API Documentation

API Documentation

January 21, 2018

Contents

Contents **1**

1 Package src **5**
- 1.1 Modules . . . 5
- 1.2 Functions . . . 6
- 1.3 Variables . . . 6

2 Module src.__main__ **7**
- 2.1 Variables . . . 7

3 Module src.b_exceptions **8**
- 3.1 Variables . . . 8
- 3.2 Class BadFormatException . . . 8
  - 3.2.1 Methods . . . 8
  - 3.2.2 Properties . . . 8
- 3.3 Class BadValueException . . . 9
  - 3.3.1 Methods . . . 9
  - 3.3.2 Properties . . . 9
- 3.4 Class BenzlimException . . . 10
  - 3.4.1 Methods . . . 10
  - 3.4.2 Properties . . . 10
- 3.5 Class PriceNotFoundException . . . 11
  - 3.5.1 Methods . . . 11
  - 3.5.2 Properties . . . 11
- 3.6 Class StationNotFoundException . . . 12
  - 3.6.1 Methods . . . 12
  - 3.6.2 Properties . . . 12
- 3.7 Class TrainingDataMissingException . . . 13
  - 3.7.1 Methods . . . 13
  - 3.7.2 Properties . . . 13

4 Module src.benchmark **14**
- 4.1 Functions . . . 14
- 4.2 Variables . . . 14

5 Package src.compat **15**
- 5.1 Modules . . . 15
- 5.2 Variables . . . 15

6 Module src.compat.py2 **16**
6.1 Functions . . . 16
6.2 Variables . . . 16

7 Module src.compat.py3 **17**
7.1 Functions . . . 17

8 Module src.config **18**
8.1 Variables . . . 18
8.2 Class Configuration . . . 18
8.2.1 Methods . . . 18
8.2.2 Class Variables . . . 18

9 Package src.coverage' **19**
9.1 Functions . . . 19
9.2 Variables . . . 19

10 Package src.dao **20**
10.1 Modules . . . 20
10.2 Class CSVDAO . . . 20
10.2.1 Methods . . . 20
10.2.2 Properties . . . 21
10.3 Class StationDAO . . . 22
10.3.1 Methods . . . 22
10.3.2 Properties . . . 23
10.3.3 Class Variables . . . 23
10.4 Class DBManager . . . 23
10.4.1 Methods . . . 24
10.4.2 Properties . . . 24
10.4.3 Class Variables . . . 25

11 Module src.dao.csv_ **26**
11.1 Variables . . . 26
11.2 Class CSVDAO . . . 26
11.2.1 Methods . . . 26
11.2.2 Properties . . . 27

12 Module src.dao.db **28**
12.1 Class DBManager . . . 28
12.1.1 Methods . . . 28
12.1.2 Properties . . . 29
12.1.3 Class Variables . . . 29
12.2 Class StationDAO . . . 29
12.2.1 Methods . . . 30
12.2.2 Properties . . . 30
12.2.3 Class Variables . . . 31

13 Module src.exceptions_ **32**
13.1 Variables . . . 32
13.2 Class BenzlimException . . . 32
13.2.1 Methods . . . 32
13.2.2 Properties . . . 33
13.3 Class StationNotFoundException . . . 33

13.3.1 Methods . . . 33
13.3.2 Properties . . . 34
13.4 Class PriceNotFoundException . . . 34
13.4.1 Methods . . . 34
13.4.2 Properties . . . 35
13.5 Class TrainingDataMissingException . . . 35
13.5.1 Methods . . . 35
13.5.2 Properties . . . 36
13.6 Class BadFormatException . . . 36
13.6.1 Methods . . . 36
13.6.2 Properties . . . 37
13.7 Class BadValueException . . . 37
13.7.1 Methods . . . 37
13.7.2 Properties . . . 38

14 Package src.prediction 39
14.1 Modules . . . 39
14.2 Functions . . . 39
14.3 Variables . . . 39

15 Module src.prediction.classification 40
15.1 Variables . . . 40
15.2 Class CSClassifier . . . 40
15.2.1 Methods . . . 40
15.2.2 Properties . . . 41
15.3 Class Classifier . . . 41
15.3.1 Methods . . . 41
15.3.2 Properties . . . 42

16 Module src.prediction.db 43
16.1 Functions . . . 43
16.2 Variables . . . 43
16.3 Class DBManager . . . 43
16.3.1 Methods . . . 43
16.3.2 Properties . . . 44
16.3.3 Class Variables . . . 44
16.4 Class PriceDAO . . . 45
16.4.1 Methods . . . 45
16.4.2 Class Variables . . . 45
16.5 Class StationDAO . . . 45
16.5.1 Methods . . . 45
16.5.2 Class Variables . . . 46

