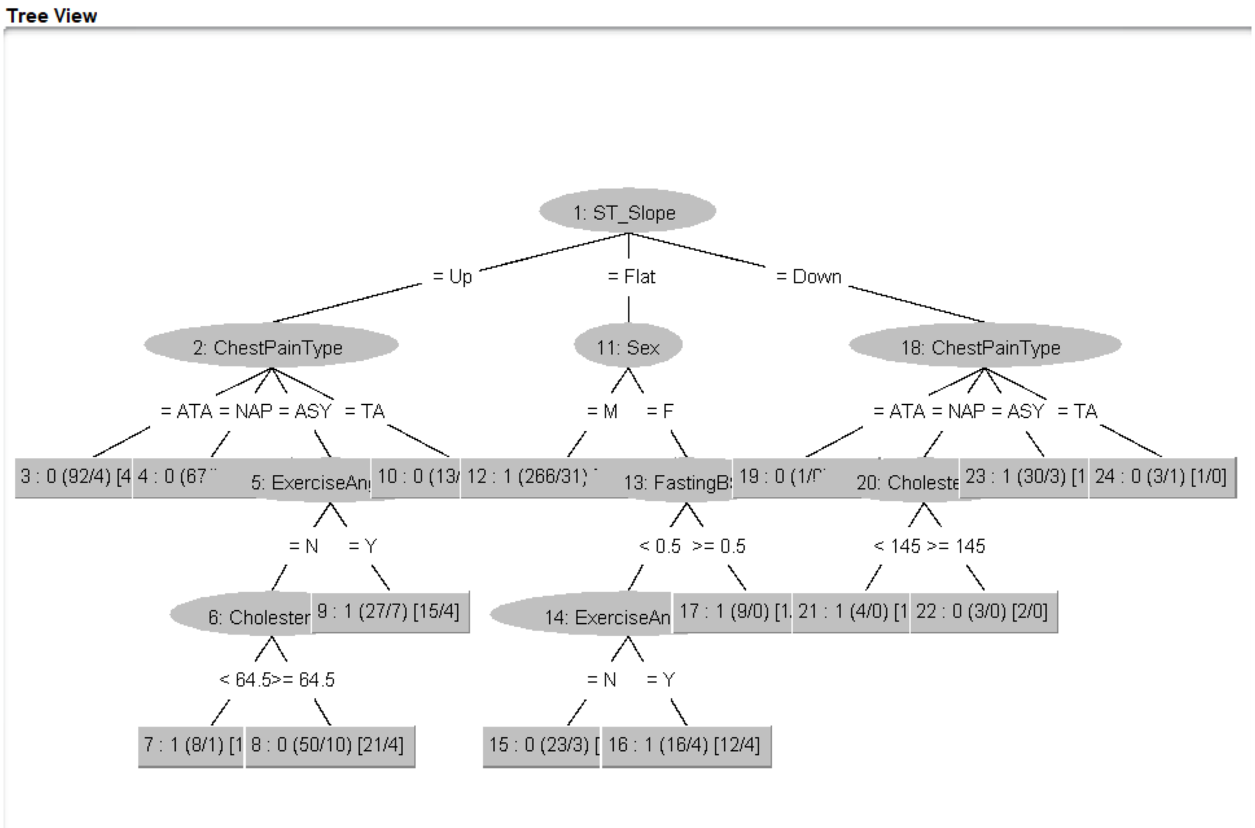
THEORY:

Decision Tree User: REPTree (Reduced Error Pruning Tree) a fast decision tree learner that builds a decision/ regression tree using Information gain as the splitting criterion and prunes it using reduced error pruning algorithm.

