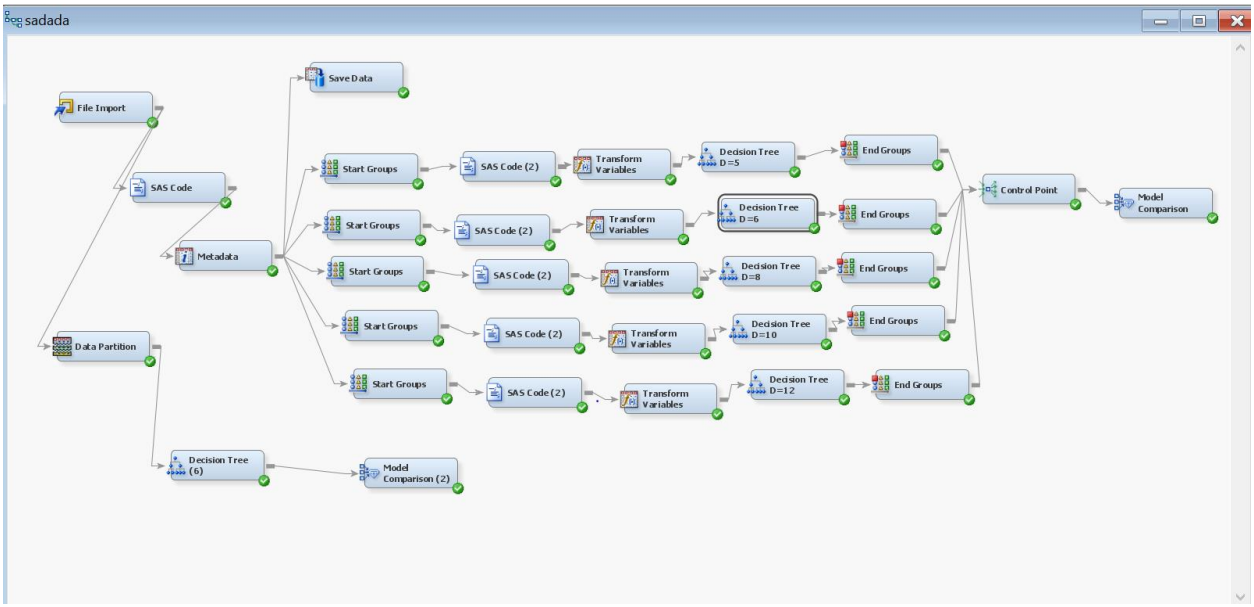
Part 1 – SAS



2. A screen shot or listing of ALL SAS code used in your diagram.

CODE 1

```
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_ /10.0);
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 and _N_ <=4*fold_size then fold='D';
if _N_ > 4*fold_size and _N_ <=5*fold_size then fold='E';
if _N_ > 5*fold_size and _N_ <=6*fold_size then fold='F';
if _N_ > 6*fold_size and _N_ <=7*fold_size then fold='G';
if _N_ > 7*fold_size and _N_ <=8*fold_size then fold='H';
if _N_ > 8*fold_size and _N_ <=9*fold_size then fold='I';
if _N_ > 9*fold_size then fold='J';
```

```
proc means data=&em_export_train;
by fold;
var result;
run;
```

