Virtual bidding in NYISO's markets

Deadline: Nov 27 2023

In this lab, we will implement a simple virtual trading strategy in New York ISO's electricity markets. The goal is to maximize profits. We shall train our model on price data from one year, and implement the strategy on the data from the next year. How much can you earn with a certain daily budget? Say $250K?

We will present a trading strategy. You are welcome to try other strategies and compare the gains over multiple runs.

Let's start with customary imports.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import choice
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import pickle

from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
```

Load the day-ahead and real-time prices from 11 zones in New York.

The day-ahead prices are defined every hour. The real-time market runs every 5 minutes. For each zone, an average of these prices over an hour is published.

Store the list of zones in the variable 'listOfZones'. Also, store the number of options as the number of zones times the 24 hours available for trading. Finally create another list containing the option names (zone + hour).

In [3]:
```
listOfZones = ['CAPITL', 'CENTRL', 'DUNWOD', 'GENESE', 'HUD VL', 'LONGIL', 'MHK VL',
               'MILLWD', 'N.Y.C.', 'NORTH', 'WEST']

nOptions = len(listOfZones) * 24
optionNames = [zone + "_Hour_" + str(t) for zone in listOfZones for t in range(24)]
```

Parse the files with DA and RT prices along with DA load forecast.

Define a function that parses three files containing DA and RT prices, along with DA load predictions from a year from all different zones in the list defined before. This function will be used for to both load the data for training the classifiers and testing them. This function has 3 outputs: they are DA prices,

difference between DA and RT prices, and finally DA load predictions. The outputs are pandas data frames whose columns are the options, and rows are the days in the year.

In [4]:
```
def loadNYISOData(year):

    # Open the relevant files for DA prices, RT prices, DA load.

    dfPriceDA = pd.read_csv("DAM_NYISO_Zonal_LBMP_" + str(year) + ".csv")
    dfPriceRT = pd.read_csv("RTM_NYISO_Zonal_LBMP_" + str(year) + ".csv")
    dfLoadDA = pd.read_csv("DAM_NYISO_LoadForecast_" + str(year) + ".csv")

    # Collect the DA and RT prices from each zone from each hour and create a pandas
    # The data should have prices and loads from all days of a year, where each day
    # contributes 24 rows, corresponding to each hour.

    priceDA = pd.DataFrame({zone: (dfPriceDA.loc[dfPriceDA['Zone Name'] == zone,
                                                 'DAM Zonal LBMP']).values
                            for zone in listOfZones})
    priceRT = pd.DataFrame({zone: (dfPriceRT.loc[dfPriceRT['Zone Name'] == zone,
                                                 'TWI Zonal LBMP']).values
                            for zone in listOfZones})
    loadDA = pd.DataFrame({zone: (dfLoadDA.loc[dfLoadDA['Zone Name'] == zone,
                                               'DAM Forecast Load']).values
                           for zone in listOfZones})

    numberOfDays = int(len(priceDA.index)/24)

    # Compute the price differences between DA and RT prices for all options on
    # all days of the year. Store it as a pandas data frame where the 24 rows for
    # each day is flattened into one row. This operation essentially allows us to
    # independently think of each zone in each hour as a separate option. Also,
    # reshape the prices for the DA market in the same manner.

    priceDART = pd.DataFrame([priceRT.sub(priceDA).loc[day * 24:
                                                       (day + 1) * 24 - 1,
                              listOfZones].values.flatten()
                              for day in range(numberOfDays)],
                             columns=optionNames)

    priceDA = pd.DataFrame([priceDA.loc[day * 24: (day + 1) * 24 - 1,
                            listOfZones].values.flatten()
                            for day in range(numberOfDays)],
                           columns=optionNames)

    return priceDA, priceDART, loadDA
```

Create a function that creates the inputs for training a classifier

Create a function that takes the price and load data and creates two arrays 'X' and 'Y'. Essentially, the rows of 'X' contains all information relevant to predicting the sign of the price difference on the various options on the next day. It takes as an input, three pandas frames corresponding to the DA prices,

price differences, and the DA load predictions, and produces three outputs: the arrays 'X', 'Y', and the

In [5]:
```
def createClassifierIO(priceDA, priceDART, loadDA):

    # Define how many past days of prices to use for classification.

    pastPrices = range(1, 3)

    # Define how many past days of load predictions to use for classification.

    pastLoad = range(1, 3)

    # Define a date range within the year to create the arrays 'X' and 'Y' in a way
    # that past price and load data for the first day is within the date range in th
    # pandas frames passed as inputs.

    rangeOfDays = range(3, len(priceDA.index))

    # 'X' will contain three sets of variables:
    #   1. the DA prices from past days in the list 'pastDays',
    #   2. the differences between DA and RT prices from the same past days,
    #   3. the load predictions from past days in the list 'pastLoad'

    X = [np.concatenate((
        priceDA.loc[[(day - h) for h in pastPrices]].values.flatten(),
        priceDART.loc[[(day - h) for h in pastPrices]].values.flatten(),
        loadDA.loc[[(day - h) for h in pastLoad]].values.flatten()
    )) for day in rangeOfDays]

    # Scale the array 'X' to make its data zero mean and unit variance.
    X = StandardScaler().fit_transform(X)

    # 'Y' will contain zeros and ones, where a one indicates that the price in DA is
    # higher than in RT for a particular option. Recall that an option corresponds t
    # a zone at a particular hour of the day.

    Y = np.array([(priceDART.loc[day].values > 0).astype(int)
                  for day in rangeOfDays])

    # Return the arrays 'X' and 'Y', and finally the range of days from the year tha
    # will be utilized for training or testing the classifier.
    return X, Y, rangeOfDays
```

Design the training module.

The training module utilizes a year's worth of data to determine the following for each option, i.e., for each zone for each hour of the day:

1. Classifiers that predict the sign of the difference between DA and RT prices.
2. Statistics of the mean of the price difference.
3. A quantile of the day-ahead prices that we will use as our bid for each option. You will either train the classifiers here or load them from the folder './Classifiers'. Storing the classifiers from time to time allows you to only vary the bidding strategy and observe the annual reward rather than having to train the classifiers every time.

Define and train the classifiers or load pre-trained classifiers.

In [6]:
```
classifiers = []

# We have two options here. Use previous training experience, or learn anew.
useSavedClassifiers = False

if not useSavedClassifiers:

    print("Starting training module...\n")
    trainPriceDA, trainPriceDART, trainLoadDA = loadNYISOData(2015)

    numberOfDays =  int(len(trainPriceDA.index))
    print("Loaded hourly prices from 2015 for %d days." % numberOfDays)