17 Module src.prediction.predict 47
17.1 Functions . . . 47
17.2 Variables . . . 48

18 Package src.routing 49
18.1 Modules . . . 49
18.2 Functions . . . 49
18.3 Variables . . . 49

19 Module src.routing.graph **50**
19.1 Variables 50
19.2 Class Graph 50
19.2.1 Methods 50

20 Module src.routing.node **51**
20.1 Variables 51
20.2 Class Node 51
20.2.1 Methods 51
20.2.2 Properties 52

21 Package src.tests **53**
21.1 Functions 53
21.2 Variables 53

22 Module src.train **54**
22.1 Variables 54
22.2 Class Trainer 54
22.2.1 Methods 54
22.2.2 Properties 54

23 Module src.utils **55**
23.1 Functions 55
23.2 Variables 55

Index **56**

1 Package src

Benzlim

1.1 Modules

- **__main__** *(Section 2, p. 7)*
- **b_exceptions** *(Section 3, p. 8)*
- **benchmark**: benchmark.py - Benchmarking tool
  (Section 4, p. 14)
- **compat**: compat - Compatibility packages for ython2 and python3
  (Section 5, p. 15)
  - **py2**: py2 - Python2 compatibility module
    (Section 6, p. 16)
  - **py3**: py3 - Python3 compatibility module
    (Section 7, p. 17)
- **config**: config.py - access benzlim's instance configuration
  (Section 8, p. 18)
- **coverage** *(Section ??, p. ??)*
- **coverage'**: coverage - Coverage informations generation about benzlim
  (Section 9, p. 19)
- **dao**: dao - Data Access Object packages for IO tasks
  (Section 10, p. 20)
  - **csv_**: csv_.py - read/write/investigate csv related files
    (Section 11, p. 26)
  - **db**: db.py - access station related informations
    (Section 12, p. 28)
- **exceptions_**: exceptions_.py - benzlim exceptions
  (Section 13, p. 32)
- **prediction**: prediction - The benzlim core prediction
  (Section 14, p. 39)
  - **classification**: classfication.py - gas stations classfication tools
    (Section 15, p. 40)
  - **db** *(Section 16, p. 43)*
  - **predict**: predict.py - core prediction tools
    (Section 17, p. 47)
- **routing**: routing - gas tank strategy manager
  (Section 18, p. 49)
  - **graph**: graph.y - Tank strategy optimizer for graph based routes
    (Section 19, p. 50)
  - **node**: node.py - Nodes for graph based representation of gas stations in a route
    (Section 20, p. 51)
- **tests**: tests - Tests runner
  (Section 21, p. 53)
- **train**: train.py - manage the whole training
  (Section 22, p. 54)
- **utils**: utils.py - usefool tools
  (Section 23, p. 55)

1.2 Functions

main()

1.3 Variables

Name	Description
__package__	**Value:** `'src'`
lvl	**Value:** `0`

2 Module src.__main__

2.1 Variables

Name	**Description**
__package__	**Value:** `None`

3 Module src.b_exceptions

3.1 Variables

Name	Description
__package__	**Value:** `None`

3.2 Class BadFormatException

object
exceptions.BaseException
exceptions.Exception
src.b_exceptions.BenzlimException
src.b_exceptions.BadFormatException

3.2.1 Methods

__init__(*self*, **args*, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

3.2.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object	

continued on next page

Name	Description
__class__	

3.3 Class BadValueException

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
src.b_exceptions.BenzlimException ─┐
                                   src.b_exceptions.BadValueException
```

3.3.1 Methods

__init__(*self*, **args*, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

3.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

3.4 Class BenzlimException

```
object ─┐
        │
exceptions.BaseException ─┐
                          │
        exceptions.Exception ─┐
                              │
                              src.b_exceptions.BenzlimException
```

Known Subclasses: src.b_exceptions.BadFormatException, src.b_exceptions.BadValueException, src.b_exceptions.PriceNotFoundException, src.b_exceptions.StationNotFoundException, src.b_exceptions.T

3.4.1 Methods

__init__(*self*, **args*, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

3.4.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

3.5 Class PriceNotFoundException

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
 src.b_exceptions.BenzlimException ─┐
                                     src.b_exceptions.PriceNotFoundException
```

3.5.1 Methods

__init__(*self*, **args*, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

3.5.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

3.6 Class StationNotFoundException

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
src.b_exceptions.BenzlimException ─┐
                                  src.b_exceptions.StationNotFoundException
```

3.6.1 Methods

__init__(*self*, **args*, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

3.6.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

3.7 Class TrainingDataMissingException

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
src.b_exceptions.BenzlimException ─┐
                                   src.b_exceptions.TrainingDataMissingException
```

3.7.1 Methods

__init__(*self*, **args*, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

3.7.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

4 Module src.benchmark

benchmark.py - Benchmarking tool

4.1 Functions

evaluate_prediction(*station_id*, *ts*, *ground_ts*, *end_train_timestamp*, *dir_prices*, *nb_predictions*)

process_benchmark_prediction(*args*)

benchmark_with_prices(*nb_stations*, *nb_predictions*, *dir_prices*)

benchmark_without_prices(*nb_stations*, *nb_predictions*, *dir_prices*)

benchmark_predictions(*nb_stations*, *nb_predictions*, *dir_prices*)

benchmark_routing(*nb_stations*, *nb_predictions*, *dir_prices*)

process_benchmark(*dir_prices*, *nb_stations*=`1`, *nb_predictions*=`5`)

4.2 Variables

Name	Description
__package__	**Value:** `'src'`

5 Package src.compat

compat - Compatibility packages for ython2 and python3

5.1 Modules

- **py2**: py2 - Python2 compatibility module
  (Section 6, p. 16)
- **py3**: py3 - Python3 compatibility module
  (Section 7, p. 17)

