KRIT GUPTA
UIN: 927001565

WEEK 8 – RARE EVENT MODELLING

PYTHON SCRIPT

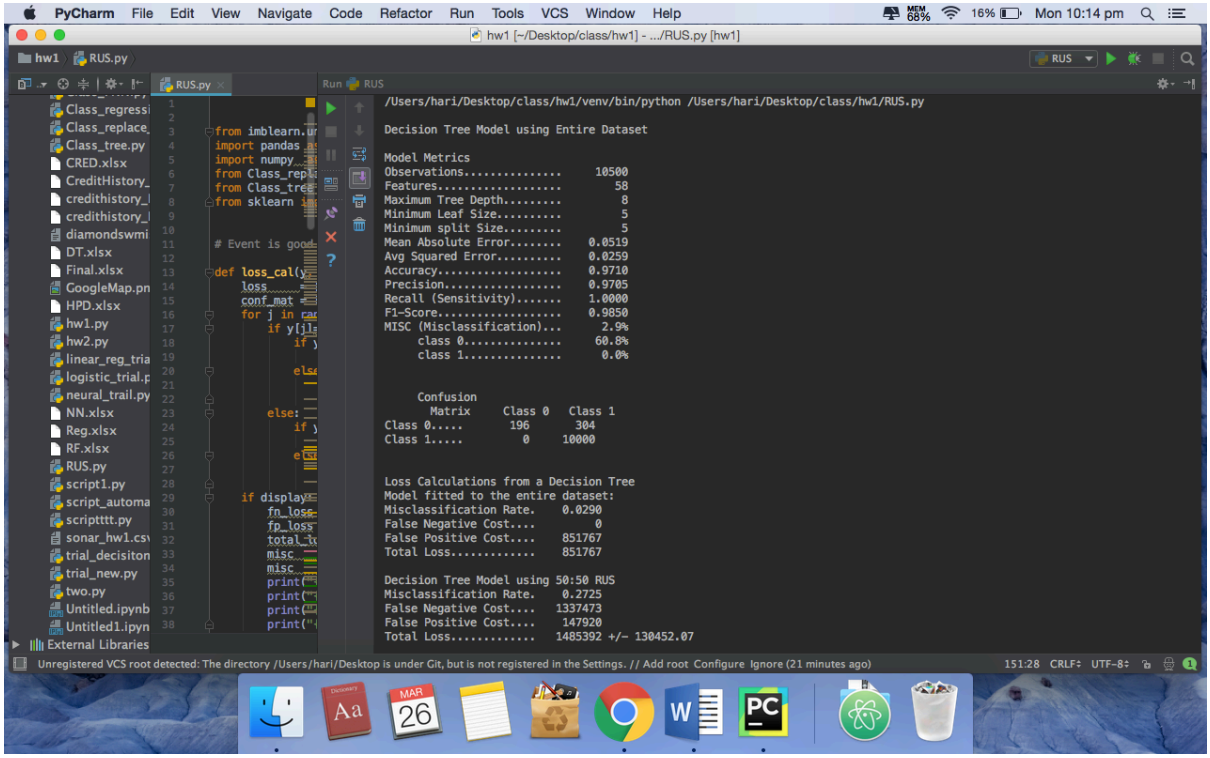
```
from imblearn.under_sampling import RandomUnderSampler
import pandas as pd
import numpy  as np
from Class_replace_impute_encode import ReplaceImputeEncode
from Class_tree import DecisionTree
from sklearn import tree


# Event is good credit = 1 o.e false positive will have greatest loss.

def loss_cal(y, y_predict, fp_cost, fn_cost, display=True):
    loss     = [0, 0]       #False Neg Cost, False Pos Cost
    conf_mat = [0, 0, 0, 0] #tn, fp, fn, tp
    for j in range(len(y)):
        if y[j]==0:
            if y_predict[j]==0:
                conf_mat[0] += 1 #True Negative
            else:
                conf_mat[1] += 1 #False Positive
                loss[1] += fp_cost[j]
        else:
            if y_predict[j]==1:
                conf_mat[3] += 1 #True Positive
            else:
                conf_mat[2] += 1 #False Negative
                loss[0] += fn_cost[j]
    if display:
        fn_loss = loss[0]
        fp_loss = loss[1]
        total_loss = fn_loss + fp_loss
        misc    = conf_mat[1] + conf_mat[2]
        misc    = misc/len(y)
        print("{:.<23s}{:10.4f}".format("Misclassification Rate", misc))
        print("{:.<23s}{:10.0f}".format("False Negative Cost", fn_loss))
        print("{:.<23s}{:10.0f}".format("False Positive Cost", fp_loss))
        print("{:.<23s}{:10.0f}".format("Total Loss", total_loss))
    return loss, conf_mat


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

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

df = pd.read_excel("CRED.xlsx")
# Encode for Logistic Regression, drop last one-hot column
rie = ReplaceImputeEncode(data_map=attribute_map, nominal_encoding='one-hot', \
                          interval_scale = 'std', drop=False, display=False)
encoded_df = rie.fit_transform(df)
# Create X and y, numpy arrays
# bad=0 and good=1
y = np.asarray(encoded_df['good_bad']) # The target is not scaled or imputed
X = np.asarray(encoded_df.drop('good_bad',axis=1))

# Setup false positive and false negative costs for each transaction
fp_cost = np.array(df['amount'])
fn_cost = np.array(0.15*df['amount'])

treeclassifier = tree.DecisionTreeClassifier(criterion='gini', max_depth=8,
min_samples_split=5, min_samples_leaf=5)
treeclassifier = treeclassifier.fit(X, y)

print("\nDecision Tree Model using Entire Dataset")
col = rie.col
col.remove('good_bad')
DecisionTree.display_binary_metrics(treeclassifier, X, y)
print("\nLoss Calculations from a Decision Tree")
print("Model fitted to the entire dataset:")
loss, conf_mat = loss_cal(y, treeclassifier.predict(X), fp_cost, fn_cost)

# Setup random number seeds
rand_val = np.array([1, 12, 123, 1234, 12345, 654321, 54321, 4321, 321, 21])
# Ratios of Majority:Minority Events
ratio = [ '50:50', '60:40', '70:30', '75:25', '80:20', '85:15' ]
# Dictionaries contains number of minority and majority events in each ratio sample
# n_majority = ratio x n_minority
rus_ratio = ({0:500, 1:500}, {0:500, 1:750}, {0:500, 1:1166}, {0:500, 1:1500},
{0:500, 1:2000}, {0:500, 1:2833})

# Best model is one that minimizes the loss
min_loss   = 9e+15
best_ratio = 0
for k in range(len(rus_ratio)):
    rand_vals = (k+1)*rand_val
    print("\nDecision Tree Model using " + ratio[k] + " RUS")
    fn_loss = np.zeros(len(rand_vals))
    fp_loss = np.zeros(len(rand_vals))
    misc    = np.zeros(len(rand_vals))
    for i in range(len(rand_vals)):
        rus = RandomUnderSampler(ratio=rus_ratio[k], \
                random_state=rand_vals[i], return_indices=False, \
                replacement=False)
        X_rus, y_rus = rus.fit_sample(X, y)
        dtree = tree.DecisionTreeClassifier(criterion='gini', max_depth=8, \
      min_samples_split=5, min_samples_leaf=5)
        dtree.fit(X_rus, y_rus)
        loss, conf_mat = loss_cal(y, dtree.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]
    misc = np.sum(misc)/(10500 * len(rand_vals))
    fn_avg_loss = np.average(fn_loss)
    fp_avg_loss = np.average(fp_loss)
    total_loss  = fn_loss + fp_loss
    avg_loss    = np.average(total_loss)
```

```
    std_loss    = np.std(total_loss)
    print("{:.<23s}{:10.4f}".format("Misclassification Rate", misc))
    print("{:.<23s}{:10.0f}".format("False Negative Cost", fn_avg_loss))
    print("{:.<23s}{:10.0f}".format("False Positive Cost", fp_avg_loss))
    print("{:.<23s}{:10.0f}{:5s}{:<10.2f}".format("Total Loss", avg_loss, \
                  " +/- ", std_loss))
    if avg_loss < min_loss:
        min_loss   = avg_loss
        best_ratio = k


# Ensemble Modeling - Averaging Classification Probabilities
avg_prob = np.zeros((len(y),2))
# Setup 100 random number seeds for use in creating random samples
np.random.seed(12345)
max_seed = 150000
rand_value = np.random.randint(1, high=max_seed, size=10)
# Model 100 random samples, each with a 70:30 ratio
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)
    dtree = tree.DecisionTreeClassifier(criterion='gini', max_depth=8, \
      min_samples_split=5, min_samples_leaf=5)
    dtree.fit(X_rus, y_rus)
    avg_prob += dtree.predict_proba(X)
avg_prob = avg_prob/len(rand_value)
# Set y_pred equal to the predicted classification
y_pred = avg_prob[0:,0] < 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 = loss_cal(y, y_pred,fp_cost,fn_cost)
```

