```
C:\Users\Frede\Anaconda3\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2018.3.3\helpers\pydev\pydevconsole.py" --mode=client --port=54816

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\Frede\\Dropbox\\Master\\DM\\Assignments\\2\\DM2', 'C:\\Users\\Frede\\Dropbox\\Master\\DM\\Assignments\\2\\DM2\\Expedia-Ranking-Competition', 'C:/Users/Frede/Dropbox/Master/DM/Assignments/2/DM2'])

Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.
PyDev console: using IPython 7.4.0

Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] on win32
In[2]: import numpy as np
   ...: import pandas as pd
   ...: from sklearn.feature_selection import RFE
   ...: from sklearn.ensemble import RandomForestRegressor
   ...: from sklearn.ensemble import RandomForestClassifier
   ...: from sklearn.linear_model import LogisticRegression
   ...: from model_test import impute_na, get_sample, split_train_test
   ...: from feature_engineering import extract_train_features
   ...: from scoring import score_prediction
   ...:
   ...:
   ...: def feature_selection(data, estimator, n_features=None):
   ...:
   ...:     X_train = data.drop(columns=["target", "booking_bool", "click_bool", "position", "random_bool"])
   ...:     y_train = data["target"]
   ...:     selector = RFE(estimator=estimator, n_features_to_select=n_features)
   ...:     selector.fit(X_train, y_train)
   ...:     cols = selector.support_
   ...:     print(X_train.loc[:, cols].columns)
```

```
31   ...:
32   ...:
33   ...: def decreasing_features_select(data, estimator,
   target):
34   ...:
35   ...:     train, test = split_train_test(data, split=4)
36   ...:
37   ...:     X_train = train.drop(columns=["target", "
   booking_bool", "click_bool", "position", "random_bool"])
38   ...:     y_train = train["target"]
39   ...:     X_test = test.drop(columns=["target", "
   booking_bool", "click_bool", "position", "random_bool"])
40   ...:     y_test = test[["target", "srch_id", "prop_id
   ", "booking_bool", "click_bool"]]
41   ...:
42   ...:     n_features = len(X_train.columns)
43   ...:     for n in range(n_features,10, -2):
44   ...:         print("#################################")
45   ...:         print(f"Number of features used: {n}.")
46   ...:         selector = RFE(estimator=estimator,
   n_features_to_select=n)
47   ...:         selector.fit(X_train, y_train)
48   ...:         cols = selector.support_
49   ...:         print("Features used: ")
50   ...:         print(X_train.loc[:, cols].columns)
51   ...:         reduced_train = X_train.loc[:, cols]
52   ...:         estimator.fit(reduced_train, y_train)
53   ...:         reduced_test = X_test.loc[:, cols]
54   ...:         pred = clf_to_predict(estimator,
   reduced_test, target)
55   ...:         score_prediction(pred, y_test, to_print=
   True)
56   ...:
57   ...:
58   ...: def clf_to_predict(estimator, X_test, target):
59   ...:     if target == "book":
60   ...:         prediction = estimator.predict_proba(X_test
   )[:, 1]
61   ...:     elif target == "score":
62   ...:         predict_array = estimator.predict_proba(
   X_test)
63   ...:         predict_array[:, 2] = predict_array[:, 2
   ] * pred_weight
64   ...:         prediction = predict_array[:, [1, 2]].sum(
   axis=1)
```

```
65   ...:     else:
66   ...:         print("ERROR. no using classification with
    score_rank!")
67   ...:         return
68   ...:
69   ...:     return prediction
70   ...:
71   ...:
72   ...: if __name__ == "main":
73   ...:     pd.options.mode.chained_assignment = None
74   ...:     data = pd.read_csv("C:/Users/Frede/Dropbox/
   Master/DM/Assignments/2/DM2/final_training_fixed_data.csv
   ")
75   ...:     impute_na(data)
76   ...:     sample = get_sample(data=data, size=0.1)
77   ...:     estimator = RandomForestClassifier(
   n_estimators=100, n_jobs=-1)
78   ...:     targets = ["score"]
79   ...:     max_rank = 10
80   ...:     pred_weight = 3
81   ...:
82   ...:     for target in targets:
83   ...:         print(f"\nCURRENT CONFIGURATION")
84   ...:         print
   ("##############################################################
   ##################")
85   ...:         print(f"Target = {target}")
86   ...:         print(f"Max_rank = {max_rank}")
87   ...:         print(f"Pred_weight = {pred_weight}")
88   ...:         print
   ("##############################################################
   ##################")
89   ...:
90   ...:         extract_train_features(data=sample, target
   =target, max_rank=max_rank)
91   ...:         #decreasing_features_select(data=sample,
   estimator=estimator, target=target)
92   ...:         feature_selection(data=sample, estimator=
   estimator, n_features=10)
93   ...:
94   ...:
95   ...:
96   ...:
97   ...:
98   ...:
```

```
 99   ...:
100   ...:
101   ...:
102   ...:
103 Backend Qt5Agg is interactive backend. Turning
    interactive mode on.
