EKT Module Test

Zach Wolpe u15094813

Question 1: Monte Carlo Integration

1.1)

Area under the curve: $Area = 65183.019$

1.2)

The maximum value of $P_t = 5160$, the equation does not exceed this value.

1.3)

$\alpha = 5160$ is the initial population size (number of people)
$\beta = 0.8$ is the rate of decay of the growth rate
$\gamma = -0.1$ is the continuous (negative) growth rate of the population

Question 2: KNN modelling

2.1)

$Y \in [1,2]$ therefore 2 unique elements

2.2)

See code

2.3)

Optimal number is $K =$

& See code

2.2)

$\{0.6693767; 0.9058344\}$

& See code

SAS code: Question 1

```
* ____________________ Question 1: Monte Carlo Integration ________________________;


proc iml;
title "Question 1: Monte Carlo Integration";
title2 "question 1.1";
start function(t);
	fx = 5160*(1 - (0.8*exp(-0.1*t)));
	return fx;
finish function;


start MC_integration(function,x1,x2,iters);

	t = do(x1,x2,0.05)`;
	y = function(t);

	y2 = max(y) + 20;
	y1 = 0;

	* question 1.1;

	do i=1 to iters;
		sample_y =  (y2 - y1)*rand('uniform');
		sample_x = x1 + (x2 - x1)*rand('uniform');

		* evaluate point;
		fx = function(sample_x);
		if sample_y < fx then; results = results // (sample_x || sample_y ||1);
		if sample_y > fx then; results = results // (sample_x || sample_y ||0);
	end;

	total_area = (x2-x1)*(y2-y1);

	q1_1 = mean(results[,3])*total_area;
	print q1_1;
finish MC_integration;

call MC_integration(function,2,20,10000);


title "Question 1: Monte Carlo Integration";
title2 "question 1.2";

t = do(200,400,0.05)`;
y = function(t);
max = max(y);

tt= t||y||J(nrow(t), 1, max);
```

```
print max;

create xy from tt[colname={'x' 'y' 'max'}];
append from tt;

print "Based on the functional form it is clear that the growth rate is
        maximized when the second terms is zero;
        The max value for y is Y=5160";

proc sgplot data=xy;
        series x=x y=y;
        refline max / axis=x;
```

SAS code: Question 2

```
FILENAME REFFILE '/folders/myfolders/sasuser.v94/EKT 720/EKT module test/data/q2.csv';

PROC IMPORT DATAFILE=REFFILE
        DBMS=CSV
        OUT=WORK.q2;
        GETNAMES=YES;
RUN;


proc sgplot data=q2;
        scatter x=x1 y=x2 / group=y;
run;

proc print data=q2 (obs=10);



proc iml;
use q2;
read all into xy;
n=nrow(xy);
x1=xy[,1];
x2=xy[,2];
y=xy[,3];

print n;
```

```
* question 2.1;
print (unique(y));
print "Number of unique parameters" (ncol(unique(y)));



start knn(xy,xy_train,k);

n_train = nrow(xy_train); n=nrow(xy);

* Compute Distances;
do i=1 to n;
        dist = J(n_train,1,888);
        do j=1 to n_train;
                dist[j] = sqrt((xy[i,1]-xy_train[j,1])**2 + (xy[i,2]-xy_train[j,2])**2);
        end;
        dis = dis || dist;
end;

* predict;
pred = J(n,2,888);
do i=1 to n;
        p = dis[,i]||xy_train[,3];
        call sort(p,{1});
        freq = p[1:k,2];
        weight = 1/p[1:k,1];

        a=0; b=0;
        do j=1 to k;
                if freq[j]=1 then;
                        if weight[j]^=. then; a=a+weight[j];
                if freq[j]=2 then;
                        if weight[j]^=. then; b=b+weight[j];
        end;
        pred[i,] = a || b;
end;


do i=1 to nrow(pred);
        yh = yh // pred[i,][<:>];
end;

return yh;
finish knn;


start accuracy(yh,y);
        a=0;
        do i=1 to nrow(yh);
                if yh[i]=y[i] then; a=a+1;
        end;
        a = a/nrow(yh);
        return a;
finish accuracy;




title2 "question 2.2";
```

```
xy_train = xy;
yh = knn(xy, xy_train,3);
accuracy= accuracy(yh,xy[,3]);
print accuracy;




title2 "question 2.3";
xy_o = xy;
train = round(uniform(J(n,1,1))*10);

*acct = J(nrow(xy), 10, 888);

scores = J(100,10,0);
do i=1 to 100; * for each possible k;
    acc = J(10,1,0);
    do f=1 to 10; * each K-Folds;
        xy = xy_o || train;
        xy_train = xy[loc(xy[,4]^=f)`,];
        xy_test = xy[loc(xy[,4]=f)`,];
        yh = knn(xy_test, xy_train,i);
        *acc[f,1] = accuracy(yh,xy_test[,3]);
        scores[i,f] = accuracy(yh,xy_test[,3]);
    end;

end;
create accuracy_table from scores;
append from scores;




proc iml;
use accuracy_table;
read all into scores;


do i=1 to nrow(scores);
    ks = ks // (sum(scores[i,])/ncol(scores));
end;

* best k;
in = ks[<:>];
print in;

print (mean(scores[20,]`));


* slice best K;
scor = scores[in,]`;


* CI;
upper = mean(scor) + 1.96*std(scor);
low = mean(scor) - 1.96*std(scor);
```

```
print low upper;
```