    # We will implement a trading strategy, where we bid a particular quantile of th
    # DA prices for an option. If you do not know what a quantile means, refer to th
    # article on it. Essentially, a 95% quantile of the DA prices equals that value
    # 95% of the DA prices are below it. Store all quantiles starting from 50% in st
    # 5% in a dictionary. Store them in a pickle file.

    quantilesToStore = [0.70, 0.75, 0.80, 0.85, 0.90, 0.95]
    offerPrices = trainPriceDA.quantile(q=quantilesToStore).transpose().to_dict()
    pickle.dump(offerPrices, open("./Training/OfferPrices", 'wb'))

    # Calculate the average price spread for each option over the entire year. This
    # us in choosing our portfolio. Store it as a dictionary. Our bid will choose th
    # options that our classifier indicates that they will be profitable and histor
    # have higher average price differences, indicating that they have higher rate o
    # Store them using pickle.

    averagePriceSpread = trainPriceDART.mean(axis=0).transpose().to_dict()
    pickle.dump(averagePriceSpread, open("./Training/AveragePriceSpread", 'wb'))

    # Create the training dataset using the function 'createClassifierIO' on the pr
    # loads, and store them in 'trainX', and 'trainY'.

    trainX, trainY, _ = createClassifierIO(trainPriceDA, trainPriceDART, trainLoadDA

    # Define a collection of classifiers, one for each option. You can try differen
    # as that based on an SVM, logistic regression, multilayer perceptron based, etc
    # measure training accuracy to indicate how well the classifier works on the tra
    # However, good training accuracy does not always indicate good test performance
    # Avoid over-fitting.


    classifiers = [MLPClassifier(hidden_layer_sizes=(20, 10), max_iter=200)
                   for _ in range(nOptions)]

    trainingAccuracy = 0

    for ii in range(nOptions):
        classifiers[ii].fit(trainX, trainY[:, ii])
        print("Classifier trained for option " + optionNames[ii])
        trainingAccuracy += classifiers[ii].score(trainX, trainY[:, ii])

        # Store the classifier.
        pickle.dump(classifiers[ii], open("./Training/Classifier_" + optionNames[ii]

    print("\nOverall training accuracy = %1.2f percent." % (100 * trainingAccuracy/n
```

```
    del numberOfDays, trainPriceDA, trainLoadDA, trainPriceDART, trainX, trainY
else:

    # Load the classifiers, the offer prices at various quantiles, and the average p

    print("Loading previously trained variables...\n")
    classifiers = [pickle.load(open("./Training/Classifier_" + optionNames[ii], 'rb
                   for ii in range(nOptions)]
    offerPrices = pickle.load(open("./Training/OfferPrices", 'rb'))
    averagePriceSpread = pickle.load(open("./Training/AveragePriceSpread", 'rb'))

    print("All training variables were loaded successfully...\n")
```

```
Starting training module...

Loaded hourly prices from 2015 for 365 days.
Classifier trained for option CAPITL_Hour_0
Classifier trained for option CAPITL_Hour_1
Classifier trained for option CAPITL_Hour_2
Classifier trained for option CAPITL_Hour_3
Classifier trained for option CAPITL_Hour_4

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option CAPITL_Hour_5
Classifier trained for option CAPITL_Hour_6
Classifier trained for option CAPITL_Hour_7
Classifier trained for option CAPITL_Hour_8
Classifier trained for option CAPITL_Hour_9
Classifier trained for option CAPITL_Hour_10
Classifier trained for option CAPITL_Hour_11
Classifier trained for option CAPITL_Hour_12

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option CAPITL_Hour_13
Classifier trained for option CAPITL_Hour_14
Classifier trained for option CAPITL_Hour_15
Classifier trained for option CAPITL_Hour_16
Classifier trained for option CAPITL_Hour_17

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
Classifier trained for option CAPITL_Hour_18
Classifier trained for option CAPITL_Hour_19
Classifier trained for option CAPITL_Hour_20
Classifier trained for option CAPITL_Hour_21
Classifier trained for option CAPITL_Hour_22
Classifier trained for option CAPITL_Hour_23
Classifier trained for option CENTRL_Hour_0
Classifier trained for option CENTRL_Hour_1

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option CENTRL_Hour_2
Classifier trained for option CENTRL_Hour_3
Classifier trained for option CENTRL_Hour_4
Classifier trained for option CENTRL_Hour_5
Classifier trained for option CENTRL_Hour_6
Classifier trained for option CENTRL_Hour_7
Classifier trained for option CENTRL_Hour_8
Classifier trained for option CENTRL_Hour_9
Classifier trained for option CENTRL_Hour_10
Classifier trained for option CENTRL_Hour_11
Classifier trained for option CENTRL_Hour_12
Classifier trained for option CENTRL_Hour_13
Classifier trained for option CENTRL_Hour_14
Classifier trained for option CENTRL_Hour_15
Classifier trained for option CENTRL_Hour_16
Classifier trained for option CENTRL_Hour_17
Classifier trained for option CENTRL_Hour_18
Classifier trained for option CENTRL_Hour_19

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option CENTRL_Hour_20
Classifier trained for option CENTRL_Hour_21
Classifier trained for option CENTRL_Hour_22
Classifier trained for option CENTRL_Hour_23

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option DUNWOD_Hour_0
Classifier trained for option DUNWOD_Hour_1
Classifier trained for option DUNWOD_Hour_2
Classifier trained for option DUNWOD_Hour_3
Classifier trained for option DUNWOD_Hour_4
Classifier trained for option DUNWOD_Hour_5
Classifier trained for option DUNWOD_Hour_6
Classifier trained for option DUNWOD_Hour_7
Classifier trained for option DUNWOD_Hour_8
Classifier trained for option DUNWOD_Hour_9
Classifier trained for option DUNWOD_Hour_10
```

```
C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option DUNWOD_Hour_11
Classifier trained for option DUNWOD_Hour_12

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option DUNWOD_Hour_13
Classifier trained for option DUNWOD_Hour_14

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option DUNWOD_Hour_15
Classifier trained for option DUNWOD_Hour_16

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option DUNWOD_Hour_17
Classifier trained for option DUNWOD_Hour_18

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option DUNWOD_Hour_19
Classifier trained for option DUNWOD_Hour_20
Classifier trained for option DUNWOD_Hour_21
Classifier trained for option DUNWOD_Hour_22

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option DUNWOD_Hour_23
Classifier trained for option GENESE_Hour_0

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option GENESE_Hour_1

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
Classifier trained for option GENESE_Hour_2
Classifier trained for option GENESE_Hour_3

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option GENESE_Hour_4
Classifier trained for option GENESE_Hour_5
Classifier trained for option GENESE_Hour_6
Classifier trained for option GENESE_Hour_7
Classifier trained for option GENESE_Hour_8

