Cory Suzuki

STAT 574

Dr. Olga

17 March 2025

Homework 3

Problem 1

SAS

```
proc import out=cardtrans
 datafile="C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.csv" dbms=csv
 replace;
 /*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
 proc surveyselect data=cardtrans rate=0.8 seed=210925
 out=cardtrans outall method=srs;
 run;
 data train (drop=selected);
 set cardtrans;
 if selected=1;
 run;
 data test (drop=selected);
 set cardtrans;
 if selected=0;
 run;
 /*COMPUTING PRIOR PROBABILITIES*/
 proc freq data=train noprint;
 table fraud/out=priors;
 run;
 data priors;
 set priors;
 percent=percent/100;
 if fraud=0 then
 call symput('prior_no', percent);
 if fraud=1 then
 call symput('prior_yes', percent);
 run;
 /*COMPUTING POSTERIOR PROBABILITIES FOR CATEGORICAL PREDICTORS*/
 proc freq data=train noprint;
 table fraud*repeat_retailer/out=repeat_retailer_perc
 nocum list;
 run;
```

```
data repeat_retailer_perc;
set repeat_retailer_perc;
percent=percent/100;
if fraud=0 and repeat_retailer=0 then
call symput('repeat_retailer_no', percent);
if fraud=0 and repeat_retailer=1 then
call symput('repeat_retailer_yes', percent);
if fraud=1 and repeat_retailer=0 then
call symput('repeat_retailer_no', percent);
if fraud=1 and repeat_retailer=1 then
call symput('repeat_retailer_yes', percent);
run;
proc freq data=train noprint;
table fraud*used_chip/out=used_chip_perc
nocum list;
run;
data used_chip_perc;
set used_chip_perc;
percent=percent/100;
if fraud=0 and used_chip=0 then
call symput('used_chip_no', percent);
if fraud=0 and used_chip=1 then
call symput('used_chip_yes', percent);
if fraud=1 and used_chip=0 then
call symput('used_chip_no', percent);
if fraud=1 and used_chip=1 then
call symput('used_chip_yes', percent);
run;
proc freq data=train noprint;
table fraud*used_pin_number/out=used_pin_number_perc
nocum list;
run;
data used_pin_number_perc;
set used_pin_number_perc;
percent=percent/100;
if fraud=0 and used_pin_number=0 then
call symput('used_pin_number_no', percent);
if fraud=0 and used_pin_number=1 then
call symput('used_pin_number_yes', percent);
if fraud=1 and used_pin_number=0 then
call symput('used_pin_number_no', percent);
if fraud=1 and used_pin_number=1 then
call symput('used_pin_number_yes', percent);
run;
proc freq data=train noprint;
```

```
table fraud*online_order/out=online_order_perc
nocum list;
run;
data online_order_perc;
set online_order_perc;
percent=percent/100;
if fraud=0 and online_order=0 then
call symput('online_order_no', percent);
if fraud=0 and online_order=1 then
call symput('online_order_yes', percent);
if fraud=1 and online_order=0 then
call symput('online_order_no', percent);
if fraud=1 and online_order=1 then
call symput('online_order_yes', percent);
run;
/*COMPUTING MEAN AND STANDARD DEVIATION FOR
NUMERICAL PREDICTORS*/ proc means data=train mean std
noprint;
class fraud;
var distance_from_home distance_from_last_transaction
ratio_to_median_purchase_price;
output out=stats;
run;
data stats;
set stats;
if fraud=0 and _stat_='MEAN' then
do;
call symput('dist_home_mean_no',distance_from_home);
call symput('dist_trans_mean_no',distance_from_last_transaction);
call symput('ratio_price_mean_no',ratio_to_median_purchase_price);
end;
if fraud=0 and _stat_='STD' then
do;
call symput('dist_home_std_no',distance_from_home);
call symput('dist_trans_std_no',distance_from_last_transaction);
call symput('ratio_price_std_no',ratio_to_median_purchase_price);
end;
if fraud=1 and _stat_='MEAN' then
do;
call symput('dist_home_mean_yes',distance_from_home);
call symput('dist_trans_mean_yes',distance_from_last_transaction);
call symput('ratio_price_mean_yes',ratio_to_median_purchase_price);
end;
if fraud=1 and _stat_='STD' then
do;
```

```
 call symput('dist_home_std_yes',distance_from_home);
 call symput('dist_trans_std_yes',distance_from_last_transaction);
 call symput('ratio_price_std_yes',ratio_to_median_purchase_price);
 end;
 run;
 /*COMPUTING POSTERIOR PROBABILITIES FOR TESTING DATA*/
 data test;
 set test;
 if (repeat_retailer=0 and used_chip=0 and used_pin_number=0 and
 online_order=0) then do;
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_no*&used_pin_number_no
 *&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no
 *&ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
 &dist_trans_mean_no)**2/(2*&dist_trans_std_no**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_no)**2/
 (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_no*&used_pin_number_no
 *&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes
 *&ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)*
 *2/ (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)
 (ratio_to_median_purchase_price-
&ratio_price_mean_yes)**2/(2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=1 and used_chip=0 and used_pin_number=0 and online_order=0)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_no*&used_pin_number_no
*&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-&dist_trans_mean_no)**
 2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-&ratio_price_mean_no)
 **2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_no*&used_pin_number_no*
&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=0 and used_chip=1 and used_pin_number=0 and online_order=0)
 then do;
```

```
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_yes*&used_pin_number_no
*&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_yes*&used_pin_number_no*
&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=0 and used_chip=0 and used_pin_number=1 and online_order=0)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_no*&used_pin_number_yes
*&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-&dist_trans_mean_no)**
 2/(2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)
 **2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_no*&used_pin_number_yes*
&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*&ratio
_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=0 and used_chip=0 and used_pin_number=0 and online_order=1)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_no*&used_pin_number_no
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-&dist_trans_mean_no)**
 2/(2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_no*&used_pin_number_no
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*&ra
tio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
```

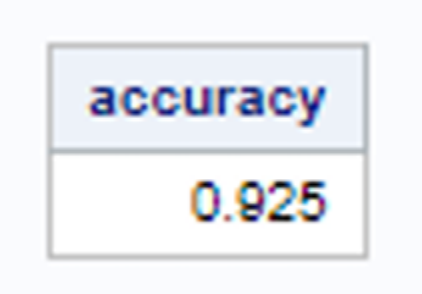
```
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=1 and used_chip=1 and used_pin_number=0 and online_order=0)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_no
 *&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/(2*&dist_trans_std_no**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_no)**2/
(2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_no
 *&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*&rat
io_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=1 and used_chip=0 and used_pin_number=1 and online_order=0)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_no*&used_pin_number_yes
*&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_no*&used_pin_number_yes
 *&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=1 and used_chip=0 and used_pin_number=0 and online_order=1)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_no*&used_pin_number_no
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
```

```
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-&ratio_price_mean_no)
 **2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_no*&used_pin_number_no
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes
 * &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=0 and used_chip=1 and used_pin_number=1 and online_order=0)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_yes*&used_pin_number_yes
*&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_yes*&used_pin_number_yes
 *&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=0 and used_chip=1 and used_pin_number=0 and online_order=1)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_yes*&used_pin_number_no
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_yes*&used_pin_number_no
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes
 * &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
```

```
 end;
 if(repeat_retailer=0 and used_chip=0 and used_pin_number=1 and online_order=1)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_no*&used_pin_number_yes
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_no*&used_pin_number_yes
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes
 * &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=1 and used_chip=1 and used_pin_number=1 and online_order=0)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_yes
*&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_ye
s
 *&online_order_no*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=1 and used_chip=1 and used_pin_number=0 and online_order=1)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_no
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
```

```
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_no
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=1 and used_chip=0 and used_pin_number=1 and online_order=1)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_no*&used_pin_number_yes
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_no*&used_pin_number_yes
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if(repeat_retailer=0 and used_chip=1 and used_pin_number=1 and online_order=1)
 then do;
 pred_prob_no=&prior_no*&repeat_retailer_no*&used_chip_yes*&used_pin_number_yes
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_no*&used_chip_yes*&used_pin_number_yes
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
 end;
 if (repeat_retailer=1 and used_chip=1 and used_pin_number=1 and
 online_order=1) then do;
```

```
 pred_prob_no=&prior_no*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_yes
*&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_no*&dist_trans_std_no*
 &ratio_price_std_no)*exp(-(distance_from_home-&dist_home_mean_no)**2/
 (2*&dist_home_std_no**2)-(distance_from_last_transaction-
&dist_trans_mean_no)**2/
 (2*&dist_trans_std_no**2)-(ratio_to_median_purchase_price-
&ratio_price_mean_no)**2/ (2*&ratio_price_std_no**2));
 pred_prob_yes=&prior_yes*&repeat_retailer_yes*&used_chip_yes*&used_pin_number_ye
s
 *&online_order_yes*1/(2*3.14)**1.5*1/(&dist_home_std_yes*&dist_trans_std_yes*
 &ratio_price_std_yes)*exp(-(distance_from_home-&dist_home_mean_yes)**2/
 (2*&dist_home_std_yes**2)-(distance_from_last_transaction-
&dist_trans_mean_yes)**2/(2*&dist_trans_std_yes**2)-
(ratio_to_median_purchase_price-&ratio_price_mean_yes)**2/
 (2*&ratio_price_std_yes**2));
end;
run;

/*COMPUTING PREDICTION ACCURACY*/
data test;
set test;
if pred_prob_no < pred_prob_yes then pred_class=1;
else pred_class=0;
if fraud=pred_class then pred=1; else pred=0;
run;

proc sql;
select mean(pred) as accuracy
from test;
quit;
```



| accuracy |
| --- |
| 0.925 |

Python

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

card_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.
csv")
X = card_data.iloc[:, 0:7].values
y = card_data.iloc[:, 7].values

# Splitting the data into 80% training and 20% testing sets.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
                                                    random_state=121406)

# Fitting a binary Naive Bayes Classifier.

gauss_nb = GaussianNB()
gauss_nb.fit(X_train, y_train)

# Computing prediction accuracy on testing data.

nb_pred = gauss_nb.predict(X_test)
accuracy_nb = metrics.accuracy_score(y_test, nb_pred) * 100
print("Accuracy score:", round(accuracy_nb, 2), '%')
```

```
Accuracy score: 55.75 %
```

R

```
library(readr)
library(e1071)

card_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.csv
",
header=T, sep=",")

# Splitting the data into 80% training and 20% testing sets.

set.seed(111009)
sample = sample(c(T,F), nrow(card_data), replace=T, prob=c(0.8, 0.2))
train = card_data[sample,]
test = card_data[!sample,]
```

```r
# Fitting binary Naive Bayes classifier.

nb_binary =
naiveBayes(as.factor(fraud)~distance_from_home+distance_from_last_transaction
+ratio_to_median_purchase_price+repeat_retailer+used_chip+used_pin_number
+online_order, data=train)

# Computing prediction accuracy for testing data.

y_pred = predict(nb_binary, newdata=test)
len = nrow(test)
test = cbind(test, y_pred)
match = c()

for (i in 1:len) {
    match[i] = ifelse(test$fraud[i]==test$y_pred[i], 1, 0)
}

print(paste('Accuracy:', round(mean(match)*100, 2), '%'))
```

```
[1] "Accuracy: 50.51 %"
```

Problem 2

SAS

```
proc import out=concussions
datafile="C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_data.c
sv"
dbms=csv replace;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
proc surveyselect data=concussions rate=0.8 seed=267363
out=concussions outall method=srs;
run;
data train (drop=selected);
set concussions;
if selected=1;
run;
data test (drop=selected);
set concussions;
if selected=0;
run;

/*COMPUTING PRIOR PROBABILITIES*/
```

```
proc freq data=train noprint;
table concussion/out=priors;
run;
data priors;
set priors;
percent=percent/100;
if concussion='mild' then
call symput('prior_mild', percent);
if concussion='moderate' then
call symput('prior_mod', percent);
if concussion='severe' then
call symput('prior_sev', percent);
run;

/*COMPUTING POSTERIOR PROBABILITIES FOR CATEGORICAL PREDICTORS*/
proc freq data=train noprint;
table concussion*position/out=position_perc;
run;
data position_perc;
set position_perc;
percent=percent/100;
if concussion='mild' and position='Cornerback' then
call symput('Cornerback_mild', percent);
if concussion='moderate' and position='Cornerback' then
call symput('Cornerback_mod', percent);
if concussion='severe' and position='Cornerback' then
call symput('Cornerback_sev', percent);
if concussion='mild' and position='Offensive Lineman' then
call symput('Offensive_Lineman_mild', percent);
if concussion='moderate' and position='Offensive Lineman' then
call symput('Offensive_Lineman_mod', percent);
if concussion='severe' and position='Offensive Lineman' then
call symput('Offensive_Lineman_sev', percent);
if concussion='mild' and position='Quarterback' then
call symput('Quarterback_mild', percent);
if concussion='moderate' and position='Quarterback' then
call symput('Quarterback_mod', percent);
if concussion='severe' and position='Quarterback' then
call symput('Quarterback_sev', percent);
if concussion='mild' and position='Running Back' then
call symput('Running_Back_mild', percent);
if concussion='moderate' and position='Running Back' then
call symput('Running_Back_mod', percent);
if concussion='severe' and position='Running Back' then
call symput('Running_Back_sev', percent);
```

