TABLE DES MATIERES

KNN (100 WORDS BY CODE) :

FITTING 3 FOLDS FOR EACH OF 3 CANDIDATES, TOTALLING 9 FITS

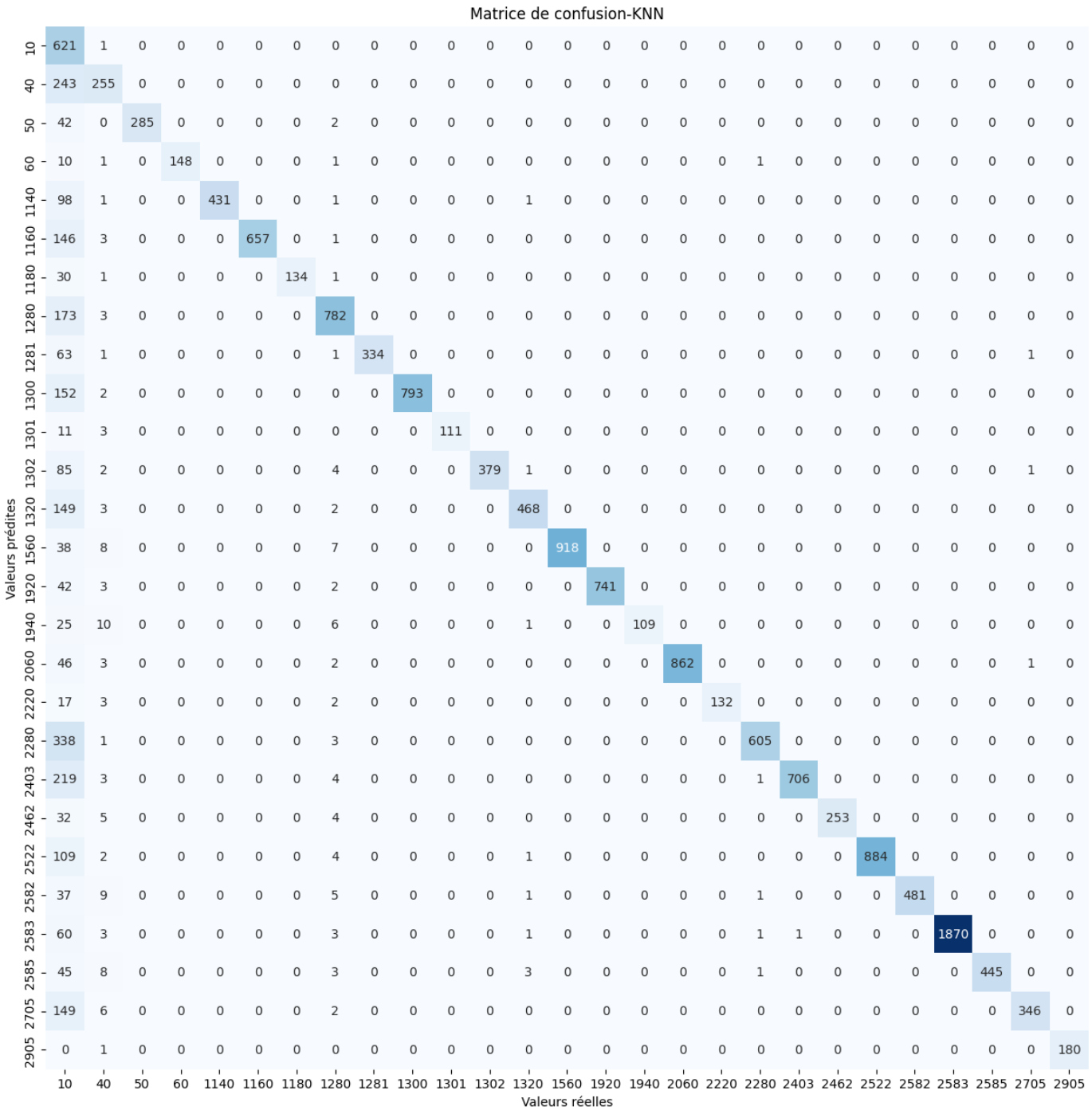
TRAIN_R2_SCORE = 0.8541603355011245

TEST_R2_SCORE = 0.8466541056342308

==========CONFUSION MATRIX==========

3. Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :



KNN (100 WORDS BY CODE)