Parameters for the Decision tree:



Visualization:



RESULTS:

```
=== Summary ===

Correctly Classified Instances         760               82.7887 %
Incorrectly Classified Instances       158               17.2113 %
Kappa statistic                          0.6492
Mean absolute error                      0.2357
Root mean squared error                  0.3623
Relative absolute error                 47.684  %
Root relative squared error             72.8761 %
Total Number of Instances              918

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.768    0.124    0.833      0.768    0.799      0.651    0.876     0.843     0
                 0.876    0.232    0.824      0.876    0.849      0.651    0.876     0.854     1
Weighted Avg.    0.828    0.184    0.828      0.828    0.827      0.651    0.876     0.849

=== Confusion Matrix ===

   a   b   <-- classified as
 315  95 |   a = 0
  63 445 |   b = 1
```

CONCLUSION:

By using a decision tree classifier, an accuracy of 82.8% was obtained.

EXPERIMENT 6

AIM:

Train a Naïve Bayes Classifier

DATASET:

Heart Disease Detection

PERFORMANCE:

Upon performing the classification through use of naïve bayes classifier: bayes.NaiveBayes the following result was obtained:

```
=== Summary ===

Correctly Classified Instances         795               86.6013 %
Incorrectly Classified Instances       123               13.3987 %
Kappa statistic                          0.7282
Mean absolute error                      0.1566
Root mean squared error                  0.338
Relative absolute error                 31.6738 %
Root relative squared error             67.9811 %
Total Number of Instances              918

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.837    0.110    0.860      0.837    0.848      0.728    0.919     0.911     0
                 0.890    0.163    0.871      0.890    0.880      0.728    0.919     0.916     1
Weighted Avg.    0.866    0.140    0.866      0.866    0.866      0.728    0.919     0.914

=== Confusion Matrix ===

   a   b   <-- classified as
 343  67 |   a = 0
  56 452 |   b = 1
```

RESULT:

An accuracy of 86.6% was obtained. This result is better than the classification accuracy of 82.8% obtained by decision trees.

EXPERIMENT 7

AIM:

Train a Bayesian Belief Network Classifier

DATASET:

Heart Disease Detection

PERFORMANCE:

Classification is performed using the inbuilt bayes.BayesNet

```
=== Summary ===

Correctly Classified Instances         779               84.8584 %
Incorrectly Classified Instances       139               15.1416 %
Kappa statistic                          0.6935
Mean absolute error                      0.1682
Root mean squared error                  0.3459
Relative absolute error                 34.0285 %
Root relative squared error             69.58   %
Total Number of Instances              918

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.827    0.134    0.833      0.827    0.830      0.693    0.915     0.908     0
                 0.866    0.173    0.861      0.866    0.864      0.693    0.915     0.919     1
Weighted Avg.    0.849    0.156    0.848      0.849    0.849      0.693    0.915     0.914

=== Confusion Matrix ===

   a   b   <-- classified as
 339  71 |   a = 0
  68 440 |   b = 1
```

RESULT:

Accuracy of 84.8 % was obtained.

EXPERIMENT 8:

AIM:

Train a Linear Regression Model

DATASET:

Heart Disease Detection

PERFORMANCE:

```
Linear Regression Model

HeartDisease =

      0.003  * Age +
      0.1618 * Sex=M +
      0.2357 * ChestPainType=ASY +
     -0.0005 * Cholesterol +
      0.1317 * FastingBS +
      0.1396 * ExerciseAngina=Y +
      0.0499 * Oldpeak +
      0.2196 * ST_Slope=Down,Flat +
      0.164  * ST_Slope=Flat +
     -0.0996

Time taken to build model: 0.09 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.7476
Mean absolute error                      0.2444
Root mean squared error                  0.3302
Relative absolute error                 49.4224 %
Root relative squared error             66.3966 %
Total Number of Instances              918
```

RESULT:

Since the task was that of classification, the class labels had a value of 0 or 1, a prediction anywhere in between would count as an error thus such a performance was expected since a regression model was trained for a classification task.