104 In[3]: import numpy as np
105   ...: import pandas as pd
106   ...: from sklearn.feature_selection import RFE
107   ...: from sklearn.ensemble import RandomForestRegressor
108   ...: from sklearn.ensemble import
    RandomForestClassifier
109   ...: from sklearn.linear_model import
    LogisticRegression
110   ...: from model_test import impute_na, get_sample,
    split_train_test
111   ...: from feature_engineering import
    extract_train_features
112   ...: from scoring import score_prediction
113   ...:
114   ...:
115   ...: def feature_selection(data, estimator, n_features=
    None):
116   ...:
117   ...:     X_train = data.drop(columns=["target", "
    booking_bool", "click_bool", "position", "random_bool"])
118   ...:     y_train = data["target"]
119   ...:     selector = RFE(estimator=estimator,
    n_features_to_select=n_features)
120   ...:     selector.fit(X_train, y_train)
121   ...:     cols = selector.support_
122   ...:     print(X_train.loc[:, cols].columns)
123   ...:
124   ...:
125   ...: def decreasing_features_select(data, estimator,
    target):
126   ...:
127   ...:     train, test = split_train_test(data, split=4)
128   ...:
129   ...:     X_train = train.drop(columns=["target", "
    booking_bool", "click_bool", "position", "random_bool"])
130   ...:     y_train = train["target"]
131   ...:     X_test = test.drop(columns=["target", "
    booking_bool", "click_bool", "position", "random_bool"])
132   ...:     y_test = test[["target", "srch_id", "prop_id
```

```
", "booking_bool", "click_bool"]]
  ...:
  ...:     n_features = len(X_train.columns)
  ...:     for n in range(n_features,10, -2):
  ...:         print
("#################################")
  ...:         print(f"Number of features used: {n}.")
  ...:         selector = RFE(estimator=estimator,
n_features_to_select=n)
  ...:         selector.fit(X_train, y_train)
  ...:         cols = selector.support_
  ...:         print("Features used: ")
  ...:         print(X_train.loc[:, cols].columns)
  ...:         reduced_train = X_train.loc[:, cols]
  ...:         estimator.fit(reduced_train, y_train)
  ...:         reduced_test = X_test.loc[:, cols]
  ...:         pred = clf_to_predict(estimator,
reduced_test, target)
  ...:         score_prediction(pred, y_test, to_print=
True)
  ...:
  ...:
  ...: def clf_to_predict(estimator, X_test, target):
  ...:     if target == "book":
  ...:         prediction = estimator.predict_proba(
X_test)[:, 1]
  ...:     elif target == "score":
  ...:         predict_array = estimator.predict_proba(
X_test)
  ...:         predict_array[:, 2] = predict_array[:, 2
] * pred_weight
  ...:         prediction = predict_array[:, [1, 2]].sum(
axis=1)
  ...:     else:
  ...:         print("ERROR. no using classification with
 score_rank!")
  ...:         return
  ...:
  ...:     return prediction
  ...:
In[4]: pd.options.mode.chained_assignment = None
  ...: data = pd.read_csv("C:/Users/Frede/Dropbox/Master/
DM/Assignments/2/DM2/final_training_fixed_data.csv")
  ...: impute_na(data)
  ...: sample = get_sample(data=data, size=0.1)
```

```
167   ...: estimator = RandomForestClassifier(n_estimators=
    100, n_jobs=-1)
168   ...: targets = ["score"]
169   ...: max_rank = 10
170   ...: pred_weight = 3
171   ...:
172   ...: for target in targets:
173   ...:     print(f"\nCURRENT CONFIGURATION")
174   ...:     print
    ("##########################################################
    ##################")
175   ...:     print(f"Target = {target}")
176   ...:     print(f"Max_rank = {max_rank}")
177   ...:     print(f"Pred_weight = {pred_weight}")
178   ...:     print
    ("##########################################################
    ##################")
179   ...:
180   ...:     extract_train_features(data=sample, target=
    target, max_rank=max_rank)
181   ...:     #decreasing_features_select(data=sample,
    estimator=estimator, target=target)
182   ...:     feature_selection(data=sample, estimator=
    estimator, n_features=10)
183   ...:
184   ...:
185   ...:
186   ...:
187   ...:
188   ...:
189   ...:
190
191 CURRENT CONFIGURATION
192 ##########################################################
    ###############
193 Target = score
194 Max_rank = 10
195 Pred_weight = 3
196 ##########################################################
    ###############
197 Index(['prop_location_score1', 'prop_location_score2',
198        'orig_destination_distance', 'gross_bookings_usd
    ', 'srch_average_loc1',
199        'srch_diff_price', 'srch_diff_locscore1', '
    srch_diff_locscore2',
```

```
200         'srch_diff_prop_review_score', '
    norm_srch_diff_locscore2'],
201        dtype='object')
202
```