Abhijeet Kumar
201302197

CSE471: Statistical Methods in AI
Assignment #1: K Nearest Neighbor Classifier

DATASETS AND THEIR DESCRIPTIONS

IRIS DATA SET

Number of Features :: 4
Number of Instances :: 150
Number of Classes :: 3
Class Distribution :: Equal (33.33% for each)

Attribute(Feature + Class) Information ::
1) sepal length in cm
2) sepal width in cm
3) petal length in cm
4) petal width in cm
5) class:
   a) Iris Setosa :: 1
   b) Iris Versicolour :: 2
   c) Iris Virginica :: 3

POKER HAND DATA SET

Number of Features :: 10
Number of Instances :: 25010
Number of Classes :: 10

Class Distribution ::
0:Nothing in hand,12493instances(49.95202%/50.117739%)

1: One pair, 10599 instances, (42.37905%/42.256903%)
2: Two pairs, 1206 instances, (4.82207% / 4.753902%)
3: Three of a kind, 513 instances, (2.05118%/2.112845%)
4: Straight, 93 instances, (0.37185% / 0.392465%)
5: Flush, 54 instances, (0.21591% / 0.19654%)
6: Full house, 36 instances, (0.14394% / 0.144058%)
7: Four of a kind, 6 instances, (0.02399% / 0.02401%)
8: Straight flush, 5 instances, (0.01999%/0.001385%)
9: Royal flush, 5 instances, (0.01999%/0.000154%)

Attribute (Feature + Class) Information ::

1) S1 “Suit of card #1”
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

2) C1 “Rank of card #1”
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

3) S2 “Suit of card #2”
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

4) C2 “Rank of card #2”
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

5) S3 “Suit of card #3”
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

6) C3 “Rank of card #3”
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

7) S4 “Suit of card #4”
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

8) C4 “Rank of card #4”

Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

9) S5 "Suit of card #5"
Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

10) C5 "Rank of card 5"
Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

11) CLASS "Poker Hand"(NUMBERED FROM 0-9)

- Nothing in hand; not a recognized poker hand
- One pair; one pair of equal ranks within five cards
- Two pairs; two pairs of equal ranks within five cards
- Three of a kind; three equal ranks within five cards
- Straight; five cards, sequentially ranked with no gaps
- Flush; five cards with the same suit
- Full house; pair + different rank three of a kind
- Four of a kind; four equal ranks within five cards
- Straight flush; straight + flush
- Royal flush; {Ace, King, Queen, Jack, Ten} + flush

TIC-TAC-TOE DATA SET (ENDGAME)

Number of Features :: 9(positions on the the board)
Number of Instances :: 958
Number of Classes :: 3
Class Distribution :: Un-Equal
1:positive ::65.3%
2:negative ::24.7%

Attribute (Feature + Class) Information ::
(x=player x has taken, o=player o has taken, b=blank)

- top-left-square: {x,o,b}
- top-middle-square: {x,o,b}
- top-right-square: {x,o,b}
- middle-left-square: {x,o,b}
- middle-middle-square: {x,o,b}
- middle-right-square: {x,o,b}
- bottom-left-square: {x,o,b}
- bottom-middle-square: {x,o,b}
- bottom-right-square: {x,o,b}
- Class: {positive,negative}

BALANCE SCALE WEIGHT AND DISTANCE DATA SET

Number of Features :: 4
Number of Instances :: 625
Number of Classes :: 3
Class Distribution :: UnEqual

- Balanced :: 8%
- Left :: 46%
- Right :: 46%

Attribute(Feature + Class) Information ::

- Class Name: 3 (L, B, R)
- Left-Weight: 5 (1, 2, 3, 4, 5)
- Left-Distance: 5 (1, 2, 3, 4, 5)
- Right-Weight: 5 (1, 2, 3, 4, 5)
- Right-Distance: 5 (1, 2, 3, 4, 5)

SPECIFICATIONS IN CODEBASE

DISTANCE FUNCTIONS

- EULIDEAN
- MAHALANOBIS
- CITY-BLOCK
- SEUCLIDEAN

TARGET FUNCTION

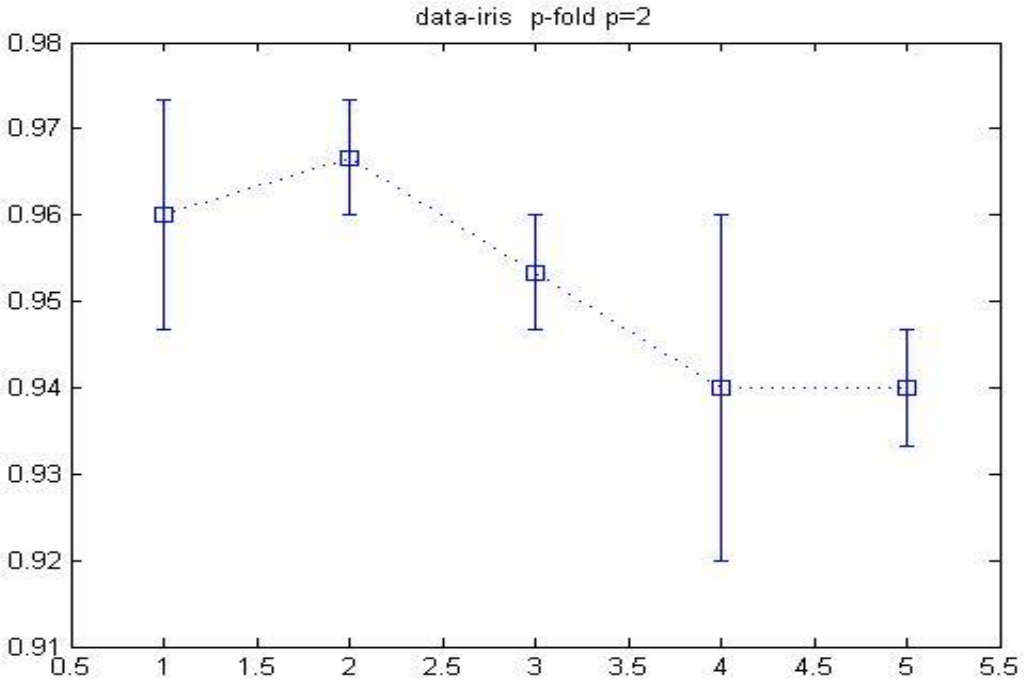
- CONTINUOUS
- INVERSE DISTANCE

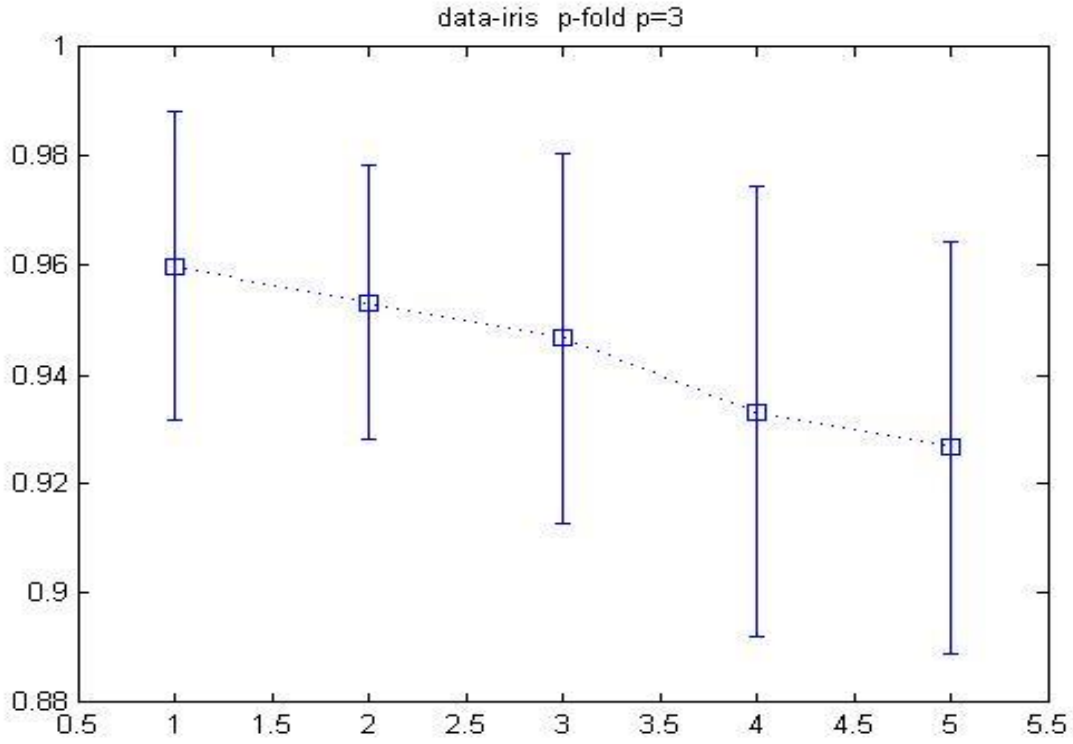
KFOLD EVALUATIONS :: 2-5
K IN KNN :: 1-5
TIE BREAK :: RANDOM
WEIGHT FUNCTION :: DEFAULT (All dimensions treated equally)