EXPERIMENT 9:

AIM:

Logistic Regression Model

DATASET:

Heart Disease Detection

PERFORMANCE:

```
=== Summary ===

Correctly Classified Instances         795               86.6013 %
Incorrectly Classified Instances       123               13.3987 %
Kappa statistic                          0.7277
Mean absolute error                      0.2007
Root mean squared error                  0.3207
Relative absolute error                 40.5949 %
Root relative squared error             64.506  %
Total Number of Instances              918

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.827    0.102    0.867      0.827    0.846      0.728    0.924     0.916     0
                 0.898    0.173    0.865      0.898    0.881      0.728    0.924     0.929     1
Weighted Avg.    0.866    0.142    0.866      0.866    0.866      0.728    0.924     0.923

=== Confusion Matrix ===

   a   b   <-- classified as
 339  71 |   a = 0
  52 456 |   b = 1
```

RESULT:

An accuracy of 86.6% was obtained along with a recall of 86.6%.

EXPERIMENT 10:

AIM:

KNN Model

DATASET:

Heart Disease Detection

PERFORMANCE:

```
=== Summary ===

Correctly Classified Instances         735               80.0654 %
Incorrectly Classified Instances       183               19.9346 %
Kappa statistic                          0.5979
Mean absolute error                      0.2001
Root mean squared error                  0.4459
Relative absolute error                 40.4746 %
Root relative squared error             89.7013 %
Total Number of Instances              918

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.793    0.193    0.768      0.793    0.780      0.598    0.798     0.702     0
                 0.807    0.207    0.828      0.807    0.818      0.598    0.798     0.781     1
Weighted Avg.    0.801    0.201    0.802      0.801    0.801      0.598    0.798     0.746

=== Confusion Matrix ===

   a   b   <-- classified as
 325  85 |   a = 0
  98 410 |   b = 1
```

RESULT:

An accuracy of 80% was obtained.

EXPERIMENT 11:

AIM:

K-Means Clustering Model

DATASET:

Heart Disease Detection

PERFORMANCE:

```
Final cluster centroids:
                                     Cluster#
Attribute           Full Data            0            1
                      (918.0)      (502.0)      (416.0)
==================================================
Age                   53.5109       56.012      50.4928
Sex                         M            M            M
ChestPainType             ASY          ASY          ATA
Cholesterol          198.7996     175.0896     227.4111
FastingBS              0.2331       0.3367       0.1082
MaxHR                136.8094     125.8486     150.0361
ExerciseAngina              N            Y            N
Oldpeak                0.8874       1.3131       0.3736
ST_Slope                 Flat         Flat           Up
HeartDisease                1            1            0




Time taken to build model (full training data) : 0.02 second

=== Model and evaluation on training set ===

Clustered Instances

0       502 ( 55%)
1       416 ( 45%)
```

EXPERIMENT 12:

Q. Knowing the Types of data – ordinal, nominal, ratio, interval.

- **Nominal**
  - Nominal scales are used for labeling variables, without any quantitative value. “Nominal” scales could simply be called “labels.”
  - all of the scales shown below are mutually exclusive (no overlap) and none of them have any numerical significance.
  - Properties: distinctness (= != are meaningful)
  - Ex. ID Numbers, eye color, Zip Codes

- **Ordinal**
  - With ordinal scales, the order of the values is what’s important and significant, but the differences between each one is not really known.
  - Ordinal scales are typically measures of non-numeric concepts like satisfaction, happiness, discomfort, etc.
  - distinctness and order, (= != < > are meaningful)
  - Ex. Grades, ranks etc.

- **Interval**
  - Interval scales are numeric scales in which we know both the order and the exact differences between the values.

o For interval attributes, differences between values are meaningful.

o Properties: distinctness, order and differences, (= != < > + - are meaningful)

o Ex. Calender dates, temperature in Celsius

• Ratio

o For ratio attributes, both differences and ratio are meaningful.

o They have an absolute 0.

o Properties: distinctness, order, differences and ratios (= != < > + - * / are meaningful)

o Ex. Temperature in Kelvin, age, mass.