CODE 2

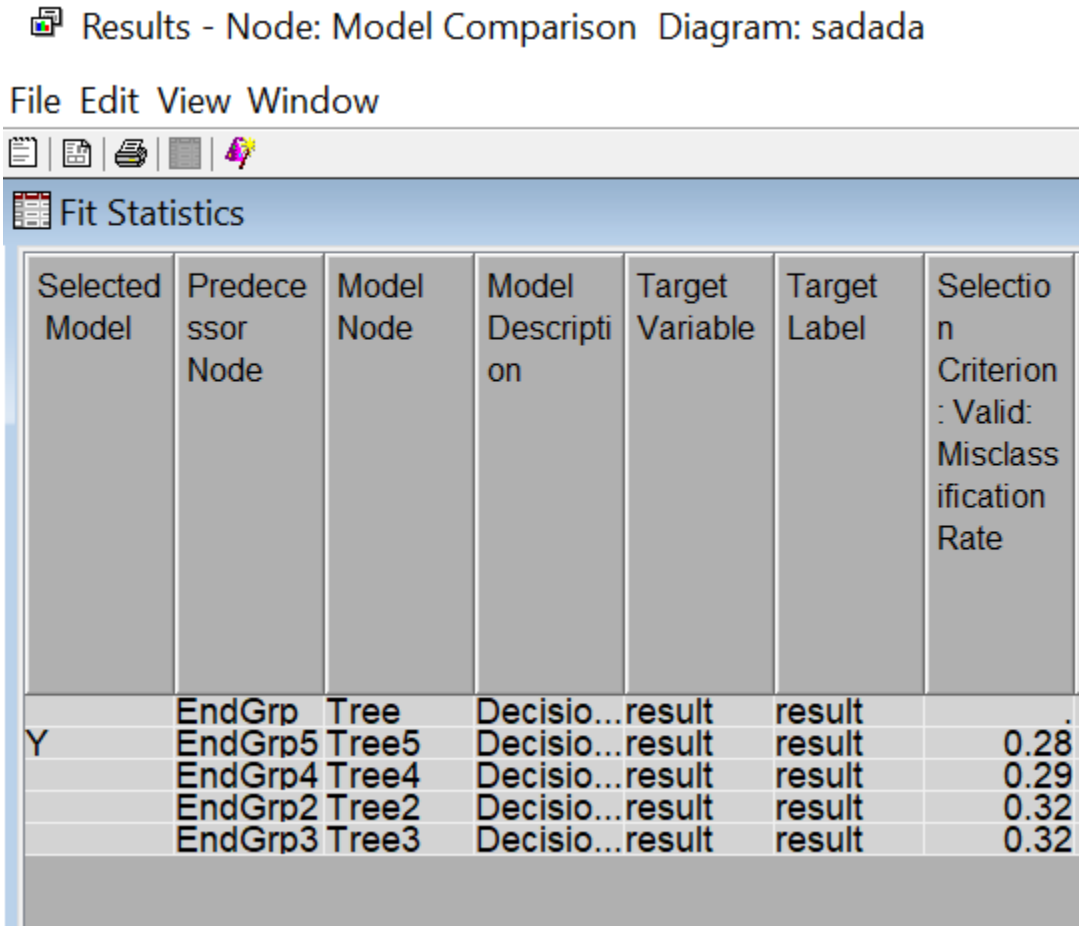
```
data mylib.temp1;
retain c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 0;
keep c1 c2 c3 c4 c5 c6 c7 c8 c9 c10;
set &em_import_data end=eof;
if fold='A' then c1 = c1 + 1;
if fold='B' then c2 = c2 + 1;
if fold='C' then c3 = c3 + 1;
if fold='D' then c4 = c4 + 1;
if fold='E' then c5 = c5 + 1;
if fold='F' then c6 = c6 + 1;
if fold='G' then c7 = c7 + 1;
if fold='H' then c8 = c8 + 1;
if fold='I' then c9 = c9 + 1;
if fold='J' then c10 = c10 + 1;
if eof then output;
data &em_export_validate;
drop c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 rfold;
retain rfold '0';
set mylib.AllData_Train;
if rfold ='0' then do;
        set mylib.temp1;
        if c1=0 then rfold='A';
        if c2=0 then rfold='B';
        if c3=0 then rfold='C';
        if c4=0 then rfold='D';
        if c5=0 then rfold='E';
        if c6=0 then rfold='F';
        if c7=0 then rfold='G';
        if c8=0 then rfold='H';
        if c9=0 then rfold='I';
        if c10=0 then rfold='J';
end;
if fold= rfold then output;
run;
```

3. A table of the metrics for each of the 10 cross-validation folds & 5. A table of the same metric for the 70/30 test of your selected model.