```
if concussion='mild' and position='Wide Receiver' then
call symput('Wide_Receiver_mild', percent);
if concussion='moderate' and position='Wide Receiver' then
call symput('Wide_Receiver_mod', percent);
if concussion='severe' and position='Wide Receiver' then
call symput('Wide_Receiver_sev', percent);
run;

proc freq data=train noprint;
table concussion*prevconc/out=prevconc_perc
nocum list;
run;
data prevconc_perc;
set prevconc_perc;
percent=percent/100;
if concussion='mild' and prevconc=0 then call symput('prevconc0_mild', percent);
if concussion='moderate' and prevconc=0 then call symput('prevconc0_mod',
percent);
if concussion='severe' and prevconc=0 then call symput('prevconc0_sev', percent);
if concussion='mild' and prevconc=1 then call symput('prevconc1_mild', percent);
if concussion='moderate' and prevconc=1 then call symput('prevconc1_mod',
percent);
if concussion='severe' and prevconc=1 then call symput('prevconc1_sev', percent);
if concussion='mild' and prevconc=2 then call symput('prevconc2_mild', percent);
if concussion='moderate' and prevconc=2 then call symput('prevconc2_mod',
percent);
if concussion='severe' and prevconc=2 then call symput('prevconc2_sev', percent);
if concussion='mild' and prevconc=3 then call symput('prevconc3_mild', percent);
if concussion='moderate' and prevconc=3 then call symput('prevconc3_mod',
percent);
if concussion='severe' and prevconc=3 then call symput('prevconc3_sev', percent);
run;

/*COMPUTING MEAN AND STANDARD DEVIATION FOR NUMERICAL PREDICTORS*/
proc means data=train mean std noprint;
class concussion;
var age nyearsplaying;
output out=stats;
run;
data stats;
set stats;
if concussion='mild' and _stat_='MEAN' then
do;
call symput('age_mean_mild',age);
call symput('nyearsplaying_mean_mild',nyearsplaying);
```

```
end;
if concussion='mod' and _stat_='MEAN' then
do;
call symput('age_mean_mod',age);
call symput('nyearsplaying_mean_mod',nyearsplaying);
end;
if concussion='severe' and _stat_='MEAN' then
do;
call symput('age_mean_sev',age);
call symput('nyearsplaying_mean_sev',nyearsplaying);
end;
if concussion='mild' and _stat_='STD' then
do;
call symput('age_std_mild',age);
call symput('nyearsplaying_std_mild',nyearsplaying);
end;
if concussion='mod' and _stat_='STD' then
do;
call symput('age_std_mod',age);
call symput('nyearsplaying_std_mod',nyearsplaying);
end;
if concussion='severe' and _stat_='STD' then
do;
call symput('age_std_sev',age);
call symput('sev',nyearsplaying);
end;
run;

/*COMPUTING POSTERIOR PROBABILITIES FOR TESTING DATA*/
data test;
set test;
if (position='Cornerback' and prevconc=0) then do;
pred_prob_mild =
&prior_mild*&Cornerback_mild*&prevconc0_mild*1/(2*3.14)*1/(&age_std_mild*&nyearsp
laying_std_mild)
*exp(-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyear
splaying_std_mild**2));
pred_prob_mod =
&prior_mod*&Cornerback_mod*&prevconc0_mod*1/(2*3.14)*1/(&age_std_mod*&nyearsplayi
ng_std_mod)*exp(-(
age-&age_m
**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nyearsplaying_std_mod**2)));
pred_prob_sev =
```

```
&prior_sev*&Cornerback_sev*&prevconc0_sev*1/(2*3.14)*1/(&age_std_sev*&nyearsplayi
ng_std_sev)
*exp(-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyear
splaying_std_sev**2)); end;
if (position='Cornerback' and prevconc=1) then do;
pred_prob_mild =
&prior_mild*&Cornerback_mild*&prevconc1_mild*1/(2*3.14)*1/(&age_std_mild*&nyearsp
laying_std_mild)
*exp(-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyear
splaying_std_mild**2));
pred_prob_mod =
&prior_mod*&Cornerback_mod*&prevconc1_mod*1/(2*3.14)*1/(&age_std_mod*&nyearsplayi
ng_std_mod)
*exp(-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&ny
earsplaying_std_mod**2));
pred_prob_sev =
&prior_sev*&Cornerback_sev*&prevconc1_sev*1/(2*3.14)*1/(&age_std_sev*&nyearsplayi
ng_std_sev)
*exp(-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyear
splaying_std_sev**2)); end;
if (position='Cornerback' and prevconc=2) then do;
pred_prob_mild =
&prior_mild*&Cornerback_mild*&prevconc2_mild*1/(2*3.14)*1/(&age_std_mild*&nyearsp
laying_std_mild)
*exp(-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyear
splaying_std_mild**2));
pred_prob_mod=&prior_mod*&Cornerback_mod*&prevconc2_mod*1/(2*3.14)*1/(&age_std_mo
d*&nyearsplayin
g_std_mod)
*exp(-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&ny
earsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Cornerback_sev*&prevconc2_sev*1/(2*3.14)*1/(&age_std_se
v*&nyearsplaying_st
d_sev)
*exp(-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyear
splaying_std_sev**2)); end;
if (position='Cornerback' and prevconc=3) then do;
pred_prob_mild =
```

```
&prior_mild*&Cornerback_mild*&prevconc3_mild*1/(2*3.14)*1/(&age_std_mild*&nyearsp
laying_std_mild)
*exp(-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyear
splaying_std_mild**2)
pred_prob_mod =
&prior_mod*&Cornerback_mod*&prevconc3_mod*1/(2*3.14)*1/(&age_std_mod*&nyearsplayi
ng_std_mod)
*exp(-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&ny
earsplaying_std_mod**2));
pred_prob_sev =
&prior_sev*&Cornerback_sev*&prevconc3_sev*1/(2*3.14)*1/(&age_std_sev*&nyearsplayi
ng_std_sev)
*exp(-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyear
splaying_std_sev**2)); end;
if (position='Offensive Lineman' and prevconc=0) then do;
pred_prob_mild =
&prior_mild*&Offensive_Lineman_mild*&prevconc0_mild*1/(2*3.14)*1/(&age_std_mild*&
nyearsplaying_std_mild)*
exp (2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearsplaying_std_mild**2));
pred_prob_mod = &prior_mod*&Offensive_Lineman_mod*
&prevconc0_mod*1/(2*3.14)*1/(&age_std_mod*&nyearsplaying_std_mod)*exp(-(age-
&age_mean_mod)**2/(2*&a
ge_std_mod**2)-(nyearsplay -
&nyearsplaying_mean_mod)**2/(2*&nyearsplaying_std_mod**2));
pred_prob_sev =
&prior_sev*&Offensive_Lineman_sev*&prevconc0_sev*1/(2*3.14)*1/(&age_std_sev*&nyea
rsplaying_std_sev)*exp(-(age (2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyearsplaying_std_sev**2)));
end;
if (position='Offensive Lineman' and prevconc=1) then do;
pred_prob_mild=&prior_mild*&Offensive_Lineman_mild*&prevconc1_mild*1/(2*3.14)*1/(
&age_std_mild*&nyears
playing_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspla
ying_std_mild**2));
pred_prob_mod=&prior_mod*&Offensive_Lineman_mod*&prevconc1_mod*1/(2*3.14)*1/(&age
_std_mod*&
nyearsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
```

```
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Offensive_Lineman_sev*&prevconc1_sev*1/(2*3.14)*1/(&age
_std_sev*&nyea
rsplaying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Offensive Lineman' and prevconc=2) then do;
pred_prob_mild=&prior_mild*&Offensive_Lineman_mild*&prevconc2_mild*1/(2*3.14)*1/(
&age_std_mild*&nyears
playing_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspla
ying_std_mild**2));
pred_prob_mod=&prior_mod*&Offensive_Lineman_mod*&prevconc2_mod*1/(2*3.14)*1/(&age
_std_mod*&
nyearsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Offensive_Lineman_sev*&prevconc2_sev*1/(2*3.14)*1/(&age
_std_sev*&nyea
rsplaying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Offensive Lineman' and prevconc=3) then do;
pred_prob_mild=&prior_mild*&Offensive_Lineman_mild*&prevconc3_mild*1/(2*3.14)*1/(
&age_std_mild*&nyears
playing_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspla
ying_std_mild**2));
pred_prob_mod=&prior_mod*&Offensive_Lineman_mod*&prevconc3_mod*1/(2*3.14)*1/(&age
_std_mod*&
nyearsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Offensive_Lineman_sev*&prevconc3_sev*1/(2*3.14)*1/(&age
_std_sev*&nyea
rsplaying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
```

```
if (position='Quarterback' and prevconc=0) then do;
pred_prob_mild=&prior_mild*&Quarterback_mild*&prevconc0_mild*1/(2*3.14)*1/(&age_s
td_mild*&nyearsplayi
ng_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Quarterback_mod*&prevconc0_mod*1/(2*3.14)*1/(&age_std_m
od*&nyear
splaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&ny
earsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Quarterback_sev*&prevconc0_sev*1/(2*3.14)*1/(&age_std_s
ev*&nyearsplayi
ng_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Quarterback' and prevconc=1) then do;
pred_prob_mild=&prior_mild*&Quarterback_mild*&prevconc1_mild*1/(2*3.14)*1/(&age_s
td_mild*&nyearsplayi
ng_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Quarterback_mod*&prevconc1_mod*1/(2*3.14)*1/(&age_std_m
od*&nyears
playing_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Quarterback_sev*&prevconc1_sev*1/(2*3.14)*1/(&age_std_s
ev*&nyearsplayi
ng_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Quarterback' and prevconc=2) then do;
pred_prob_mild=&prior_mild*&Quarterback_mild*&prevconc2_mild*1/(2*3.14)*1/(&age_s
td_mild*&nyearsplayi
ng_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
```

```
pred_prob_mod=&prior_mod*&Quarterback_mod*&prevconc2_mod*1/(2*3.14)*1/(&age_std_m
od*&nyears
playing_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Quarterback_sev*&prevconc2_sev*1/(2*3.14)*1/(&age_std_s
ev*&nyearsplayi
ng_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Quarterback' and prevconc=3) then do;
pred_prob_mild=&prior_mild*&Quarterback_mild*&prevconc3_mild*1/(2*3.14)*1/(&age_s
td_mild*&nyearsplayi
ng_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Quarterback_mod*&prevconc3_mod*1/(2*3.14)*1/(&age_std_m
od*&nyears
playing_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Quarterback_sev*&prevconc3_sev*1/(2*3.14)*1/(&age_std_s
ev*&nyearsplayi
ng_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Running Back' and prevconc=0) then do;
pred_prob_mild=&prior_mild*&Running_Back_mild*&prevconc0_mild*1/(2*3.14)*1/(&age_
std_mild*&nyearspla
ying_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Running_Back_mod*&prevconc0_mod*1/(2*3.14)*1/(&age_std_
mod*&nyea
rsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
```

```
pred_prob_sev=&prior_sev*&Running_Back_sev*&prevconc0_sev*1/(2*3.14)*1/(&age_std_
sev*&nyearspl
aying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Running Back' and prevconc=1) then do;
pred_prob_mild=&prior_mild*&Running_Back_mild*&prevconc1_mild*1/(2*3.14)*1/(&age_
std_mild*&nyearspla
ying_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Running_Back_mod*&prevconc1_mod*1/(2*3.14)*1/(&age_std_
mod*&nyea
rsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Running_Back_sev*&prevconc1_sev*1/(2*3.14)*1/(&age_std_
sev*&nyearspl
aying_std_sev)*exp(-(age
-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyearsplayi
ng_std_sev**2)); end;
if (position='Running Back' and prevconc=2) then do;
pred_prob_mild=&prior_mild*&Running_Back_mild*&prevconc2_mild*1/(2*3.14)*1/(&age_
std_mild*&nyearspla
ying_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Running_Back_mod*&prevconc2_mod*1/(2*3.14)*1/(&age_std_
mod*&nyea
rsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Running_Back_sev*&prevconc2_sev*1/(2*3.14)*1/(&age_std_
sev*&nyearspl
aying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Running Back' and prevconc=3) then do;
```

```
pred_prob_mild=&prior_mild*&Running_Back_mild*&prevconc3_mild*1/(2*3.14)*1/(&age_
std_mild*&nyearspla
ying_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Running_Back_mod*&prevconc3_mod*1/(2*3.14)*1/(&age_std_
mod*&nye
arsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&ny
earsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Running_Back_sev*&prevconc3_sev*1/(2*3.14)*1/(&age_std_
sev*&nyearspl
aying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Wide Receiver' and prevconc=0) then do;
pred_prob_mild=&prior_mild*&Wide_Receiver_mild*&prevconc0_mild*1/(2*3.14)*1/(&age
_std_mild*&nyearspl
aying_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
pred_prob_mod=&prior_mod*&Wide_Receiver_mod*&prevconc0_mod*1/(2*3.14)*1/(&age_std
_mod*&nye
arsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Wide_Receiver_sev*&prevconc0_sev*1/(2*3.14)*1/(&age_std
_sev*&nyearspl
aying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Wide Receiver' and prevconc=1) then do;
pred_prob_mild=&prior_mild*&Wide_Receiver_mild*&prevconc1_mild*1/(2*3.14)*1/(&age
_std_mild*&nyearspl
aying_std_mild)*exp(
-(age-&age_mean_mild)**2/(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearspl
aying_std_mild**2));
```

```
pred_prob_mod=&prior_mod*&Wide_Receiver_mod*&prevconc1_mod*1/(2*3.14)*1/(&age_std
_mod*&nye
arsplaying_std_mod)*exp(
-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nye
arsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Wide_Receiver_sev*&prevconc1_sev*1/(2*3.14)*1/(&age_std
_sev*&nyearspl
aying_std_sev)*exp(
-(age-&age_mean_sev)**2/(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyears
playing_std_sev**2)); end;
if (position='Wide Receiver' and prevconc=2) then do;
pred_prob_mild=&prior_mild*&Wide_Receiver_mild*&prevconc2_mild*1/(2*3.14)*1/(&age
_std_mild*&nyearsplayi
ng_std_mild)*exp(-(age
(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearsplaying_std_mild**2)));
pred_prob_mod=&prior_mod*&Wide_Receiver_mod*&prevconc2_mod*1/(2*3.14)*1/(&age_std
_mod*&nyearsplay
ing_std_mod)*exp(-(age-&age_
(2*&age_std_mod**2)-(nyearsplaying-
&nyearsplaying_mean_mod)**2/(2*&nyearsplaying_std_mod**2)));
pred_prob_sev=&prior_sev*&Wide_Receiver_sev*&prevconc2_sev*1/(2*3.14)*1/(&age_std
_sev*&nyearsplaying_
std_sev)*exp(-(age-&age_
(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyearsplaying_std_sev**2))); end;
if (position='Wide Receiver' and prevconc=3) then do;
pred_prob_mild=&prior_mild*&Wide_Receiver_mild*&prevconc3_mild*1/(2*3.14)*1/(&age
_std_mild*&nyearsplayi
ng_std_mild)*exp(-(age
(2*&age_std_mild**2)-(nyearsplaying-
&nyearsplaying_mean_mild)**2/(2*&nyearsplaying_std_mild**2)));
pred_prob_mod=&prior_mod*&Wide_Receiver_mod*&prevconc3_mod*1/(2*3.14)*1/(&age_std
_mod*
&nyearsplaying_std_mod)*exp(-(age-&age_mean_mod)**2/(2*&age_std_mod**2)-
(nyearsplaying
-&nyearsplaying_mean_mod)**2/(2*&nyearsplaying_std_mod**2));
pred_prob_sev=&prior_sev*&Wide_Receiver_sev*&prevconc3_sev*1/(2*3.14)*1/(&age_std
_sev*&nyearsplaying_
std_sev)*exp(-(age-&age_
(2*&age_std_sev**2)-(nyearsplaying-
&nyearsplaying_mean_sev)**2/(2*&nyearsplaying_std_sev**2))); end;
run;
```