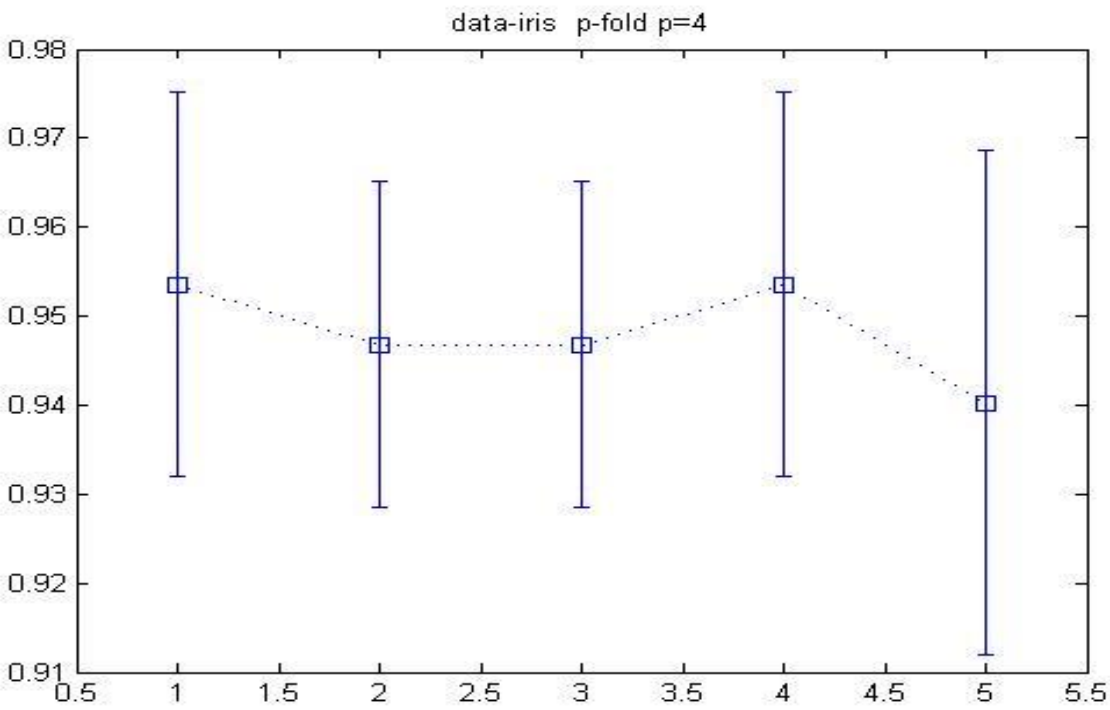
KERNEL METHODS :: NOT USED

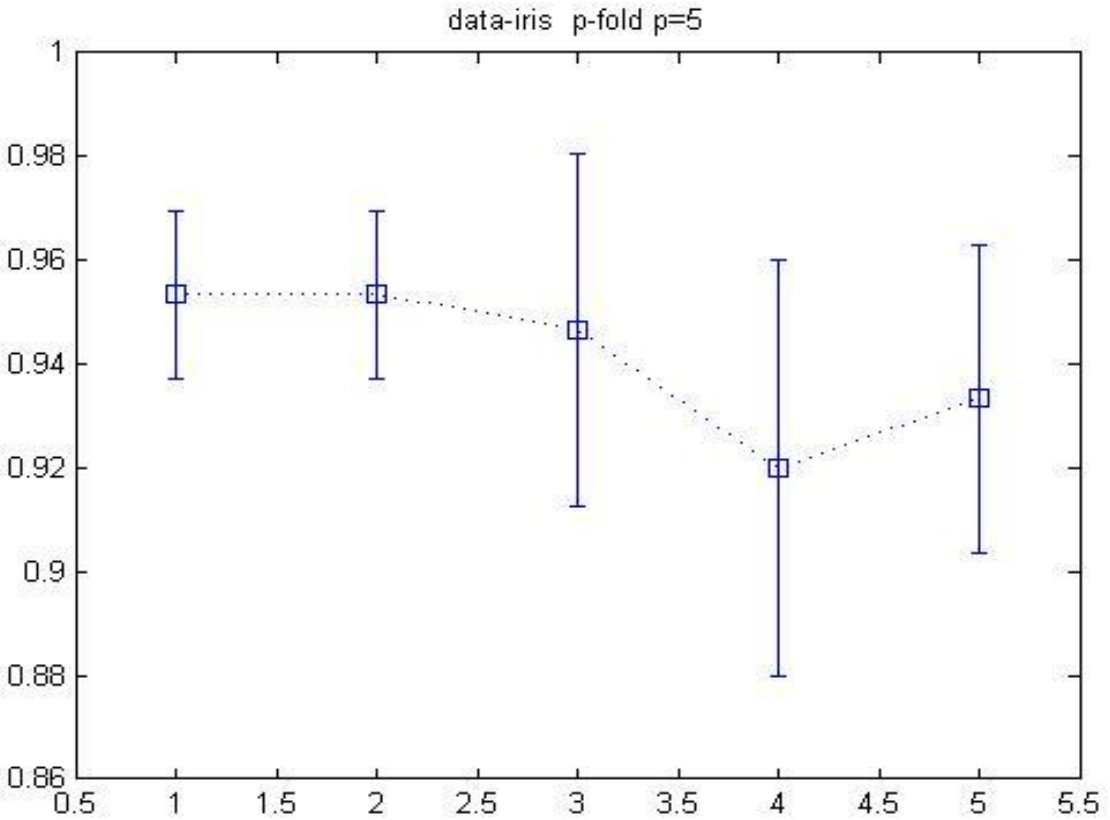
PLOTS

IRIS DATA (Euclidean distance function)

