My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 StarCastRecommend Namespace Reference

Classes

- class Knap
- class ReadFiles

Functions

- def eval_func (a)

  This function converts a value to a string and returns it.
- def mul_func (a)

  This function converts a value to an integer and multiplies it by 500.
- def make_list (a)
- def div_func (a)
- def recommendation ()

  This is the core function that reads files, takes user input, applies association rules, and uses knapsack to finally output a good star cast for a movie. It uses other classes to achieve this functionality.
- def actorEvaluation ()

Variables

- **app** = QtWidgets.QApplication([ ])
- **dig** = uic.loadUi("projectUI.ui")
- **res** = uic.loadUi("result.ui")
- list **genreList** = ['Adventure','Action','Comedy','Crime','Drama','Family','Fantasy','Thriller','Romance','Horror','Musical']

3.1.1 Detailed Description

```
@file File Documented
```

3.1.2 Function Documentation

3.1.2.1 actorEvaluation()

```
def StarCastRecommend.actorEvaluation ( )
```

```
Function that communicates with the user interface.
```

3.1.2.2 div_func()

```
def StarCastRecommend.div_func (
              a )
```

```
This function divides the number a by 500 and returns it.
```

3.1.2.3 eval_func()

```
def StarCastRecommend.eval_func (
              a )
```

This function converts a value to a string and returns it.

Parameters

a	Value to be converted to string

3.1.2.4 make_list()

```
def StarCastRecommend.make_list (
              a )
```

```
This function creates a list out of the parameter and returns it.
```

3.1.2.5 mul_func()

```
def StarCastRecommend.mul_func (
              a )
```

This function converts a value to an integer and multiplies it by 500.

Parameters

a	Value to be multiplied by 500

Chapter 4

Class Documentation

4.1 StarCastRecommend.Knap Class Reference

Public Member Functions

- def __init__ (self, weights, total, profit)

  Constructor used to initialize variables used everywhere in the function.
- def getItemsUsed (self)

  Once the table of the Knapsack Algorithm is constructed, this function can be used to determine which actors were used to get this table.
- def algorithm (self)

  This function is used to compute the table in the Knapsack Algorithm.

Public Attributes

- weights

  Contains cost of picking each element.
- total

  Contains total weight of bag allowed.
- profit

  Profit associated with picking an element.
- n

  Number of elements to be picked from.
- selected

  A matrix of size N X (W + 1)
- marked

  Boolean List which will indicate which actor is selected after doing 0/1 knapsack.

4.1.1 Detailed Description

```
Class used for Knapsack implementation. \ Consists of two functions used for computing table, and one for eval
```

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
def StarCastRecommend.Knap.__init__ (
              self,
              weights,
              total,
              profit )
```

Constructor used to initialize variables used everywhere in the function.

Parameters

weights	Cost of each actor is stored in this list.
total	This variable is used to denote the total budget.
profit	This list indicates the profit of choosing an actor.

4.1.3 Member Function Documentation

4.1.3.1 algorithm()

```
def StarCastRecommend.Knap.algorithm (
              self )
```

This function is used to compute the table in the Knapsack Algorithm.

This table `self.selected` indicates the best profit for a set of actors.

Returns

List containing the maximum profit, and the set of actors used to get this profit

4.1.3.2 getItemsUsed()

```
def StarCastRecommend.Knap.getItemsUsed (
              self )
```

Once the table of the Knapsack Algorithm is constructed, this function can be used to determine which actors were used to get this table.

Returns

Set of actors (in 0s and 1s) that maximize profit and keep the total cost in the budget as determined by Knapsack.

4.1.4 Member Data Documentation

4.1.4.1 marked

`StarCastRecommend.Knap.marked`

Boolean List which will indicate which actor is selected after doing 0/1 knapsack.

4.1.4.2 total

`StarCastRecommend.Knap.total`

Contains total weight of bag allowed.

4.1.4.3 weights

`StarCastRecommend.Knap.weights`

Contains cost of picking each element.

The documentation for this class was generated from the following file:

- StarCastRecommend.py

4.2 StarCastRecommend.ReadFiles Class Reference

Public Member Functions

- def __init__ (self, apr, top, req)

    Constructor that is used to read the CSV files related to the project and store them into Dataframes.
- def apply (self)
- def support (self, budget, input_genre)

    This function determines the set of supporting actors as determined by the association rules.

Public Attributes

- **apriori**
- **topactors**
- **req**
- **genre_list**
- **rules**
- **supporting_actors**
- **daa**

4.2.1 Detailed Description

```
This class is used to read CSV files related to this module, perform preprocessing if necessary, get associati
```

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
def StarCastRecommend.ReadFiles.__init__ (
              self,
              apr,
              top,
              req )
```

Constructor that is used to read the CSV files related to the project and store them into Dataframes.

Parameters

apr	String which contains the path of the CSV file output after Apriori algorithm is run.
topactors	String which contains the path of the CSV file containing the actors/actresses who have worked most in each Genre.
req	String which contains the path of the CSV file containing relevant information of each actor separately.

4.2.3 Member Function Documentation

4.2.3.1 apply()

```
def StarCastRecommend.ReadFiles.apply (
              self )
```

```
    Function that preprocesses the results from the apriori algorithm. \ It then determines the association ru
```

4.2.3.2 support()

```
def StarCastRecommend.ReadFiles.support (
              self,
              budget,
              input_genre )
```

This function determines the set of supporting actors as determined by the association rules.

\ It uses these association rules to determine the supporting actors for the top actor selected randomly.

Parameters

budget	Total budget of the movie
input_genre	The genre that the user inputs

The documentation for this class was generated from the following file:

- StarCastRecommend.py

Index