In [17]:
```
#Random Oversampling: Randomly duplicate examples in the minority class.
# https://www.kaggle.com/residentmario/undersampling-and-oversampling-imbalanced-data

import pandas as pd
from sklearn.model_selection import train_test_split
from dmba import classificationSummary

from sklearn.neural_network import MLPClassifier # Neural Net
from sklearn.linear_model import LogisticRegression #Logistic Regression
from sklearn.model_selection import GridSearchCV #Needed for Decision Tree
from sklearn.tree import DecisionTreeClassifier #Needed for Decision Tree
```

In [18]:
```
flip_df = pd.read_csv('Renesas.csv')
flip_df.columns = [s.strip().replace(' ', '_') for s in flip_df.columns]
```

In [19]:
```
flip_df.columns
```

Out[19]:
```
Index(['Booking_Part_Number', 'Product_BU', 'Region', 'Category_1',
       'Category_2', 'Quantity', 'Unit_Price', 'Left_10', 'IC_Source',
       'IHS_ECCN', 'Heat_2', 'Distributors', 'IHS_EOL', 'IHS_Cost',
       'Flip_Stock', 'Sales_History', 'Total_Quote', 'Total_Quote_$$',
       'Authorized_Availability', 'Authorized_Price',
       'Authorized_Availability', 'Authorized_Price.1', 'Flip_Stock.1',
       'Sales_History.1', 'Total_Quotes', 'Quoted_Amount'],
      dtype='object')
```

In [20]:
```
#cold=1354
#warm=20
#Classification should exceed 98.54%

predictors = ['Category_1', 'Quantity', 'Unit_Price', 'Distributors']
#predictors = ['Region', 'Quantity', 'Unit_Price']
outcome = 'Heat_2'

X=flip_df[predictors].loc[0:1372]
y= flip_df[outcome].loc[0:1372]
```

In [21]:
```
X = pd.get_dummies(X, prefix_sep='_', drop_first=False)
```

In [22]:
```
X.columns = [s.strip().replace(' ', '_') for s in X.columns]
X.columns = [s.strip().replace('.', '_') for s in X.columns]
X.columns = [s.strip().replace('-', '_') for s in X.columns]
X.columns = [s.strip().replace('/', '_') for s in X.columns]
X.columns = [s.strip().replace('(', '_') for s in X.columns]
X.columns = [s.strip().replace(')', '_') for s in X.columns]

train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=10) # test_size is validation da
```

In [23]:
```
#y.to_csv(r'C:\Users\jalaluddin\Desktop\GBC B412\Data Project Capstone BUS 4045\Assignments\imblance\y.csv', index = Fal
```

In [24]:
```
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler

ros = RandomOverSampler(random_state=5) #(sampling_strategy=2.0, random_state=5)
```

In [25]:
```
train_X_re, train_y_re = ros.fit_resample(train_X, train_y)
```

In [26]:
```
#train_y.to_csv(r'C:\Users\jalaluddin\Desktop\GBC B412\Data Project Capstone BUS 4045\Assignments\imbalance\train_y.csv'
```

In [27]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

rf = RandomForestClassifier(n_estimators=1000, random_state=1)
rf.fit(train_X_re, train_y_re)
classificationSummary(valid_y, rf.predict(valid_X), class_names=['Cold', 'Warm'])
```

```
Confusion Matrix (Accuracy 0.9873)

       Prediction
Actual Cold Warm
  Cold  542    1
  Warm    6    1
```

In [28]:
```
boost = GradientBoostingClassifier()
boost.fit(train_X_re, train_y_re)
classificationSummary(valid_y, boost.predict(valid_X), class_names=['Cold', 'Warm'])
```

```
Confusion Matrix (Accuracy 0.9764)

       Prediction
Actual Cold Warm
  Cold  535    8
  Warm    5    2
```

In [29]:
```
# Start with an initial guess for parameters
param_grid = {
'max_depth': [10, 20, 30, 40],
'min_samples_split': [20, 40, 60, 80, 100],
'min_impurity_decrease': [0, 0.0005, 0.001, 0.005, 0.01],
}

gridSearch = GridSearchCV(DecisionTreeClassifier(random_state=1), param_grid, cv=5, n_jobs=-1) # n_jobs=-1 will utilize
gridSearch.fit(train_X_re, train_y_re)

#print('Initial score: ', gridSearch.best_score_)
#print('Initial parameters: ', gridSearch.best_params_)

# Adapt grid based on result from initial grid search
param_grid = {
'max_depth': list(range(2, 16)), # 14 values
'min_samples_split': list(range(10, 22)), # 11 values
'min_impurity_decrease': [0.0009, 0.001, 0.0011], # 3 values
}
gridSearch = GridSearchCV(DecisionTreeClassifier(random_state=1), param_grid, cv=5, n_jobs=-1)
gridSearch.fit(train_X_re, train_y_re)
print('Improved score: ', gridSearch.best_score_)
print('Improved parameters: ', gridSearch.best_params_)
bestClassTree = gridSearch.best_estimator_
```