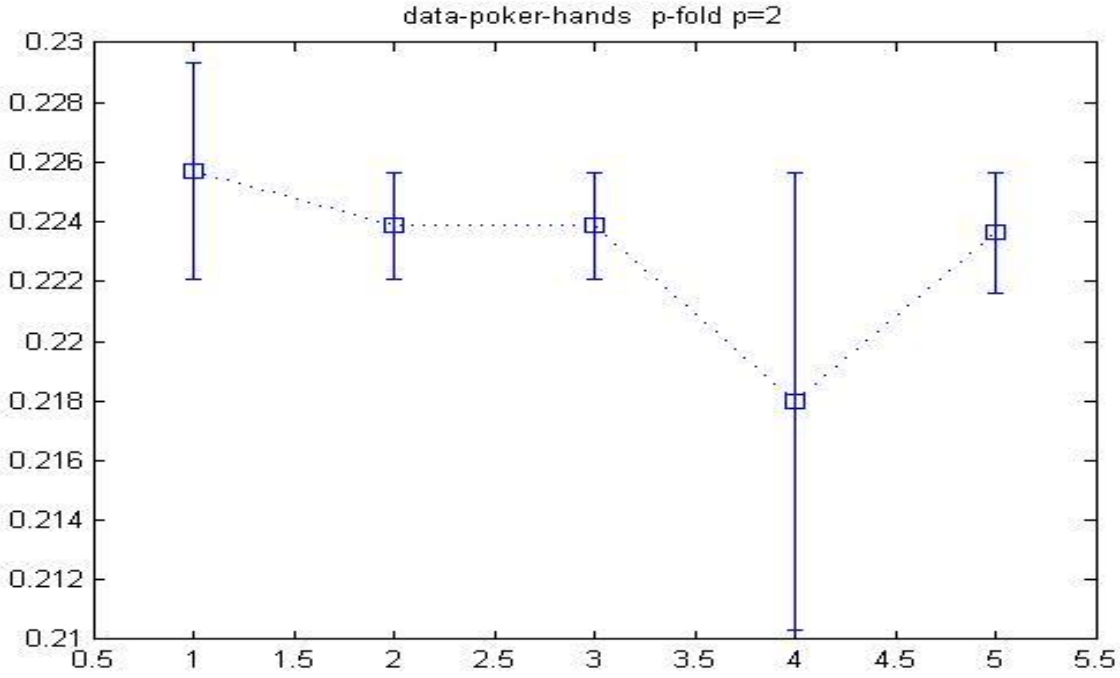


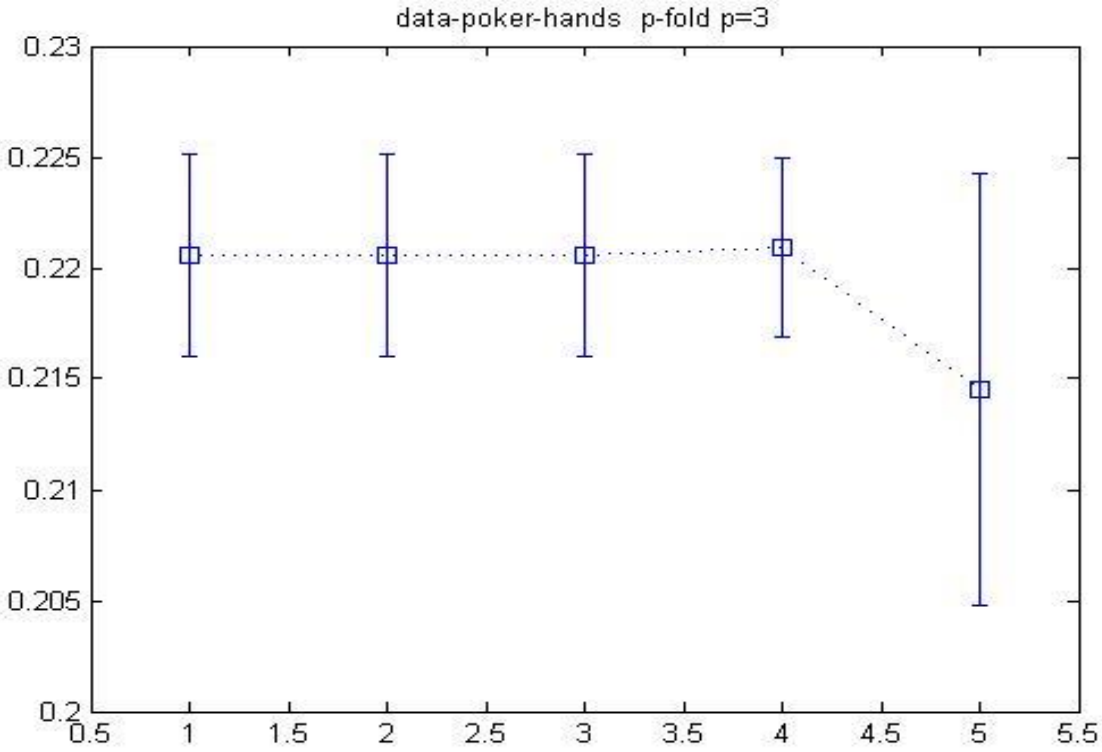


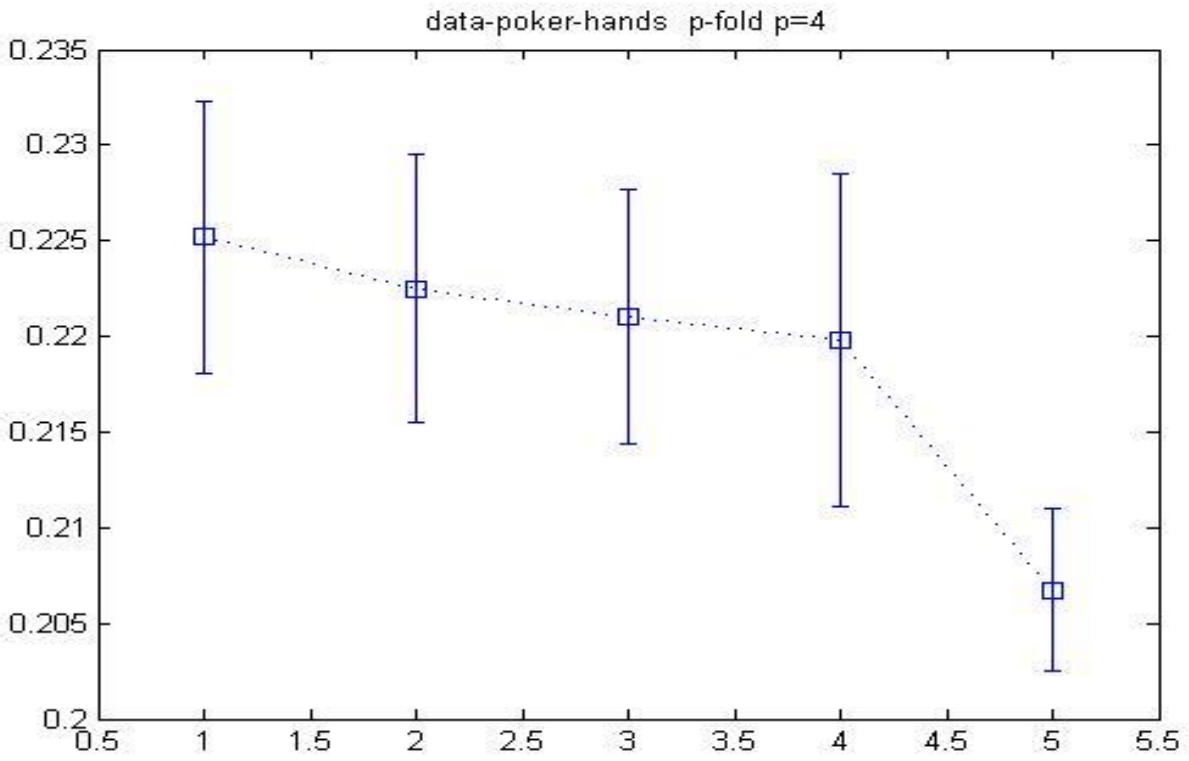

data-iris p-fold p=4
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

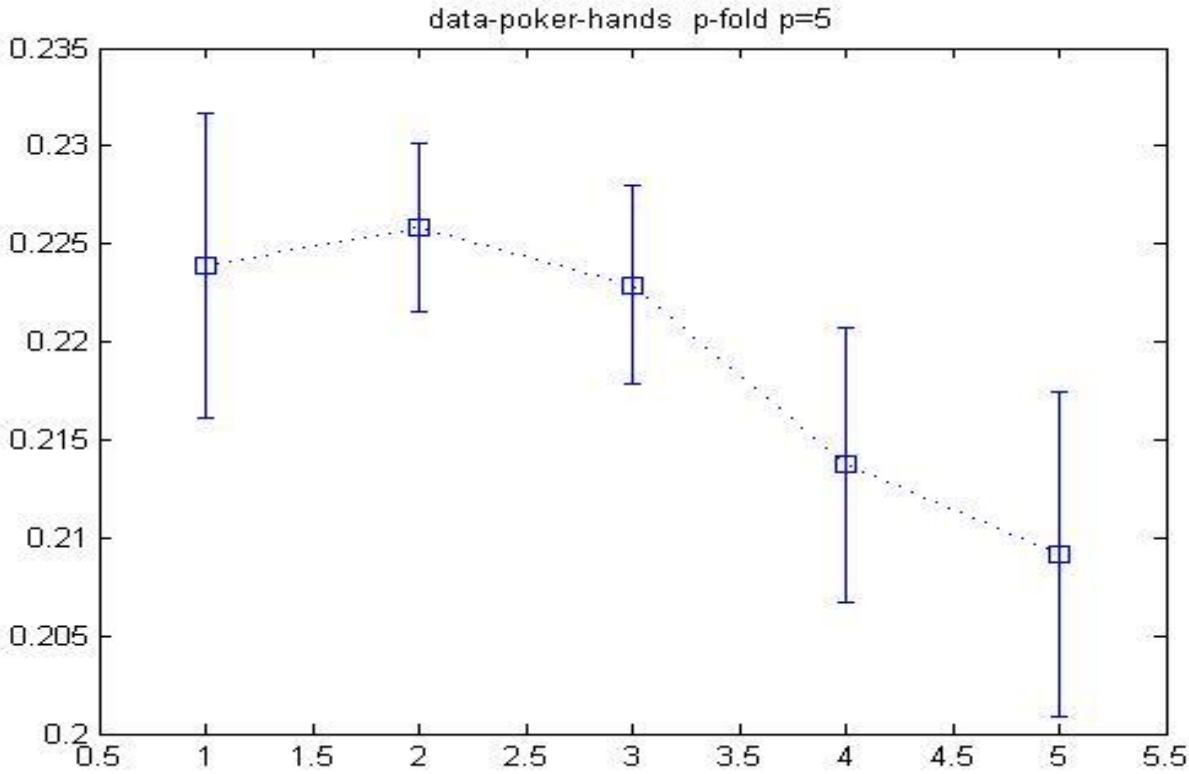

data-iris p-fold p=5
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

POKER DATA (Seuclidian distance function)

