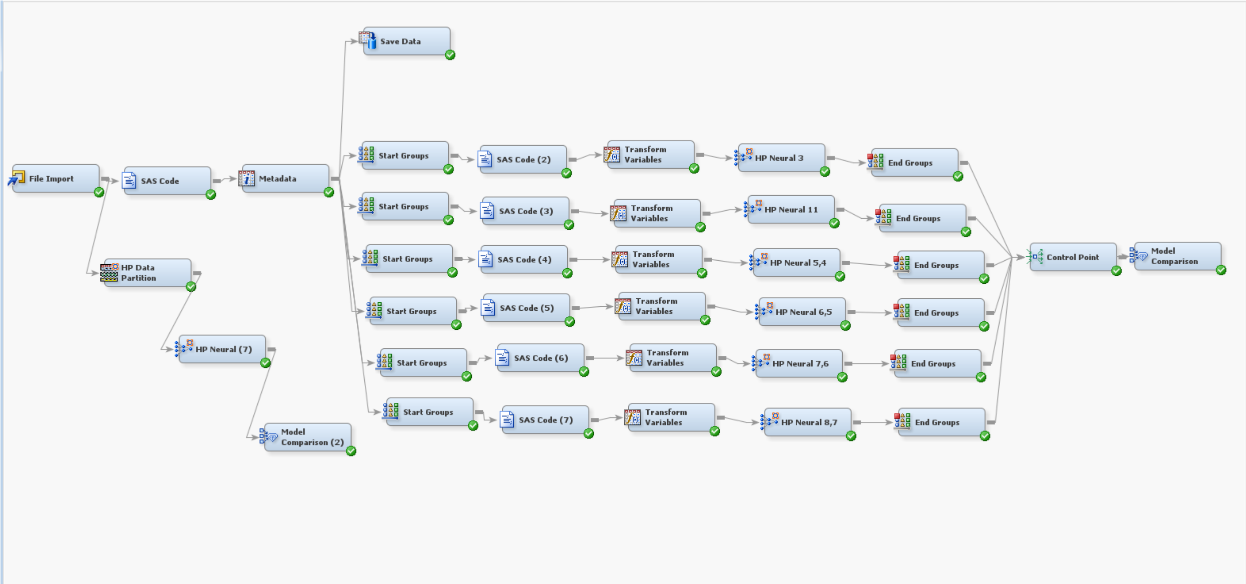
STAT 656
Homework 5

Name-Mayank Jaggi

UIN-526005299

PART 1 SAS EM

1) Project Diagram



2)

SAS Code 1

Training Code

```
data mylib.selection;
call streaminit(12345);
set &em_import_data;
urand = rand('uniform');
proc sort data=mylib.selection;
by urand;
data &em_export_train;
drop fold_size urand;
set mylib.selection NOBS=nobs_;
fold_size = round(nobs_ /4.0);
if _N_ <= fold_size then fold='A';
if _N_ > fold_size and _N_ <=2*fold_size then fold='B';
if _N_ > 2*fold_size and _N_ <=3*fold_size then fold='C';
if _N_ > 3*fold_size then fold='D';
proc means data=&em_export_train;
by fold;
var result;
run;
```

SAS Code 2

Training Code

```
data mylib.templ;
retain cl c2 c3 c4 0;
keep cl c2 c3 c4;
set &em_import_data end=eof;
if fold='A' then cl = cl + 1;
if fold='B' then c2 = c2 + 1;
if fold='C' then c3 = c3 + 1;
if fold='D' then c4 = c4 + 1;
if eof then output;
data &em_export_validate;
drop cl c2 c3 c4 rfold;
retain rfold '0';
set mylib.AllData_Train;
if rfold ='0' then do;
set mylib.templ;
if cl=0 then rfold='A';
if c2=0 then rfold='B';
if c3=0 then rfold='C';
if c4=0 then rfold='D';
end;
if fold= rfold then output;
run;
```