```
/*COMPUTING PREDICTION ACCURACY*/
data test;
set test;
max_prob=max(pred_prob_mild, pred_prob_mod,
pred_prob_sev);
if max_prob=pred_prob_mod then pred_class='moderate';
if max_prob=pred_prob_mild then pred_class='mild';
if max_prob=pred_prob_sev then pred_class='severe';
if pred_class=concussion then pred=1; else pred=0;
run;

proc sql;
select mean(pred) as accuracy
from test;
quit;
```

| accuracy |
| --- |
| 0.211538 |

Python

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from statistics import mean

concussion_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_dat
a.csv")
position_code = {'Offensive Lineman':0, 'Cornerback':1, 'Running Back':2,
'Quarterback':3, 'Wide Receiver':4}
concussion_code = {'mild':0, 'moderate':1, 'severe':2}
concussion_data['position'] = concussion_data['position'].map(position_code)
concussion_data['concussion'] =
concussion_data['concussion'].map(concussion_code)
X = concussion_data.iloc[:, 0:4].values
y = concussion_data.iloc[:, 4].values
```

```
# Splitting the data into 80% training and 20% testing sets.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
                                        random_state=432504)

# Fitting a multinomial Naive Bayes Classifier.

gaussiannb_multi = GaussianNB()
gaussiannb_multi.fit(X_train, y_train)

# Computing prediction accuracy for testing set.

nb_pred = gaussiannb_multi.predict(X_test)
y_test = pd.DataFrame(y_test, columns=['concussion'])
nb_pred = pd.DataFrame(nb_pred, columns=['predicted'])
df = pd.concat([y_test, nb_pred], axis=1)

match = []
for i in range(len(df)):
    if df['concussion'][i] == df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print("Accuracy:", round(mean(match), 2)* 100, '%')
```

```
Accuracy: 90.0 %
```

R

```
[1] "Accuracy: 91 %"
```

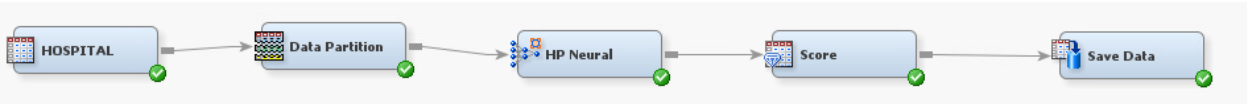
```
library(readr)
library(e1071)

concussion_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_data.c
sv",
header=T, sep=",")

# Splitting the data into 80% training and 20% testing sets.

set.seed(250217)
sample = sample(c(T,F), nrow(concussion_data), replace=T, prob=c(0.8, 0.2))
```

```
train = concussion_data[sample,]
test = concussion_data[!sample,]

# Fitting a multinomial Naive Bayes classifier.

nb_multi = naiveBayes(as.factor(concussion)~age+nyearsplaying+position+prevconc,
data=train)

# Computing prediction accuracy for testing data.

y_pred = predict(nb_multi, newdata=test)
len = nrow(test)
test = cbind(test, y_pred)
match = c()

for (i in 1:len) {
    match[i] = ifelse(test$concussion[i]==test$y_pred[i], 1, 0)
}

print(paste("Accuracy:", round(mean(match), 2)*100, '%'))
```

Problem 3

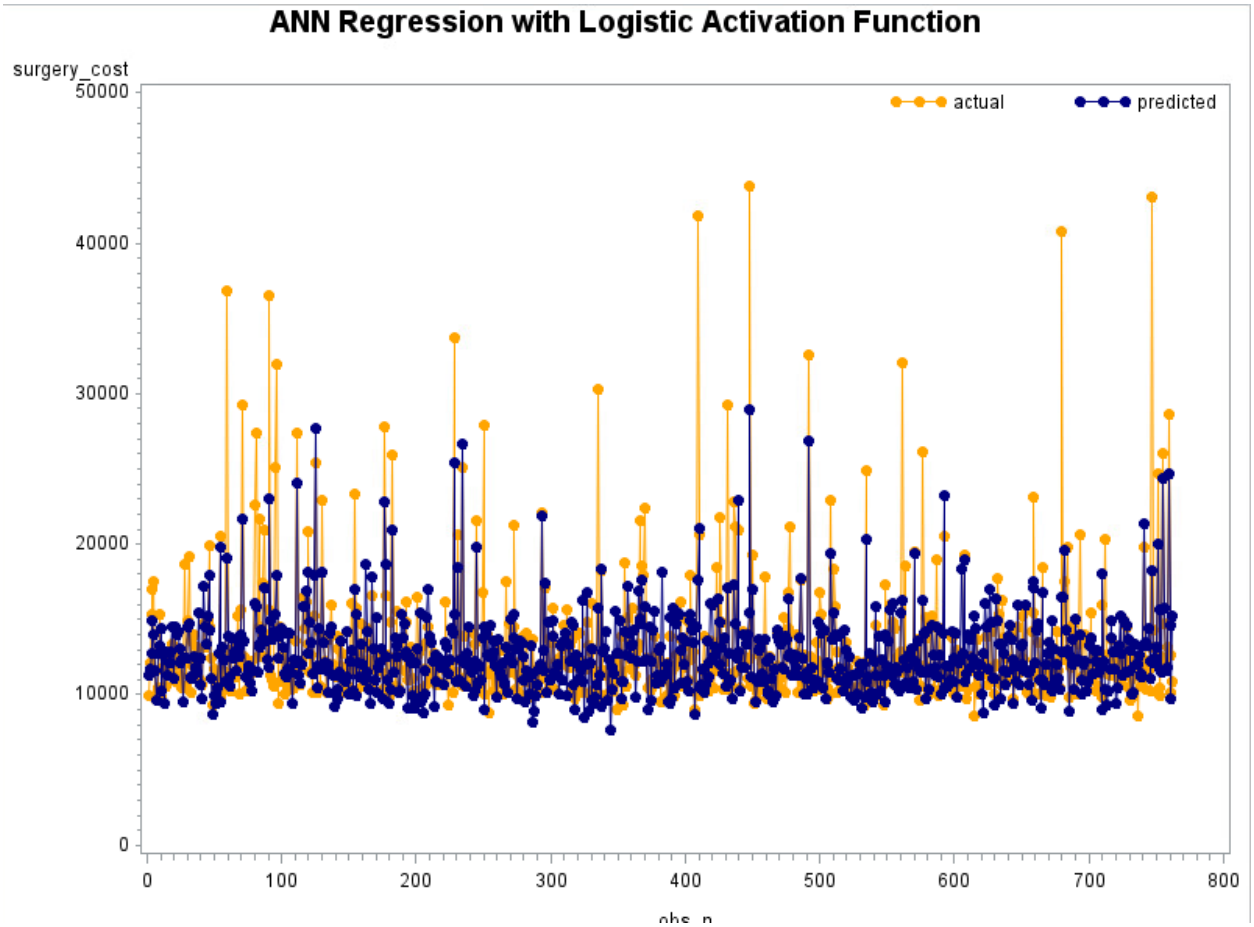
SAS and SAS Enterprise Miner

Logistic Activation Function



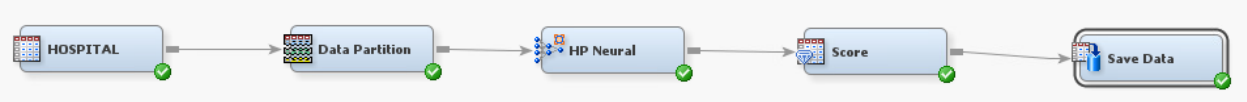

The SAS System

| accuracy10 | accuracy15 | accuracy20 |
|---|---|---|
| 0.452756 | 0.641732 | 0.765092 |


ANN Regression with Logistic Activation Function
surgery_cost
50000
40000
30000
20000
10000
0
actual
predicted
0
100
200
300
400
500
600
700
800

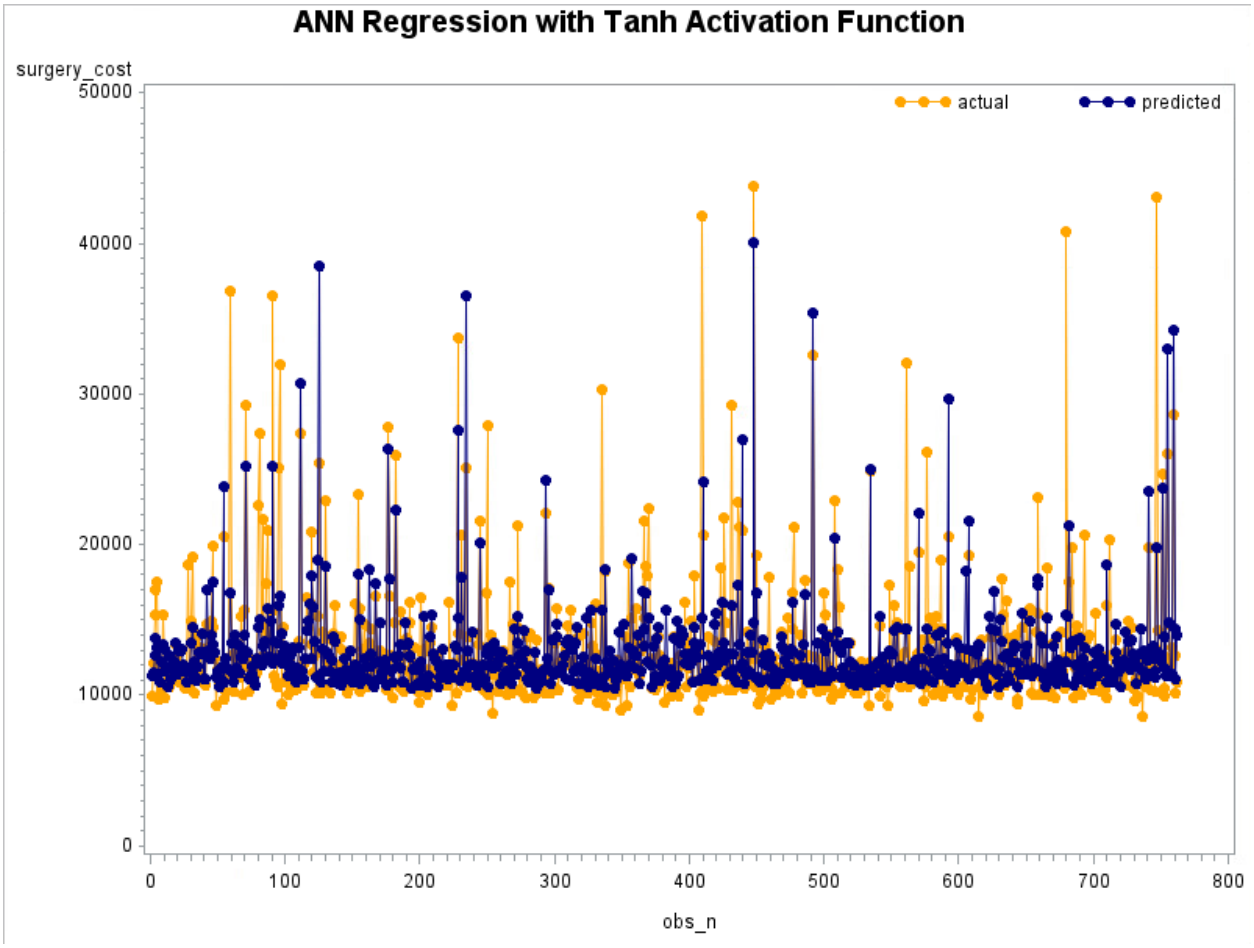
```
data accuracy;
set tmp1.em_save_test;
ind10 = (abs(R_surgery_cost)<0.10*surgery_cost);
ind15 = (abs(R_surgery_cost)<0.15*surgery_cost);
ind20 = (abs(R_surgery_cost)<0.20*surgery_cost);
obs_n = _N_;
run;

proc sql;
select mean(ind10) as accuracy10,
mean(ind15) as accuracy15,
mean(ind20) as accuracy20
from accuracy;
quit;

goptions reset=all border;
title1 "ANN Regression with Logistic Activation Function";
symbol1 interpol=join value=dot color=orange;
symbol2 interpol=join value=dot color=navy;
legend1 value=("actual" "predicted")
position=(top right inside) label=none;
proc gplot data=accuracy;
plot surgery_cost*obs_n
EM_PREDICTION*obs_n/ overlay legend=legend1;
run;
```

Tanh Activation Function




**The SAS System**

| accuracy10 | accuracy15 | accuracy20 |
|---|---|---|
| 0.52231 | 0.708661 | 0.824147 |