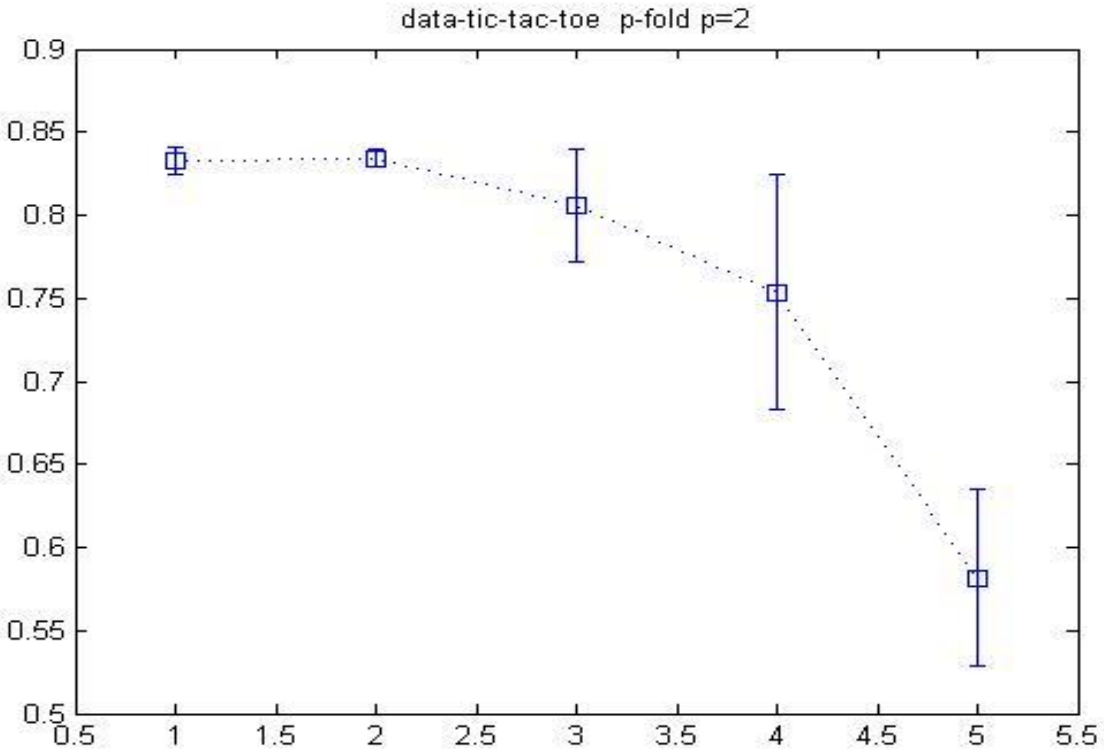


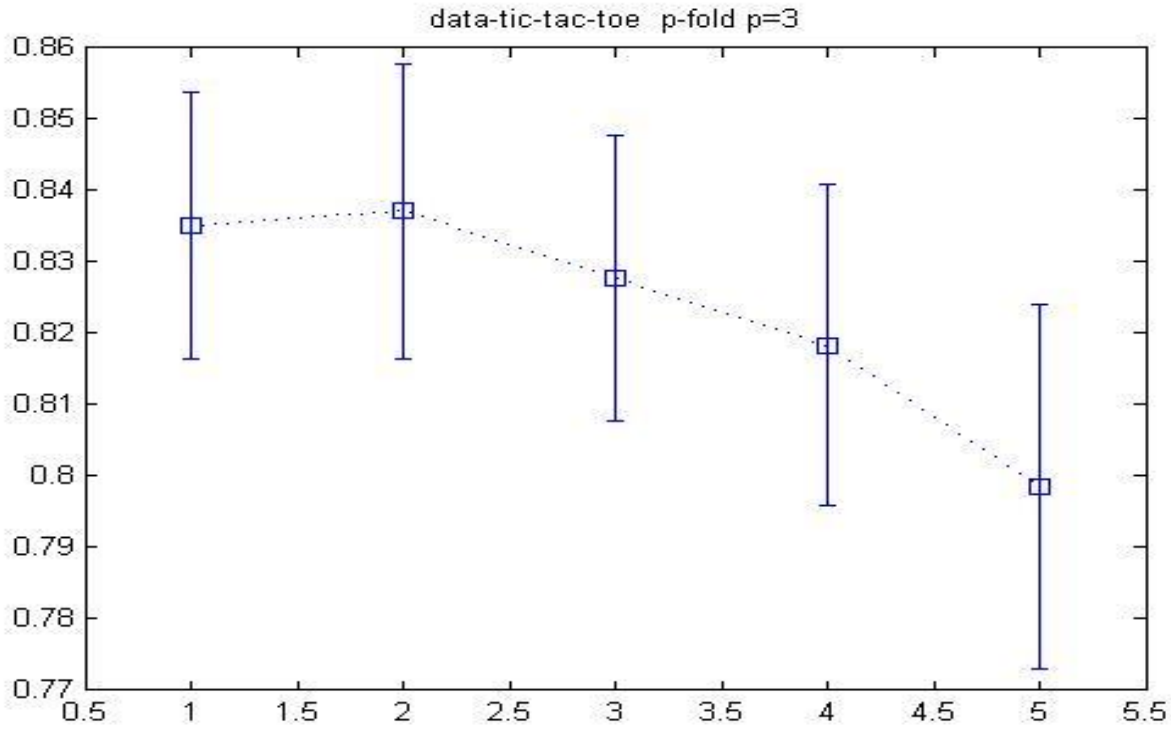


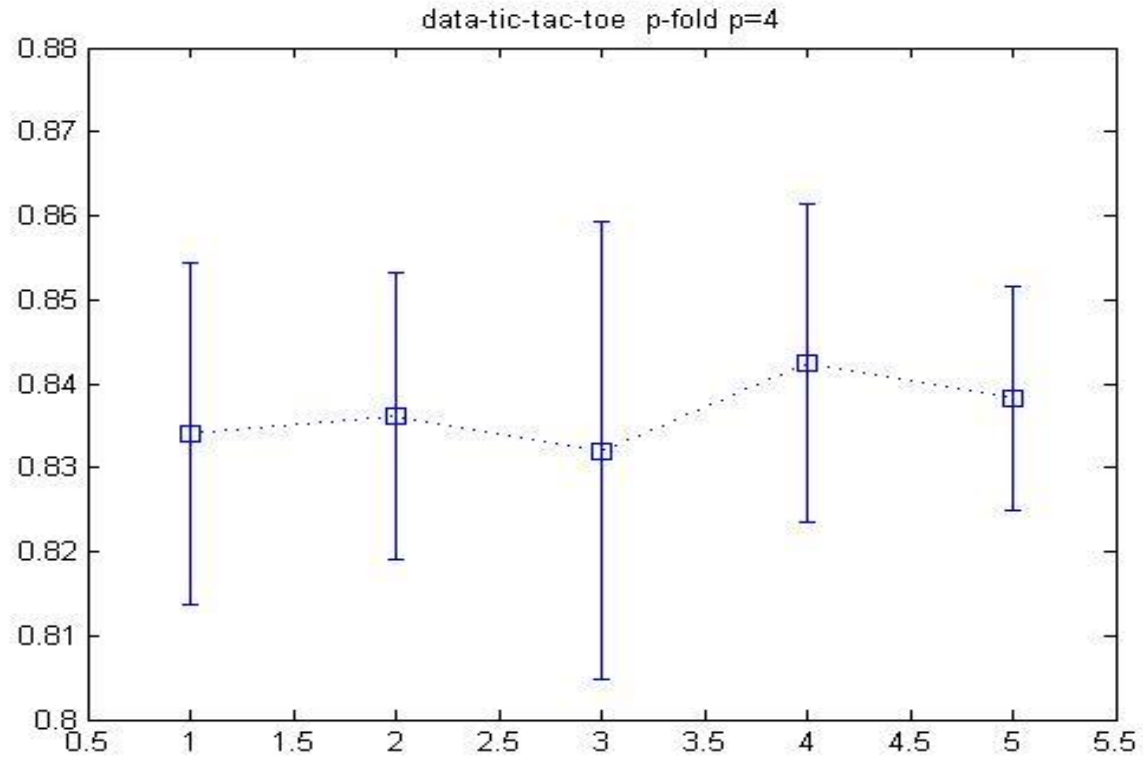

data-poker-hands p-fold p=4
0.235
0.23
0.225
0.22
0.215
0.21
0.205
0.2
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