Depth=5	Predicted Negative	Predicted Positive	Depth=6	Predicted Negative	Predicted Positive	Depth=8	Predicted Negative	Predicted Positive
Actual Negative	142	158	Actual Negative	137	163	Actual Negative	157	143
Actual Positive	82	618	Actual Positive	60	640	Actual Positive	70	630

Depth=10	Predicted Negative	Predicted Positive	Depth=12	Predicted Negative	Predicted Positive	Validation	Predicted Negative	Predicted Positive
Actual Negative	161	139	Actual Negative	167	133	Actual Negative	169	131
Actual Positive	72	628	Actual Positive	70	630	Actual Positive	75	625

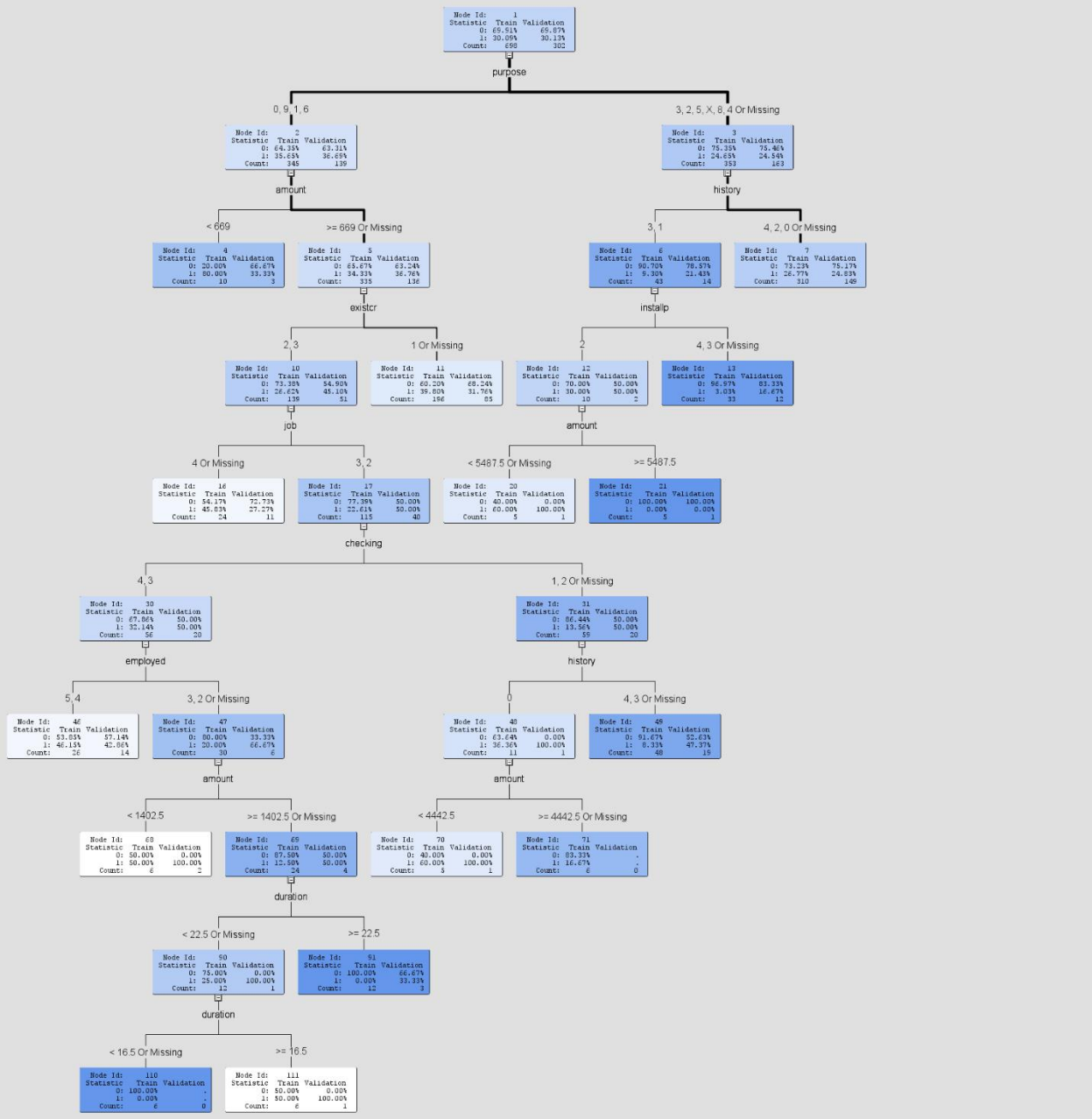
4. Describe which model you selected and why.



Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Misclassification Rate
	EndGrp	Tree	Decisio...	result	result	.
Y	EndGrp5	Tree5	Decisio...	result	result	0.28
	EndGrp4	Tree4	Decisio...	result	result	0.29
	EndGrp2	Tree2	Decisio...	result	result	0.32
	EndGrp3	Tree3	Decisio...	result	result	0.32

We select the model with the lowest misclassification rate, i.e. With a decision tree of Depth=12

6. A screen shot of your tree



Python Part

1. A listing of your python code.

```
from AdvancedAnalytics import DecisionTree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from sklearn.model_selection import cross_val_score
from pydotplus import graph_from_dot_data
from sklearn.model_selection import train_test_split
import graphviz as show_tree
import pandas as pd
import numpy as np

df = pd.read_excel(r"C:\Users\haris\Documents\stat 656\week4\CreditHistory_Clean.xlsx")

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)],
        'telephon':['B', (1,2)] }

rie = ReplaceImputeEncode(data_map=attribute_map, interval_scale=None, nominal_encoding='one-hot',
df_rie = rie.fit_transform(df)
print("\nData after replacing outliers, imputing missing and encoding:")
print(df_rie.head())

#good_bad is the name of the binary target
varlist = ['good_bad']
X = np.asarray(df_rie.drop(varlist, axis=1))
y = np.asarray(df_rie['good_bad'])

#10-fold CV
score_list = ['accuracy', 'recall', 'precision', 'f1']
search_depths = [5,6,7,8,10,12,15,20,25]
for d in search_depths:
    dtc = DecisionTreeClassifier(criterion='gini', max_depth=d, min_samples_split=5, min_samples_le
    mean_score = []
```

```
    std_score = []
    print("max_depth=", d)
    print("{:.<13s}{:>6s}{:>13s}".format("Metric", "Mean", "Std. Dev."))
    for s in score_list:
        dtc_10 = cross_val_score(dtc, X, y, scoring=s, cv=10)
        mean = dtc_10.mean()
        std = dtc_10.std()
        mean_score.append(mean)
        std_score.append(std)
        print("{:.<13s}{:>7.4f}{:>10.4f}".format(s, mean, std))

#the optimum decision tree
dtc = DecisionTreeClassifier(criterion='gini', max_depth=5,min_samples_split=5, min_samples_leaf=5)
X_train, X_validate, y_train, y_validate = \
           train_test_split(X,y,test_size = 0.3, random_state=7)
dtc = dtc.fit(X_train,y_train)

classes = [ 'good','bad']
col = rie.col
col.remove('good_bad')
DecisionTree.display_importance(dtc, col)
DecisionTree.display_binary_split_metrics(dtc, X_train, y_train, X_validate, y_validate)

'''
### TEXTBOOK WAY OF GETTINGA TREE - NOT WORKING

dot_data = export_graphviz(dtc, filled=True, rounded=True, \
                           class_names=classes, feature_names = col, out_file=None)

'''
### TEXTBOOK WAY OF GETTINGA TREE - NOT WORKING

dot_data = export_graphviz(dtc, filled=True, rounded=True, \
                           class_names=classes, feature_names = col, out_file=None)
#write tree to png file 'homework_tree'
graph_png = graph_from_dot_data(dot_data)
graph_path = r'C:\Users\haris\Documents\stat 656\week4'
graph_png.write_png(r"C:\Users\haris\Documents\stat 656\week4\homework_tree.png")
graph_pdf = graphviz.Source(dot_data)
graph_pdf.view("tree")
'''
# from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
featureNames=df_rie[0:68]
export_graphviz(dtc,out_file=dot_data, class_names= ['1:Good','0:Bad'], filled=True, rounded=True,
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

2. A table of the metrics (recall, accuracy, precision and F1) calculated for each of your 10 cross-validation folds.