5.2 Variables

Name	Description
__package__	**Value:** `'src.compat'`

6 Module src.compat.py2

py2 - Python2 compatibility module

6.1 Functions

printf(**args*, ***kwargs*)

str2unicode(*value*)

convert a str to unicode

6.2 Variables

Name	Description
__package__	**Value:** `'src.compat'`

7 Module src.compat.py3

py3 - Python3 compatibility module

7.1 Functions

printf(**args*, ***kwargs*)

str2unicode(*value*)

convert a str to unicode

8 Module src.config

config.py - access benzlim's instance configuration

8.1 Variables

Name	Description
__package__	**Value:** `'src'`

8.2 Class Configuration

Contains the configuration to run a benzlim instance

8.2.1 Methods

__init__(*self*, ***kwargs*)

get_instance(***kwargs*)

config(***kwargs*)

get_pool(*self*)

8.2.2 Class Variables

Name	Description
RESOURCE_DIR	**Value:** `'resources'`
OUTPUT_DIR	**Value:** `'out'`
CLASSIFIER_FILENAM-E	**Value:** `'classifier.pkl'`
DATABASE_FILENAME	**Value:** `'db.sqlite3'`
TIME_BINS	**Value:** `['00:00', '01:00', '02:00', '03:00', '04:00', '05:00', '0...`
h	**Value:** 23

9 Package src.coverage'

coverage - Coverage informations generation about benzlim

9.1 Functions

clean_benzlim()

clean benzim files, database file classifier file and coverage files

clean_mp_coverages(*empty_only*=`True`)

clean coverage files generated by multiprocessing runs

coverage()

Run coverage

execute_coverage()

Run coverage for the whole project

9.2 Variables

Name	Description
__package__	**Value:** `'src.coverage'`

10 Package src.dao

dao - Data Access Object packages for IO tasks

10.1 Modules

- **csv_**: csv_.py - read/write/investigate csv related files
  (Section 11, p. 26)
- **db**: db.py - access station related informations
  (Section 12, p. 28)

10.2 Class CSVDAO

```
object ─┐
        │
    src.dao.csv_.CSVDAO
```

10.2.1 Methods

get_station_filename(*cls*, *station_id*, *prices_dir*=`None`)

return the filename containing prices for the station <station_id>

is_prices_available(*cls*, *station_id*)

return True if prices are avaiable for the given station else False

get_station_dataframe(*cls*, *station_id*, *dir_prices*)

return a DataFrame containing timestamps and prices of the station <station_id>

get_all_extended_stations_infos(*cls*)

return station informations: id: int => Station id name: str => Station name mark: str => Mark name street: str => streetname street-number: int => house number/ street number zipcode: int => zipcode town: str latitude: float longitude: float prices_available: bool => if prices are available begin_timestamp: str => the first price timestamp

get_all_stations_infos(*cls*)

return station informations: id: int => Station id name: str => Station name mark: str => Mark name street: str => streetname street-number: int => house number/ street number zipcode: int => zipcode town: str latitude: float longitude: float

get_predict_params(*cls*, *filename*)

return [<end_timestamp>, <prediction_timestamp>, <station_id>]

get_route_params(*cls*, *filename*)

return <capacity>, [<timestamp>, <station_id>]

get_route_prices_params(*cls*, *filename*)

return [<timestamp>, <station_id>, <pred_price>]

get_route_as_predict_params(*cls*, *filename*)

get_predicted_prices(*cls*, *filename*)

return [<end_timestamp>, <prediction_timestamp>, <station_id>, <pred_price>]

export_to_csv(*cls*, *filename*, *rows*, *header*=`None`)

rows in the file <filename> as csv

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

10.2.2 Properties

Name	Description
Inherited from object	
__class__	

10.3 Class StationDAO

object ─┐
src.dao.db.StationDAO

Data Access object manager for gas stations

10.3.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

get_all(*cls*)

return all stations from the database

get(*cls*, *pk*)

return the station with the id <pk>

get_all_before(*cls*, *timestamp*)

Return all stations with prices available before <timestamp>

get_latitude_longitude(*cls*, *pk*)

Return the latitude and longitude of the station with id <pk>

is_prices_available(*cls*, *pk*)

Return True if the station <pk> has prices else False

get_all_with_prices(*cls*)

Return all stations with prices available

get_all_without_prices(*cls*)

Return all stations without prices available

populate(*cls*, *data*)

Populate the corresponding table with items in data.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

10.3.2 Properties

Name	Description
Inherited from object	
__class__	

10.3.3 Class Variables

Name	Description
table	**Value:** `'stations'`
schema	**Value:** `'\nCREATE TABLE IF NOT EXISTS stations(\n id INTEGER P...`
indexes	**Value:** `('CREATE INDEX IF NOT EXISTS prices_index on stations(pri...`
select_all_before_sql	**Value:** `'select * from stations where datetime(begin_timestamp||\...`
select_all_query_sql	**Value:** `'select * from stations'`
select_query_sql	**Value:** `'select * from stations where id=?'`
select_all_prices_available_s-ql	**Value:** `'select * from stations where prices_available'`
select_all_prices_missing_sq-l	**Value:** `'select * from stations where not (prices_available)'`
select_prices_is_available_s-ql	**Value:** `'select prices_available from stations where id=?'`
select_latitude_longitude	**Value:** `'select latitude, longitude from stations where id=?'`
insert_station_sql	**Value:** `'insert into stations (id, name, mark, street, street_num...`