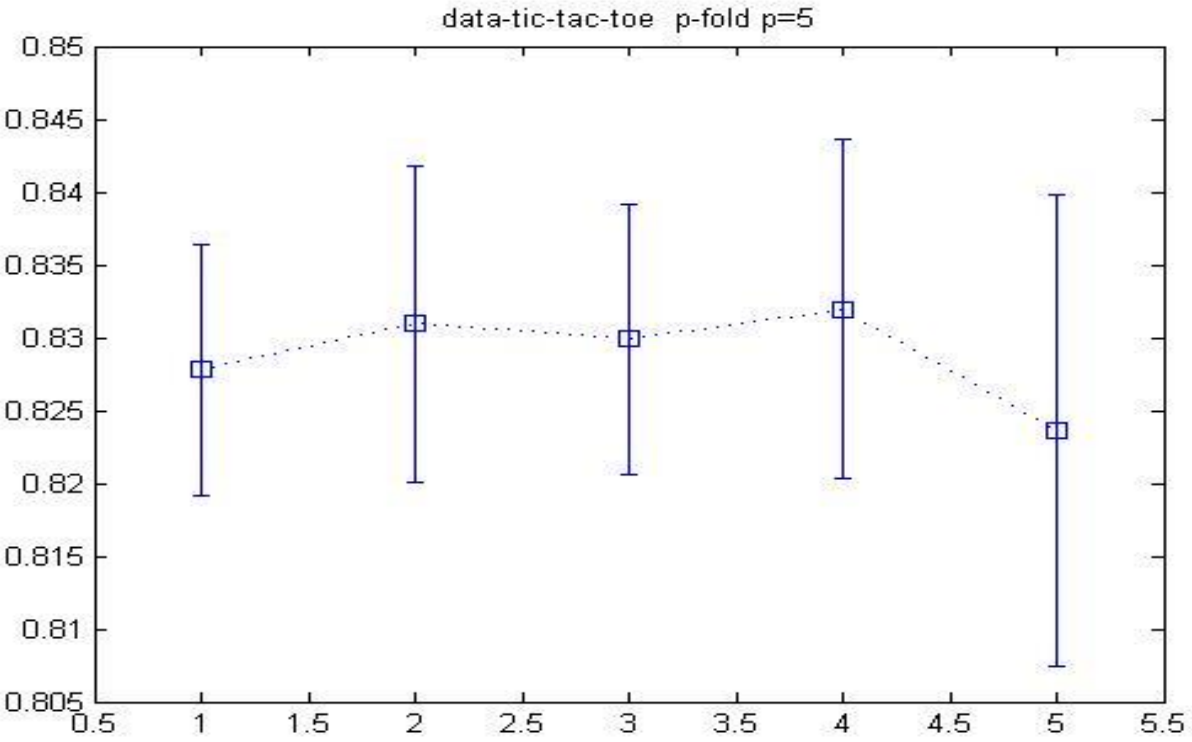

data-poker-hands p-fold p=5
0.235
0.23
0.225
0.22
0.215
0.21
0.205
0.2
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

TIC TAC TOE DATA (Mahalanobis Distance function)

