app.TransactionDataset

index
d:\projects\summer-project-msc\pythonserver\app\transactiondataset.py

Modules

numpy pandas sklearn

Classes

builtins.object

FrequencyOption
TransactionDataset

enum.Enum(builtins.object)

DistanceMeasure
FrequencyUniqueKey
LinkageMethod

class **DistanceMeasure**(enum.Enum)

```
DistanceMeasure(value, names=None, *, module=None, qualname=None, type=None, start=1)

An enumeration.
```

Method resolution order:
DistanceMeasure
enum.Enum
builtins.object

Data and other attributes defined here:

DAMERAU_LEVENSHTEIN = <DistanceMeasure.DAMERAU_LEVENSHTEIN: 'damerauLevenshtein'>

HAMMING = <DistanceMeasure.HAMMING: 'hamming'>

JARO_SIMILARITY = <DistanceMeasure.JARO_SIMILARITY: 'jaroSimilarity'>

JARO_WINKLER_SIMILARITY = <DistanceMeasure.JARO_WINKLER_SIMILARITY: 'jaroWinklerSimilarity'>

LEVENSHTEIN = <DistanceMeasure.LEVENSHTEIN: 'levenshtein'>

MATCH_RATING_APPROACH = <DistanceMeasure.MATCH_RATING_APPROACH: 'MatchRatingApproach'>

Data descriptors inherited from enum.Enum:

name
```
The name of the Enum member.
```

value
```
The value of the Enum member.
```

Readonly properties inherited from enum.EnumMeta:

__members__
```
Returns a mapping of member name->value.

This mapping lists all enum members, including aliases. Note that this
is a read-only view of the internal mapping.
```

class **FrequencyOption**(builtins.object)

```
FrequencyOption(uniqueKey: app.TransactionDataset.FrequencyUniqueKey, distanceMeasure: Optional[app.TransactionDataset.DistanceMeasure] = None, linkageMethod: Optional[app.TransactionDataset.LinkageMethod] = None, numberOfCluster: Optional[int] = None, per: Literal['month', 'day'] = 'month')

a class wrap the information for the frequency option
```

Methods defined here:

__init__(self, uniqueKey: app.TransactionDataset.FrequencyUniqueKey, distanceMeasure: Optional[app.TransactionDataset.DistanceMeasure] = None, linkageMethod: Optional[app.TransactionDataset.LinkageMethod] = None, numberOfCluster: Optional[int] = None, per: Literal['month', 'day'] = 'month')
```
initialise the frequencyOption object
uniqueKey: category or transactionDescription or clusteredTransactionDescription

if uniqueKey = CLUSTERED_TRANSACTION_DESCRIPTION, the following parameters need to be provided
distanceMeasure: levenshtein or damerauLevenshtein or hamming or jaroSimilarity or jaroWinklerSimilarity or MatchRatingApproach
linkageMethod: single or complete or average or weighted or centroid or median or ward
numberOfCluster: greater than 1
```

getDistanceMeasure(self)

getLinkageMethod(self)

getNumberOfCluster(self)

getPer(self) -> Literal['month', 'day']

getUniqueKey(self)

Data descriptors defined here:

__dict__
dictionary for instance variables (if defined)

__weakref__
list of weak references to the object (if defined)

class **FrequencyUniqueKey**(enum.Enum)

FrequencyUniqueKey(value, names=None, *, module=None, qualname=None, type=None, start=1)

An enumeration.

Method resolution order:
FrequencyUniqueKey
enum.Enum
builtins.object

Data and other attributes defined here:

CATEGORY = <FrequencyUniqueKey.CATEGORY: 'category'>

CLUSTERED_TRANSACTION_DESCRIPTION = <FrequencyUniqueKey.CLUSTERED_TRANSACTION_DESCRIPTION: 'clusteredTransactionDescription'>

TRANSACTION_DESCRIPTION = <FrequencyUniqueKey.TRANSACTION_DESCRIPTION: 'transactionDescription'>

Data descriptors inherited from enum.Enum:

name
The name of the Enum member.

value
The value of the Enum member.

Readonly properties inherited from enum.EnumMeta:

__members__
Returns a mapping of member name->value.

This mapping lists all enum members, including aliases. Note that this
is a read-only view of the internal mapping.

class **LinkageMethod**(enum.Enum)

LinkageMethod(value, names=None, *, module=None, qualname=None, type=None, start=1)

An enumeration.

Method resolution order:
LinkageMethod
enum.Enum
builtins.object

Data and other attributes defined here:

AVERAGE = <LinkageMethod.AVERAGE: 'average'>

CENTROID = <LinkageMethod.CENTROID: 'centroid'>

COMPLETE = <LinkageMethod.COMPLETE: 'complete'>

MEDIAN = <LinkageMethod.MEDIAN: 'median'>

SINGLE = <LinkageMethod.SINGLE: 'single'>

WARD = <LinkageMethod.WARD: 'ward'>

WEIGHTED = <LinkageMethod.WEIGHTED: 'weighted'>

Data descriptors inherited from enum.Enum:

name
The name of the Enum member.

value
The value of the Enum member.

Readonly properties inherited from enum.EnumMeta:

__members__
Returns a mapping of member name->value.

This mapping lists all enum members, including aliases. Note that this
is a read-only view of the internal mapping.

class **TransactionDataset**(builtins.object)

TransactionDataset(csvPath: str)

a dataset with the following columns: transactionNumber (str), transactionDate (datetime),
transactionType (str), transactionDescription(str), balance (float), category (str), locationCity (str), locationCountry (str),
isCredit (boolean), transactionAmount (float), frequency(float), frequencyUniqueKey()

Methods defined here:

__init__(self, csvPath: str)
read transactions from csv file, the data will be initialised

clusterByKMeans(self, metric1, metric2, numberOfCluster, maxIteration: int = 300, nInit=10)
assume metric1 and metric2 exist in the column names.
run KMean clustering algorithm based on the two metrics
if any metric is not int or float, they will be try to convert to float.
set the clusterId column to be the result of clustering algorithm
this method will mutate self.**dataframe**
maxIteration should >VALID_KMEAN_ITERATION[0] and <VALID_KMEAN_ITERATION[1]

getClusterIdOfTransactionNumber(self) -> dict
if the cluster algorithm has runned return a dictionary where the key is transactionNumber and the value is the cluster.

getColumn(self, columnName, toNumerical=False) -> pandas.core.series.Series
assume the columnName exist
if the column is categorical and toNumerical==True, return a list of number represents the category of the columnName,
otherwise return a list of value of the columnName

getColumnNames(self) -> list
Return a list of valid columnNames

getDataframe(self) -> pandas.core.frame.DataFrame
return the internal dataframe for the transaction with the following columns: transactionNumber (str), transactionDate (datetime),
transactionType (str), transactionDescription(str), balance (float), category (str), locationCity (str), locationCountry (str),
isCredit (boolean), transactionAmount (float)

isValidColumnName(self, columnNameToCheck: str) -> bool
columnNameToCheck: the column name to check
return True if the columnName is valid
return False if not

setFrequencyOption(self, newFrequencyOption)
Set the frequency option, update the frequency column, frequency Unique key Column. if frequency is based on clustering, algorithm will be run

---

Data descriptors defined here:

__dict__
dictionary for instance variables (if defined)

__weakref__
list of weak references to the object (if defined)

Data

Literal = typing.Literal
Union = typing.Union
VALID_KMEAN_ITERATION = [1, 2000]
VALID_KMEAN_N_INIT = [10, 1000]