ANN Regression with Tanh Activation Function
surgery_cost
50000
40000
30000
20000
10000
0
actual
predicted
0
100
200
300
400
500
600
700
800
obs_n

```
data accuracy;
set tmp1.em_save_test;
ind10 = (abs(R_surgery_cost)<0.10*surgery_cost);
ind15 = (abs(R_surgery_cost)<0.15*surgery_cost);
ind20 = (abs(R_surgery_cost)<0.20*surgery_cost);
obs_n = _N_;
run;

proc sql;
select mean(ind10) as accuracy10,
mean(ind15) as accuracy15,
mean(ind20) as accuracy20
from accuracy;
quit;

goptions reset=all border;
title1 "ANN Regression with Tanh Activation Function";
symbol1 interpol=join value=dot color=orange;
symbol2 interpol=join value=dot color=navy;
legend1 value=("actual" "predicted")
position=(top right inside) label=none;
proc gplot data=accuracy;
plot surgery_cost*obs_n
EM_PREDICTION*obs_n/ overlay legend=legend1;
run;
```

Python

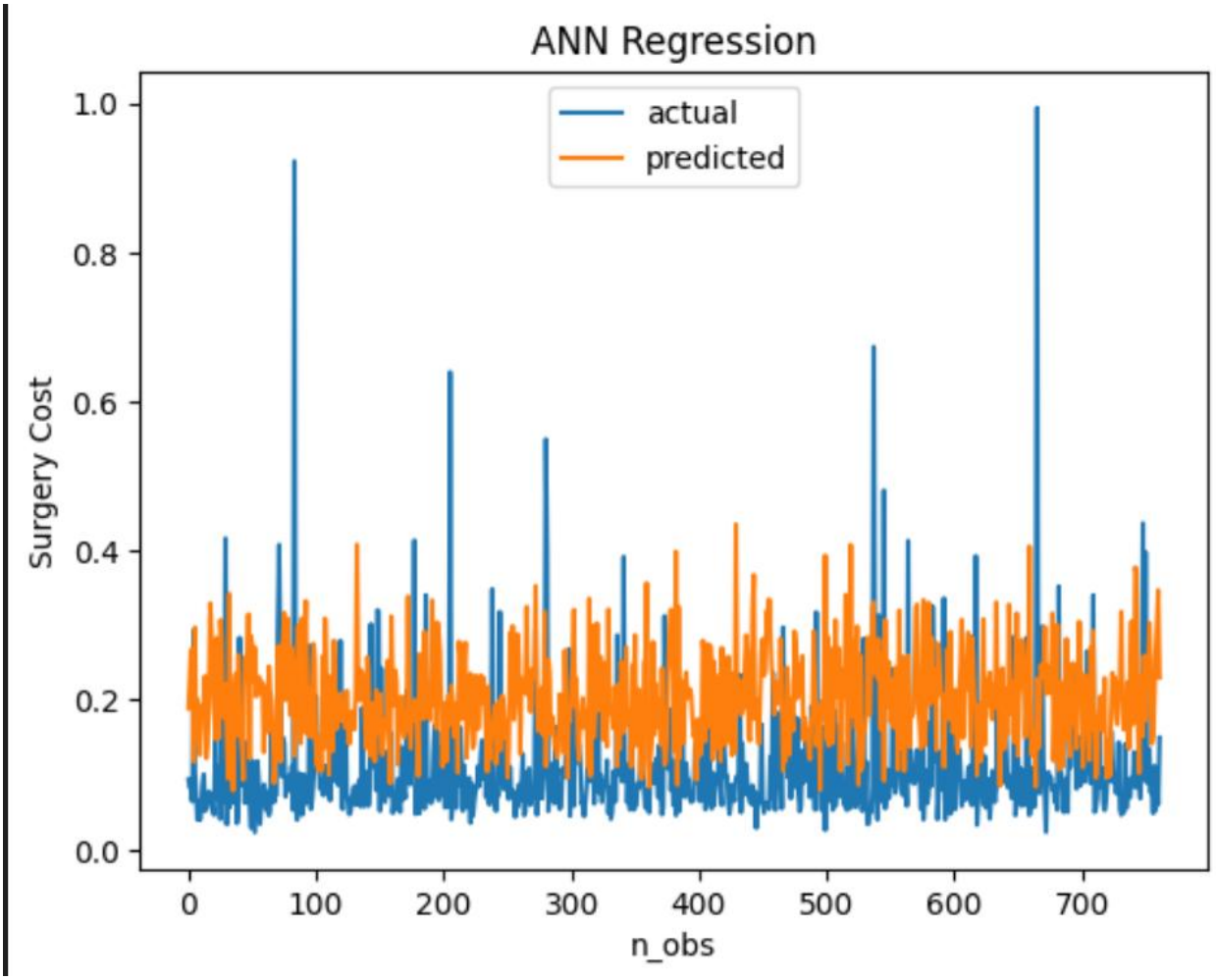
Logistic Activation Function

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from statistics import mean
import tensorflow as tf
import keras as keras
import matplotlib.pyplot as plt

hospital_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/hospital_data.c
sv")
```

```
gender_code = {'M':1, 'F':0}
hospital_data["gender"] = hospital_data["gender"].map(gender_code)

# Scaling the data to fall within [0,1].

scaler = preprocessing.MinMaxScaler()
scaler_fit = scaler.fit_transform(hospital_data)
scaled_hospital = pd.DataFrame(scaler_fit, columns = hospital_data.columns)

X = scaled_hospital.iloc[:, 0:6].values
y = scaled_hospital.iloc[:, 6].values

# Splitting the data into 80% training and 20% testing sets.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
                                    random_state=315205)

# Constructing an ANN for regression.
# Using the sequential model, we initialize the input layer using the sigmoid
activation
# function in the output layer.

tf.random.set_seed(112113)

reg_model = tf.keras.Sequential([
    tf.keras.layers.Dense(3, activation="relu", input_shape=(6,)),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

# Compile the model with the Adam optimizer, a built-in optimizer from Keras.
# Traditionally, learning rates are set to 0.001 but can be adjusted in projects
# during hyperparameter tuning. We use 0.001 for this exercise.

reg_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss=['mean_squared_error'])

reg_model.fit(X_train, y_train)

y_pred = reg_model.predict(X_test)

ind10 = []
ind15 = []
ind20 = []

for sub1, sub2 in zip(y_pred, y_test):
```

```
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

# Accuracy within 10%

accuracy10 = mean(ind10)
print("Accuracy within 10%:", round(accuracy10, 4))

# Accuracy within 15%

accuracy15 = mean(ind15)
print("Accuracy within 15%:", round(accuracy15, 4))

# Accuracy within 20%

accuracy20 = mean(ind20)
print("Accuracy within 20%:", round(accuracy20, 4))

# Plotting actual vs. predicted observations and observation number.

n_obs=list(range(0,len(y_test)))
plt.plot(n_obs, y_test, label="actual")
plt.plot(n_obs, y_pred, label="predicted")
plt.xlabel('n_obs')
plt.ylabel('Surgery Cost')
plt.title('ANN Regression')
plt.legend()
plt.show()
```

```
    super().__init__(activity_regularizer=activity_regularizer, **kwargs)
  96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0374
  24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
  Accuracy within 10%: 0.0499
  Accuracy within 15%: 0.0787
  Accuracy within 20%: 0.0997
```

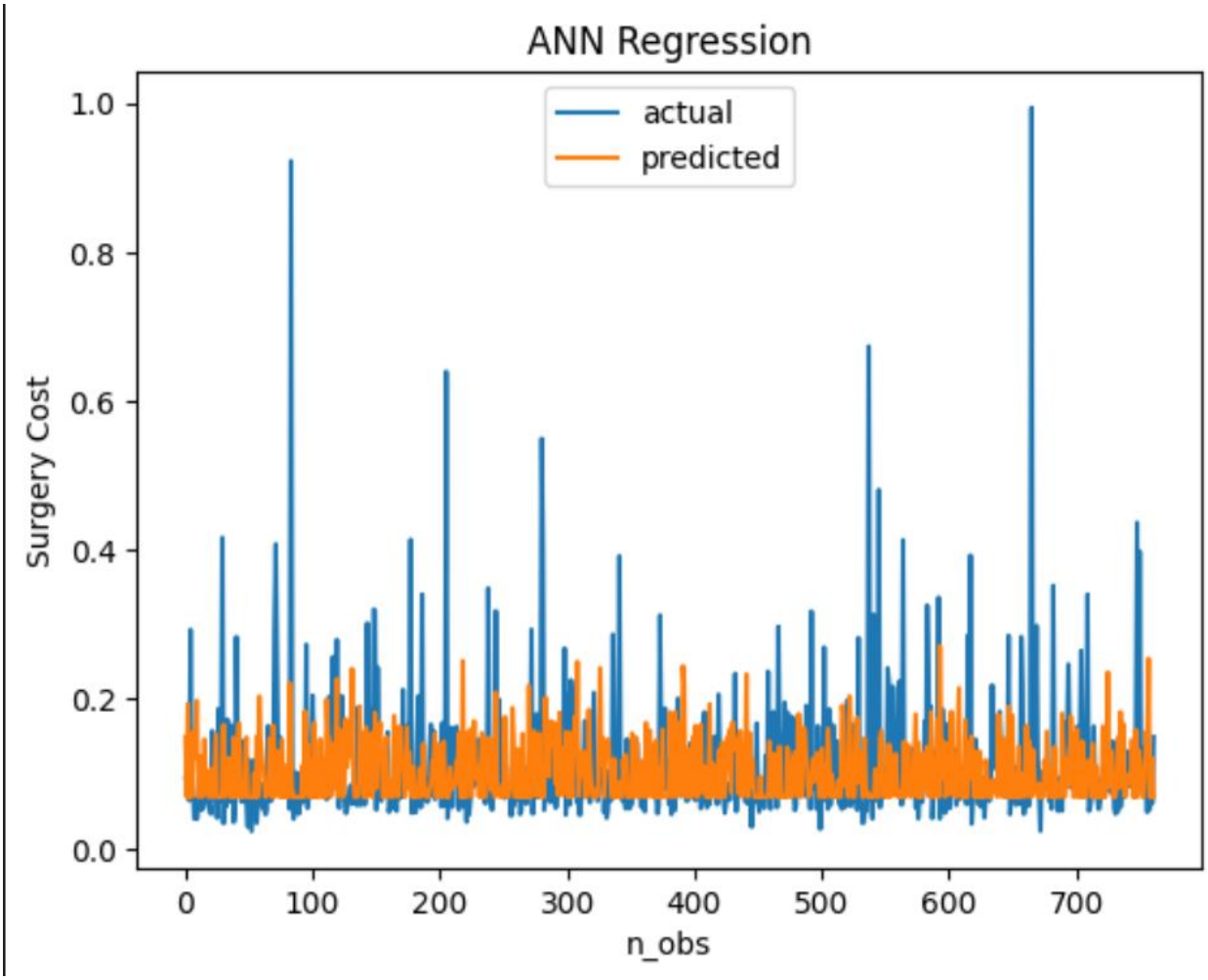



Tanh Activation Function

```
tf.random.set_seed(450560)

reg_model2 = tf.keras.Sequential([
    tf.keras.layers.Dense(3, activation="relu", input_shape=(6,)),
    tf.keras.layers.Dense(1, activation="tanh")
])

# Compile the model with the Adam optimizer, a built-in optimizer from Keras.
# Traditionally, learning rates are set to 0.001 but can be adjusted in projects
# during hyperparameter tuning. We use 0.001 for this exercise.

reg_model2.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                   loss=['mean_squared_error'])

reg_model2.fit(X_train, y_train)

y_pred = reg_model2.predict(X_test)
```

```python
ind10 = []
ind15 = []
ind20 = []

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

# Accuracy within 10%

accuracy10 = mean(ind10)
print("Accuracy within 10%:", round(accuracy10, 4))

# Accuracy within 15%

accuracy15 = mean(ind15)
print("Accuracy within 15%:", round(accuracy15, 4))

# Accuracy within 20%

accuracy20 = mean(ind20)
print("Accuracy within 20%:", round(accuracy20, 4))

# Plotting actual vs. predicted observations and observation number.

n_obs=list(range(0,len(y_test)))
plt.plot(n_obs, y_test, label="actual")
plt.plot(n_obs, y_pred, label="predicted")
plt.xlabel('n_obs')
plt.ylabel('Surgery Cost')
plt.title('ANN Regression')
plt.legend()
plt.show()
```