3) Metrics for cv=4

3	Predicted Negative	Predicted Positive
Actual Negative	146	154
Actual Positive	85	615

Accuracy	0.761
Precision	0.799739922
Recall	0.878571429
F1	0.837304289

6,5	Predicted Negative	Predicted Positive
Actual Negative	162	138
Actual Positive	100	600

Accuracy	0.762
Precision	0.81300813
Recall	0.857142857
F1	0.83449235

11	Predicted Negative	Predicted Positive
Actual Negative	145	155
Actual Positive	88	612

Accuracy	0.757
Precision	0.79791395
Recall	0.874285714
F1	0.834355828

7,6	Predicted Negative	Predicted Positive
Actual Negative	168	132
Actual Positive	112	588

Accuracy	0.756
Precision	0.816666667
Recall	0.84
F1	0.828169014

5,4	Predicted Negative	Predicted Positive
Actual Negative	155	145
Actual Positive	99	601

Accuracy	0.756
Precision	0.805630027
Recall	0.858571429
F1	0.831258645

8,7	Predicted Negative	Predicted Positive
Actual Negative	163	137
Actual Positive	106	594

Accuracy	0.757
Precision	0.812585499
Recall	0.848571429
F1	0.830188679

4) **We select the network with a hidden layer combination of 6,5 because it has the highest accuracy.**

5) **Metrics for the validation set after 70/30 partition**

Validation	Predicted Negative	Predicted Positive
Actual Negative	29	40
Actual Positive	17	144

Accuracy	0.752173913
Precision	0.782608696
Recall	0.894409938
F1	0.834782609

PART 2
##1 PYTHON PROGRAM

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 20 10:02:26 2019

@author: mayank
"""

from AdvancedAnalytics import NeuralNetwork
from AdvancedAnalytics import ReplaceImputeEncode
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import math


df2 = pd.read_excel("CreditHistory_Clean.xlsx")     #data file name


df2.rename(columns={'telephon':'telephone'},inplace = True)   # renaming attribute


attribute_map = {
        'age':['I', (19, 120)],
        'amount':['I', (0, 20000)],
        'checking':['N',(1,2,3,4)],
        'coapp':['N',(1, 2, 3)],
        'depends':['B',(1, 2)],
        'duration':['I',(1,72)],
        'employed':['N',(1,2,3,4,5)],
        'existcr':['N',(1,2,3,4)],
        'foreign':['B', (1,2)],
        'good_bad':['B', ('bad','good')],
        'history':['N', (0,1,2,3,4)],
        'housing':['N', (1,2,3)],
        'installp':['N', (1,2,3,4)],
        'job':['N', (1,2,3,4)],
        'marital':['N', (1,2,3,4)],
        'other':['N', (1,2,3)],
        'property':['N', (1,2,3,4)],
        'purpose':['N',('0','1','2','3','4','5','6','8','9','X')],
        'resident':['N', (1,2,3,4)],
        'savings':['N', (1,2,3,4,5)],
        'telephone':['B', (1,2)] }
```

```
# Data Preprocessing, Replace outlier, impute missing values and encode
rie = ReplaceImputeEncode(data_map=attribute_map,
nominal_encoding='one-hot',interval_scale=None, drop=False, display=True)
# Now request replace-impute-encode for your dataframe
encoded_df = rie.fit_transform(df2)
print("\nData after replacing outliers, impute missing and encoding:")
print(encoded_df.head())


# Defining target and input variables

y = encoded_df['good_bad']   #target
x = encoded_df.drop('good_bad',axis=1)  #input
np_y=np.ravel(y)     # convertig  y into flat array