10.4 Class DBManager

```
object ─┐
        │
        src.dao.db.DBManager
```

Base manager for interactions with the database

10.4.1 Methods

open(*cls*)

open the database

get_conn(*cls*)

get the connection from the database

close(*cls*)

close the database

init_db(*cls*)

Init the database if it doesn't exist yet.

force_init_db(*cls*)

Force the initialisation of the database and overwrite it.

execute(*cls*, *sql*, *data*=`None`)

execute the query <sql> with <data> and return the result

executemany(*cls*, *sql*, *data*=`None`)

execute the query <sql> which each value in data and return the result

populate_db(*cls*, *data*, *sql_query*=`None`)

Populate the database with items in data.

set_auto_commit(*cls*, *value*=`True`)

enable/disable auto_commits to speed-up batch queries

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

10.4.2 Properties

Name	Description
Inherited from object	

continued on next page

Name	Description
__class__	

10.4.3 Class Variables

Name	Description
sql_schemas	**Value:** `('\nCREATE TABLE IF NOT EXISTS stations(\n id INTEGER ...`
sql_indexes	**Value:** `('CREATE INDEX IF NOT EXISTS prices_index on stations(pri...`
conn	**Value:** `None`
table	**Value:** `'stations'`
table_stations	**Value:** `'stations'`
sql_insert_station_sql	**Value:** `'insert into stations (id, name, mark, street, street_num...`
sql_get	**Value:** `''`
sql_update	**Value:** `''`
sql_delete	**Value:** `''`
sql_save	**Value:** `''`

11 Module src.dao.csv_

csv_.py - read/write/investigate csv related files

11.1 Variables

Name	Description
__package__	**Value:** `'src.dao'`

11.2 Class CSVDAO

```
object ─┐
        │
        src.dao.csv_.CSVDAO
```

11.2.1 Methods

get_station_filename(*cls*, *station_id*, *prices_dir*=`None`)

return the filename containing prices for the station <station_id>

is_prices_available(*cls*, *station_id*)

return True if prices are avaiable for the given station else False

get_station_dataframe(*cls*, *station_id*, *dir_prices*)

return a DataFrame containing timestamps and prices of the station <station_id>

get_all_extended_stations_infos(*cls*)

return station informations: id: int => Station id name: str => Station name mark: str => Mark name street: str => streetname street-number: int => house number/ street number zipcode: int => zipcode town: str latitude: float longitude: float prices_available: bool => if prices are available begin_timestamp: str => the first price timestamp

get_all_stations_infos(*cls*)

return station informations: id: int => Station id name: str => Station name mark: str => Mark name street: str => streetname street-number: int => house number/ street number zipcode: int => zipcode town: str latitude: float longitude: float

get_predict_params(*cls*, *filename*)

return [<end_timestamp>, <prediction_timestamp>, <station_id>]

get_route_params(*cls*, *filename*)

return <capacity>, [<timestamp>, <station_id>]

get_route_prices_params(*cls*, *filename*)

return [<timestamp>, <station_id>, <pred_price>]

get_route_as_predict_params(*cls*, *filename*)

get_predicted_prices(*cls*, *filename*)

return [<end_timestamp>, <prediction_timestamp>, <station_id>, <pred_price>]

export_to_csv(*cls*, *filename*, *rows*, *header*=`None`)

rows in the file <filename> as csv

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

11.2.2 Properties

Name	Description
Inherited from object	
__class__	

12 Module src.dao.db

db.py - access station related informations

12.1 Class DBManager

```
object ─┐
        src.dao.db.DBManager
```

Base manager for interactions with the database

12.1.1 Methods

open(*cls*)

open the database

get_conn(*cls*)

get the connection from the database

close(*cls*)

close the database

init_db(*cls*)

Init the database if it doesn't exist yet.

force_init_db(*cls*)

Force the initialisation of the database and overwrite it.

execute(*cls*, *sql*, *data*=`None`)

execute the query <sql> with <data> and return the result

executemany(*cls*, *sql*, *data*=`None`)

execute the query <sql> which each value in data and return the result

populate_db(*cls*, *data*, *sql_query*=`None`)

Populate the database with items in data.

set_auto_commit(*cls*, *value*=`True`)
enable/disable auto_commits to speed-up batch queries

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

12.1.2 Properties

Name	Description
Inherited from object	
__class__	

12.1.3 Class Variables

Name	Description
sql_schemas	**Value:** `('\nCREATE TABLE IF NOT EXISTS stations(\n id INTEGER ...`
sql_indexes	**Value:** `('CREATE INDEX IF NOT EXISTS prices_index on stations(pri...`
conn	**Value:** `None`
table	**Value:** `'stations'`
table_stations	**Value:** `'stations'`
sql_insert_station_sql	**Value:** `'insert into stations (id, name, mark, street, street_num...`
sql_get	**Value:** `''`
sql_update	**Value:** `''`
sql_delete	**Value:** `''`
sql_save	**Value:** `''`

12.2 Class StationDAO

```
object ─┐
        │
        src.dao.db.StationDAO
```

Data Access object manager for gas stations

12.2.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

get_all(*cls*)

return all stations from the database

get(*cls*, *pk*)

return the station with the id <pk>

get_all_before(*cls*, *timestamp*)

Return all stations with prices available before <timestamp>

get_latitude_longitude(*cls*, *pk*)

Return the latitude and longitude of the station with id <pk>

is_prices_available(*cls*, *pk*)

Return True if the station <pk> has prices else False

get_all_with_prices(*cls*)

Return all stations with prices available

get_all_without_prices(*cls*)

Return all stations without prices available

populate(*cls*, *data*)

Populate the corresponding table with items in data.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

12.2.2 Properties

Name	Description
Inherited from object __class__	

12.2.3 Class Variables

Name	Description
table	**Value:** `'stations'`
schema	**Value:** `'\nCREATE TABLE IF NOT EXISTS stations(\n id INTEGER P...`
indexes	**Value:** `('CREATE INDEX IF NOT EXISTS prices_index on stations(pri...`
select_all_before_sql	**Value:** `'select * from stations where datetime(begin_timestamp||\...`
select_all_query_sql	**Value:** `'select * from stations'`
select_query_sql	**Value:** `'select * from stations where id=?'`
select_all_prices_available_s-ql	**Value:** `'select * from stations where prices_available'`
select_all_prices_missing_sq-l	**Value:** `'select * from stations where not (prices_available)'`
select_prices_is_available_s-ql	**Value:** `'select prices_available from stations where id=?'`
select_latitude_longitude	**Value:** `'select latitude, longitude from stations where id=?'`
insert_station_sql	**Value:** `'insert into stations (id, name, mark, street, street_num...`

13 Module src.exceptions_

exceptions_.py - benzlim exceptions

13.1 Variables

Name	Description
__package__	**Value:** `None`

13.2 Class BenzlimException

```
 object ─┐
         │
 exceptions.BaseException ─┐
                           │
          exceptions.Exception ─┐
                                │
                   src.exceptions_.BenzlimException
```

Known Subclasses: src.exceptions_.BadFormatException, src.exceptions_.BadValueException, src.exceptions_.PriceNotFoundException, src.exceptions_.StationNotFoundException, src.exceptions_.Train

13.2.1 Methods

__init__(*self*, *message*, **args*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

13.2.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

13.3 Class StationNotFoundException

```
object ─┐
        │
exceptions.BaseException ─┐
                          │
          exceptions.Exception ─┐
                                │
  src.exceptions_.BenzlimException ─┐
                                    │
                                    src.exceptions_.StationNotFoundException
```

13.3.1 Methods

__init__(*self*, *message*, **args*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from src.exceptions_.BenzlimException(Section 13.2)

__str__()

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

13.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

13.4 Class PriceNotFoundException

```
object ─┐
exceptions.BaseException ─┐
            exceptions.Exception ─┐
  src.exceptions_.BenzlimException ─┐
                              src.exceptions_.PriceNotFoundException
```

13.4.1 Methods

__init__(*self*, *message*, **args*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from src.exceptions_.BenzlimException(Section 13.2)

__str__()

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

13.4.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

13.5 Class TrainingDataMissingException

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
  src.exceptions_.BenzlimException ─┐
                                     src.exceptions_.TrainingDataMissingException
```

13.5.1 Methods

__init__(*self*, *message*, **args*)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

Inherited from src.exceptions_.BenzlimException(Section 13.2)

__str__()

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

13.5.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

13.6 Class BadFormatException

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
  src.exceptions_.BenzlimException ─┐
                                     src.exceptions_.BadFormatException
```

13.6.1 Methods

__init__(*self*, *message*, **args*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from src.exceptions_.BenzlimException(Section 13.2)

__str__()

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

13.6.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

13.7 Class BadValueException

```
object ─┐
        │
exceptions.BaseException ─┐
                          │
          exceptions.Exception ─┐
                                │
  src.exceptions_.BenzlimException ─┐
                                    │
                                    src.exceptions_.BadValueException
```

13.7.1 Methods

__init__(*self*, *message*, **args*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from src.exceptions_.BenzlimException(Section 13.2)

__str__()

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

13.7.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	
Inherited from object	
__class__	

14 Package src.prediction

prediction - The benzlim core prediction

14.1 Modules

- **classification**: classfication.py - gas stations classfication tools
  (Section 15, p. 40)
- **db** *(Section 16, p. 43)*
- **predict**: predict.py - core prediction tools
  (Section 17, p. 47)

14.2 Functions

process_task(*args*)

predict_prices_timestamps_x2_stations(*timestamps_x2_stations*, *dir_prices*, *nb_workers*=`None`)

return [<end_timestamp>, <timestamp>, <station_id>, <pred_price>],
timestamps_x2_stations: list[<end_timestamp>, <timestamp>, <station_id>]
dir_prices: directory path

process_predictions(*filename*, *dir_prices*, *out_filename*=`None`, *nb_workers*=`None`)

process_routing(*filename*, *dir_prices*, *out_filename*=`None`, *gas_prices_file*=`None`, *nb_workers*=`None`, *auto_end_timestamp*=`True`)