EXPERIMENT 13:

Q. Find the mean, median, variance, and standard deviation of data.

```
#include <bits/stdc++.h>
using namespace std;

int main() {

    ifstream input_file("numeric_data.csv");

    if (!input_file.is_open())
        cout<<"ERROR\n";


    vector<vector<string>> data;

    while (input_file.good()) {
        string str;
        getline(input_file, str, '\n');
        string s = str;
        vector<string> v;

        stringstream ss(s);

        while (ss.good()) {
            string substr;
            getline(ss, substr, ',');
            v.push_back(substr);
        }
        data.push_back(v);
    }

    input_file.close();
    int n = data.size();
    int m = data[0].size();

    cout<<"This input csv file is an "<<n<<"x"<<m<<" dataset\n";

    for(int j=0;j<m;j++){


        double mean = 0;
```

```
        for (int i = 0; i < n; i++) {
            mean += stoi(data[i][j]);
        }
        mean = mean * 1.0/n;


        double variance = 0;
        for (int i = 0; i < n; i++) {
            double x = abs(mean - stoi(data[i][j]));
            x = x * x;
            variance += x;
        }
        variance /= (double)n;


        double std_deviation = sqrt(variance);
        vector<int> v;
        for (int i = 0; i < n; i++) {
            v.push_back(stoi(data[i][j]));
        }
        sort(v.begin(), v.end());
        double median = 0;
        if (n & 1) {
            int idx = (n + 1) / 2;
            idx--;
            median = v[idx];
        }
        else {
            int idx = n/ 2;
            median = v[idx];
            median += v[idx - 1];
            median /= (double)2;
        }
        cout<<"Measures for "<<j<<" feature : "<<endl;
        cout << "mean" << " : " << mean << endl;
        cout << "median" << " : " << median << endl;
        cout << "variance" << " : " << variance << endl;
        cout << "standard deviation" << " : " << std_deviation << endl;
    }
    return 0;
}
```

Input File:

“numeric_data.csv”

6,11,12,1

125,15,3,5

42,23,63,21

1,63,123,6

4,27,63,3

82,23,95,27

58,74,12,234

2,12,753,8

1,411,2,12

47,69,6,7

OUTPUT:

```
This input csv file is an 3x5 dataset
Measures for 0 feature :
mean : 6
median : 6
variance : 16.6667
standard deviation : 4.08248
Measures for 1 feature :
mean : 7
median : 7
variance : 16.6667
standard deviation : 4.08248
Measures for 2 feature :
mean : 8
median : 8
variance : 16.6667
standard deviation : 4.08248
Measures for 3 feature :
mean : 9
median : 9
variance : 16.6667
standard deviation : 4.08248
Measures for 4 feature :
mean : 9.66667
median : 10
variance : 20.2222
standard deviation : 4.49691
```

DATA MINING PROJECT

FIFA MAN OF THE MATCH PREDICTION

Team Members :

1) Sugam Rohilla (2019UCO1541)
2) Mohit (2019UCO1578)
3) Ajay Kumar (2019UCO1565)
4) Kunal Mathur (2019UCO1581)

ABOUT THE PROJECT

– The objective of this project is to predict the Team with Man Of The Match award before the official announcement that will be made right after the match.

– Dataset consists of 26 Predictor Variables (Independent) and one Outcome Variable (Dependent).

– The outcome variable value is either Yes or No indicating whether a person from the team wins the man of the match or not.

FEATURES

Column	Description
Date	match Date
Team	Playing Team
Opponent	Opponent Team
Goal Scored	Number of goals scored by this team
Ball Possession %	Amount of time ball was in control by the team
Attempts	Number of attempts to score goal
On-Target	Number of shots on-target
Off-Target	Number of shots that went off-target
Blocked	Number of attempts blocked by the opponent's team
Corners	Number of corner shots used
Offsides	Number of off-side events
Free Kicks	Number of free-kicks used
Saves	Number of saves by the goal keeper
Pass Accuracy %	Percentage of passes that reached the same team player as aimed
Passes	Total number of passes by the team
Distance Covered (Kms)	Total distance covered by the team members in this game
Fouls Committed	Number of fouls committed by the team members
Yellow Card	Number of Yellow warning received
Yellow & Red	Number of Yellow & Red warning received
Red	Number of Red cards received

Features

Man of the Match	Did this team member win Man of the Match?
1st Goal	When did the team score the 1st goal?
Round	Stage of the match
PSO	Was there a penalty shootout (PSO) in this match?
Goals in PSO	Number of goals scored in the Penalty shootout
Own goals	Number of own goals
Own goal Time	When did the team score own goal?