# 4 fold Cross validation
list1 = ['accuracy','recall', 'precision', 'f1']
network = [(3),(11),(5,4),(6,5),(7,6),(8,7)]
max_f1 = 0
best_network = (0)


for nn in network:
    fnn = MLPClassifier(hidden_layer_sizes=nn, activation='relu', solver='lbfgs',
max_iter=2000,tol=1e-64,random_state=12345)
    fnn=fnn.fit(x,np_y)
    print("\nNetwork with Hidden_Layer_Sizes=", nn)
    mean_score=[]
    std_score=[]
    for i in range(len(list1)):
        fnn_4 = cross_val_score(fnn, x, np_y, cv=4, scoring=list1[i])
        mean_score.append(fnn_4.mean())
        std_score.append (fnn_4.std())
    print("{:.<13s}{:>6s}{:>13s}".format("Metric", "Mean", "Std. Dev."))
    for i in range(len(list1)):
        mean=math.fabs(mean_score[i])
        std=std_score[i]
        print("{:.<13s}{:>7.4f}{:>10.4f}".format(list1[i], mean, std))
    if mean>max_f1:      #ideally we want f1 to be 1
        max_f1=mean
        best_network = nn
print("\n{:<23s}{:>5.3f}{:<7s}".format("The highest f1 is",max_f1, " for
hidden_layer_sizes="), best_network)


# Evaluating the neural network with the best network

x_train, x_validate, y_train, y_validate = train_test_split(x, y, test_size=0.3,
random_state=12345)
```

```
fnn1 =
MLPClassifier(hidden_layer_sizes=best_network,activation='relu',solver='lbfgs',
max_iter=2000,tol=1e-64,random_state=12345)
fnn1 = fnn1.fit(x_train, y_train)
print("\nTraining Data\nRandom Selection of 70% of Original Data")
NeuralNetwork.display_binary_split_metrics(fnn1, x_train,
y_train,x_validate,y_validate)
```

##2 Table of the metrics

```
Network with Hidden_Layer_Sizes= 3
Metric.......  Mean    Std. Dev.
accuracy..... 0.7550    0.0203
recall....... 0.8757    0.0148
precision.... 0.7959    0.0231
f1........... 0.8336    0.0116

Network with Hidden_Layer_Sizes= 11
Metric.......  Mean    Std. Dev.
accuracy..... 0.7000    0.0000
recall....... 1.0000    0.0000
precision.... 0.7000    0.0000
f1........... 0.8235    0.0000

Network with Hidden_Layer_Sizes= (5, 4)
Metric.......  Mean    Std. Dev.
accuracy..... 0.7000    0.0000
recall....... 1.0000    0.0000
precision.... 0.7000    0.0000
f1........... 0.8235    0.0000

Network with Hidden_Layer_Sizes= (6, 5)
Metric.......  Mean    Std. Dev.
accuracy..... 0.7510    0.0337
recall....... 0.9086    0.0283
precision.... 0.7772    0.0367
f1........... 0.8367    0.0174

Network with Hidden_Layer_Sizes= (7, 6)
Metric.......  Mean    Std. Dev.
accuracy..... 0.7000    0.0000
recall....... 1.0000    0.0000
precision.... 0.7000    0.0000
f1........... 0.8235    0.0000

Network with Hidden_Layer_Sizes= (8, 7)
```

Metric.......	Mean	Std. Dev.
accuracy.....	0.7240	0.0185
recall.......	0.9471	0.0187
precision....	0.7356	0.0177
f1...........	0.8278	0.0097

##3 Model selection

The highest f1 is 0.837 for hidden_layer_sizes= (6, 5)

##4 Table of metrics for 70/30 training, validation data

Training Data
Random Selection of 70% of Original Data

Model Metrics..........	Training	Validation
Observations...........	700	300
Features...............	68	68
Number of Layers.......	2	2
Number of Outputs......	1	1
Number of Neurons......	11	11
Number of Weights......	455	455
Number of Iterations...	608	608
Activation Function....	logistic	logistic
Mean Absolute Error....	0.3078	0.3008
Avg Squared Error......	0.1563	0.1555
Accuracy...............	0.7657	0.7700
Precision..............	0.8031	0.8283
Recall (Sensitivity)...	0.8703	0.8694
F1-score...............	0.8353	0.8484
MISC (Misclassification)...	23.4%	23.0%
class 0..............	45.9%	51.3%
class 1..............	13.0%	13.1%

Training

Confusion Matrix	Class 0	Class 1
Class 0.....	120	102
Class 1.....	62	416

Validation

Confusion Matrix	Class 0	Class 1
Class 0.....	38	40
Class 1.....	29	193