```
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.0150
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Accuracy within 10%: 0.1181
Accuracy within 15%: 0.1942
Accuracy within 20%: 0.2546
```




R

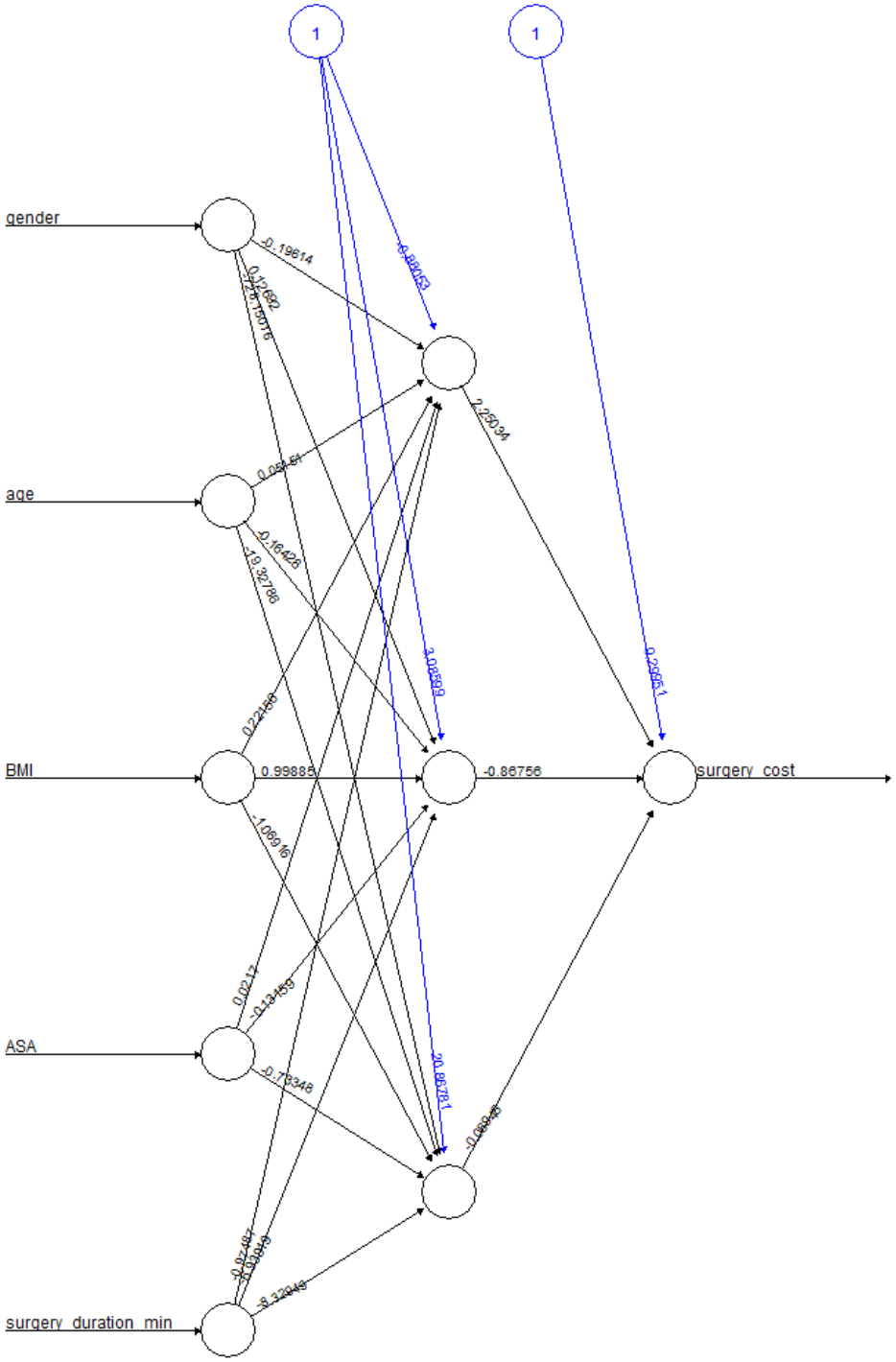
Logistic Activation Function

```
library(readr)
library(dplyr)
library(caTools)
library(neuralnet)

hospital_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/hospital_data.csv"
,
header=T, sep=",")

hospital_data$gender = ifelse(hospital_data$gender=='M', 1, 0)

# Scaling the variables to fall within [0,1].

scale01 = function(x) {
```

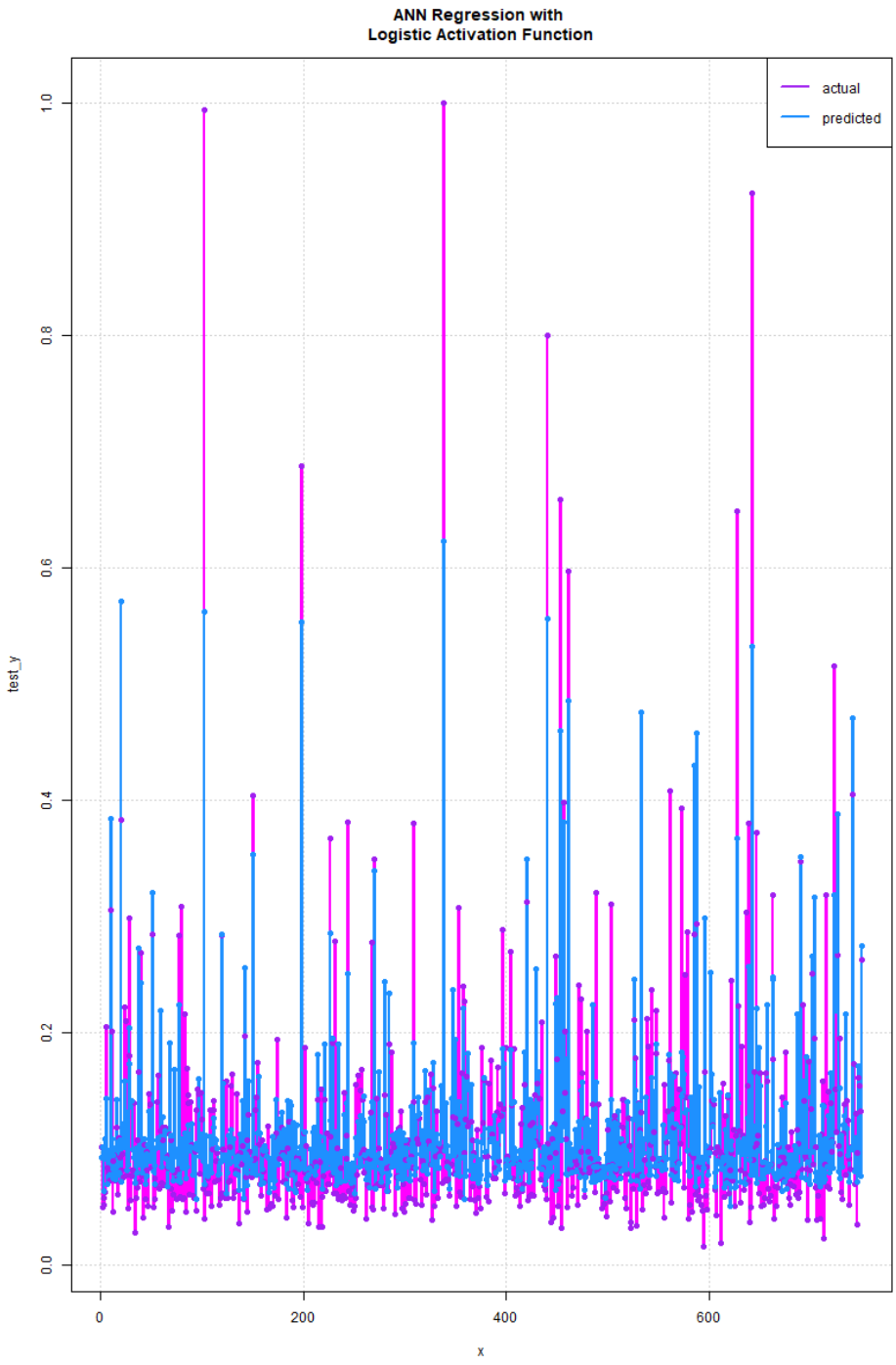
```
    (x-min(x))/(max(x)-min(x))
}

hospital_data = hospital_data %>% mutate_all(scale01)

# Splitting the data into 80% training and 20% testing sets.

set.seed(566409)

sample = sample(c(T,F), nrow(hospital_data), replace=T,
prob=c(0.8, 0.2))
train = hospital_data[sample,]
test = hospital_data[!sample,]

test_x = data.matrix(test[2:6])
test_y = data.matrix(test[7])

# Fitting an ANN for regression.

ann_reg = neuralnet(surgery_cost~gender+age+BMI+ASA
+surgery_duration_min, data=train, hidden=3, act.fct="logistic",
stepmax=1e7)

# Plotting the diagram.

plot(ann_reg)

# Computing prediction accuracy for testing data.

pred_y = predict(ann_reg, test_x)

# Accuracy within 10%

accuracy10 = ifelse(abs(test_y-pred_y)<0.10*test_y, 1, 0)

# Accuracy within 15%

accuracy15 = ifelse(abs(test_y-pred_y)<0.15*test_y, 1, 0)

# Accuracy within 20%

accuracy20 = ifelse(abs(test_y-pred_y)<0.20*test_y, 1, 0)

print('Prediction Accuracy for Logistic Activation Function')
print(paste('within 10%:', round(mean(accuracy10),4)))
```

```
print(paste('within 15%:', round(mean(accuracy15),4)))
print(paste('within 20%:', round(mean(accuracy20),4)))

# Plotting the actual vs predicted values for testing data.

x = 1:length(test_y)
plot(x, test_y, type="l", lwd=2, col="magenta", main="ANN Regression with
Logistic Activation Function", panel.first=grid())
lines(x, pred_y, lwd=2, col="dodgerblue")
points(x, test_y, pch=16, col="purple")
points(x, pred_y, pch=16, col="dodgerblue")
legend("topright", c("actual", "predicted"), lty=1, lwd=2,
col=c("purple","dodgerblue"))
```

```
[1] "Prediction Accuracy for Logistic Activation Function"
[1] "within 10%: 0.2053"
[1] "within 15%: 0.3013"
[1] "within 20%: 0.396"
```




Error: 5.498137  Steps: 15197

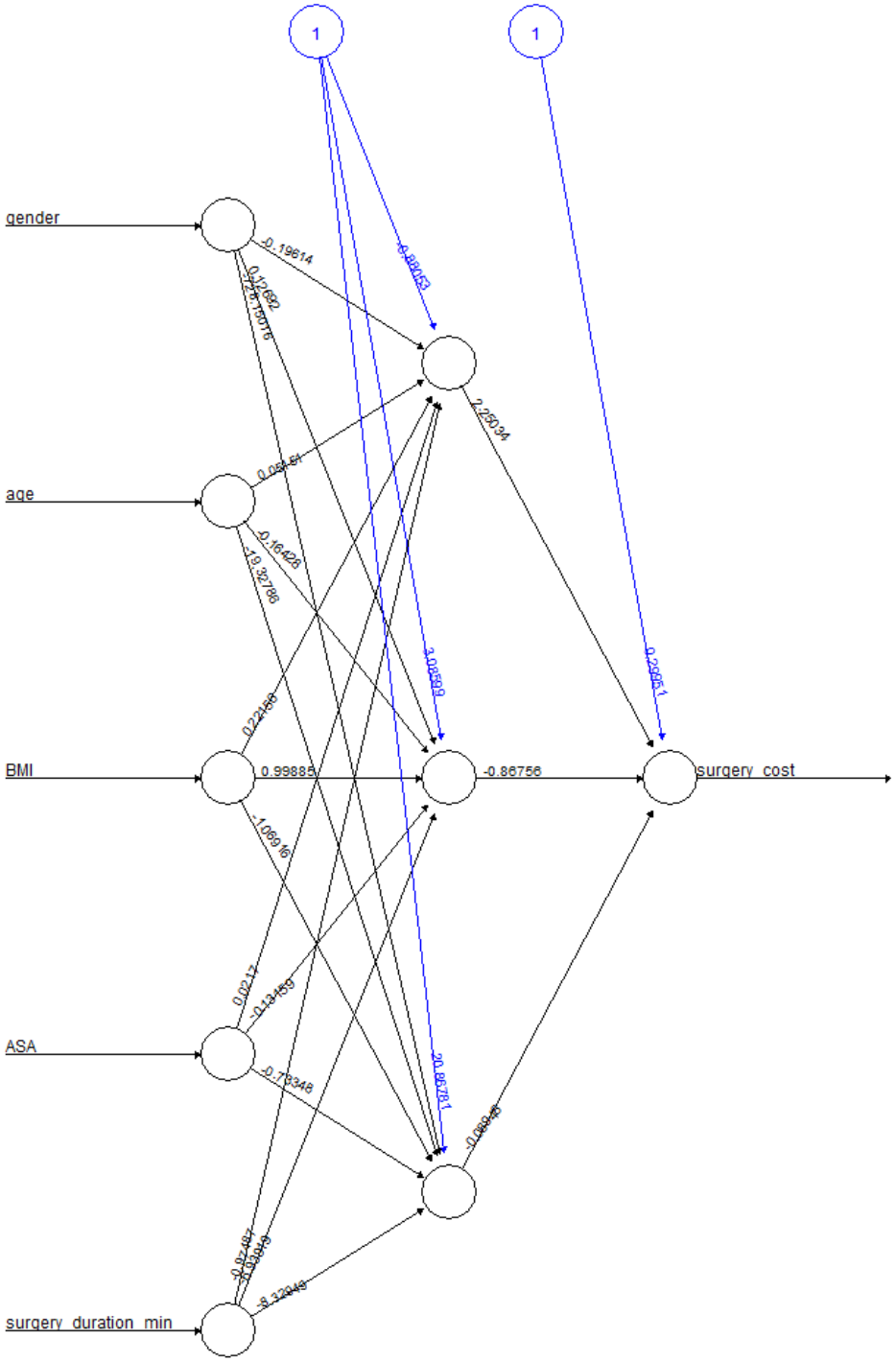



Tanh Activation Function

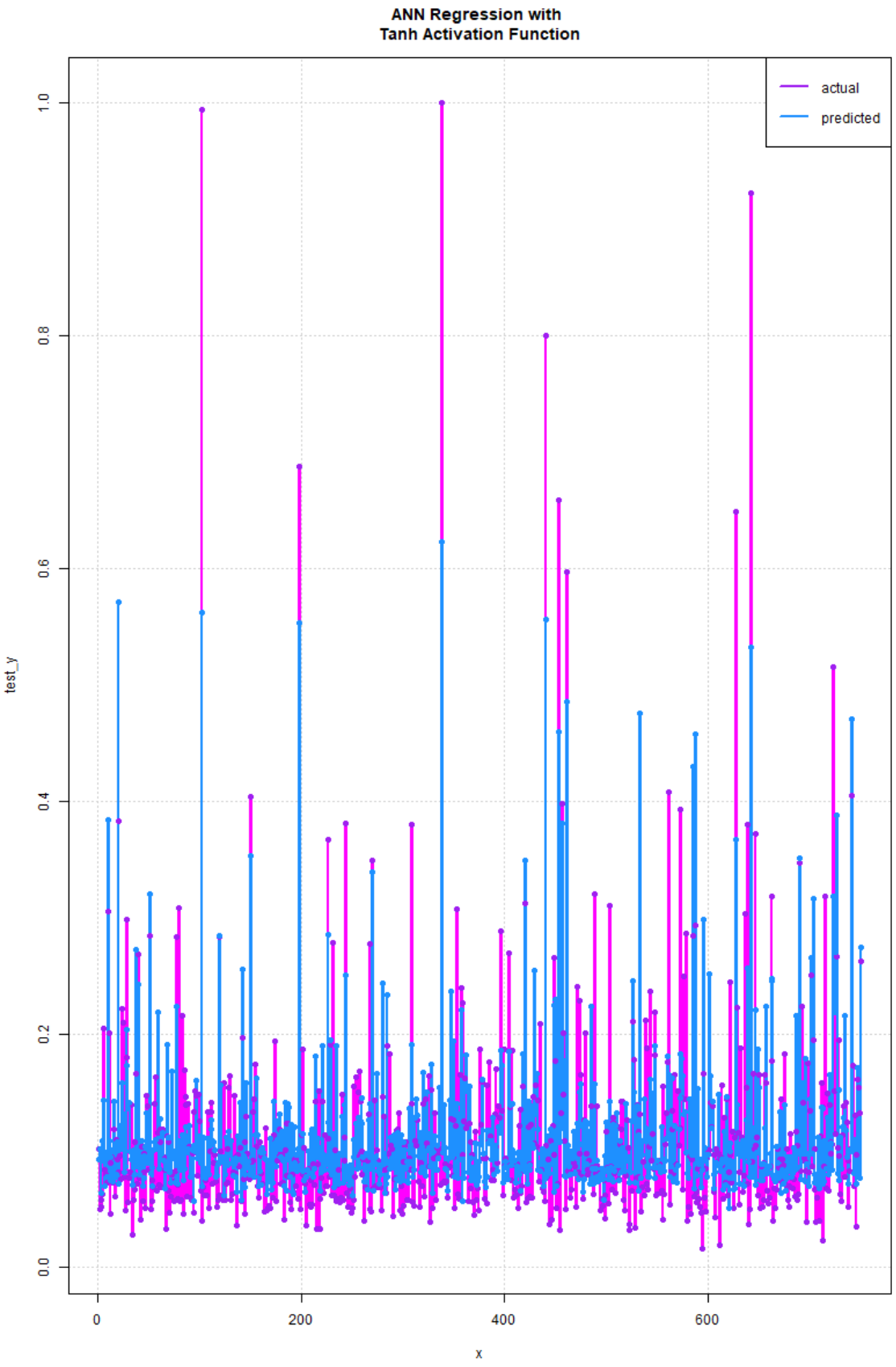
```
# Fitting an ANN for regression using a tanh activation function.