Target Feature :- Man of the Match (Did this team member win Man of the Match ? .

Exploratory Data Analysis

In [30]:
```python
data_df = pd.read_csv('fifa.csv')
data_df.head()
```

Out[30]:

	Date	Team	Opponent	Goal Scored	Ball Possession %	Attempts	On-Target	Off-Target	Blocked	Corners	...	Yellow Card	Yellow & Red	Red	Man of the Match	1st Goal	Round	PSO	Goals in PSO	O go
0	14-06-2018	Russia	Saudi Arabia	5	40	13	7	3	3	6	...	0	0	0	Yes	12.0	Group Stage	No	0	N
1	14-06-2018	Saudi Arabia	Russia	0	60	6	0	3	3	2	...	0	0	0	No	NaN	Group Stage	No	0	N
2	15-06-2018	Egypt	Uruguay	0	43	8	3	3	2	0	...	2	0	0	No	NaN	Group Stage	No	0	N
3	15-06-2018	Uruguay	Egypt	1	57	14	4	6	4	5	...	0	0	0	Yes	89.0	Group Stage	No	0	N
4	15-06-2018	Morocco	Iran	0	64	13	3	6	4	5	...	1	0	0	No	NaN	Group Stage	No	0	

5 rows × 27 columns

In [31]:
```python
data_df.columns
```

Out[31]:
```
Index(['Date', 'Team', 'Opponent', 'Goal Scored', 'Ball Possession %',
       'Attempts', 'On-Target', 'Off-Target', 'Blocked', 'Corners', 'Offsides',
       'Free Kicks', 'Saves', 'Pass Accuracy %', 'Passes',
       'Distance Covered (Kms)', 'Fouls Committed', 'Yellow Card',
       'Yellow & Red', 'Red', 'Man of the Match', '1st Goal', 'Round', 'PSO',
       'Goals in PSO', 'Own goals', 'Own goal Time'],
      dtype='object')
```

Data Preprocessing

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Major Tasks in Data Preprocessing:

1. Data Cleaning
2. Data integration
3. Data reduction
4. Data transformation

DATA CLEANING

Data cleaning is the process to remove incorrect data, incomplete data and inaccurate data from the datasets, and it also replaces the missing values.

Removing Duplicates:

```
In [32]: data_df.drop_duplicates()
```

Handling Missing/NULL values:

```
In [29]: training_features.isnull().sum()

Out[29]: Goal Scored             0
         Corners                0
         Pass Accuracy %        0
         Man of the Match       0
         Efficient Attempts     0
         Fouls Advantage        0
         dtype: int64
```

Data Reduction

This process helps in the reduction of the volume of the data which makes the analysis easier yet produces the same or almost the same result. This reduction also helps to reduce storage space.

DELETING UNWANTED COLUMNS:

```
In [34]: training_features = data_df.drop(['Date', 'Team', 'Opponent', 'Ball Possession %', 'Round', 'Offsides','Passes','Saves'
         'Distance Covered (Kms)', 'Yellow Card', 'Yellow & Red', 'Red', 'PSO', 'Goals in PSO','Own goal Time', '1st Goal',
         'Own goals'], axis = 1)

         training_features.head(3)
```

Out[34]:

	Goal Scored	Attempts	On-Target	Off-Target	Blocked	Corners	Free Kicks	Pass Accuracy %	Fouls Committed	Man of the Match
0	5	13	7	3	3	6	11	78	22	Yes
1	0	6	0	3	3	2	25	86	10	No
2	0	8	3	3	2	0	7	78	12	No

Replacing Yes & No with 1 & 0 respectively(Man of the match :-

```
In [7]: training_features['Man of the Match'].replace(['Yes', 'No'], [1, 0], inplace=True)
```

Out[7]:

