```
 1 import DataPackageSupport as dps
 2
 3
 4 class DataPackage:
 5     __version = 0.1
 6
 7     def __init__(self,
 8                  origData,
 9                  uniqueColumn,
10                  targetColumn
11                  ):
12         self.uniqueColumn = uniqueColumn
13         self.targetColumn = targetColumn
14         self.__setOrigData(origData)
15
16
17     def __setOrigData(self, origData):
18         self.origData = origData
19         self.isOrigDataLoaded = True
20
21         #Get and set features listing, removing unique
   and target columns
22         self.dataFeatures = list(origData.columns)
23         # Remove unique and target columnm
24         self.dataFeatures.remove(self.uniqueColumn)
25         self.dataFeatures.remove(self.targetColumn)
26
27         # A new dataframe means we need to reset our
   work
28         self.__resetWork()
29
30     # if the data gets changed then we need to
31     # invalidate all the results/work done previously
32     def __resetWork(self):
33         self.isBalanced = False
34
35         self.__clearTrainTestData()
36
```

```
37      def __setTrainData(self, trainData):
38          self.trainData = trainData
39          self.isTrainDataLoaded = True
40
41      def getTrainData(self):
42          return self.trainData
43
44      def __setTestData(self, testData):
45          self.testData = testData
46          self.isTestDataLoaded = True
47
48      def getTestData(self):
49          return self.testData
50
51      def __clearOrigData(self):
52          self.origData = None
53          self.isOrigDataLoaded = False
54
55      def __clearTrainTestData(self):
56          self.isTrainTestSplit = False
57          self.isTrainDataLoaded = False
58          self.trainData = None
59
60          self.isTestDataLoaded = False
61          self.testData = None
62
63      def splitTrainTest(self,
64                         stratifyColumn=None,
65                         test_size=0.2,
66                         random_state=765,
67                         shuffle=True
68                         ):
69
70          if stratifyColumn is None:
71              stratifyColumn = self.targetColumn
72
73          train, test = dps.trainTestSplit(dataFrame=self
    .getOrigData(),
```

```
                                                test_size=
    test_size,
                                                random_state=
    random_state,

    stratifyColumn=stratifyColumn,
                                                shuffle=
    shuffle)

            self.__setTrainData(train)
            self.__setTestData(test)
            self.isTrainTestSplit = True
            self.__clearOrigData()

        def getOrigData(self):
            if self.isOrigDataLoaded == False:
                display("Original data frame is not loaded
    ")
            return self.origData

        def display(self):
            emptySpace = '    '
            indent = emptySpace + '---> '

            print(f'{emptySpace}DataPackage summary:')
            print(f'{emptySpace}Attributes:')
            print(f'{indent}uniqueColumn: {self.
    uniqueColumn}')
            print(f'{indent}targetColumn: {self.
    targetColumn}')

            print(f'{emptySpace}Process:')
            print(f'{indent}isBalanced: {self.isBalanced}'
    )
            print(f'{indent}isTrainTestSplit: {self.
    isTrainTestSplit}')

            print(f'{emptySpace}Data:')
```

```
103         print(f'{indent}isOrigDataLoaded: {self.
    isOrigDataLoaded}')
104         print(f'{indent}isTrainDataLoaded: {self.
    isTrainDataLoaded}')
105         print(f'{indent}isTestDataLoaded: {self.
    isTrainDataLoaded}')
106 
107     def displayClassBalance(self, columnName=None,
    verbose=False, showRecords=5):
108         if columnName is None:
109             columnName = self.targetColumn
110 
111         dps.displayClassBalance(dataFrame=self.
    getOrigData(),
112                                 columnName=columnName,
113                                 showRecords=
    showRecords,
114                                 verbose=verbose)
115 
116     def classBalanceUndersample(self,
117                                 columnName=None):
118 
119 
120         if columnName is None:
121             columnName = self.targetColumn
122 
123         # Needs to be balanced
124         dfBalanced = dps.classBalanceUndersample(
    dataFrame=self.getOrigData(),
125 
    columnName=columnName)
126 
127         if not self.isBalanced:
128             self.__setOrigData(dfBalanced)
129             self.isBalanced = True
130 
131     def showClassBalance(self,
132                          columnName=None):
```

```
133         if columnName is None:
134             columnName = self.targetColumn
135 
136         # Needs to be balanced
137         dfBalanced = dps.classBalanceUndersample(
    dataFrame=self.getOrigData(),
138 
    columnName=columnName,
139 
    alreadyBalanced=True)
140 
141 
```