ann_reg_tanh = neuralnet(surgery_cost~gender+age+BMI+ASA
+surgery_duration_min, data=train, hidden=3, act.fct="logistic",
stepmax=1e7)
```

```r
# Plotting the diagram.

plot(ann_reg_tanh)

# Computing the prediction accuracy for testing data.

y_pred_tanh = predict(ann_reg_tanh, test_x)

# Accuracy within 10%

accuracy10_tanh = ifelse(abs(test_y-y_pred_tanh)<0.10*test_y, 1, 0)

# Accuracy within 15%

accuracy15_tanh = ifelse(abs(test_y-y_pred_tanh)<0.15*test_y, 1, 0)

# Accuracy within 20%

accuracy20_tanh = ifelse(abs(test_y-y_pred_tanh)<0.20*test_y, 1, 0)

print("Prediction accuracy for Tanh activation function")
print(paste('within 10%:', round(mean(accuracy10_tanh),4)))
print(paste('within 15%:', round(mean(accuracy15_tanh),4)))
print(paste('within 20%:', round(mean(accuracy20_tanh),4)))

# Plotting the actual vs predicted values for testing data.

x<- 1:length(test_y)
plot(x, test_y, type="l", lwd=2, col="magenta", main="ANN Regression with
Tanh Activation Function", panel.first=grid())
lines(x, y_pred_tanh, lwd=2, col="dodgerblue")
points(x,test_y, pch=16, col="purple")
points(x, y_pred_tanh, pch=16, col="dodgerblue")
legend("topright", c("actual", "predicted"), lty=1, lwd=2,
col=c("purple","dodgerblue"))
```

```
[1] "Prediction accuracy for Tanh activation function"
[1] "within 10%: 0.2053"
[1] "within 15%: 0.3013"
[1] "within 20%: 0.396"
```



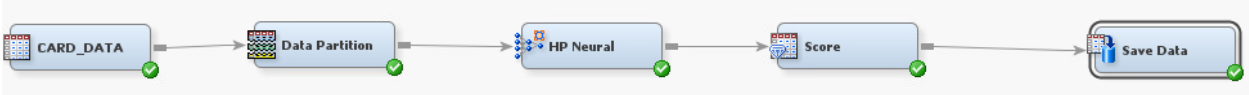

Error: 5.498137  Steps: 15197


ANN Regression with
Tanh Activation Function
actual
predicted
test_y
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
x

Problem 4

SAS and SAS Enterprise Miner

Logistic Activation Function




The SAS System

| accuracy |
| --- |
| 0.955224 |

```
data accuracy;
set tmp1.em_save_test;
match=(em_classification=em_classtarget);
run;

proc sql;
select mean(match) as accuracy
from accuracy;
quit;
```

Tanh Activation Function


The SAS System

| accuracy |
| --- |
| 0.977612 |

```
data accuracy;
 set tmp1.em_save_test;
 match=(em_classification=em_classtarget);
 run;

proc sql;
 select mean(match) as accuracy
 from accuracy;
 quit;
```

Python

Logistic Activation Function

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from statistics import mean
from sklearn import preprocessing
import tensorflow as tf
import keras as keras
from sklearn import metrics

card_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.
csv")

# Scaling the data to fall within [0, 1].

scaler = preprocessing.MinMaxScaler()
scaler_fit = scaler.fit_transform(card_data)
scaled_card = pd.DataFrame(scaler_fit, columns=card_data.columns)

# Splitting data into 80% training and 20% testing sets.

X = scaled_card.iloc[:, 0:7].values
y = scaled_card.iloc[:, 7].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
                                random_state=248561)

# Constructing ANN for binary classification using the sigmoid activation
function in the
```

```
# output layer.

tf.random.set_seed(308618)

sigmoid_model = tf.keras.Sequential([
    tf.keras.layers.Dense(3, activation="relu", input_shape=(7,)),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

sigmoid_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                      loss=['binary_crossentropy'])

sigmoid_model.fit(X_train, y_train)

# Computing prediction accuracy for testing data.

sigmoid_pred = sigmoid_model.predict(X_test)
sigmoid_pred = np.round(sigmoid_model.predict(X_test), 0)

print("Accuracy:", round(metrics.accuracy_score(y_test, sigmoid_pred)*100, 2),
'%')
```

```
    super().__init__(activity_regularizer=activity_regularizer, **kwargs)
  50/50 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.6485
  13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
  13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
  Accuracy: 90.5 %
```

Tanh Activation Function

```
tf.random.set_seed(120460)

tanh_model = tf.keras.Sequential([
    tf.keras.layers.Dense(3, activation="relu", input_shape=(7,)),
    tf.keras.layers.Dense(1, activation="tanh")
])

tanh_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                      loss=['binary_crossentropy'])

tanh_model.fit(X_train, y_train)

# Computing prediction accuracy for testing data.
```

```
tanh_pred = tanh_model.predict(X_test)
tanh_pred = np.round(tanh_model.predict(X_test), 0)

print("Accuracy:", round(metrics.accuracy_score(y_test, tanh_pred)*100, 2), '%')
```

```
    super().__init__(activity_regularizer=activity_regularizer, **kwargs)
  50/50 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.1486
  13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
  13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
  Accuracy: 32.75 %
```

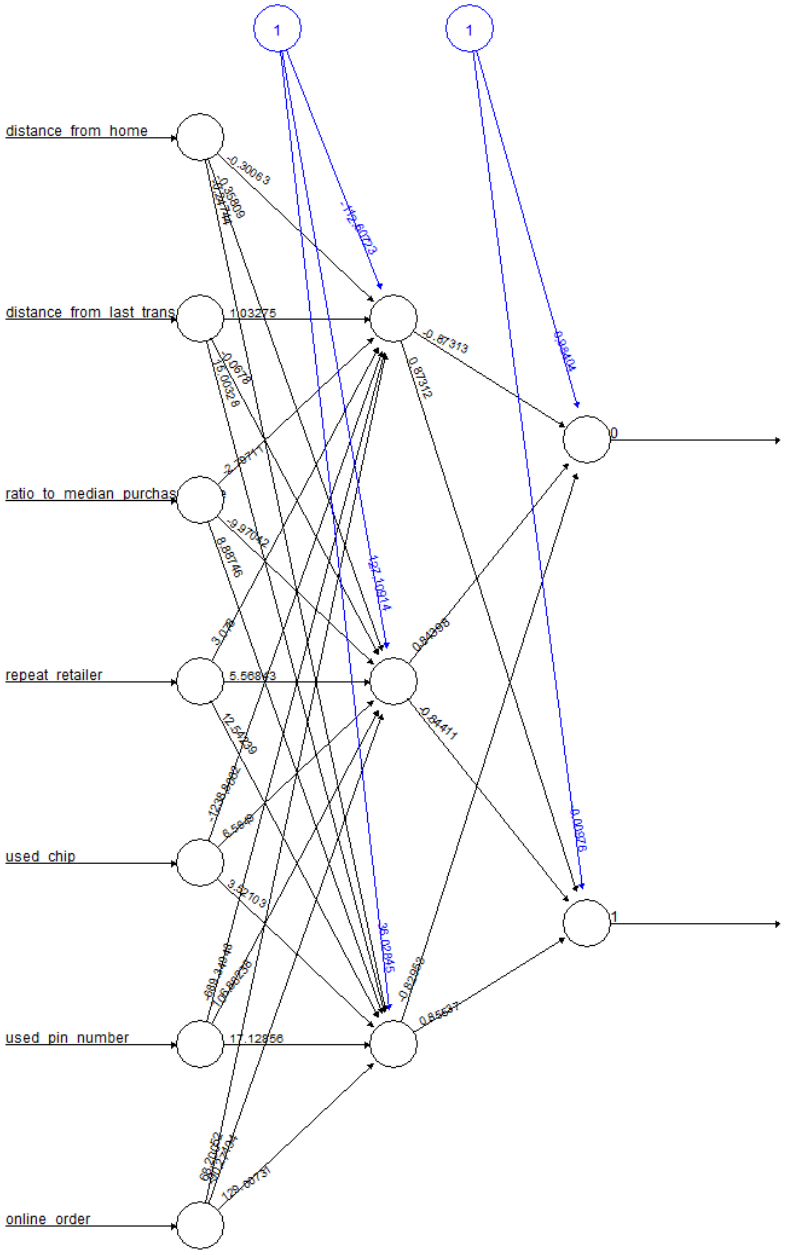
R

Logistic Activation Function

```
library(readr)
library(dplyr)
library(caTools)
library(neuralnet)

card_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.csv
",
header=T, sep=",")

set.seed(111082)

sample = sample.split(card_data, SplitRatio=0.8)
train = subset(card_data, sample==T)
test = subset(card_data, sample==F)

train_x = data.matrix(train[-4])
train_y = data.matrix(train[4])
test_x = data.matrix(test[-4])
test_y = data.matrix(test[4])

# Fitting an ANN for binary classification with logistic activation function.

ann_logistic =
neuralnet(as.factor(fraud)~distance_from_home+distance_from_last_transaction
+ratio_to_median_purchase_price+repeat_retailer+used_chip+used_pin_number
+online_order, data=train, hidden=3, act.fct="logistic", stepmax=1e7)
```

```r
plot(ann_logistic)

# Computing prediction accuracy for testing data.

pred_prob = predict(ann_logistic, test_x)[,1]
pred_y = c()
match = c()
for (i in 1:length(test_y)) {
    pred_y[i] = ifelse(pred_prob[i]>=0.5,1,0)
    match[i] = ifelse(test_y[i]==pred_y[i],1,0)
}

print(paste("Accuracy:", round(mean(match), 4)))
```

```
[1] "Accuracy: 0.874"
```

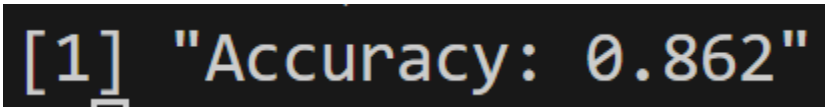


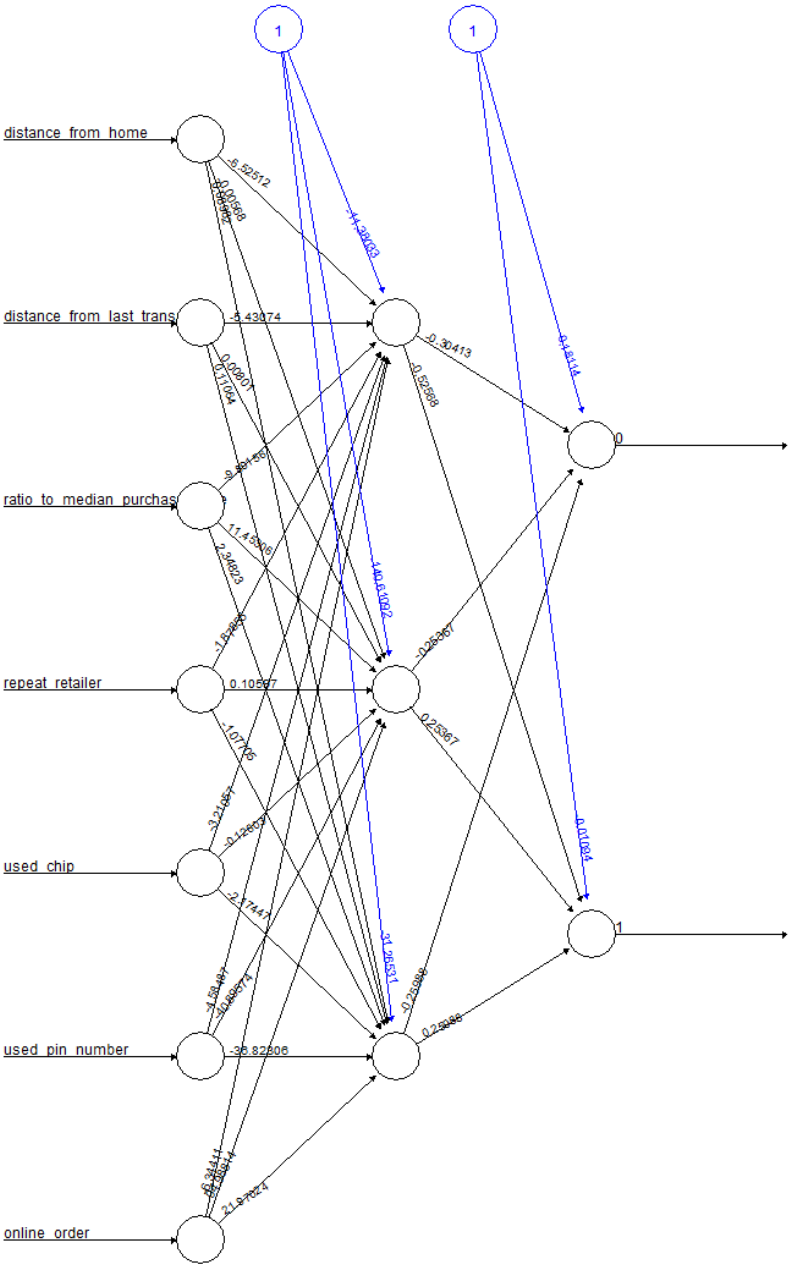

Error: 26.0414 Steps: 65215

Tanh Activation Function

```
# Fitting an ANN binary classifier using the Tanh activation function.

ann_tanh_bin =
neuralnet(as.factor(fraud)~distance_from_home+distance_from_last_transaction
```

```
+ratio_to_median_purchase_price+repeat_retailer+used_chip+used_pin_number
+online_order, data=train, hidden=3, act.fct="tanh", stepmax=1e7)

plot(ann_tanh_bin)

# Computing prediction accuracy for testing data.

pred_prob = predict(ann_tanh_bin, test_x)[,1]
pred_y = c()
match = c()
for (i in 1:length(test_y)){
    pred_y[i] = ifelse(pred_prob[i]>=0.5,1,0)
    match[i] = ifelse(test_y[i]==pred_y[i],1,0)
}

print(paste("Accuracy:", round(mean(match), 4)))
```



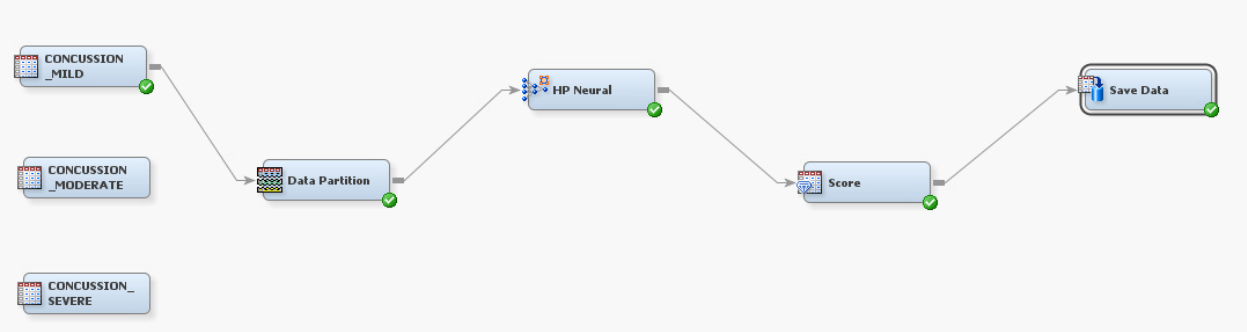


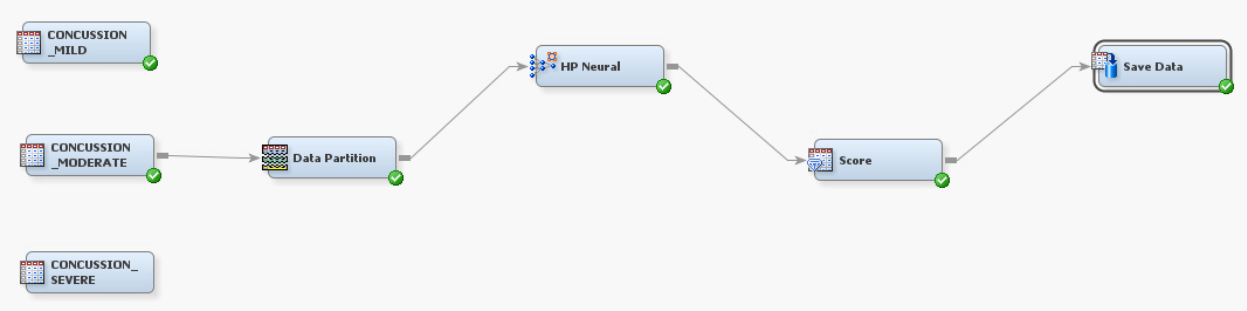

Error: 23.226783 Steps: 53609

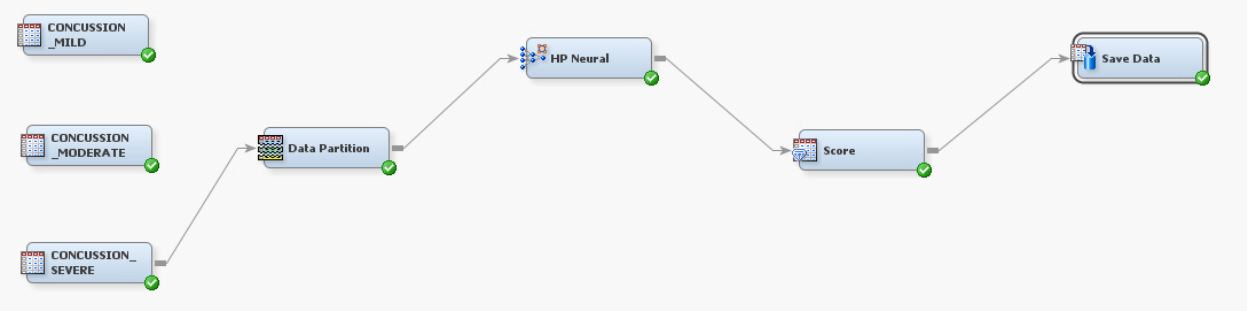
Problem 5

SAS and SAS Enterprise Miner

Logistic Activation Function




**The SAS System**

| accuracy |
| --- |
| 0.242105 |

```
data concussions_data;
set sasuser.Concussion_data;
_dataobs_ = _N_;
run;


proc sort data=concussions_data;
```

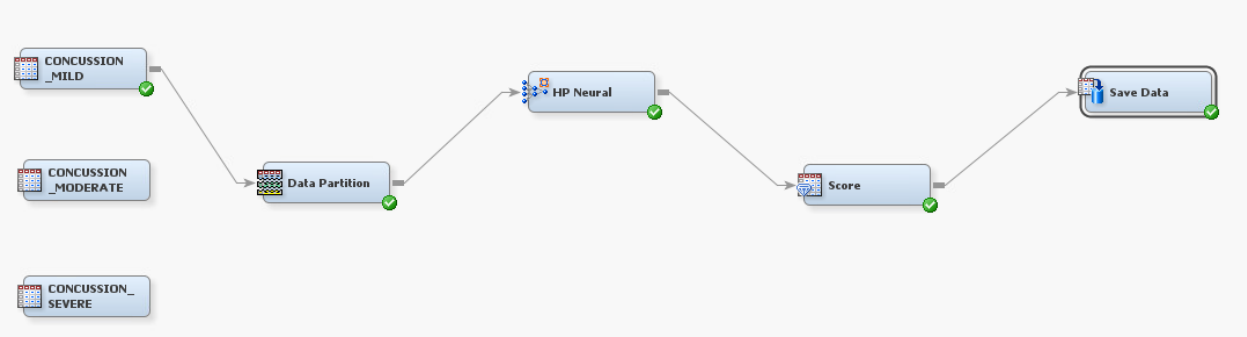
```
by _dataobs_;
run;

data concussion_mild;
set tmp1.mild_results_test;
predprob_mild=em_eventprobability;
class_mild=em_classtarget;
keep _dataobs_ class_mild predprob_mild;
run;

proc sort;
by _dataobs_;
run;

data concussion_moderate;
set tmp2.moderate_results_test;
predprob_moderate=em_eventprobability;
class_moderate=em_classtarget;
keep _dataobs_ class_moderate predprob_moderate;
run;

proc sort;
by _dataobs_;
run;

data concussion_severe;
set tmp3.severe_results_test;
```

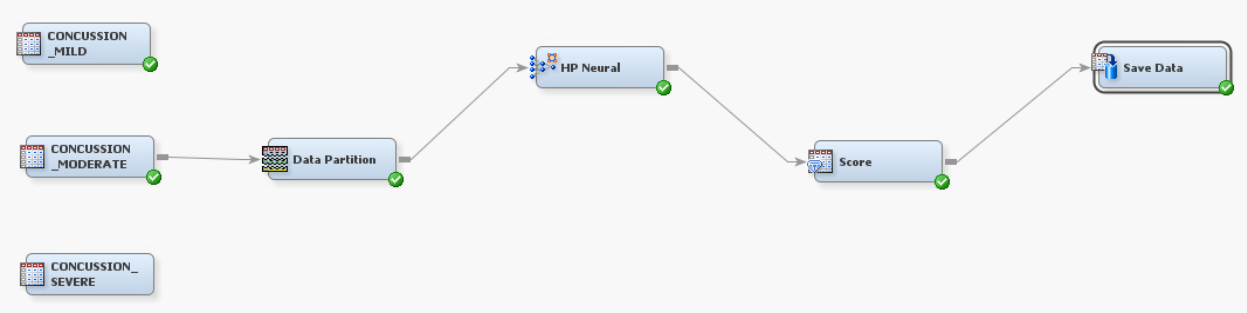
```
predprob_severe=em_eventprobability;
class_severe=em_classtarget;
keep _dataobs_ class_severe predprob_severe;
run;

data all_data;
merge concussions_data concussion_mild concussion_moderate concussion_severe;
by _dataobs_;
if CMISS(predprob_mild, predprob_moderate, predprob_severe)=0;
run;

data all_data;
set all_data;
predprob_max = MAX(predprob_mild, predprob_moderate, predprob_severe);
if (predprob_mild=predprob_max) then pred_class='mild';
if (predprob_moderate=predprob_max) then pred_class='moderate';
if (predprob_severe=predprob_max) then pred_class='severe';
keep concussion pred_class;
run;

data all_data;
set all_data;
match=(concussion=pred_class);
run;

proc sql;
select mean(match) as accuracy
```

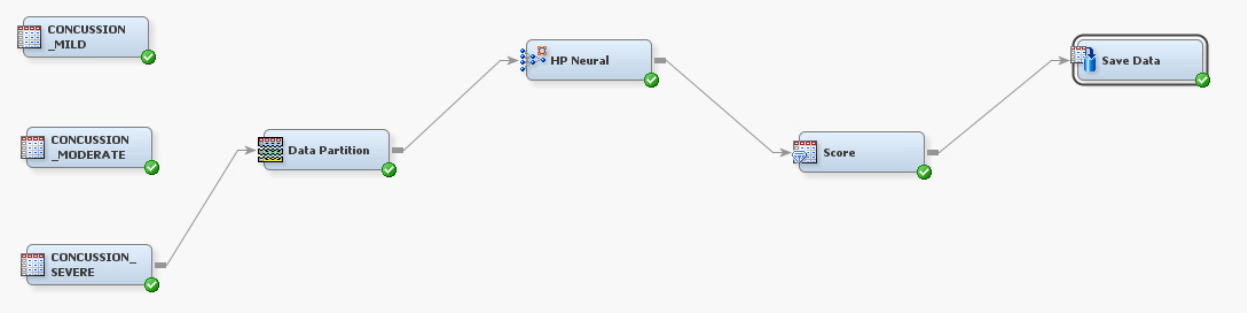
from all_data;

quit;

Tanh Activation Function




The SAS System

| accuracy |
| --- |
| 0.242105 |

data concussions_data;

set sasuser.Concussion_data;

```
_dataobs_ = _N_;
run;

proc sort data=concussions_data;
by _dataobs_;
run;

data concussion_mild;
set tmp1.mild_tanh_test;
predprob_mild=em_eventprobability;
class_mild=em_classtarget;
keep _dataobs_ class_mild predprob_mild;
run;

proc sort;
by _dataobs_;
run;

data concussion_moderate;
set tmp2.moderate_tanh_test;
predprob_moderate=em_eventprobability;
class_moderate=em_classtarget;
keep _dataobs_ class_moderate predprob_moderate;
run;

data concussion_severe;
set tmp3.severe_tanh_test;
```

```
predprob_severe=em_eventprobability;
class_severe=em_classtarget;
keep _dataobs_ class_severe predprob_severe;
run;

proc sort;
by _dataobs_;
run;

data all_data;
merge concussions_data concussion_mild concussion_moderate concussion_severe;
by _dataobs_;
if CMISS(predprob_mild, predprob_moderate, predprob_severe)=0;
run;

data all_data;
set all_data;
predprob_max=MAX(predprob_mild, predprob_moderate, predprob_severe);
if (predprob_mild=predprob_max) then pred_class='mild';
if (predprob_moderate=predprob_max) then pred_class='moderate';
if (predprob_severe=predprob_max) then pred_class='severe';
keep concussion pred_class;
run;

data all_data;
set all_data;
match=(concussion=pred_class);
```

run;

proc sql;

select mean(match) as accuracy

from all_data;

quit;

Python

Logistic Activation Function

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import preprocessing
import tensorflow as tf
import keras as keras
from statistics import mean

concussion_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_dat
a.csv")
position_code = {'Offensive Lineman':0, 'Cornerback':1, 'Running Back':2,
'Quarterback':3, 'Wide Receiver':4}
concussion_code = {'mild':0, 'moderate':1, 'severe':2}
concussion_data['position'] = concussion_data['position'].map(position_code)
concussion_data['concussion'] =
concussion_data['concussion'].map(concussion_code)
X = concussion_data.drop(['concussion'], axis=1)
y = concussion_data.drop(['age', 'nyearsplaying', 'position', 'prevconc'],
axis=1)

# Scaling the data to fall within [0, 1].

scaler = preprocessing.MinMaxScaler()
scaler_fit = scaler.fit_transform(X)
scaled_concussion = pd.DataFrame(scaler_fit, columns=['age', 'nyearsplaying',
'position', 'prevconc'])
new_data = pd.concat([scaled_concussion, y], axis=1)
```

```
# Splitting data into 80% training and 20% testing sets.