	Goal Scored	Attempts	On-Target	Off-Target	Blocked	Corners	Free Kicks	Pass Accuracy %	Fouls Committed	Man of the Match
0	5	13	7	3	3	6	11	78	22	1
1	0	6	0	3	3	2	25	86	10	0

Merging and Merging Attribute :-

```
In [38]: training_features['Efficient Attempts'] = round((training_features['On-Target'] / training_features['Attempts'] * 100),
training_features.head(3)
```

Out[38]:

	Goal Scored	Attempts	On-Target	Off-Target	Blocked	Corners	Free Kicks	Pass Accuracy %	Fouls Committed	Man of the Match	Fouls Advantage	Efficient Attempts
0	5	13	7	3	3	6	11	78	22	1	-11	53.8
1	0	6	0	3	3	2	25	86	10	0	15	0.0
2	0	8	3	3	2	0	7	78	12	0	-5	37.5

```
In [39]: training_features['Fouls Advantage'] = training_features['Free Kicks']  - training_features['Fouls Committed']
training_features.head(3)
```

Out[39]:

	Goal Scored	Attempts	On-Target	Off-Target	Blocked	Corners	Free Kicks	Pass Accuracy %	Fouls Committed	Man of the Match	Fouls Advantage	Efficient Attempts
0	5	13	7	3	3	6	11	78	22	1	-11	53.8
1	0	6	0	3	3	2	25	86	10	0	15	0.0
2	0	8	3	3	2	0	7	78	12	0	-5	37.5

 # DELETE UNWANTED COLUMNS AFTER FEATURE ENGINEERING

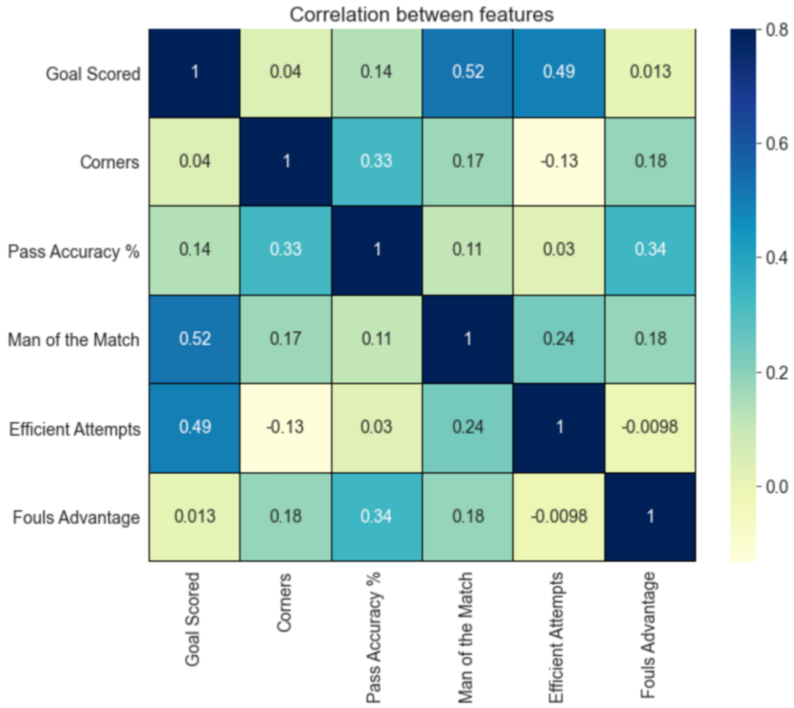
```
In [40]: training_features.drop(['On-Target', 'Off-Target', 'Blocked', 'Attempts', 'Free Kicks', 'Fouls Committed'], axis = 1, i
training_features.head(3)
```

Out[40]:

	Goal Scored	Corners	Pass Accuracy %	Man of the Match	Fouls Advantage	Efficient Attempts
0	5	6	78	1	-11	53.8
1	0	2	86	0	15	0.0
2	0	0	78	0	-5	37.5

Data Visualisation

In [12]:
```
corr = training_features.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr, vmax=.8, linewidths=0.01, square=False, annot=True, cmap='YlGnBu', linecolor='Black')
plt.title('Correlation between features');
```



 # Visualizing correlation between features using heatmap

Scaling/Normalization

It is basically Data transformation in data preprocessing. It is done to normalize range of independent variables or feature of data.