14.3 Variables

Name	Description
__package__	**Value:** `'src.prediction'`

15 Module src.prediction.classification

classfication.py - gas stations classfication tools

15.1 Variables

Name	Description
LATITUDE_MAX	**Value:** `90.0`
LONGITUDE_MAX	**Value:** `180.0`
HASH_MAX	**Value:** `982451653`
NB_CHARS	**Value:** `45`
__package__	**Value:** `'src.prediction'`

15.2 Class CSClassifier

object ⌐
src.prediction.classification.CSClassifier

15.2.1 Methods

__init__(*self*, *scoring_function*=`None`, *partition_index*=`0`)

Basic classifier, The classification is done by using partitioning on partition_index and Near Neighbour selection scoring_function: function, the scoring fuction for the NNS partition_index: int, the index to use for partitioning

Overrides: object.__init__

fit(*self*, *x_values*, *labels*)

Train the classifier

predict(*self*, *x*, *try_skip_id*=`None`)

predict class for features x based on the training data

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

15.2.2 Properties

Name	Description
Inherited from object	
__class__	

15.3 Class Classifier

object ┐
src.prediction.classification.Classifier

Main classifier of gas stations

15.3.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

get_category(*cls*, *station_row*)

Return a category for the given station

station_id2id(*cls*, *station_id*, *end_train_timestamp*=`None`, *ignore_station*=`False`)

Return a usable station id

station_row2id(*cls*, *station_row*, *end_train_timestamp*=`None`, *ignore_station*=`False`)

Return a usable (with prices available) id

get_station_features(*cls*, *station_row*)

Return features for the given station

get_prepared_data(*cls*, *ext_stations*=`None`)

return features with corresponding classes

train(*cls*, *features*=`None`, *classes*=`None`)

train the classifier using

dump(*cls*, *classifier*, *filename*=`None`)

save a classifier to the file <filename>

load(*cls*, *filename*=`None`, *create_on_error*=`True`)

Load a classifier from the file <filename>

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

15.3.2 Properties

Name	Description
Inherited from object __class__	

16 Module src.prediction.db

16.1 Functions

icompare(*text1*, *text2*)

16.2 Variables

Name	Description
DB_PATH	**Value:** `'resources/db/db.sqlite3'`
DB_SQL_INDEX_MARK	**Value:** `'CREATE INDEX IF NOT EXISTS mark_index on stations(mark C...`
DB_SQL_INDEX_NAME	**Value:** `'CREATE INDEX IF NOT EXISTS word_index on stations(name C...`
DB_SQL_INDEX_PLACE	**Value:** `'CREATE INDEX IF NOT EXISTS place_index on stations(place...`
DB_SQL_INDEX_STATI-ON	**Value:** `'CREATE INDEX IF NOT EXISTS station_index on prices(stati...`
DB_SQL_SCHEMA_PRIC-ES	**Value:** `'CREATE TABLE IF NOT EXISTS prices(\n id INTEGER PRIMA...`
DB_SQL_SCHEMA_STAT-IONS	**Value:** `'\nCREATE TABLE IF NOT EXISTS stations(\n id INTEGER P...`
__package__	**Value:** `'src.prediction'`

16.3 Class DBManager

object
└ **src.prediction.db.DBManager**

16.3.1 Methods

close(*cls*)

execute(*cls*, *sql*, *data*=`None`)

executemany(*cls*, *sql*, *data*=`None`)

force_init_db(*cls*)

Force the initialisation of the database and overwrite it.

getConn(*cls*)

init_db(*cls*)

Init the database if it doesn't exist yet.

open(*cls*)

populate_db(*cls*, *data*, *sql_query*=`None`)

Populate the database with items in data.

set_auto_commit(*cls*, *value*=`True`)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

16.3.2 Properties

Name	Description
Inherited from object __class__	

16.3.3 Class Variables

Name	Description
conn	**Value:** `None`
filename	**Value:** `'resources/db/db.sqlite3'`
sql_delete	**Value:** `''`
sql_get	**Value:** `''`
sql_indexes	**Value:** `('CREATE INDEX IF NOT EXISTS mark_index on stations(mark ...`
sql_insert_price	**Value:** `'insert into Prices (id, station_id, timestamp, price) va...`
sql_insert_station	**Value:** `'insert into Stations (id, name, mark, street, street_num...`
sql_save	**Value:** `''`

continued on next page

Name	Description
sql_schemas	**Value:** `('\nCREATE TABLE IF NOT EXISTS stations(\n id INTEGER ...`
sql_update	**Value:** `''`
table	**Value:** `'Stations'`
table_prices	**Value:** `'Prices'`
table_stations	**Value:** `'Stations'`

16.4 Class PriceDAO

16.4.1 Methods

get(*cls*, *pk*=`None`)

getAll(*cls*)

16.4.2 Class Variables

Name	Description
select_all_query	**Value:** `'select * from prices'`
select_query	**Value:** `'select * from prices where id=?'`
table	**Value:** `'prices'`

16.5 Class StationDAO

16.5.1 Methods

get(*cls*, *pk*)

getAll(*cls*)

get_all_with_prices(*cls*)

get_all_without_prices(*cls*)

is_prices_missing(*cls*, *pk*)