FITTING 3 FOLDS FOR EACH OF 1 CANDIDATES, TOTALLING 3 FITS

TRAIN_R2_SCORE = 0.8645383820579834

TEST_R2_SCORE = 0.8550416337446058

train_mse_result = 451240.9234030268

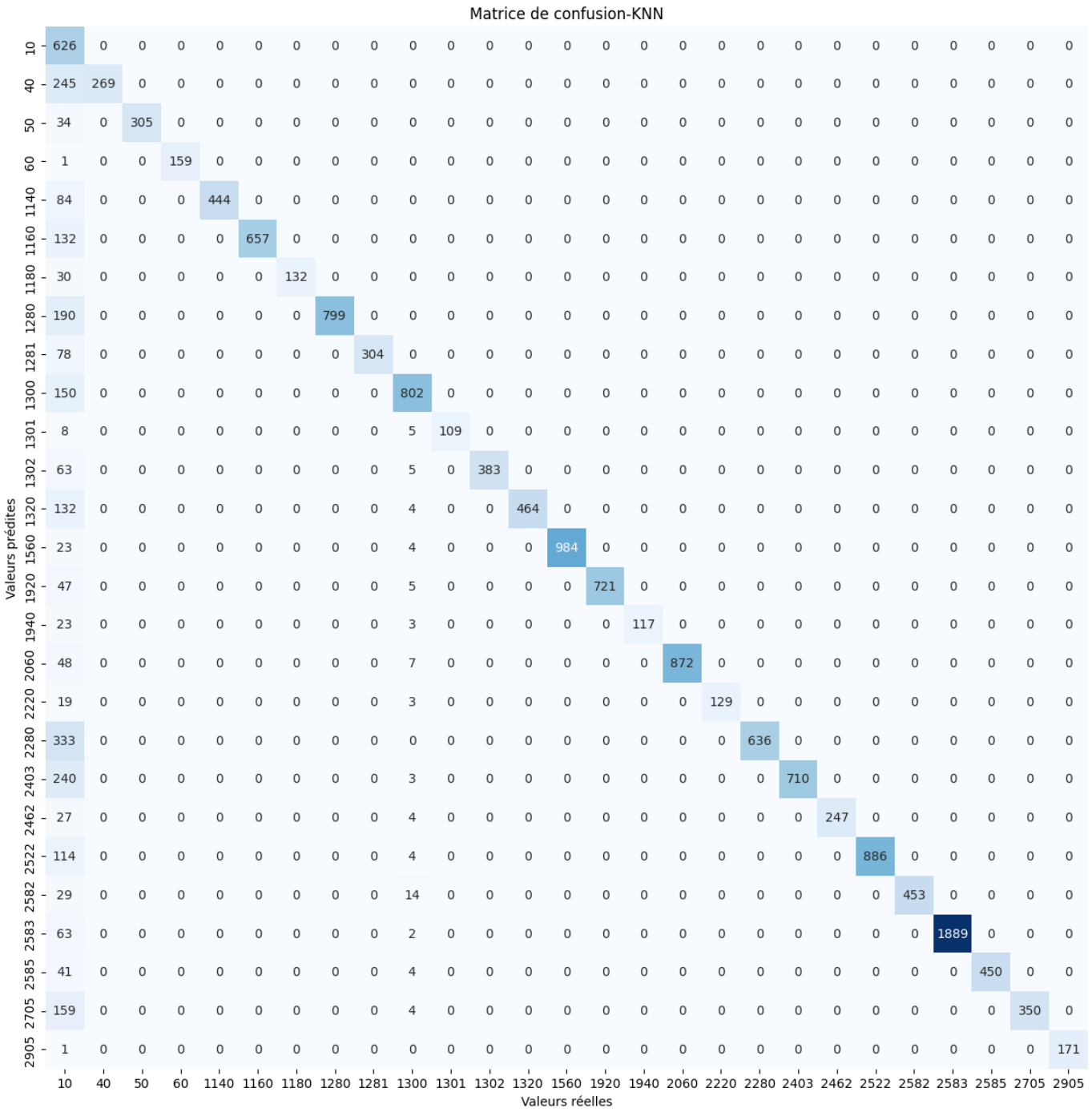
test_mse_result = 488600.12933811464

best_params: [{'n_neighbors': 2}]

==========CONFUSION MATRIX==========

3. Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :



RF (100 WORDS BY CODE):