In [43]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(training_features[numeric_feature_names])
training_features[numeric_feature_names] = scaler.transform(training_features[numeric_feature_names])
training_features.head()
```

Out[43]:

	Goal Scored	Corners	Pass Accuracy %	Man of the Match	Fouls Advantage	Efficient Attempts
0	3.194193	0.525857	-0.770604	1	-1.955757	1.624259
1	-1.146112	-1.115843	0.582910	0	2.163711	-2.239935
2	-1.146112	-1.936693	-0.770604	0	-1.005111	0.453509
3	-0.278051	0.115432	0.582910	1	0.896182	-0.185735
4	-1.146112	0.115432	0.582910	0	-1.480434	-0.580774

SPLITTING DATA

Split training features into X & y using pandas)

```
In [44]: X = training_features.loc[:, training_features.columns != 'Man of the Match']
         y = training_features['Man of the Match']
```

Split X & y for training and testing

```
In [67]: from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
         print('Train dataset shape: {} {}'.format(X_train.shape, y_train.shape))
         print('Test dataset shape: {} {}'.format(X_test.shape, y_test.shape))

         Train dataset shape: (102, 5) (102,)
         Test dataset shape: (26, 5) (26,)
```

Split the data into

1. Training set (80%)
2. Test set (20%)

LEARNING ALGORITHMS

We used following learning models:

1. Logistic Regression
2. Decision Tree classifier
3. RandomForest classifier

```
In [68]: from sklearn.linear_model import LogisticRegression
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
         from sklearn.ensemble import RandomForestClassifier
```

Logistic Regression

Logistic regression is used for solving the classification problems.It is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.Since it predicts the output of a categorical dependent variable, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

```
In [69]: model_1 = LogisticRegression()
         model_1.fit(X_train, y_train)

Out[69]: LogisticRegression()
```

DECISION TREE CLASSIFIER

It is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In this there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

```
In [70]: model_2 = DecisionTreeClassifier()
         model_2.fit(X_train, y_train)

Out[70]: DecisionTreeClassifier()
```

RANDOM FOREST CLASSIFIER

- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

```
In [71]: model_3 = RandomForestClassifier()
         model_3.fit(X_train, y_train)

Out[71]: RandomForestClassifier()
```

MODEL PREDICTION AND EVALUATION

We precdicted Man of the match using all three models on Test dataset(X_test).

We evaluated our models by calculating **ACCURACY, PRECISION, RECALL** and **F1 SCORE.**

ACCURACY - The simplest way of reporting the effectiveness of an algorithm is by calculating its accuracy. Accuracy is the ratio of number of correct predictions to the total number of predictions made.
RECALL - Recall measures the percentage of relevant items that your classifier found.
PRECISION - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations
F1 SCORE -F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is , as well as how robust it is (it does not miss a significant number of instances).

Model evaluation scores

In [498]:
```
model_evaluation_data = {
    'Model' : models,
    'Accuracy' : accuracy_scores,
    'Precision' : precision_scores,
    'Recall' : recall_scores,
    'F1 score' : f1_scores
}

model_evaluation_df = pd.DataFrame(model_evaluation_data)

model_evaluation_df
```

Out[498]:

	Model	Accuracy	Precision	Recall	F1 score
0	Logistic Regression	0.846154	0.769231	0.909091	0.833333
1	Decision Tree	0.730769	0.666667	0.727273	0.695652
2	Random Forest	0.730769	0.642857	0.818182	0.720000

CONCLUSION

AFTER EXPERIMENTING WITH THE CLASSIFICATION MODELS, FOR FIFA 2018 TO PREDICT THE MAN OF THE MATCH AWARD, LOGISTIC REGRESSION MODEL PERFORMS THE BEST WITH

- ACCURACY = 84.60%
- F1 SCORE = 83.33%

THANK YOU