16.5.2 Class Variables

Name	Description
select_all_prices_available	**Value:** `'select * from stations where prices_available=1'`
select_all_prices_missing	**Value:** `'select * from stations where prices_available=0'`
select_all_query	**Value:** `'select * from stations'`
select_prices_is_available	**Value:** `'select * from stations where id=?'`
select_query	**Value:** `'select * from stations where id=?'`
table	**Value:** `'stations'`

17 Module src.prediction.predict

predict.py - core prediction tools

17.1 Functions

get_time_range(*timestamp*)

return beginning and ending range of a given timestamp

get_freq_avg(*ts*, *freq*=`'10T'`, *fill_method*=`'pad'`, *fill_method2*=`None`)

resample the timeserie with the new frequence <freq> using the fill methods for NANs

get_time(*timestamp*, *field*=`None`)

return the corresponding value of the attribut corresponding to <field> timestamp: <pd.Timestamp> field: <str> Y, M, W, D, H, T

get_price_predictor(*station_id*, *dir_prices*, *ts*=`None`, *time_begin*=`None`, *time_end*=`None`, *end_train_timestamp*=`None`, *poly_deg*=2)

Generate a price predictor for gas station <station_id> of the timeserie <ts>, station_id: str, the id of the station ts: DataFrame|Serie, the price's timeserie of as gas station time_begin: str, time_end: str, end_train_timestamp: str, the last usable timestamp for learning, poly_deg: int, the degree of polynomial approximation [1,2,3,4,5] return the callable prediction(timestamp)

if station_id is submitted, the predictor is cached resp. recovered from the cache if available

If the difference between the predicted value and the average is bigger than 20% of the average, the predictor will return the average instead of the predicted value

get_price_predictor2(*station_id*, *dir_prices*, *ts*=`None`, *time_begin*=`None`, *time_end*=`None`, *end_train_timestamp*=`None`, *poly_deg*=`2`)

Generate a price predictor for gas station <station_id> of the timeserie <ts>, station_id: str, the id of the station ts: DataFrame|Series, the price's timeserie of as gas station end_train_timestamp: str, the last usable timestamp for learning, poly_deg: int, the degree of polynomial approximation return the predictor as a numpy.poly1d

if station_id is submitted, the predictor is cached resp. recovered from the cache

If the difference between the predicted value and the average is bigger than 20% of the average, the predictor will return the average instead of the predicted value

predict_price(*station_id*, *timestamp*, *end_train_timestamp*, *dir_prices*, *bench_ts*=`None`)

predict the price of a given station a at given time, station_id: int, the id of the station timestamp: str, the moment at which the price is needed end_train_timestamp: str, the last date at which data should be used for training dir_prices: str, path to the directory containing prices bench_ts: DataFrame, data frame for testing in benchmark mode

17.2 Variables

Name	Description
MAX_MARGIN_COEF	**Value:** `0.2`
CACHE_PREDICTORS	**Value:** {}
__package__	**Value:** `'src.prediction'`

18 Package src.routing

routing - gas tank strategy manager

18.1 Modules

- **graph**: graph.y - Tank strategy optimizer for graph based routes
  (Section 19, p. 50)
- **node**: node.py - Nodes for graph based representation of gas stations in a route
  (Section 20, p. 51)

18.2 Functions

generate_tank_infos(*capacity*, *timestamps_stations_prices*)

generate routing informations, capacity: int, the tank capacity timestamps_stations_prices: lst<str, int, int>, the predicted price informations return routing informations according to the Intellitank format

18.3 Variables

Name	Description
__package__	**Value:** `'src.routing'`

19 Module src.routing.graph

graph.y - Tank strategy optimizer for graph based routes

19.1 Variables

Name	Description
__package__	**Value:** `'src.routing'`

19.2 Class Graph

19.2.1 Methods

__init__(*self*, *capacity*)

Graph for a graph based represention of the routes, capacity: vehicle capacity

gas_for_km(*self*, *km*)

give the amount of gas needed for the given distance <km>

km_for_gas(*self*, *gas*)

give the amount of km one can travel for the given <gas>

find_prevs(*self*)

find cheapest predecessor for all nodes

find_nexts(*self*)

find cheapest successor for all nodes

generate_refuel_infos(*self*)

generate routing informations for these route

20 Module src.routing.node

node.py - Nodes for graph based representation of gas stations in a route

20.1 Variables

Name	Description
__package__	**Value:** `'src.routing'`

20.2 Class Node

20.2.1 Methods

__init__(*self*, *id_*, *lat*, *lon*, *price*=`0`, *timestamp*=`''`)

Node for representing a gas station id_: int, station id lat: float, the station latitude lon: float, the station longitude price: int, the gas price at the station timestamp: str, the time of visiting this station

__lt__(*self*, *other*)

__le__(*self*, *other*)

__eq__(*self*, *other*)

__str__(*self*)

__repr__(*self*)

distance_to(*self*, *other*, *g*, *use_tolerance*=`False`)

determine the distance between this node and <other>, if <use_tolerance> is set to True, the tolerance set in <g> is used. other: Node, the other node to determine distance to g: Graph, the graph containing all nodes of the route use_tolerance: bool, weither to use tolerance or not

return the calculated distance

price_for_gas(*self*, *amount*)

return the cost for <amount> of gas at this station

set_price(*self*, *price*)

set the price for this station

20.2.2 Properties

Name	Description
key	a unique station identifier at a given time

21 Package src.tests

tests - Tests runner

21.1 Functions

diff_prices(*data1*, *data2*)

return min, max and average difference between data1 and data2

get_route_files_prices()

return all route file with their ground truth files

get_predict_files_prices()

return all prediction files with their ground truth files

verify_route(*route_filename*, *route_prices_filename*, *nb_runs*=20)