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option GENESE_Hour_9
Classifier trained for option GENESE_Hour_10
Classifier trained for option GENESE_Hour_11

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option GENESE_Hour_12
Classifier trained for option GENESE_Hour_13

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option GENESE_Hour_14
Classifier trained for option GENESE_Hour_15
Classifier trained for option GENESE_Hour_16

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option GENESE_Hour_17
Classifier trained for option GENESE_Hour_18
Classifier trained for option GENESE_Hour_19
Classifier trained for option GENESE_Hour_20
Classifier trained for option GENESE_Hour_21
Classifier trained for option GENESE_Hour_22
Classifier trained for option GENESE_Hour_23
Classifier trained for option HUD VL_Hour_0
Classifier trained for option HUD VL_Hour_1
Classifier trained for option HUD VL_Hour_2
Classifier trained for option HUD VL_Hour_3
```

```
C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option HUD VL_Hour_4

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option HUD VL_Hour_5
Classifier trained for option HUD VL_Hour_6

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option HUD VL_Hour_7
Classifier trained for option HUD VL_Hour_8
Classifier trained for option HUD VL_Hour_9
Classifier trained for option HUD VL_Hour_10

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option HUD VL_Hour_11
Classifier trained for option HUD VL_Hour_12
Classifier trained for option HUD VL_Hour_13
Classifier trained for option HUD VL_Hour_14
Classifier trained for option HUD VL_Hour_15

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option HUD VL_Hour_16
Classifier trained for option HUD VL_Hour_17

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option HUD VL_Hour_18
Classifier trained for option HUD VL_Hour_19
Classifier trained for option HUD VL_Hour_20
Classifier trained for option HUD VL_Hour_21
Classifier trained for option HUD VL_Hour_22

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option HUD VL_Hour_23
```

```
C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option LONGIL_Hour_0

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option LONGIL_Hour_1
Classifier trained for option LONGIL_Hour_2
Classifier trained for option LONGIL_Hour_3

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option LONGIL_Hour_4
Classifier trained for option LONGIL_Hour_5
Classifier trained for option LONGIL_Hour_6
Classifier trained for option LONGIL_Hour_7
Classifier trained for option LONGIL_Hour_8
Classifier trained for option LONGIL_Hour_9

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option LONGIL_Hour_10

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option LONGIL_Hour_11
Classifier trained for option LONGIL_Hour_12
Classifier trained for option LONGIL_Hour_13
Classifier trained for option LONGIL_Hour_14

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
Classifier trained for option LONGIL_Hour_15
Classifier trained for option LONGIL_Hour_16
Classifier trained for option LONGIL_Hour_17
Classifier trained for option LONGIL_Hour_18
Classifier trained for option LONGIL_Hour_19
Classifier trained for option LONGIL_Hour_20
Classifier trained for option LONGIL_Hour_21
Classifier trained for option LONGIL_Hour_22
Classifier trained for option LONGIL_Hour_23
Classifier trained for option MHK VL_Hour_0
Classifier trained for option MHK VL_Hour_1
Classifier trained for option MHK VL_Hour_2
Classifier trained for option MHK VL_Hour_3
Classifier trained for option MHK VL_Hour_4
Classifier trained for option MHK VL_Hour_5
Classifier trained for option MHK VL_Hour_6
Classifier trained for option MHK VL_Hour_7

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MHK VL_Hour_8
Classifier trained for option MHK VL_Hour_9
Classifier trained for option MHK VL_Hour_10

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MHK VL_Hour_11

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MHK VL_Hour_12
Classifier trained for option MHK VL_Hour_13
Classifier trained for option MHK VL_Hour_14

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MHK VL_Hour_15
Classifier trained for option MHK VL_Hour_16
Classifier trained for option MHK VL_Hour_17
Classifier trained for option MHK VL_Hour_18
Classifier trained for option MHK VL_Hour_19

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MHK VL_Hour_20
```

```
C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MHK VL_Hour_21
Classifier trained for option MHK VL_Hour_22
Classifier trained for option MHK VL_Hour_23
Classifier trained for option MILLWD_Hour_0

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_1

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_2

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_3

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_4

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_5

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_6
Classifier trained for option MILLWD_Hour_7
Classifier trained for option MILLWD_Hour_8
Classifier trained for option MILLWD_Hour_9
Classifier trained for option MILLWD_Hour_10
Classifier trained for option MILLWD_Hour_11
Classifier trained for option MILLWD_Hour_12
Classifier trained for option MILLWD_Hour_13
Classifier trained for option MILLWD_Hour_14
Classifier trained for option MILLWD_Hour_15
Classifier trained for option MILLWD_Hour_16
```

```
C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_17

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option MILLWD_Hour_18
Classifier trained for option MILLWD_Hour_19
Classifier trained for option MILLWD_Hour_20
Classifier trained for option MILLWD_Hour_21
Classifier trained for option MILLWD_Hour_22
Classifier trained for option MILLWD_Hour_23
Classifier trained for option N.Y.C._Hour_0
Classifier trained for option N.Y.C._Hour_1

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option N.Y.C._Hour_2
Classifier trained for option N.Y.C._Hour_3
Classifier trained for option N.Y.C._Hour_4
Classifier trained for option N.Y.C._Hour_5

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option N.Y.C._Hour_6
Classifier trained for option N.Y.C._Hour_7
Classifier trained for option N.Y.C._Hour_8

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option N.Y.C._Hour_9

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
Classifier trained for option N.Y.C._Hour_10
Classifier trained for option N.Y.C._Hour_11
Classifier trained for option N.Y.C._Hour_12
Classifier trained for option N.Y.C._Hour_13
Classifier trained for option N.Y.C._Hour_14
Classifier trained for option N.Y.C._Hour_15
Classifier trained for option N.Y.C._Hour_16
Classifier trained for option N.Y.C._Hour_17
Classifier trained for option N.Y.C._Hour_18
Classifier trained for option N.Y.C._Hour_19
Classifier trained for option N.Y.C._Hour_20
Classifier trained for option N.Y.C._Hour_21
Classifier trained for option N.Y.C._Hour_22
Classifier trained for option N.Y.C._Hour_23
Classifier trained for option NORTH_Hour_0
Classifier trained for option NORTH_Hour_1
Classifier trained for option NORTH_Hour_2

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option NORTH_Hour_3

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option NORTH_Hour_4
Classifier trained for option NORTH_Hour_5
Classifier trained for option NORTH_Hour_6
Classifier trained for option NORTH_Hour_7
Classifier trained for option NORTH_Hour_8
Classifier trained for option NORTH_Hour_9
Classifier trained for option NORTH_Hour_10
Classifier trained for option NORTH_Hour_11
Classifier trained for option NORTH_Hour_12
Classifier trained for option NORTH_Hour_13
Classifier trained for option NORTH_Hour_14
Classifier trained for option NORTH_Hour_15
Classifier trained for option NORTH_Hour_16

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option NORTH_Hour_17
Classifier trained for option NORTH_Hour_18
Classifier trained for option NORTH_Hour_19
Classifier trained for option NORTH_Hour_20

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option NORTH_Hour_21
```

```
C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option NORTH_Hour_22
Classifier trained for option NORTH_Hour_23
Classifier trained for option WEST_Hour_0
Classifier trained for option WEST_Hour_1
Classifier trained for option WEST_Hour_2
Classifier trained for option WEST_Hour_3

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_4

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_5

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_6
Classifier trained for option WEST_Hour_7
Classifier trained for option WEST_Hour_8

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_9
Classifier trained for option WEST_Hour_10

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_11

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
Classifier trained for option WEST_Hour_12
Classifier trained for option WEST_Hour_13
Classifier trained for option WEST_Hour_14
Classifier trained for option WEST_Hour_15
Classifier trained for option WEST_Hour_16
Classifier trained for option WEST_Hour_17
Classifier trained for option WEST_Hour_18
Classifier trained for option WEST_Hour_19

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_20

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_21

C:\Users\matt3\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\n
eural_network\_multilayer_perceptron.py:679: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Classifier trained for option WEST_Hour_22
Classifier trained for option WEST_Hour_23

Overall training accuracy = 99.96 percent.
```

Test the classifier's accuracy on test data.

In [7]:
```
# First, load the test data from NYISO for the year 2016. Again, utilize our functio
# named 'loadNYISOData'.

print("Starting the testing module...\n")
testPriceDA, testPriceDART, testLoadDA = loadNYISOData(2016)

# Create the data for the classifier using the function 'createClassifierIO'.
testX, testY, rangeOfDays = createClassifierIO(testPriceDA, testPriceDART, testLoadD

# The next step is not useful for implementing the trading strategy, but quickly che
# your trained classifiers are for the test data. Training accuracy is not always in
# test accuracy.

testingAccuracy = [classifiers[ii].score(testX, testY[:, ii]) for ii in range(nOptio
print("Test Accuracy Stats: Min = %.2f%%, Avg = %.2f%%, Max = %.2f%%" %
      (100 * np.min(testingAccuracy),
       100 * np.mean(testingAccuracy),
       100 * np.max(testingAccuracy)))

# Utilize the classifiers to predict the sign of DA - RT prices for each day in 'ran
# the test data. Store the result in a pandas data frame with columns as the option
# day in year as index.
predictedY = pd.DataFrame(np.column_stack([classifiers[ii].predict(testX) for ii in
                          columns=optionNames, index=rangeOfDays)
```

```
Starting the testing module...

Test Accuracy Stats: Min = 49.04%, Avg = 58.94%, Max = 69.70%
```

Design and implement the trading strategy.

We define a fairly simple trading strategy. Define a total budget that you are willing to spend in the DA market. Recall that we only invest in options where we buy at the DA market and sell at the RT market. When your bid for one unit of an option clears in the DA market, you have to pay the DA price for that option. There are two possibilities:

1. Your bid clears: Therefore, your offer price was higher than the DA price.
2. Your bid does not clear: Then, the DA price was higher than your bid. In both these cases, the maximum you have to pay is your bid. Therefore, we will enforce that your bids across all options in a day does not exceed your total budget. Keep track of how rewards grow (or fall) through the year as you utilize your strategy. Also, keep track of how much rewards you get from each option. We shall visualize these results after implementing the trading strategy over the NYISO data for 2016.

Choose the bid prices as a suitable quantile of the historical DA prices. The higher the quantile, the better your chances are that your bid will be cleared. However, a higher quantile also indicates that you are budgeting more money for each option, and hence, you will buy fewer options.

In [11]:
```
dailyBudget = 250000
quantileOffer =0.8

# Keep track of your rewards in each day, a cumulative reward over the year, and a t
# year. Also, keep track of the total reward from each option. Store them as diction
# indexed by day of the year.
reward = {}
cumulativeReward = {}
totalReward = 0
optionReturn = dict((option, 0) for option in optionNames)

# Implement the trading strategy on each day!

for day in rangeOfDays:

    reward[day] = 0

    # Find the options that your classifier says that should be profitable. Store th
    # names in chosenOptions.
    chosenOptionNumbers = np.ravel(list(np.nonzero(predictedY.loc[day].values)))
    if np.size(chosenOptionNumbers) == 0:
        continue
    chosenOptions = [optionNames[i] for i in chosenOptionNumbers]

    # Design the portfolio based on average price spreads. Our strategy is that as l
    # exceeded your daily budget, pick an option from the list of 'chosenOptions' pr
    # the probability of choosing it is proportional to exponential(historical rewar
    # is, a historically profitable option is chosen more often than one that is not
    # decreasing your budget with each bid.

    # Start with an empty portfolio.

    portfolio = dict((option, 0) for option in chosenOptions)

    # Calculate the probabilities of choosing each option among the list 'chosenOpti
    # 'chosenOptions' contains the options that your classifier indicates as being p
    priceSpreads = [1.0 * averagePriceSpread[option] for option in chosenOptions]
    probabilityOptions = [np.exp(p) for p in priceSpreads]
    probabilityOptions /= np.sum(probabilityOptions)

    # Start with your daily budget.
    budget = dailyBudget

    # Sampling among the profitable options and bid based on them.
    while budget > np.median([offerPrices[quantileOffer][option] for option in chose

        optionToBuy = choice(chosenOptions, p=probabilityOptions)

        if budget >= offerPrices[quantileOffer][optionToBuy]:
            portfolio[optionToBuy] += 1
            budget -= offerPrices[quantileOffer][optionToBuy]

    # Compute the reward from the day. Go through each of the options you have decid
    # If the DA price is lower than the bid price, then your bid is cleared. For eac
    # have bought, you get a reward equal to the DA - RT price.

    for option in chosenOptions:
        if testPriceDA.at[day, option] < offerPrices[quantileOffer][option]:
```

```
            rewardOptionDay = testPriceDART.at[day, option] * portfolio[option]
            optionReturn[option] += rewardOptionDay
            reward[day] += rewardOptionDay

    totalReward += reward[day]

    # Calculate the cumulative reward in millions of dollars.
    cumulativeReward[day] = totalReward/1000000

    print("Day " + str(day) + ": Reward (in $) = " + "{0:,.0f}".format(reward[day]))

print("Total money earned over the year (in $) = " + "{0:,.0f}".format(totalReward))
```

```
Day 3: Reward (in $) = -77,997
Day 4: Reward (in $) = 1,414,047
Day 5: Reward (in $) = 25,178
Day 6: Reward (in $) = -28,369
Day 7: Reward (in $) = -12,575
Day 8: Reward (in $) = -39,926
Day 9: Reward (in $) = -42,636
Day 10: Reward (in $) = -104,316
Day 11: Reward (in $) = -43,041
Day 12: Reward (in $) = -56,447
Day 13: Reward (in $) = -39,753
Day 14: Reward (in $) = -36,639
Day 15: Reward (in $) = 8,801
Day 16: Reward (in $) = 43,385
Day 17: Reward (in $) = -67,580
Day 18: Reward (in $) = -39,822
Day 19: Reward (in $) = 30,899
Day 20: Reward (in $) = -5,094
Day 21: Reward (in $) = 35,761
Day 22: Reward (in $) = 85,031
Day 23: Reward (in $) = 34,459
Day 24: Reward (in $) = -86,349
Day 25: Reward (in $) = -56,894
Day 26: Reward (in $) = -27,164
Day 27: Reward (in $) = -6,392
Day 28: Reward (in $) = -13,114
Day 29: Reward (in $) = -22,813
Day 30: Reward (in $) = -28,635
Day 31: Reward (in $) = -31,701
Day 32: Reward (in $) = -3,263
Day 33: Reward (in $) = -211,493
Day 34: Reward (in $) = -15,485
Day 35: Reward (in $) = 42,196
Day 36: Reward (in $) = -36,750
Day 37: Reward (in $) = -40,318
Day 38: Reward (in $) = -351
Day 39: Reward (in $) = 12,944
Day 40: Reward (in $) = -32,792
Day 41: Reward (in $) = -38,097
Day 42: Reward (in $) = -15,285
Day 43: Reward (in $) = -55,504
Day 44: Reward (in $) = -33,368
Day 45: Reward (in $) = -17,148
Day 46: Reward (in $) = 91,170
Day 47: Reward (in $) = 54,701
Day 48: Reward (in $) = -54,520
Day 49: Reward (in $) = -51,205
Day 50: Reward (in $) = -55,337
Day 51: Reward (in $) = 5,147
Day 52: Reward (in $) = -19,548
Day 53: Reward (in $) = -7,132
Day 54: Reward (in $) = 5,048
Day 55: Reward (in $) = -3,940
Day 56: Reward (in $) = -43,593
Day 57: Reward (in $) = -28,529
Day 58: Reward (in $) = -19,809
Day 59: Reward (in $) = -16,370
Day 60: Reward (in $) = -68,574
```

```
Day 61: Reward (in $) = -40,195
Day 62: Reward (in $) = 1,123
Day 63: Reward (in $) = -7,826
Day 64: Reward (in $) = -31,865
Day 65: Reward (in $) = 13,179
Day 66: Reward (in $) = 62,808
Day 67: Reward (in $) = -15,213
Day 68: Reward (in $) = -44,821
Day 69: Reward (in $) = -38,966
Day 70: Reward (in $) = 83,903
Day 71: Reward (in $) = -9,202
Day 72: Reward (in $) = -9,939
Day 73: Reward (in $) = -95,805
Day 74: Reward (in $) = 768,102
Day 75: Reward (in $) = -25,953
Day 76: Reward (in $) = -111,352
Day 77: Reward (in $) = -47,624
Day 78: Reward (in $) = -27,026
Day 79: Reward (in $) = 10,793
Day 80: Reward (in $) = -40,065
Day 81: Reward (in $) = 70,832
Day 82: Reward (in $) = 76,873
Day 83: Reward (in $) = -107,978
Day 84: Reward (in $) = 518
Day 85: Reward (in $) = 67,328
Day 86: Reward (in $) = -33,391
Day 87: Reward (in $) = 3,018,976
Day 88: Reward (in $) = 632,168
Day 89: Reward (in $) = -41,623
Day 90: Reward (in $) = -29,384
Day 91: Reward (in $) = -27,713
Day 92: Reward (in $) = -28,636
Day 93: Reward (in $) = 1,835
Day 94: Reward (in $) = 35,415
Day 95: Reward (in $) = -7,868
Day 96: Reward (in $) = 252,318
Day 97: Reward (in $) = -25,361
Day 98: Reward (in $) = -2,589
Day 99: Reward (in $) = -23,776
Day 100: Reward (in $) = 17,028
Day 101: Reward (in $) = -24,852
Day 102: Reward (in $) = -178
Day 103: Reward (in $) = 158,058
Day 104: Reward (in $) = -70,129
Day 105: Reward (in $) = -11,185
Day 106: Reward (in $) = 3,242
Day 107: Reward (in $) = 1,191
Day 108: Reward (in $) = 50,158
Day 109: Reward (in $) = -4,834
Day 110: Reward (in $) = -45,891
Day 111: Reward (in $) = -29,466
Day 112: Reward (in $) = 7,748
Day 113: Reward (in $) = -44,023
Day 114: Reward (in $) = -58,149
Day 115: Reward (in $) = -121,665
Day 116: Reward (in $) = 11,151
Day 117: Reward (in $) = -24,559
Day 118: Reward (in $) = 8,410
Day 119: Reward (in $) = -75,064
```

```
Day 120: Reward (in $) = 2,139
Day 121: Reward (in $) = 39,661
Day 122: Reward (in $) = 30
Day 123: Reward (in $) = -62,588
Day 124: Reward (in $) = 2,813
Day 125: Reward (in $) = -9,448
Day 126: Reward (in $) = -11,220
Day 127: Reward (in $) = -38,888
Day 128: Reward (in $) = -41,880
Day 129: Reward (in $) = -93,371
Day 130: Reward (in $) = -36,004
Day 131: Reward (in $) = 23,653
Day 132: Reward (in $) = 157,660
Day 133: Reward (in $) = 72,988
Day 134: Reward (in $) = -46,338
Day 135: Reward (in $) = -13,832
Day 136: Reward (in $) = -51,746
Day 137: Reward (in $) = -47,771
Day 138: Reward (in $) = -87,170
Day 139: Reward (in $) = 4,327
Day 140: Reward (in $) = -44,837
Day 141: Reward (in $) = 18,545
Day 142: Reward (in $) = 23,988
Day 143: Reward (in $) = -99,713
Day 144: Reward (in $) = -77,452
Day 145: Reward (in $) = -53,809
Day 146: Reward (in $) = -19,545
Day 147: Reward (in $) = -25,641
Day 148: Reward (in $) = 224,376
Day 149: Reward (in $) = -7,101
Day 150: Reward (in $) = 50,411
Day 151: Reward (in $) = 405,633
Day 152: Reward (in $) = 370,844
Day 153: Reward (in $) = 281,098
Day 154: Reward (in $) = -732
Day 155: Reward (in $) = -40,357
Day 156: Reward (in $) = -41,118
Day 157: Reward (in $) = -130,491
Day 158: Reward (in $) = 17,031
Day 159: Reward (in $) = -46,387
Day 160: Reward (in $) = -139,293
Day 161: Reward (in $) = 5,078
Day 162: Reward (in $) = 164,122
Day 163: Reward (in $) = -14,154
Day 164: Reward (in $) = -6,555
Day 165: Reward (in $) = 263,544
Day 166: Reward (in $) = -28,050
Day 167: Reward (in $) = 24,915
Day 168: Reward (in $) = -1,148
Day 169: Reward (in $) = -2,785
Day 170: Reward (in $) = -49,354
Day 171: Reward (in $) = -3,990
Day 172: Reward (in $) = -4,698
Day 173: Reward (in $) = 178,984
Day 174: Reward (in $) = -16,519
Day 175: Reward (in $) = 17,003
Day 176: Reward (in $) = -16,356
Day 177: Reward (in $) = -27,640
Day 178: Reward (in $) = -10,544
```

```
Day 179: Reward (in $) = -45,741
Day 180: Reward (in $) = -22,836
Day 181: Reward (in $) = -22,746
Day 182: Reward (in $) = -35,162
Day 183: Reward (in $) = -85,605
Day 184: Reward (in $) = -62,407
Day 185: Reward (in $) = -20,279
Day 186: Reward (in $) = 85,492
Day 187: Reward (in $) = 16,244
Day 188: Reward (in $) = 5,554
Day 189: Reward (in $) = -3,008
Day 190: Reward (in $) = -101,760
Day 191: Reward (in $) = -24,506
Day 192: Reward (in $) = -23,310
Day 193: Reward (in $) = -64,756
Day 194: Reward (in $) = -239
Day 195: Reward (in $) = 1,080
Day 196: Reward (in $) = 6,780
Day 197: Reward (in $) = 34,721
Day 198: Reward (in $) = 58,016
Day 199: Reward (in $) = -1,742
Day 200: Reward (in $) = -51,952
Day 201: Reward (in $) = -27,929
Day 202: Reward (in $) = 1,822
Day 203: Reward (in $) = 5,709
Day 204: Reward (in $) = 3,917
Day 205: Reward (in $) = -72,237
Day 206: Reward (in $) = -4,342
Day 207: Reward (in $) = -5,268
Day 208: Reward (in $) = 187
Day 209: Reward (in $) = 5,652
Day 210: Reward (in $) = 78,185
Day 211: Reward (in $) = 47,243
Day 212: Reward (in $) = 49,293
Day 213: Reward (in $) = -1,333
Day 214: Reward (in $) = -6,153
Day 215: Reward (in $) = -68,275
Day 216: Reward (in $) = 7,499
Day 217: Reward (in $) = 1,578
Day 218: Reward (in $) = 34,380
Day 219: Reward (in $) = -25,161
Day 220: Reward (in $) = -1,783
Day 221: Reward (in $) = -32,249
Day 222: Reward (in $) = 15,327
Day 223: Reward (in $) = 3,433
Day 224: Reward (in $) = 211
Day 225: Reward (in $) = 701
Day 226: Reward (in $) = -23,685
Day 227: Reward (in $) = -601
Day 228: Reward (in $) = 1,644
Day 229: Reward (in $) = -1,446
Day 230: Reward (in $) = -2,441
Day 231: Reward (in $) = 2,529
Day 232: Reward (in $) = -38,296
Day 233: Reward (in $) = -52,423
Day 234: Reward (in $) = -103,445
Day 235: Reward (in $) = -107,855
Day 236: Reward (in $) = 3,696
Day 237: Reward (in $) = 51,077
```

```
Day 238: Reward (in $) = -2,805
Day 239: Reward (in $) = -14,784
Day 240: Reward (in $) = -13,335
Day 241: Reward (in $) = -1,161
Day 242: Reward (in $) = -43,271
Day 243: Reward (in $) = -3,556
Day 244: Reward (in $) = -920
Day 245: Reward (in $) = -87,449
Day 246: Reward (in $) = -46,785
Day 247: Reward (in $) = -10,087
Day 248: Reward (in $) = -32,282
Day 249: Reward (in $) = -10,263
Day 250: Reward (in $) = -19,060
Day 251: Reward (in $) = 1,197
Day 252: Reward (in $) = 6,249
Day 253: Reward (in $) = 174,116
Day 254: Reward (in $) = -13,399
Day 255: Reward (in $) = -48,373
Day 256: Reward (in $) = -4,606
Day 257: Reward (in $) = -6,530
Day 258: Reward (in $) = -20,819
Day 259: Reward (in $) = -30,373
```

```
Day 260: Reward (in $) = -32,660
Day 261: Reward (in $) = 23,488
Day 262: Reward (in $) = 6,580
Day 263: Reward (in $) = -641
Day 264: Reward (in $) = -60,472
Day 265: Reward (in $) = -25,220
Day 266: Reward (in $) = -71,096
Day 267: Reward (in $) = -17,166
Day 268: Reward (in $) = -22,876
Day 269: Reward (in $) = -67,001
Day 270: Reward (in $) = -70,989
Day 271: Reward (in $) = -31,167
Day 272: Reward (in $) = -60,764
Day 273: Reward (in $) = -54,737
Day 274: Reward (in $) = 12,477
Day 275: Reward (in $) = 23,891
Day 276: Reward (in $) = -32,245
Day 277: Reward (in $) = -6,498
Day 278: Reward (in $) = 187,018
Day 279: Reward (in $) = -74,542
Day 280: Reward (in $) = 13,300
Day 281: Reward (in $) = -12,646
Day 282: Reward (in $) = 14,699
Day 283: Reward (in $) = -2,246
Day 284: Reward (in $) = 7,632
Day 285: Reward (in $) = -64,832
Day 286: Reward (in $) = -57,023
Day 287: Reward (in $) = -68,967
Day 288: Reward (in $) = 11,157
Day 289: Reward (in $) = 16,366
Day 290: Reward (in $) = 6,418
Day 291: Reward (in $) = -6,988
Day 292: Reward (in $) = -3,303
Day 293: Reward (in $) = 125,354
Day 294: Reward (in $) = 15,819
Day 295: Reward (in $) = -13,737
Day 296: Reward (in $) = -61,193
Day 297: Reward (in $) = -40,019
Day 298: Reward (in $) = -12,452
Day 299: Reward (in $) = -61,745
Day 300: Reward (in $) = -30,474
Day 301: Reward (in $) = -11,293
Day 302: Reward (in $) = -8,657
Day 303: Reward (in $) = 15,668
Day 304: Reward (in $) = -51,900
Day 305: Reward (in $) = -15,861
Day 306: Reward (in $) = -8,025
Day 307: Reward (in $) = 45,511
Day 308: Reward (in $) = -41,108
Day 309: Reward (in $) = -12,742
Day 310: Reward (in $) = 2,402
Day 311: Reward (in $) = -30,469
Day 312: Reward (in $) = 46,825
Day 313: Reward (in $) = -8,138
Day 314: Reward (in $) = -31,158
Day 315: Reward (in $) = -33,025
Day 316: Reward (in $) = -11,931
Day 317: Reward (in $) = -30,702
```

```
Day 318: Reward (in $) = -14,243
Day 319: Reward (in $) = 2,940
Day 320: Reward (in $) = 14,970
Day 321: Reward (in $) = 18,877
Day 322: Reward (in $) = 1,378
Day 323: Reward (in $) = 7,113
Day 324: Reward (in $) = 16,003
Day 325: Reward (in $) = -22,554
Day 326: Reward (in $) = 864
Day 327: Reward (in $) = 20,204
Day 328: Reward (in $) = 114,144
Day 329: Reward (in $) = 36,110
Day 330: Reward (in $) = 6,351
Day 331: Reward (in $) = -10,353
Day 332: Reward (in $) = -56,458
Day 333: Reward (in $) = -69,144
Day 334: Reward (in $) = -2,620
Day 335: Reward (in $) = -21,536
Day 336: Reward (in $) = -17,588
Day 337: Reward (in $) = 9,219
Day 338: Reward (in $) = -9,027
Day 339: Reward (in $) = -32,691
Day 340: Reward (in $) = -36,962
Day 341: Reward (in $) = -41,449
Day 342: Reward (in $) = -91,573
Day 343: Reward (in $) = -40,680
Day 344: Reward (in $) = 70,904
Day 345: Reward (in $) = 74,547
Day 346: Reward (in $) = -38,321
Day 347: Reward (in $) = 9,290
Day 348: Reward (in $) = -43,442
Day 349: Reward (in $) = 40,498
Day 350: Reward (in $) = -375
Day 351: Reward (in $) = 177,879
Day 352: Reward (in $) = 9,244
Day 353: Reward (in $) = -10,948
Day 354: Reward (in $) = -41,961
Day 355: Reward (in $) = -20,750
Day 356: Reward (in $) = 1,395
Day 357: Reward (in $) = -82,409
Day 358: Reward (in $) = -48,321
Day 359: Reward (in $) = -10,042
Day 360: Reward (in $) = -12,383
Day 361: Reward (in $) = -82,146
Day 362: Reward (in $) = 25,445
Day 363: Reward (in $) = 23,829
Day 364: Reward (in $) = -3,940
Day 365: Reward (in $) = -27,416
Total money earned over the year (in $) = 3,182,875
```

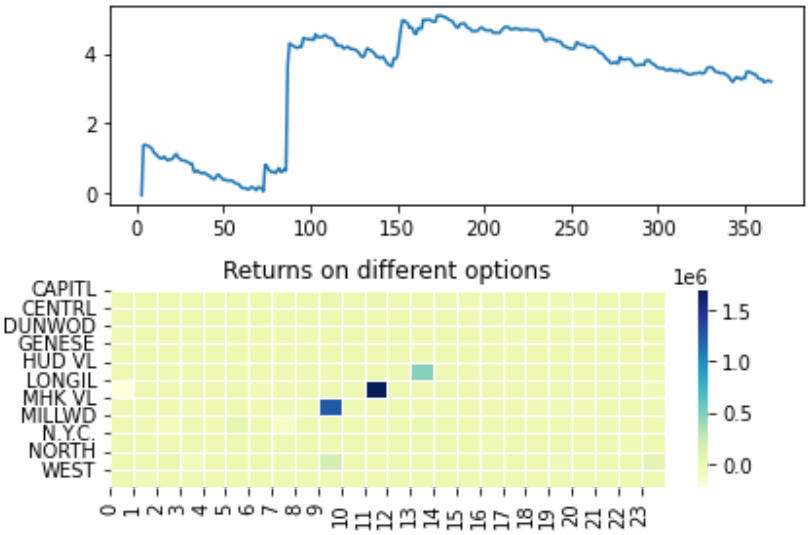
Task1: Visualize the rewards (25 points)

We would like to plot the cumulative reward over the year 2016. By cumulative reward on a particular date, we mean the total reward from the start of the year till that date.

Also, plot a heat map of the returns from each option.

Fill in the missing lines below.

In [10]:
```
# Plot the cumulative reward over the year 2016. Also, plot a heat map of the returr
# each option.

fig, axs = plt.subplots(2, 1, tight_layout=True)
axs = axs.ravel()


## Enter code here where you plot in axs[0].
axs[0].plot(*zip(*sorted(cumulativeReward.items())))


axs[1] = sns.heatmap(np.reshape(list(optionReturn.values()), (len(listOfZones), 24))
                     linewidth=0.5,
                     cmap="YlGnBu")

axs[1].set_yticks(range(len(listOfZones)))
axs[1].set_yticklabels(listOfZones, rotation=0)
axs[1].set_xticks(range(24))
axs[1].set_xticklabels(range(24), rotation=90)
axs[1].set_title('Returns on different options')

plt.show()
```



Task 2: Choosing a classifier (30 points)

We used a multilayer perceptron classifier. Your task is to try out SVM and logistic regression classifiers, and explore which one leads to more profits. Use the relevant functions from 'sklearn'.

(comments here, add a new code cell below)

In [*]:
```
def trading(dailyBudget, quantileOffer):
    # Keep track of your rewards in each day, a cumulative reward over the year, and
    # year. Also, keep track of the total reward from each option. Store them as dic
    # indexed by day of the year.
    global reward
    global cumulativeReward
    global totalReward
    global optionReturn

    reward = {}
    cumulativeReward = {}
    totalReward = 0
    optionReturn = dict((option, 0) for option in optionNames)

    # Implement the trading strategy on each day!

    for day in rangeOfDays:

        reward[day] = 0

        # Find the options that your classifier says that should be profitable. Stor
        # names in chosenOptions.
        chosenOptionNumbers = np.ravel(list(np.nonzero(predictedY.loc[day].values))
        if np.size(chosenOptionNumbers) == 0:
            continue
        chosenOptions = [optionNames[i] for i in chosenOptionNumbers]

        # Design the portfolio based on average price spreads. Our strategy is that
        # exceeded your daily budget, pick an option from the list of 'chosenOptions
        # the probability of choosing it is proportional to exponential(historical r
        # is, a historically profitable option is chosen more often than one that is
        # decreasing your budget with each bid.