KRIT GUPTA
UIN: 927001565

SCREENSHOT



OUTPUT:

Decision Tree Model using Entire Dataset

Model Metrics
Observations..... 10500
Features..... 58
Maximum Tree Depth..... 8
Minimum Leaf Size..... 5
Minimum split Size..... 5
Mean Absolute Error..... 0.0519
Avg Squared Error..... 0.0259
Accuracy..... 0.9710
Precision..... 0.9705
Recall (Sensitivity)..... 1.0000
F1-Score..... 0.9850
MISC (Misclassification)... 2.9%
class 0..... 60.8%
class 1..... 0.0%

Confusion
Matrix Class 0 Class 1
Class 0..... 196 304
Class 1..... 0 10000

KRIT GUPTA
UIN: 927001565

Loss Calculations from a Decision Tree
Model fitted to the entire dataset:
Misclassification Rate. 0.0290
False Negative Cost.... 0
False Positive Cost.... 851767
Total Loss..... 851767

Decision Tree Model using 50:50 RUS
Misclassification Rate. 0.2725
False Negative Cost.... 1337473
False Positive Cost.... 147920
Total Loss..... 1485392 +/- 130452.07

Decision Tree Model using 60:40 RUS
Misclassification Rate. 0.1962
False Negative Cost.... 949554
False Positive Cost.... 258551
Total Loss..... 1208105 +/- 132413.99

Decision Tree Model using 70:30 RUS
Misclassification Rate. 0.1021
False Negative Cost.... 468444
False Positive Cost.... 423333
Total Loss..... 891777 +/- 69286.81

Decision Tree Model using 75:25 RUS
Misclassification Rate. 0.0742
False Negative Cost.... 333280
False Positive Cost.... 515937
Total Loss..... 849217 +/- 75304.62

Decision Tree Model using 80:20 RUS
Misclassification Rate. 0.0566
False Negative Cost.... 227334
False Positive Cost.... 583129
Total Loss..... 810463 +/- 90470.82

Decision Tree Model using 85:15 RUS
Misclassification Rate. 0.0396
False Negative Cost.... 120918
False Positive Cost.... 650800
Total Loss..... 771718 +/- 75140.48

Ensemble Estimates based on averaging 10 Models
Misclassification Rate. 0.0246
False Negative Cost.... 0
False Positive Cost.... 595382
Total Loss..... 595382

```
Process finished with exit code 0
```