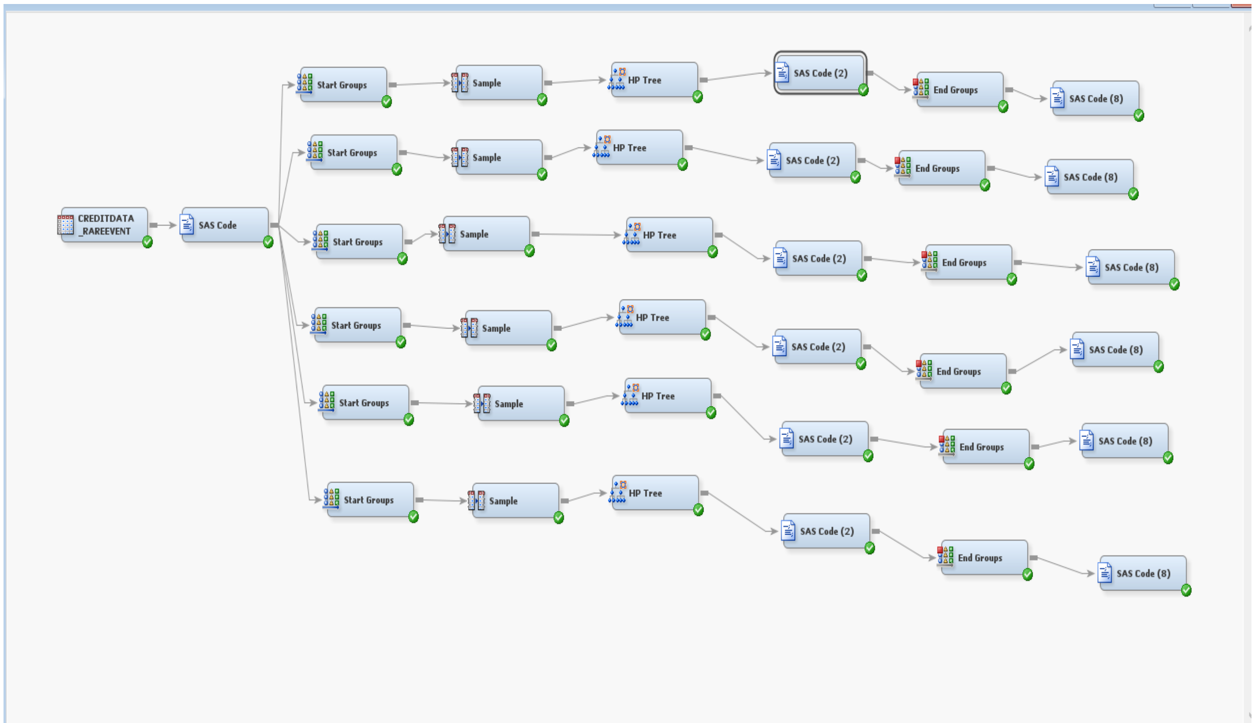
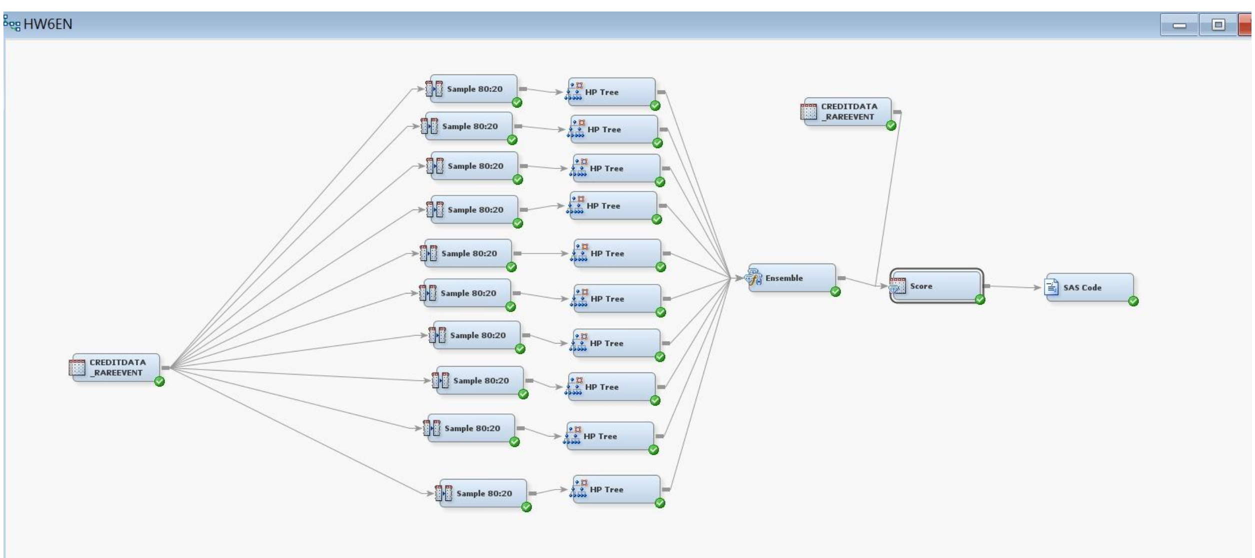
SAS solution

1. A screen shot of your project diagrams, step 1 and step 2





2. A screen shot or listing of SAS Code used inside one loop and all nodes outside the loop. Should be 3 screenshots.

SAS code 1

```
%let fprate = 1.00;
%let fnrate = 0.15;
Proc Datasets library=MyScores kill;
run;
```

SAS code 2

```
%let temp=myscores.temp50;
%let score=myscores.score50;
data &temp;
RETAIN FP FN NBad NGood FPCost FNCost;
KEEP N F_Score Sensitivity Specificity MISSCLASS MISC FN NBad
   FP NGood FNR FPR TPR TNR FPCost FNCost Loss;
set &EM_IMPORT_DATA END=EOF;
if _N_ EQ 1 then do;
   FN = 0;
   FP = 0;
   NBad   = 0;
   NGood  = 0;
   FPCost = 0;
   FNCost = 0;
end;
if Compare(Good_Bad, "bad", "i") EQ 0  then NBad  = NBad  + 1;
if Compare(Good_Bad, "good", "i") EQ 0 then NGood = Ngood + 1;
if Compare(Good_Bad, I_Good_Bad, "i") NE 0 AND
   Compare(Good_Bad, "bad", "i")  EQ 0 THEN DO;
         FP = FP + 1;
         FPCost = FPCost + &FPRate * AMOUNT;
END;
if Compare(Good_Bad, I_Good_Bad, "i") NE 0 AND
   Compare(Good_Bad, "good", "i") EQ 0 THEN DO;
         FN = FN + 1;
         FNCost = FNCost + &FNRate * AMOUNT;
END;
if EOF Then do;
   N = _N_;
   MissClass = FP + FN;
   FNR   = FN / NGood;
   FPR   = FP / Nbad;
   Misc = MissClass / N;
   TP    = NGood - FN;
   TN    = NBad  - FP;
   TPR   = TP / NGood;
   TNR   = TN / NBad;
   Sensitivity = (TP)/(TP+FN);
   Specificity = (TN)/(TN+FP);
   F_Score = 2*TP/(2*TP + FP + FN);
```

```
/*Adjust for size of "good" cases */
   FNCost  = (10000/NGood) * FNCost;
/*No adjustment needed for bad cases since
  all 500 of them are included in every
  sample */
   Loss = FPCost + FNCost ;
   output ;
end;
proc append base= &score data= &temp;
run;
```

SAS CODE 3

```
%let p=50;
proc means data=myscores.score&p;
run;
```

SAS Code 4

```
%let FPRate=1.0;
%let FNRate=0.15;
data mylib.score15;
RETAIN FP FN NBad NGood FPCost FNCost;
KEEP N F_Score Sensitivity Specificity MISSCLASS MISC FN NBad
   FP NGood FNR FPR TPR TNR FPCost FNCost Loss;
set &EM_IMPORT_SCORE END=EOF;
if _N_ EQ 1 then do;
   FN = 0;
   FP = 0;
   NBad   = 0;
   NGood  = 0;
   FPCost = 0;
   FNCost = 0;
end;
if Compare(Good_Bad, "bad", "i") EQ 0  then NBad  = NBad  + 1;
if Compare(Good_Bad, "good", "i") EQ 0 then NGood = Ngood + 1;
if Compare(Good_Bad, EM_CLASSIFICATION, "i") NE 0 AND
   Compare(Good_Bad, "bad", "i")  EQ 0 THEN DO;
         FP = FP + 1;
         FPCost = FPCost + &FPRate * AMOUNT;
END;
if Compare(Good_Bad, EM_CLASSIFICATION, "i") NE 0 AND
   Compare(Good_Bad, "good", "i") EQ 0 THEN DO;
         FN = FN + 1;
         FNCost = FNCost + &FNRate * AMOUNT;
END;
if EOF Then do;
   N = _N_;
   MissClass = FP + FN;
   FNR  = FN / NGood;
   FPR  = FP / Nbad;
   Misc = MissClass / N;
   TP   = NGood - FN;
```

```
    TN    = NBad  - FP;
    TPR   = TP / NGood;
    TNR   = TN / NBad;
    Sensitivity = (TP)/(TP+FN);
    Specificity = (TN)/(TN+FP);
    F_Score = 2*TP/(2*TP + FP + FN);
/* no need to adjust FP cost because this is scoring the
    entire dataset */
    Loss = FPCost + FNCost ;
    output ;
end;

proc means data=mylib.score15;
run;
```

3. A table showing average loss and MISC for each ratio

Ratio	**Loss**	MISC
50:50	1500830.50	0.192
60:40	1139443.10	0.179
70:30	994540.50	0.162
75:25	792165.10	0.132
80:20	746225.90	0.083
85:15	792044.98	0.112

Chose the 80:20 ratio

4. A description of the total loss and MISC for the ensemble model calculated from the entire dataset, not the smaller ratio dataset.

Variable	N	Mean	Std Dev	Minimum	Maximum
FP	1	205.0000000	.	205.0000000	205.0000000
FN	1	59.0000000	.	59.0000000	59.0000000
NBad	1	500.0000000	.	500.0000000	500.0000000
NGood	1	10000.00	.	10000.00	10000.00
FPCost	1	541125.00	.	541125.00	541125.00
FNCost	1	89398.95	.	89398.95	89398.95
N	1	10500.00	.	10500.00	10500.00
MissClass	1	264.0000000	.	264.0000000	264.0000000
FNR	1	0.0059000	.	0.0059000	0.0059000
FPR	1	0.4100000	.	0.4100000	0.4100000
Misc	1	0.0251429	.	0.0251429	0.0251429
TPR	1	0.9941000	.	0.9941000	0.9941000
TNR	1	0.5900000	.	0.5900000	0.5900000
Sensitivity	1	0.9941000	.	0.9941000	0.9941000
Specificity	1	0.5900000	.	0.5900000	0.5900000
F_Score	1	0.9868957	.	0.9868957	0.9868957
Loss	1	630523.95	.	630523.95	630523.95