FITTING 3 FOLDS FOR EACH OF 1 CANDIDATES, TOTALLING 3 FITS

TRAIN_R2_SCORE = 0.8693095484106242

TEST_R2_SCORE = 0.8640977329362426

train_mse_result = 446904.01537713484

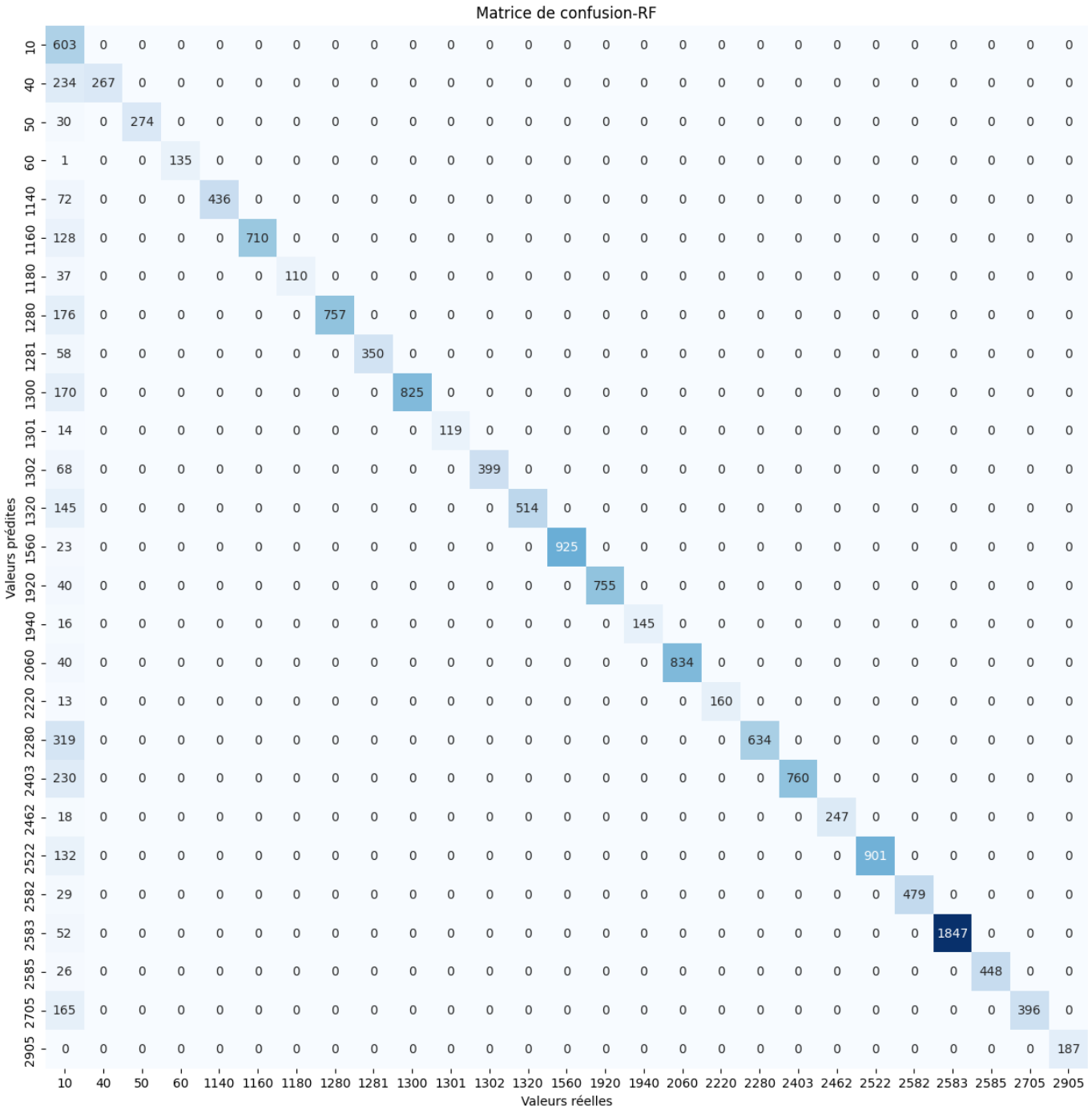
test_mse_result = 465541.4230231569

best_params: [{'max_features': 'sqrt', 'min_samples_split': 100}]

==========CONFUSION MATRIX==========

3. Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :



RF (20 WORDS BY CODE)

ESTIMATOR RANDOMFORESTCLASSIFIER()

PARAMS {'MAX_FEATURES': ['SQRT'], 'MIN_SAMPLES_SPLIT'...