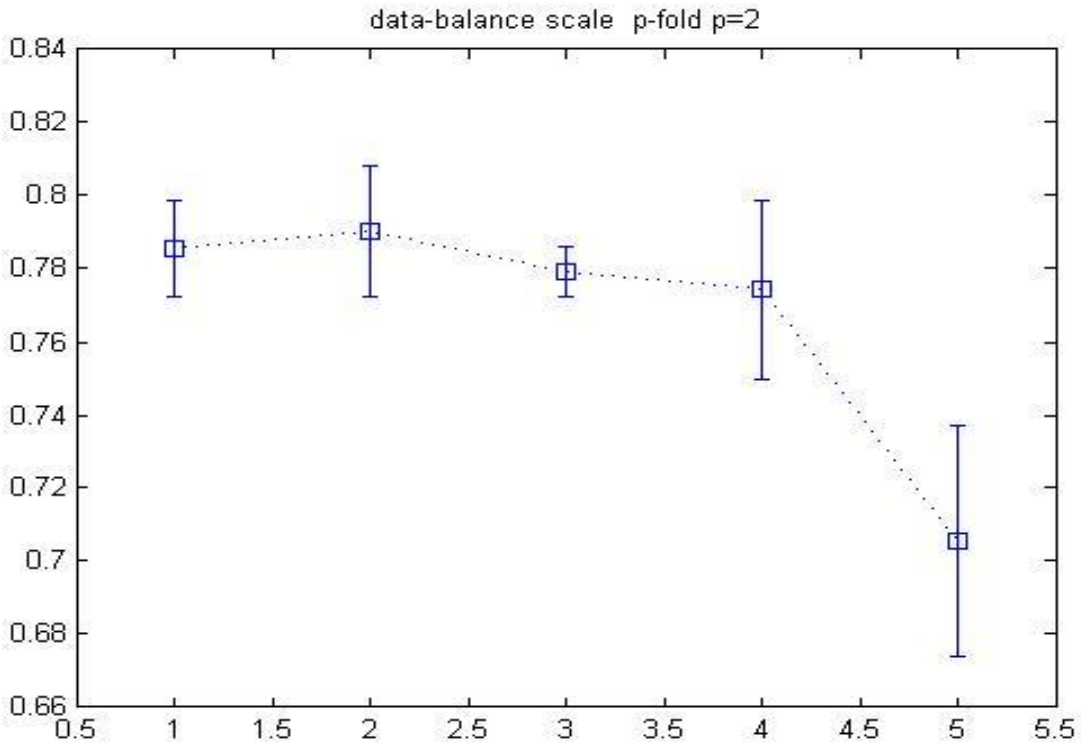


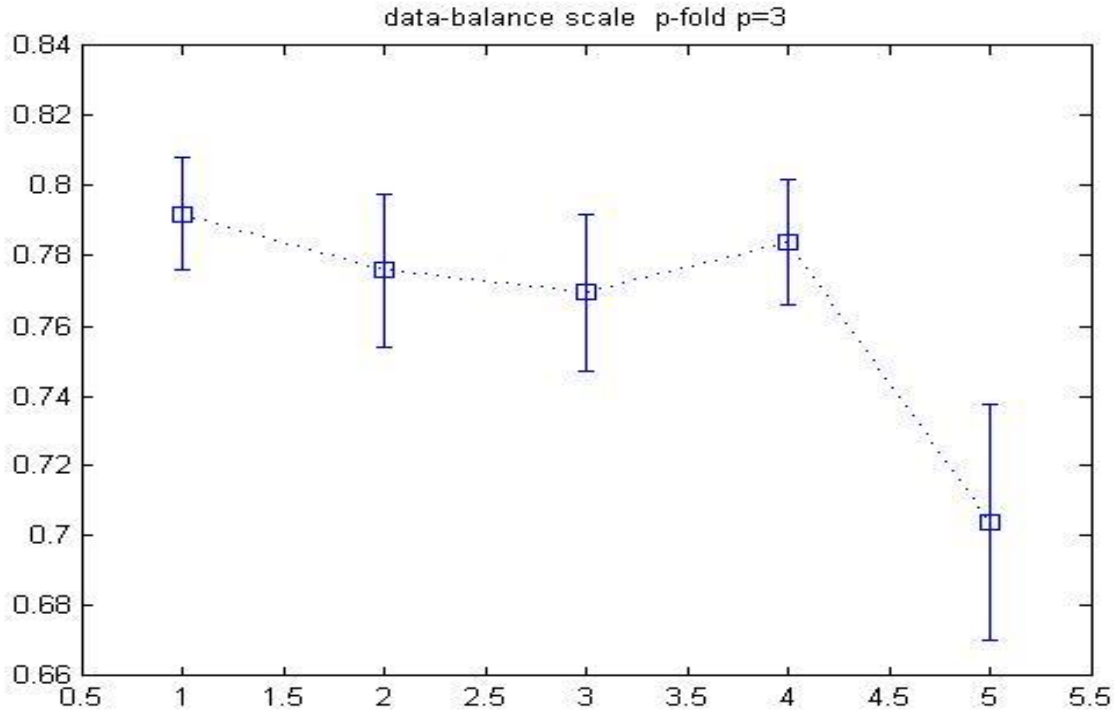


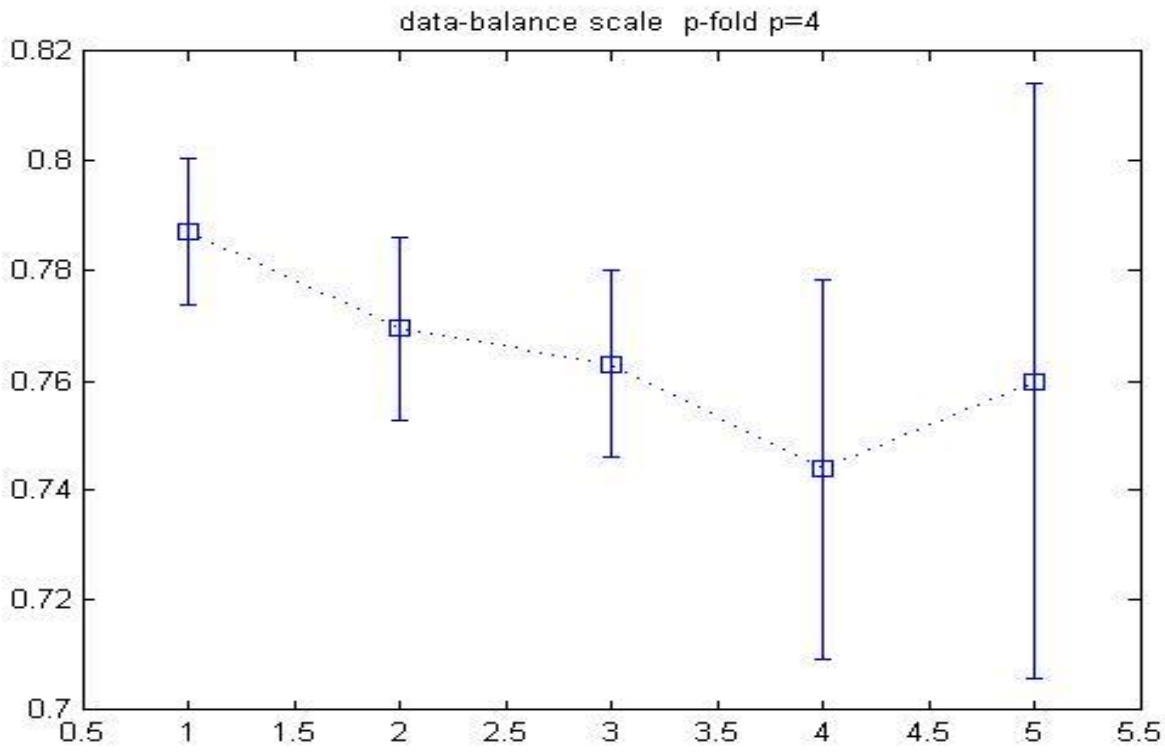

data-tic-tac-toe p-fold p=4
0.88
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.8
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

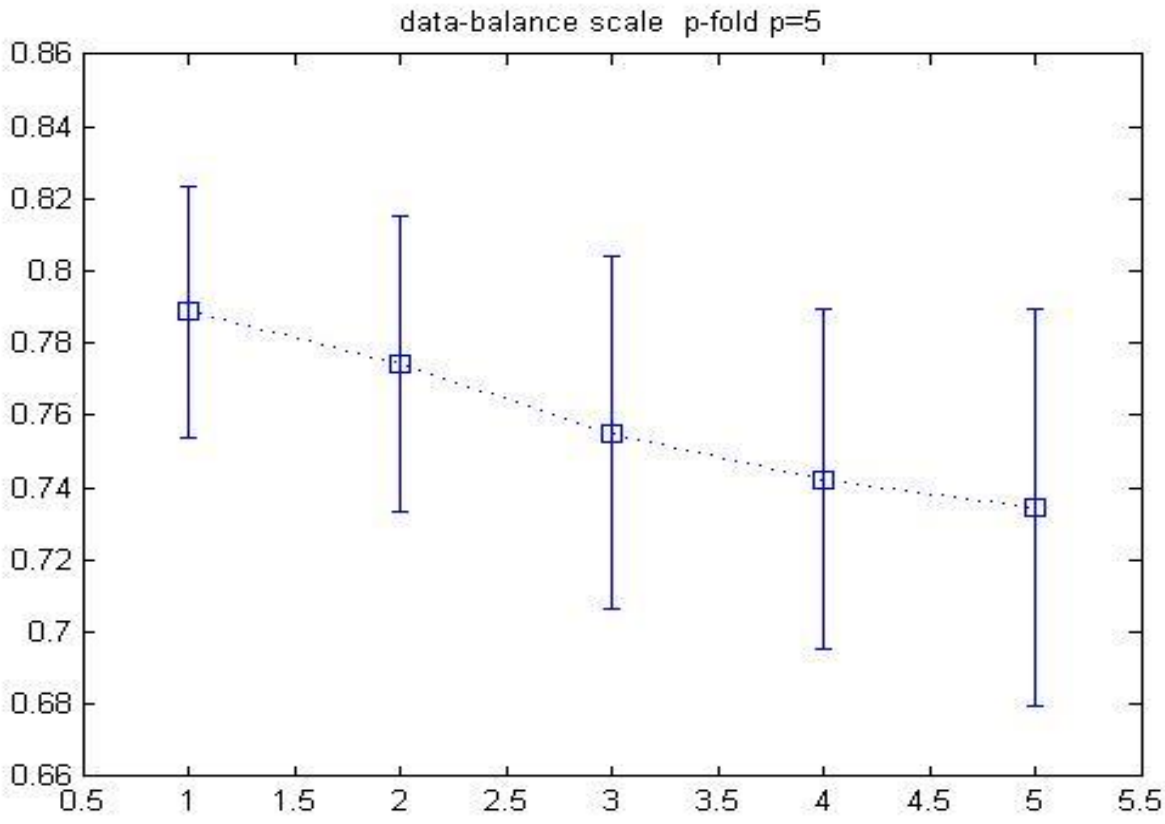

data-tic-tac-toe p-fold p=5
0.85
0.845
0.84
0.835
0.83
0.825
0.82
0.815
0.81
0.805
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