Python Solution

```
from AdvancedAnalytics import ReplaceImputeEncode,calculate
from sklearn.tree import DecisionTreeClassifier
import math
import pandas as pd
import numpy as np
from imblearn.under_sampling import RandomUnderSampler

credit_xlsx="CreditData_RareEvent.xlsx"
credit_df = pd.read_excel(credit_xlsx)

attribute_map = {
'age':['I',(1, 120),[0,0]],
'amount':['I',(0, 20000),[0,0]],
'duration':['I',(1,100),[0,0]],
'checking':['N',(1, 2, 3, 4),[0,0]],
'coapp':['N',(1,2,3),[0,0]],
'depends':['B',(1,2),[0,0]],
'employed':['N',(1,2,3,4,5),[0,0]],
'existcr':['N',(1,2,3,4),[0,0]],
'foreign':['B',(1,2),[0,0]],
'good_bad':['B',('bad', 'good'),[0,0]],
'history':['N',(0,1,2,3,4),[0,0]],
'housing':['N',(1, 2, 3), [0,0]],
'installp':['N',(1,2,3,4),[0,0]],
'job':['N',(1,2,3,4),[0,0]],
'marital':['N',(1,2,3,4),[0,0]],
```

```
'other':['N',(1,2,3),[0,0]],
'property':['N',(1,2,3,4),[0,0]],
'resident':['N',(1,2,3,4),[0,0]],
'savings':['N',(1,2,3,4,5),[0,0]],
'telephon':['B',(1,2),[0,0]] }

#Using RIE to replace and impute missing values and enocde attributes
rie = ReplaceImputeEncode(data_map=attribute_map,nominal_encoding='one-hot',
                          interval_scale='std', drop=True, display=True)
encoded_credit = rie.fit_transform(credit_df)
y = np.asarray(encoded_credit['good_bad']) # The target is not scaled or
imputed
X = np.asarray(encoded_credit.drop('good_bad',axis=1))

#Calculate potential loss for FP and FN
fp_cost = np.array(credit_df['amount'])
fn_cost = np.array(0.15 * credit_df['amount'])


#Create the ratio list for max:min
ratio = ['50:50','60:40','70:30','75:25','80:20','85:15']
rus_ratio =
({0:500,1:500},{0:500,1:750},{0:500,1:1167},{0:500,1:1500},{0:500,1:2000},{0:
500,1:2833})

#Set up 10 random sample with different random seed
np.random.seed(12345)
max_seed = 2**10 - 1
rand_val = np.random.randint(1, high=max_seed, size=10)

best_ratio = 0
min_loss = 1e64
best_decTree =0

#Get the Best Tree Depth which minimize the loss
for k in range(len(rus_ratio)):
    min_loss_d = 1e64
    best_depth = 0
    print("\nDecesion Tree using " + ratio[k] + " RUS")
    for j in range(2,21):
        d = j #Tree depth
        fn_loss = np.zeros(len(rand_val))
        fp_loss = np.zeros(len(rand_val))
        misc = np.zeros(len(rand_val))
        for i in range(len(rand_val)):
```

```
            rus =
RandomUnderSampler(ratio=rus_ratio[k],random_state=rand_val[i],
                                     return_indices=False,replacement=False)
            X_rus, y_rus = rus.fit_sample(X, y)
            dtc = DecisionTreeClassifier(criterion='gini', max_depth=d,
                             min_samples_split=5, min_samples_leaf=5)

            dtc.fit(X_rus,y_rus)
            loss, conf_mat = calculate.binary_loss(y, dtc.predict(X),
                                                       fp_cost, fn_cost,
display=False)
            fn_loss[i] = loss[0]
            fp_loss[i] = loss[1]
            misc[i] = (conf_mat[1] + conf_mat[2])/y.shape[0]
        avg_misc = np.average(misc) #Avg Missclassificaition rate
        t_loss = fp_loss+fn_loss #Total Loss
        avg_loss = np.average(t_loss) #Avg Loss
        if avg_loss < min_loss_d: #Get the least loss among the tree depth
            min_loss_d = avg_loss
            se_loss_d = np.std(t_loss)/math.sqrt(len(rand_val))
            best_depth = d
            misc_d = avg_misc
            fn_avg_loss = np.average(fn_loss)
            fp_avg_loss = np.average(fp_loss)
    if min_loss_d < min_loss:# Get the best ratio and the best depth tree
        min_loss = min_loss_d
        se_loss = se_loss_d
        best_ratio = k
        best_decTree = best_depth
    print("{:.<23s}{:d}".format("Best Depth", best_depth))
    print("{:.<23s}{:12.4f}".format("Misclassification Rate",misc_d))
    print("{:.<23s} ${:10,.0f}".format("False Negative Loss",fn_avg_loss))
    print("{:.<23s} ${:10,.0f}".format("False Positive Loss",fp_avg_loss))
    print("{:.<23s} ${:10,.0f}{:5s}${:<,.0f}".format("Total Loss",
          min_loss_d, " +/- ", se_loss_d))
print("")
print("{:.<23s}{:>12s}".format("Best RUS Ratio", ratio[best_ratio]))
print("{:.<23s}{:d}".format("Best Depth", best_decTree))
print("{:.<23s} ${:10,.0f}{:5s}${:<,.0f}".format("Lowest Loss", \
min_loss, " +/-", se_loss))

#Ensemble Modelling
n_obs = len(y)
n_rand = 100 # No of random samples to be selected out of the best
ratio(85:15)
predicted_prob = np.zeros((n_obs,n_rand))
```

```
avg_prob = np.zeros(n_obs)
predicted_prob = np.zeros((n_obs,n_rand))
avg_prob = np.zeros(n_obs)

# Setup 100 random number seeds
np.random.seed(12345)
max_seed = 2**20 - 1
rand_value = np.random.randint(1, high=max_seed, size=n_rand)

# Model 100 random samples- each with a Best ratio, which in our case is
85:15
for i in range(len(rand_value)):
    rus = RandomUnderSampler(ratio=rus_ratio[best_ratio],
random_state=rand_value[i],
                             return_indices=False, replacement=False)
    X_rus, y_rus = rus.fit_sample(X, y)
    dtc = DecisionTreeClassifier(criterion='gini', max_depth=best_decTree,
                             min_samples_split=5, min_samples_leaf=5)
    dtc.fit(X_rus,y_rus)
    predicted_prob[0:n_obs, i] = dtc.predict_proba(X)[0:n_obs, 0]

for i in range(n_obs):
    avg_prob[i] = np.mean(predicted_prob[i,0:n_rand])

# Set y_pred equal to the predicted classification
y_pred = avg_prob[0:n_obs] < 0.5
y_pred.astype(np.int)

# Calculate loss from using the ensemble predictions
print("\nEnsemble Estimates based on averaging",len(rand_value), "Models")
loss, conf_mat = calculate.binary_loss(y, y_pred, fp_cost, fn_cost)
```