Run a basic verification of the implement routing algorithm route_filename: str, the route file route_prices_filename: str, the generate prices for the route file nb_runs: int, the number of runs

test_predict()

Run the prediction tests

test_route()

Run the routing tests

test()

Run the prediction and routing tests

21.2 Variables

Name	Description
__package__	**Value:** `'src.tests'`

22 Module src.train

train.py - manage the whole training

22.1 Variables

Name	Description
__package__	**Value:** `'src'`

22.2 Class Trainer

object
└ **src.train.Trainer**

22.2.1 Methods

train(*force_train*=`False`)

autotrain()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

22.2.2 Properties

Name	Description
Inherited from object __class__	

23 Module src.utils

utils.py - usefool tools

23.1 Functions

diff_score(*v1*, *v2*)

return the norm of the difference between both vectors

str2latitute(*value*)

convert a str to valid latitude

str2longitude(*value*)

convert a str to valid longitude

str2mark(*value*)

convert a str to unicode

str2town(*value*)

convert a str to unicode

str2zipcode(*value*)

convert a str to int

create_file_dirs(*filename*)

create all directories contained in the tree to filename

create_dirs(*path*)

create all directories leading to path (inclusive itself)

23.2 Variables

Name	Description
ERROR_FILE_EXISTS	**Value:** `17`
__package__	**Value:** `'src'`

Index

src *(package)*, 5–6
  src.__main__ *(module)*, 7
  src.b_exceptions *(module)*, 8–13
    src.b_exceptions.BadFormatException *(class)*, 8–9
    src.b_exceptions.BadValueException *(class)*, 9
    src.b_exceptions.BenzlimException *(class)*, 9–10
    src.b_exceptions.PriceNotFoundException *(class)*, 10–11
    src.b_exceptions.StationNotFoundException *(class)*, 11–12
    src.b_exceptions.TrainingDataMissingException *(class)*, 12–13
  src.benchmark *(module)*, 14
    src.benchmark.benchmark_predictions *(function)*, 14
    src.benchmark.benchmark_routing *(function)*, 14
    src.benchmark.benchmark_with_prices *(function)*, 14
    src.benchmark.benchmark_without_prices *(function)*, 14
    src.benchmark.evaluate_prediction *(function)*, 14
    src.benchmark.process_benchmark *(function)*, 14
    src.benchmark.process_benchmark_prediction *(function)*, 14
  src.compat *(package)*, 15
    src.compat.py2 *(module)*, 16
    src.compat.py3 *(module)*, 17
  src.config *(module)*, 18
    src.config.Configuration *(class)*, 18
  src.coverage' *(package)*, 19
    src.coverage'.clean_benzlim *(function)*, 19
    src.coverage'.clean_mp_coverages *(function)*, 19
    src.coverage'.coverage *(function)*, 19
    src.coverage'.execute_coverage *(function)*, 19
src.dao *(package)*, 20–25
  src.dao.csv_ *(module)*, 26–27
  src.dao.db *(module)*, 28–31
src.exceptions_ *(module)*, 32–38
  src.exceptions_.BadFormatException *(class)*, 36–37
  src.exceptions_.BadValueException *(class)*, 37–38
  src.exceptions_.BenzlimException *(class)*, 32–33
  src.exceptions_.PriceNotFoundException *(class)*, 34–35
  src.exceptions_.StationNotFoundException *(class)*, 33–34
  src.exceptions_.TrainingDataMissingException *(class)*, 35–36
src.main *(function)*, 6
src.prediction *(package)*, 39
  src.prediction.classification *(module)*, 40–42
  src.prediction.db *(module)*, 43–46
  src.prediction.predict *(module)*, 47–48
  src.prediction.predict_prices_timestamps_x2_stations *(function)*, 39
  src.prediction.process_predictions *(function)*, 39
  src.prediction.process_routing *(function)*, 39
  src.prediction.process_task *(function)*, 39
src.routing *(package)*, 49
  src.routing.generate_tank_infos *(function)*, 49
  src.routing.graph *(module)*, 50
  src.routing.node *(module)*, 51–52
src.tests *(package)*, 53
  src.tests.diff_prices *(function)*, 53
  src.tests.get_predict_files_prices *(function)*, 53
  src.tests.get_route_files_prices *(function)*, 53
  src.tests.test *(function)*, 53
  src.tests.test_predict *(function)*, 53

src.tests.test_route *(function)*, 53
src.tests.verify_route *(function)*, 53
src.train *(module)*, 54
src.train.Trainer *(class)*, 54
src.utils *(module)*, 55
src.utils.create_dirs *(function)*, 55
src.utils.create_file_dirs *(function)*, 55
src.utils.diff_score *(function)*, 55
src.utils.str2latitute *(function)*, 55
src.utils.str2longitude *(function)*, 55
src.utils.str2mark *(function)*, 55
src.utils.str2town *(function)*, 55
src.utils.str2zipcode *(function)*, 55