```
Improved score:  0.9814814814814815
Improved parameters:  {'max_depth': 15, 'min_impurity_decrease': 0.0009, 'min_samples_split': 10}
```

In [30]:
```
classificationSummary(valid_y, gridSearch.predict(valid_X), class_names=['Cold', 'Warm'])
```

```
Confusion Matrix (Accuracy 0.9618)

       Prediction
Actual Cold Warm
  Cold  527   16
  Warm    5    2
```

In [31]:
```
from sklearn import ensemble

#instantiate model
nEst = 10000
depth = 8
learnRate = 0.007
maxFeatures = 4
GBMModel2 = ensemble.GradientBoostingClassifier(n_estimators=nEst, max_depth=depth, learning_rate=learnRate, max_feature

GBMModel2 = GBMModel2.fit(train_X_re, train_y_re)

classificationSummary(valid_y, GBMModel2.predict(valid_X), class_names=['Cold', 'Warm'])
```

```
Confusion Matrix (Accuracy 0.9873)

       Prediction
Actual Cold Warm
  Cold  542    1
  Warm    6    1
```

In [35]:
```
classes = sorted(y.unique())
clf1 = MLPClassifier(hidden_layer_sizes=(200), activation='logistic', solver='lbfgs', random_state=1)
clf1.fit(train_X_re, train_y_re)
#clf.predict(X)

classificationSummary(valid_y, clf1.predict(valid_X), class_names=['Cold', 'Warm'])
```

```
Confusion Matrix (Accuracy 0.6691)

       Prediction
Actual Cold Warm
  Cold  362  181
  Warm    1    6

C:\Users\jalaluddin\Anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:471: ConvergenceWarnin
g: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

In [40]:
```
classes = sorted(y.unique())
clf2 = MLPClassifier(hidden_layer_sizes=(800), activation='logistic', solver='lbfgs', random_state=1)
clf2.fit(train_X_re, train_y_re)
#clf.predict(X)

classificationSummary(valid_y, clf2.predict(valid_X), class_names=['Cold', 'Warm'])
```

```
Confusion Matrix (Accuracy 0.6691)

       Prediction
Actual Cold Warm
  Cold  362  181
  Warm    1    6

C:\Users\jalaluddin\Anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:471: ConvergenceWarnin
g: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

In [41]:
```
#Now that three classifiers have been trained use majority voting here.
from sklearn.ensemble import VotingClassifier
eclf1 = VotingClassifier(estimators=[('NN1', clf1), ('NN2', clf2), ('Boost', boost)], voting='hard')
eclf1 = eclf1.fit(train_X_re, train_y_re)

classificationSummary(valid_y, eclf1.predict(valid_X), class_names=['Cold', 'Warm'])
```

```
C:\Users\jalaluddin\Anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:471: ConvergenceWarnin
g: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\jalaluddin\Anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:471: ConvergenceWarnin
g: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)

Confusion Matrix (Accuracy 0.6909)

       Prediction
Actual Cold Warm
  Cold  374  169
  Warm    1    6
```

In [42]:
```
classificationSummary(y, eclf1.predict(X), class_names=['Cold', 'Warm'])
```

```
Confusion Matrix (Accuracy 0.6999)

       Prediction
Actual Cold Warm
  Cold  945  408
  Warm    4   16
```

In [43]:
```
boost_proba = boost.predict_proba(X)
clf_proba1 = clf1.predict_proba(X)
clf_proba2 = clf2.predict_proba(X)

sum_proba=(boost_proba+clf_proba1+clf_proba2)/3
```

In [44]:
```
final_predict=eclf1.predict(X)

final_result = pd.DataFrame({'actual': y,
'p(0)': [p[0] for p in sum_proba],
'p(1)': [p[1] for p in sum_proba],
'predicted': final_predict })

print(final_result)
```

```
     actual      p(0)      p(1) predicted
0      Cold  0.517700  0.482300      Warm
1      Cold  0.586867  0.413133      Warm
2      Cold  0.533557  0.466443      Warm
3      Cold  0.749501  0.250499      Cold
4      Cold  0.729618  0.270382      Cold
...     ...       ...       ...       ...
1368   Cold  0.556620  0.443380      Cold
1369   Cold  0.767559  0.232441      Cold
1370   Cold  0.800581  0.199419      Cold
1371   Cold  0.584257  0.415743      Cold
1372   Cold  0.677186  0.322814      Cold

[1373 rows x 4 columns]
```

In [45]:
```
v(r'C:\Users\jalaluddin\Desktop\GBC B412\Data Project Capstone BUS 4045\Assignments\imbalance\result_renesas.csv', index
```

In [ ]: