Project Phase III *by*

Saurabh Katkar

Collaborators: None

Mode of Execution:

- Place the agent_skatkar.py file in the same folder as the agents.py and the simulate_agents_phase2.py file
- Add “from agent_skatkar import *” at the beginning of simulate_agents_phase2.py file.
- Added “agent = Agent_skatkar("skatkar")” to add the user-create agent in simulate_agents_phase4.py file.
- This code was executed in IPython using the Anaconda environment.

Overview

Given two products to start with, a Good product and a Bad product, along with their features, and a market place where products with varying values, prices, and features purchase **n** products, one at a time, to maximize your wealth.

Strategy implemented:

We apply different classification algorithms to the products' features in order to get the quality of the product ie. Whether Good or Bad. We are gifted two products, one Good and one Bad which are used to train the classifier, using which a prediction is made on the product quality.

On receiving a good or a bad product:

Using a classification algorithm, we train the classifier using the gifted product using the classifiers **fit()** method.

Once the classifier is trained with the gifted product, the classifier makes prediction by using the **predict_proba()** method.

Thus we determine the probability of the product being good and also the products with value greater than 1000 is taken into consideration.

After the product is taken into consideration, it is stored in an array to be used for training for the next set of examples.

Choosing the next product:

Once examined, the product is stored for the purpose of storing it for the next set of examples that need to be tested.

So the next product under consideration is trained with the training data which has increased by the previous sample and subsequently conclusion is drawn on the quality of the product ie. Whether the product is Good or Bad

As such the good products are selected for our consideration.

Strategies selected and Simulation Results:

KMeans(clustering)

Agent_skatkar's final wealth: $95,660.20 Agent_skatkar
has 497 good products.

BernoulliNB
```
Agent_ skatkar's final wealth:      $310,468.12 Agent_
skatkar has 707 good products.
```

Logistic Regression
```
Agent_ skatkar's final wealth:      $320,175.71 Agent_
skatkar has 645 good products.
```

GaussianNB
```
Agent_ skatkar's final wealth:      $313,625.52 Agent_
skatkar has 669 good products.
```

KNeighborsClassifier
```
Agent_ skatkar's final wealth:      $312,379.11 Agent_
skatkar has 620 good products.
```

SVC
```
Agent_ skatkar's final wealth:      $311,824.03 Agent_
skatkar has 631 good products.
```

DecisionTreeClassifier
```
Agent_ skatkar's final wealth:      $270,815.76 Agent_
skatkar has 536 good products.
```

RandomForestClassifier
```
Agent_ skatkar's final wealth:      $309,632.43 Agent_
skatkar has 643 good products.
```

AdaBoostClassifier
```
Agent_ skatkar's final wealth:      $313,623.56 Agent_
skatkar has 670 good products.
```

Baseline Agents:

CheapAgent Output:
```
Agent_cheap's final wealth:     $243,838.41 Agent_cheap
has 498 good products.
```

RandomAgent Output:
```
Agent_random's final wealth:    $15,160.33 Agent_random
has 513 good products.
```

Conclusion

Thus from the output derived we see that **BernoulliNB** Classifier performs more accurately deriving the most number of Good Products.

The highest wealth however is obtained using the **Logistic Regression** algorithm in our case.

Amongst the baseline agents, RandomAgent performs most poorly and amongst the user defined agents, the clustering algorithm **KMeans** performs most poorly.