```
    ...:
max_depth= 5
Metric.......  Mean    Std. Dev.
accuracy..... 0.7180    0.0260
recall....... 0.8700    0.0358
precision.... 0.7653    0.0392
f1.......... 0.8150    0.0132
max_depth= 6
Metric.......  Mean    Std. Dev.
accuracy..... 0.7080    0.0209
recall....... 0.8414    0.0517
precision.... 0.7723    0.0449
f1.......... 0.8027    0.0094
max_depth= 7
Metric.......  Mean    Std. Dev.
accuracy..... 0.7060    0.0353
recall....... 0.8343    0.0453
precision.... 0.7695    0.0424
f1.......... 0.7993    0.0211
```

```
max_depth= 8
Metric.......  Mean    Std. Dev.
accuracy..... 0.7020    0.0264
recall....... 0.8057    0.0434
precision.... 0.7822    0.0369
f1.......... 0.7918    0.0194
max_depth= 10
Metric.......  Mean    Std. Dev.
accuracy..... 0.7060    0.0422
recall....... 0.8014    0.0569
precision.... 0.7895    0.0442
f1.......... 0.7919    0.0276
max_depth= 12
Metric.......  Mean    Std. Dev.
accuracy..... 0.6950    0.0408
recall....... 0.7743    0.0538
precision.... 0.7867    0.0343
f1.......... 0.7804    0.0321
max_depth= 15
Metric.......  Mean    Std. Dev.
accuracy..... 0.7000    0.0293
recall....... 0.7743    0.0538
precision.... 0.7935    0.0299
f1.......... 0.7844    0.0240
```

```
max_depth= 20
Metric.......  Mean    Std. Dev.
accuracy..... 0.7030    0.0361
recall....... 0.7800    0.0516
precision.... 0.7982    0.0294
f1........... 0.7842    0.0262
max_depth= 25
Metric.......  Mean    Std. Dev.
accuracy..... 0.7030    0.0380
recall....... 0.7757    0.0495
precision.... 0.7945    0.0317
f1........... 0.7852    0.0236
```

3. Max Depth = 5 is selected as this is the smallest depth at which maximum F1 occurs. This depth is also best for maximizing accuracy and recall. This suggest the best value for this parameter is max_depth = 5.

4. A table of the metrics (recall, accuracy, precision and F1) for the 70/30 split using your selected model.

Results for depth=5

```
Model Metrics.....        Training     Validation
Observations.....              700            300
Features.........               68             68
Maximum Tree Depth.....          5              5
Minimum Leaf Size......          5              5
Minimum split Size.....          5              5
Mean Absolute Error....     0.3014         0.3431
Avg Squared Error......     0.1507         0.1963
Accuracy.............       0.7714         0.6933
Precision............        0.8148         0.7907
Recall (Sensitivity)...      0.8654         0.7834
F1-score.............        0.8394         0.7870
MISC (Misclassification)...   22.9%          30.7%
     class 0.............     43.8%          54.2%
     class 1.............     13.5%          21.7%


Training
Confusion Matrix  Class 0   Class 1
Class 0.....       122        95
Class 1.....        65       418


Validation
Confusion Matrix  Class 0   Class 1
Class 0.....        38        45
Class 1.....        47       170
```

5. A screen shot of your tree