X = new_data.iloc[:, 0:4].values
y = new_data.iloc[:, 4].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
                                random_state=123055)

# Constructing ANN model for multinomial classification. Using the sigmoid
# activation function in the output layer.

tf.random.set_seed(210572)

sigmoid_model = tf.keras.Sequential([
    tf.keras.layers.Dense(3, activation="relu", input_shape=(4,)),
    tf.keras.layers.Dense(1, activation="softmax")
])

sigmoid_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                      loss=['categorical_crossentropy'])

sigmoid_model.fit(X_train, y_train)

# Computing prediction accuracy for testing data.

sigmoid_prob = sigmoid_model.predict(X_test)
sigmoid_pred = pd.DataFrame(sigmoid_prob, columns=['predicted'])
y_test = pd.DataFrame(y_test, columns=['concussion'])
df = pd.concat([sigmoid_pred, y_test], axis=1)
match = []
for i in range(len(df)):
    if df['concussion'][i] == df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print("Accuracy:", round(mean(match), 4))
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0000e+00
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
Accuracy: 0.4571
```

Tanh Activation Function

```
tf.random.set_seed(622904)
```

```
tanh_model = tf.keras.Sequential([
    tf.keras.layers.Dense(3, activation="tanh", input_shape=(4,)),
    tf.keras.layers.Dense(1, activation="softmax")
])

tanh_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                   loss=['categorical_crossentropy'])

tanh_model.fit(X_train, y_train)

# Computing prediction accuracy for testing data.

tanh_pred = pd.DataFrame(tanh_model.predict(X_test), columns=['predicted'])
y_test = pd.DataFrame(y_test, columns=['concussion'])
df2 = pd.concat([tanh_pred, y_test], axis=1)
match2 = []
for i in range(len(df2)):
    if df2['concussion'][i] == df2['predicted'][i]:
        match2.append(1)
    else:
        match2.append(0)

print("Accuracy:", round(mean(match2), 4))
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0000e+00
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
Accuracy: 0.4571
```

R

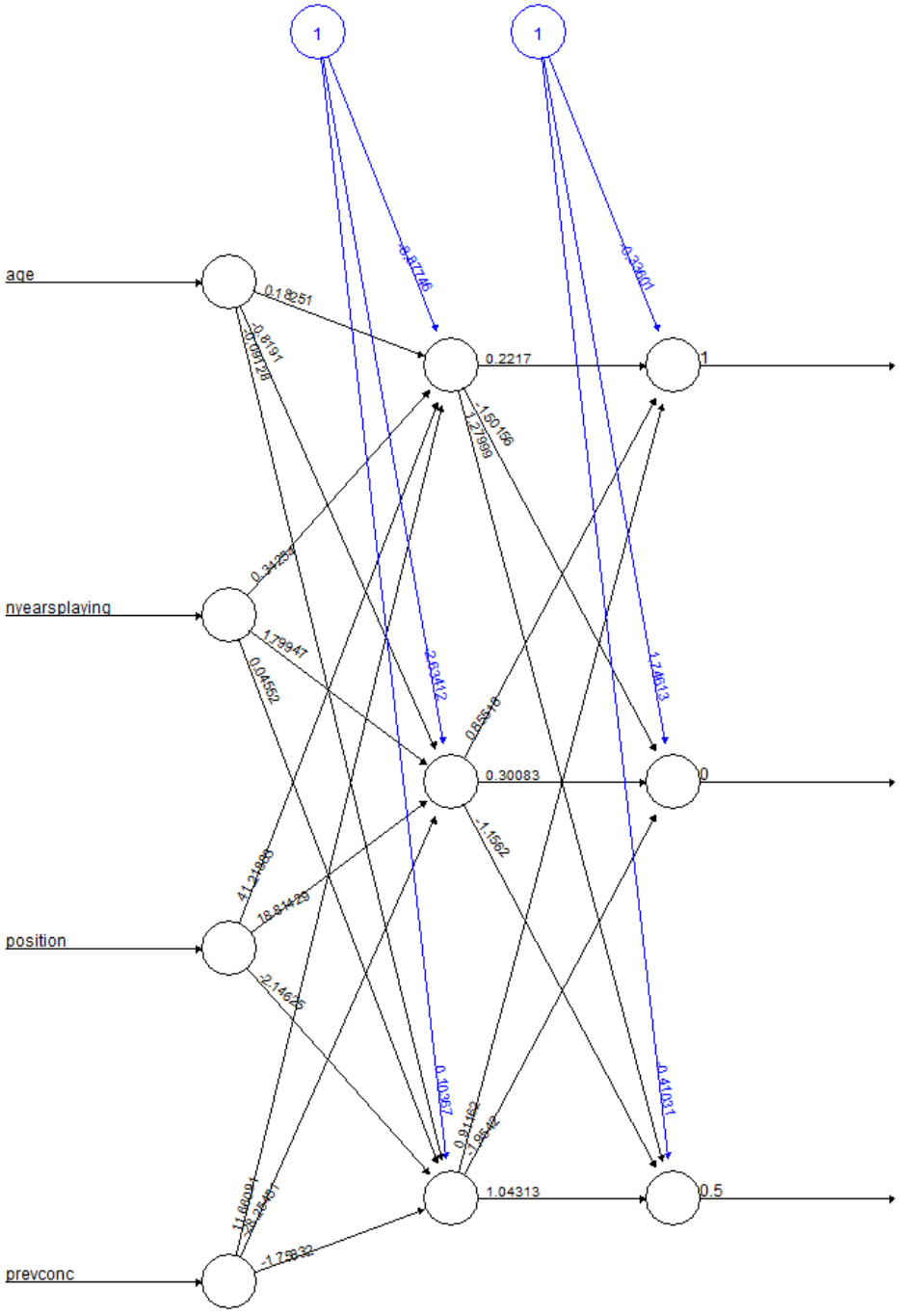
Logistic Activation Function

```
library(readr)
library(dplyr)
library(neuralnet)

concussion_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_data.csv",
header=T, sep=",")

concussion_data$position = ifelse(concussion_data$position=='Offensive Lineman',
1,
ifelse(concussion_data$position=='Cornerback', 2,
```

```
ifelse(concussion_data$position=='Wide Receiver', 3,
ifelse(concussion_data$position=='Runningback',4,5))))

concussion_data$concussion = ifelse(concussion_data$concussion=='mild',1,
ifelse(concussion_data$concussion=='moderate', 2, 3))

scale01 <- function(x){
  (x-min(x))/(max(x)-min(x))
}

concussion_data = concussion_data %>% mutate_all(scale01)

# Splitting data into 80% training and 20% testing sets.

set.seed(273194)
sample = sample(c(T,F), nrow(concussion_data), replace=T,
prob=c(0.8, 0.2))
train = concussion_data[sample,]
test = concussion_data[!sample,]

train_x = data.matrix(train[-5])
train_y = data.matrix(train[5])
test_x = data.matrix(test[-5])
test_y = data.matrix(test[5])

# Fitting an ANN Multinomial Classifier with logistic activation function.

ann_log_multi = neuralnet(as.factor(concussion)~age+nyearsplaying+position
+prevconc, data=train, hidden=3, act.fct="logistic", stepmax=1e7)

plot(ann_log_multi)

# Computing prediction accuracy for testing data.

pred_prob = predict(ann_log_multi, test_x)
pred_prob = as.data.frame(pred_prob)

colnames(pred_prob) = c(0, 0.5, 1)

pred_class = apply(pred_prob, 1, function(x) colnames(pred_prob)[which.max(x)])

match = c()
for (i in 1:length(test_y)) {
    match[i] = ifelse(pred_class[i]==as.character(test_y[i]), 1, 0)
}
```

```
print(paste("Accuracy:", round(mean(match), 4)))
```

```
[1] "Accuracy: 0.8468"
```



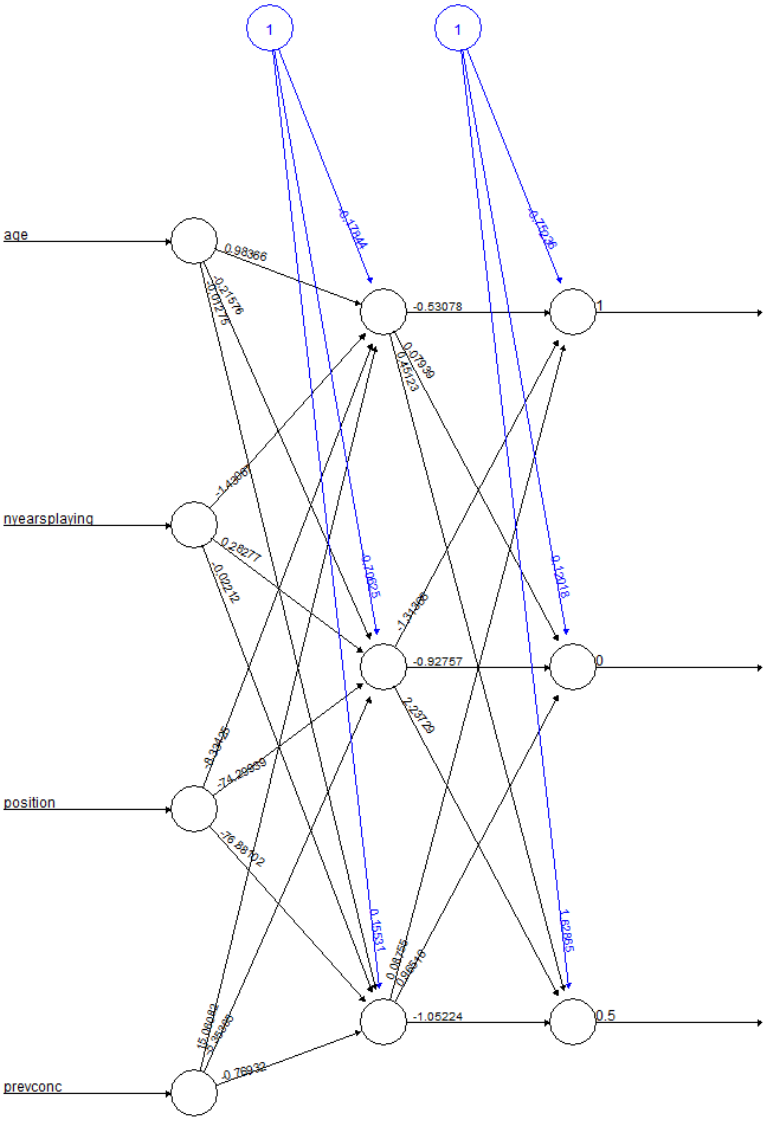

Error: 24.177906 Steps: 10001

Tanh Activation Function

```
library(readr)
library(dplyr)
library(neuralnet)

concussion_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_data.c
sv",
header=T, sep=",")

concussion_data$position = ifelse(concussion_data$position=='Offensive Lineman',
1,
ifelse(concussion_data$position=='Cornerback', 2,
ifelse(concussion_data$position=='Wide Receiver', 3,
ifelse(concussion_data$position=='Runningback',4,5))))

concussion_data$concussion = ifelse(concussion_data$concussion=='mild',1,
ifelse(concussion_data$concussion=='moderate', 2, 3))

scale01 <- function(x){
  (x-min(x))/(max(x)-min(x))
}

concussion_data = concussion_data %>% mutate_all(scale01)

# Splitting data into 80% training and 20% testing sets.

set.seed(273194)
sample = sample(c(T,F), nrow(concussion_data), replace=T,
prob=c(0.8, 0.2))
train = concussion_data[sample,]
test = concussion_data[!sample,]

train_x = data.matrix(train[-5])
train_y = data.matrix(train[5])
test_x = data.matrix(test[-5])
test_y = data.matrix(test[5])

# Fitting an ANN Multinomial Classifier with logistic activation function.

#ann_log_multi = neuralnet(as.factor(concussion)~age+nyearsplaying+position
#+prevconc, data=train, hidden=3, act.fct="logistic", stepmax=1e7)

#plot(ann_log_multi)
```

```
# Computing prediction accuracy for testing data.

#pred_prob = predict(ann_log_multi, test_x)
#pred_prob = as.data.frame(pred_prob)

#colnames(pred_prob) = c(0, 0.5, 1)

#pred_class = apply(pred_prob, 1, function(x) colnames(pred_prob)[which.max(x)])

#match = c()
#for (i in 1:length(test_y)) {
    #match[i] = ifelse(pred_class[i]==as.character(test_y[i]), 1, 0)
#}

#print(paste("Accuracy:", round(mean(match), 4)))

################################################################################

# Fitting an ANN Multinomial Classifier with tanh activation function.

ann_tanh_multi = neuralnet(as.factor(concussion)~age+nyearsplaying+position
+prevconc, data=train, hidden=3, act.fct="tanh", stepmax=1e7)

plot(ann_tanh_multi)

# Computing prediction accuracy for testing data.

pred_prob = predict(ann_tanh_multi, test_x)
pred_prob = as.data.frame(pred_prob)

colnames(pred_prob) = c(0, 0.5, 1)

pred_class = apply(pred_prob, 1, function(x) colnames(pred_prob)[which.max(x)])

match = c()
for (i in 1:length(test_y)) {
    match[i] = ifelse(pred_class[i]==as.character(test_y[i]), 1, 0)
}

print(paste("Accuracy:", round(mean(match), 4)))
```

```
[1] "Accuracy: 0.8468"
```




Error: 24.886674  Steps: 22105