PYTHON RESULTS

Decesion Tree using 50:50 RUS
Best Depth.............16
Misclassification Rate. 0.2294
False Negative Loss.... $ 1,235,474
False Positive Loss.... $ 127,819
Total Loss............ $ 1,363,293 +/- $33,292

Decesion Tree using 60:40 RUS
Best Depth.............20

Misclassification Rate.   0.1613
False Negative Loss.... $ 886,420
False Positive Loss.... $ 164,850
Total Loss............ $ 1,051,271 +/- $23,579

Decesion Tree using 70:30 RUS
Best Depth............19
Misclassification Rate.   0.0984
False Negative Loss.... $ 521,994
False Positive Loss.... $ 209,416
Total Loss............ $ 731,410 +/- $23,112

Decesion Tree using 75:25 RUS
Best Depth............20
Misclassification Rate.   0.0801
False Negative Loss.... $ 393,880
False Positive Loss.... $ 208,579
Total Loss............ $ 602,460 +/- $16,842

Decesion Tree using 80:20 RUS
Best Depth............17
Misclassification Rate.   0.0602
False Negative Loss.... $ 302,752
False Positive Loss.... $ 210,330
Total Loss............ $ 513,082 +/- $16,155

Decesion Tree using 85:15 RUS
Best Depth............18
Misclassification Rate.   0.0394
False Negative Loss.... $ 179,630
False Positive Loss.... $ 260,525
Total Loss............ $ 440,155 +/- $13,332

Best RUS Ratio.........   85:15
Best Depth............18
Lowest Loss........... $ 440,155 +/- $13,332

Ensemble Estimates based on averaging 100 Models
Misclassification Rate.   0.0031
False Negative Loss....   0
False Positive Loss....   109992
Total Loss............   109992