BALABCE AND WEIGHT DATA (Euclidean Distance function)






data-balance scale p-fold p=4
0.82
0.8
0.78
0.76
0.74
0.72
0.7
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5


data-balance scale p-fold p=5
0.86
0.84
0.82
0.8
0.78
0.76
0.74
0.72
0.7
0.68
0.66
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

RESULTS & SUMMARY

Distance Function and Variations

Further table description ::

	K=1	K=2	K=3	K=4	K=5
P=2					
P=3					
P=4					
P=5					

IRIS DATA

City-block

0.9333 0.9267 0.9333 0.9200 0.9133
0.9600 0.9333 0.9400 0.9400 0.9400
0.9465 0.9532 0.9399 0.9465 0.9330
0.9400 0.9333 0.9133 0.9200 0.9200

Min :0.91333
Max :0.96
Average :0.93429

Euclidean

0.9600 0.9600 0.9600 0.9400 0.9200
0.9667 0.9667 0.9533 0.9467 0.9267
0.9600 0.9602 0.9401 0.9399 0.9397
0.9533 0.9467 0.9267 0.9133 0.9200

Min :0.91333
Max :0.96667
Average :0.94499

Mahalanobis

0.8400	0.8400	0.8400	0.7933	0.7933
0.8667	0.8800	0.8667	0.8533	0.7667
0.8860	0.8658	0.8793	0.8860	0.8062
0.9200	0.9067	0.8800	0.8600	0.8200

Min :0.76667
Max :0.92
Average :0.8525

Seuclidean

0.9267	0.9200	0.8667	0.8533	0.8333
0.9467	0.9400	0.9200	0.9067	0.8600
0.9403	0.9403	0.9068	0.9001	0.9001
0.9400	0.9400	0.9267	0.9333	0.9067

Min :0.83333
Max :0.94667
Average :0.91037

TIC-TAC-TOE DATA

City-block

0.7484	0.7370	0.7140	0.7599	0.7422
0.7192	0.7192	0.7140	0.7036	0.7192
0.7161	0.7161	0.7140	0.7067	0.7140
0.7045	0.7045	0.7035	0.6983	0.7274

Min :0.69828
Max :0.75992
Average :0.71909

Euclidean

0.7484	0.7443	0.7296	0.7338	0.7495
0.7202	0.7182	0.7182	0.7119	0.7265
0.7151	0.7151	0.7140	0.7098	0.7161

0.7088 0.7088 0.7067 0.7057 0.7151

Min :0.70568
Max :0.74948
Average :0.72079

Mahalanobis

0.8299 0.8330 0.8194 0.7651 0.8205
0.8372 0.8414 0.8340 0.8309 0.8152
0.8382 0.8392 0.8392 0.8382 0.8299
0.8361 0.8351 0.8330 0.8288 0.8351

Min :0.76514
Max :0.84135
Average :0.82897

Seuclidean

0.8006 0.7839 0.7704 0.7025 0.6096
0.7933 0.7881 0.7819 0.7537 0.6983
0.8069 0.8058 0.7828 0.7484 0.7233
0.7975 0.7964 0.7714 0.7494 0.7316

Min :0.6096
Max :0.80687
Average :0.7598

POKER DATA

City-block

0.2117 0.2154 0.2138 0.2102 0.2015
0.2124 0.2125 0.2179 0.2061 0.2092
0.2096 0.2350 0.2150 0.2157 0.2082
0.2095 0.2277 0.2147 0.2090 0.2033

Min :0.20148
Max :0.23503
Average :0.21292

Euclidean

0.2116	0.2129	0.2093	0.2082	0.2024
0.2081	0.2100	0.2100	0.2057	0.2030
0.2104	0.2124	0.2156	0.2099	0.2045
0.2088	0.2113	0.2117	0.2049	0.2054

Min :0.2024
Max :0.21559
Average :0.20879

Mahalanobis

0.2246	0.2264	0.2268	0.2190	0.2073
0.2240	0.2224	0.2184	0.2180	0.2146
0.2240	0.2254	0.2269	0.2231	0.2119
0.2259	0.2244	0.2242	0.2167	0.2144

Min :0.20732
Max :0.22687
Average :0.22092

Seuclidean

0.2237	0.2248	0.2262	0.2262	0.2262
0.2245	0.2250	0.2177	0.2159	0.2076
0.2246	0.2262	0.2255	0.2230	0.2162
0.2213	0.2205	0.2216	0.2156	0.2114

Min :0.2076
Max :0.22623
Average :0.22119

BALANCE AND WEIGHT DATA

City-block

0.7984	0.7744	0.7744	0.7024	0.6624

0.7904 0.7824 0.7824 0.7921 0.7488
0.7888 0.7776 0.7824 0.7664 0.7487
0.7840 0.7648 0.7648 0.7376 0.7312

Min :0.66241
Max :0.79841
Average :0.76273

Euclidean

0.7968 0.7760 0.7344 0.7104 0.6976
0.7744 0.7632 0.7552 0.7488 0.7087
0.7872 0.7712 0.7568 0.7696 0.7361
0.7824 0.7600 0.7584 0.7472 0.7088

Min :0.69761
Max :0.79683
Average :0.75217

Mahalanobis

0.7648 0.7648 0.7264 0.7216 0.7216
0.7712 0.7776 0.7744 0.7712 0.7455
0.7472 0.7505 0.7393 0.7553 0.7393
0.7760 0.7776 0.7600 0.7808 0.7408

Min :0.72159
Max :0.7808
Average :0.75528

Seuclidean

0.7744 0.7744 0.7296 0.7200 0.7200
0.7520 0.7680 0.7520 0.7072 0.7183
0.7665 0.7713 0.7681 0.7248 0.6960
0.7760 0.7744 0.7472 0.7872 0.7664

Min : 0.69604
Max : 0.7872
Average: 0.74969

GENERAL

- HIGHER K IN KNN MOVES AWAY FROM REALITY
- K FOLD EVALUATION :: PROVIDES MORE GENERIC RESULTS
- OVERFITTING TO THE PARTICULAR TRAINING DATA
- WRONG CHOICE OF DISTANCE FUNCTION MAY PROVE TO BE FUTILE
- HIGH CLASSIFICATION TIME
- FAST TRAINING

DATA SPECIFIC

IRIS DATA

- VERY GOOD RESULT WITH ALL DISTANCE FUNCTION
- SIMPLE DOMAIN
- DATA EQUALLY REPSENTED
- LINEAR SEPRABILITY OF ONE CLASS FROM TWO IN EUCLIDEAN SPACE IS KNOWN
- CITY-BLOCK AND EULIDEAN DISTANCE FUNCTIONS ARE THE BEST DISTANCE FUNCTIONS

TIC-TAC-TOE DATA

- GOOD RESULTS
- HARD TO CONVERT THE PROBLEM TO KNN AS DEFINNING DISTANCE BEWEEN TWO T's WHERE T ={'X','O',B'} IS DIFFICULT
- USE OF DIFFERENT DISTANCE FUNCTIONS MAKES MUCH MORE SENSE HERE
- UNEQUAL DATA REPRSENTAION
- VARIATION IN RESLULTS WITH DIFFERENT DISTANCE FUNCTIONS
- MAHALANOBIS DISTANCE EASILY OUTPERFORMS OTHER FUNCTIONS
- COMPLEX TO UNDERSTAND AS SPACE DIVISION PROBLEM OR TO REALIZE LDF's