        # Start with an empty portfolio.

        portfolio = dict((option, 0) for option in chosenOptions)

        # Calculate the probabilities of choosing each option among the list 'chosen
        # 'chosenOptions' contains the options that your classifier indicates as bet
        priceSpreads = [1.0 * averagePriceSpread[option] for option in chosenOptions
        probabilityOptions = [np.exp(p) for p in priceSpreads]
        probabilityOptions /= np.sum(probabilityOptions)

        # Start with your daily budget.
        budget = dailyBudget

        # Sampling among the profitable options and bid based on them.
        while budget > np.median([offerPrices[quantileOffer][option] for option in c

            optionToBuy = choice(chosenOptions, p=probabilityOptions)

            if budget >= offerPrices[quantileOffer][optionToBuy]:
                portfolio[optionToBuy] += 1
                budget -= offerPrices[quantileOffer][optionToBuy]

        # Compute the reward from the day. Go through each of the options you have c
        # If the DA price is lower than the bid price, then your bid is cleared. For
        # have bought, you get a reward equal to the DA - RT price.
```

```
        for option in chosenOptions:
            if testPriceDA.at[day, option] < offerPrices[quantileOffer][option]:
                rewardOptionDay = testPriceDART.at[day, option] * portfolio[option]
                optionReturn[option] += rewardOptionDay
                reward[day] += rewardOptionDay

        totalReward += reward[day]

        # Calculate the cumulative reward in millions of dollars.
        cumulativeReward[day] = totalReward/1000000

    print("Total money earned over the year (in $) = " + "{0:,.0f}".format(totalRewa
```

SVC with the kernel set to linear makes the most profit out of all the classifiers.

Task 3: Quantile (25 points)

For the best classifier, try different quantile choices from the following list: 0.70, 0.75, 0.80, 0.85, 0.90, 0.95. Rank them in terms of profits (low to high).

How do you expect the quantile to affect the portfolio?

(comments here, add a new code cell below)

In [*]:
```
# Code for task 3
qlist = [0.70, 0.75, 0.80, 0.85, 0.90, 0.95]
profit = dict((i, 0) for i in qlist)
for q in qlist:
    trading(dailyBudget=250000, quantileOffer=q)
    profit[q] = totalReward
print("\nRank of quantiles:")
print(dict(sorted(profit.items(), key=lambda item: item[1])).keys())
```

Rank in terms of profits:

0.7, 0.95, 0.9, 0.85, 0.8, 0.75

If the quantile is too small, then the bid price would have a larger probability to be lower than the DA price which would result in lower profit. In turn if the quantile is too high, the budget would be used up faster which would also lead to a smaller chance to make a profit.

Task 4: Daily budget (25 points)

We used a daily budget of 250,000. Try different budgets, let's say, 100,000, 150,000, 200,000, and 300,000. Rank them in terms of profits (low to high). Fix the quantile to the one that led to maximum profits in the previous task.

(comments here, add a new code cell below)

In [ ]:
```
# Code for task 4
blist = [100000, 150000, 200000, 300000]
profit = dict((i, 0) for i in blist)
for b in blist:
    trading(dailyBudget=b, quantileOffer=0.75)
    profit[b] = totalReward
print("\nRank of budgets:")
print(dict(sorted(profit.items(), key=lambda item: item[1])).keys())
```

Rank of daily budget:

100,000, 150,000, 200,000, 300,000

Task 5: Beat the algorithm! (40 points)

Make changes to increase the profits further! Try implementing a different trading strategy. Explain what did for improvement and how much profits you made. Do not exceed the 250,000 budget!

(comments here, add a new code cell below)

In [*]:
```
# Code for task 5
dailyBudget = 250000
quantileOffer = 0.75
tune = 0.85

# Keep track of your rewards in each day, a cumulative reward over the year, and a t
# year. Also, keep track of the total reward from each option. Store them as dictior
# indexed by day of the year.
reward = {}
cumulativeReward = {}
totalReward = 0
optionReturn = dict((option, 0) for option in optionNames)

# Implement the trading strategy on each day!

for day in rangeOfDays:

    reward[day] = 0

    # Find the options that your classifier says that should be profitable. Store th
    # names in chosenOptions.
    chosenOptionNumbers = np.ravel(list(np.nonzero(predictedY.loc[day].values)))
    if np.size(chosenOptionNumbers) == 0:
        continue
    chosenOptions = [optionNames[i] for i in chosenOptionNumbers]

    # Design the portfolio based on average price spreads. Our strategy is that as l
    # exceeded your daily budget, pick an option from the list of 'chosenOptions' pr
    # the probability of choosing it is proportional to exponential(historical rewar
    # is, a historically profitable option is chosen more often than one that is not
    # decreasing your budget with each bid.

    # Start with an empty portfolio.

    portfolio = dict((option, 0) for option in chosenOptions)

    # Calculate the probabilities of choosing each option among the list 'chosenOpti
    # 'chosenOptions' contains the options that your classifier indicates as being p
    priceSpreads = [1.0 * averagePriceSpread[option] for option in chosenOptions]

    if tune:
        k_least = int(len(priceSpreads) * tune)
        idx = np.argpartition(np.array(priceSpreads), k_least)[:k_least]
        chosenOptions = np.delete(np.array(chosenOptions), idx)
        priceSpreads = np.delete(np.array(priceSpreads), idx)

    probabilityOptions = [np.exp(p) for p in priceSpreads]
    probabilityOptions /= np.sum(probabilityOptions)

    # Start with your daily budget.
    budget = dailyBudget

    # Sampling among the profitable options and bid based on them.
    while budget > np.median([offerPrices[quantileOffer][option] for option in chose

        optionToBuy = choice(chosenOptions, p=probabilityOptions)

        if budget >= offerPrices[quantileOffer][optionToBuy]:
```

```
            portfolio[optionToBuy] += 1
            budget -= offerPrices[quantileOffer][optionToBuy]

    # Compute the reward from the day. Go through each of the options you have decid
    # If the DA price is lower than the bid price, then your bid is cleared. For eac
    # have bought, you get a reward equal to the DA - RT price.

    for option in chosenOptions:
        if testPriceDA.at[day, option] < offerPrices[quantileOffer][option]:
            rewardOptionDay = testPriceDART.at[day, option] * portfolio[option]
            optionReturn[option] += rewardOptionDay
            reward[day] += rewardOptionDay

    totalReward += reward[day]

    # Calculate the cumulative reward in millions of dollars.
    cumulativeReward[day] = totalReward/1000000

    print("Day " + str(day) + ": Reward (in $) = " + "{0:,.0f}".format(reward[day]))

print("Total money earned over the year (in $) = " + "{0:,.0f}".format(totalReward))
```

Type *Markdown* and LaTeX: α^2

For improvement, I added a tuning option to remove some of the options in the data with lower price spreads, increasing the profits compared to the orignal implementation.