POKER DATA

- RESULT IS BAD USING ANY TYPE OF DISTANCE FUNCTION
- A SINGLE HAND SUCH AS FULL HOUSE CAN HAS A LOT OF POSSIBLE CASES
- A LOT OF MATCHING TENDENCY(SIMILARITY) WITH OTHERS SUCH AS FOUR OF A KIND ,THREE OF A KIND TWO PAIRS
- ORIGINAL DATA IS BIGGER AND HERE ONLY A PART OF IT IS USED
- DATA IS HIGHLY UNEQUALLY REPRESENTED
- ALTHOUG ALL DISTANCE FUNCTION BEHAVE VERY POORLY SECLUDIAN SEEMS TO HAVE A LIITLE ADVANTAGE ON THE OTHERS

BALANCE AND WEIGHT DATA

- GOOD RESULTS
- SIMPLE PHYSICS PROBLEM
- HARD TO SEPRATE IN SIMPLE KERNEL SPACES(SEEMS INTUTIVE)
- ALL DISTANCE FUNCTIONS PERFORM EQUALLY WELL
- CLASS DISTRIBUTION IN ACTUAL DATA IS UNEQUAL BUT STILL REPRESTS THE DATA FINE ENOUGH

CodeBase(Matlab)

```
% main caller function for each data set
data_iris;
data_poker_hands;
data_tic_tac_toe;
data_balance_scale;


%iris data
%preprocessing
file_id=fopen('Iris.data.txt');
c=textscan(file_id,'%f %f %f %f %s','delimiter',',');
fclose(file_id);

data=zeros(150,4);
for i=1:4
    data(:,i)=c{i};
end

results=c{5};
results ( find ((strcmp(results,'Iris-setosa'))==1))=
mat2cell(['1']);
results ( find ((strcmp(results,'Iris-
versicolor'))==1))= mat2cell(['2']);
results ( find ((strcmp(results,'Iris-
virginica'))==1))= mat2cell(['3']);

results=str2num(cell2mat(results));


%results on - folfd knn  using the custom myknn
function
[mean,deviation]=myknn(data,results,5,5);
```

```
%draw plots
for i=1:4
figure
errorbar( mean(i,:) , deviation(i,:) , ':bs');
title (strcat('data-iris  p-fold p=',num2str(i+1)) );
end

clear c data results

%poker hands data
%preprocessing
file_id=fopen('poker-hand-training-true.data.txt');
c=textscan(file_id,'%d %d %d %d %d %d %d %d %d %d
%d','delimiter',',');
fclose(file_id);

data=zeros(25010,10);
for i=1:10
    data(:,i)=c{i};
end

results=c{11};

%results on - folfd knn  using the custom myknn
function
[mean,deviation]=myknn(data,results,5,5);

%draw plots
for i=1:4
figure
errorbar( mean(i,:) , deviation(i,:) , ':bs');
title (strcat('data-poker-hands  p-fold
p=',num2str(i+1)) );
end

clear c data results


%tic-tac-toe data
%preprocessing
file_id=fopen('tic-tac-toe.data.txt');
```

```
c=textscan(file_id,'%c %c %c %c %c %c %c %c %c
%s','delimiter',',');
fclose(file_id);

data=zeros(958,9);
for i=1:9
    data(:,i)=c{i};
end
data ( data =='x')=1;
data ( data =='o')=2;
data ( data =='b')=3;

results=c{10};
results ( find
((strcmp(results,'positive'))==1))=mat2cell(['1']);
results ( find
((strcmp(results,'negative'))==1))=mat2cell(['2']);

results=str2num(cell2mat(results));

%results on - folfd knn  using the custom myknn
function
[mean,deviation]=myknn(data,results,5,5);

%draw plots
for i=1:4
figure
errorbar( mean(i,:) , deviation(i,:) , ':bs');
title (strcat('data-tic-tac-toe  p-fold
p=',num2str(i+1)) );
end

clear c data results


%balance scale data
%preprocessing
file_id=fopen('balance-scale.data.txt');
c=textscan(file_id,'%s %f %f %f %f','delimiter',',');
fclose(file_id);

data=zeros(625,4);
```

```
for i=1:4
    data(:,i)=c{i+1};
end

results=c{1};
results ( find ((strcmp(results,'R'))==1))=
mat2cell(['1']);
results ( find ((strcmp(results,'B'))==1))=
mat2cell(['2']);
results ( find ((strcmp(results,'L'))==1))=
mat2cell(['3']);

results=str2num(cell2mat(results));


%results on - folfd knn  using the custom myknn
function
[mean,deviation]=myknn(data,results,5,5);

%draw plots
for i=1:4
figure
errorbar( mean(i,:) , deviation(i,:) , ':bs');
title (strcat('data-balance scale  p-fold
p=',num2str(i+1)) );
end

clear c data results


%%  p fold k-nn classification %%

%input %
%caution -- all inputs shold be of type double do the
necessary pre-processing
%data - n*d matrix : n sample points , d dimensions of
a sample point
%gt - n*1 mattrix : ground truth for each corresponding
data point
% p_max - 2-p folds
% k_max - k-nearesrt
```

```
%output%

function
[accuracy,deviation]=myknn(data,gt,k_max,p_max)

[n,d]=size(data);
accuracy=zeros(p_max,k_max);
deviation=zeros(p_max,k_max);

unique_gt=unique(gt);

%partition set via kfold - test and training  ::  data
and gt
for i=2:p_max
    c=cvpartition(n,'KFold',i);
    observed_result=zeros(k_max,c.NumTestSets);
    for j=1:c.NumTestSets
        training_data = data ( find ( c.training(j) ) ,
: ) ;
        test_data = data ( find ( c.test(j) ) , : ) ;
        gt_training_data = gt ( find ( c.training(j) )
, : );
        gt_testdata = gt ( find ( c.test(j) ) ,: );

        % k_max minimum distances of each test point
from all the training points

[distance,index]=pdist2(training_data,test_data,'euclid
ean','smallest',k_max);
        index=index.';
        distance=distance.';
        prediction=gt_training_data(index);

        % loop in k
        for k=1:k_max
            temp_prediction=prediction(:,1:k);
            temp_distance=distance(:,1:k);
            %normalizing the distances in each row
            temp_distance=temp_distance./repmat(
sum(temp_distance,2),1,k );
```

```
predict_mat=zeros(c.TestSize(j),size(unique_gt,1));

predict_mat2=zeros(c.TestSize(j),size(unique_gt,1));
            % predict and compare results
            for s=1:size(unique_gt,1)
                %use normalized distance here

predict_mat(:,s)=histc(temp_prediction,unique_gt(s),2);
                for t=1:c.TestSize(j)
                    r=find ( temp_prediction(t,:) ==
unique_gt(s) & temp_prediction(t,:) >=0 );
                    if size(r,2) ~= 0
                        predict_mat2(t,s)=sum (
temp_distance ( r ) ) /size(r,2);
                    else
                        predict_mat2(t,s)=Inf;
                    end
                end
            end
            %predict_mat
            %predict_mat2
            %prediction based on

[max_val,final_predicted_k_result]=min(predict_mat2,[],
2);

            % metrics data storage%

correct_predicted=size(find(gt_testdata==final_predicte
d_k_result),1);
            wrong_predicted=c.TestSize(j)-
correct_predicted;

observed_result(k,j)=correct_predicted/c.TestSize(j);
clear temp_prediction temp_distance predict_mat
predict_mat2 max_val final_predicted_k_result
        end

        clear training_data test_data gt_training_data
gt_test_data index distance prediction
```

```
    end
    % observed_result
    % metric calculation for ith fold %
        accuracy(i,:) = mean(observed_result,2);

        deviation(i,:)= std(observed_result,1,2);
end
accuracy = accuracy(2:p_max,:);
deviation = deviation(2:p_max,:);
```