FITTING 3 FOLDS FOR EACH OF 4 CANDIDATES, TOTALLING 12 FITS

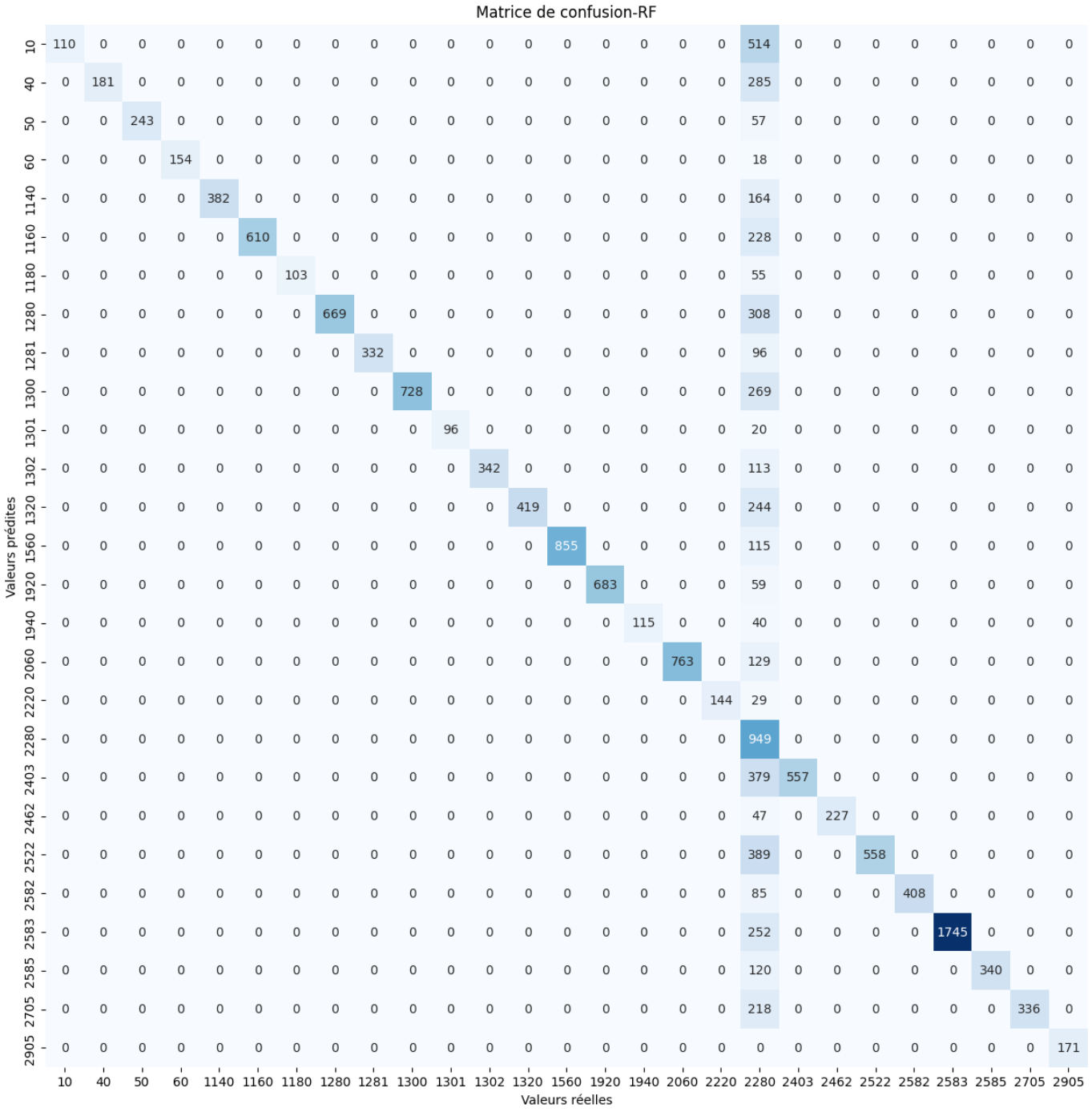
TRAIN_R2_SCORE = 0.7422506533762839

TEST_R2_SCORE = 0.7427216920926275

==========CONFUSION MATRIX==========

3. Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :



train_f1_score = [array([0.2868937 , 0.58707865, 0.90558292, 0.94570928, 0.82228999,

0.84216397, 0.7988107 , 0.79822897, 0.8501845 , 0.84218161,

0.88344988, 0.84454176, 0.80547041, 0.93720586, 0.95649241,

0.85385297, 0.92784717, 0.9112426 , 0.30963331, 0.74173927,

0.92497626, 0.74491886, 0.90461875, 0.93111803, 0.86209887,

0.74950242, 0.99928622])]

test_f1_score = [array([0.29972752, 0.55950541, 0.89502762, 0.94478528, 0.82327586,

0.84254144, 0.78927203, 0.81287971, 0.87368421, 0.84405797,

0.90566038, 0.85821832, 0.77449168, 0.9369863 , 0.95859649,

0.85185185, 0.92205438, 0.90851735, 0.30957429, 0.74614869,

0.90618762, 0.74152824, 0.90566038, 0.93265633, 0.85 ,

0.75505618, 1.])]

train_mse_result = 379914.35068072693

test_mse_result = 378075.26171518874

best_params: [{'max_features': 'sqrt', 'min_samples_split': 10}]

KNN (20 WORDS BY CODE)

ESTIMATOR KNEIGHBORSCLASSIFIER()

PARAMS {'N_NEIGHBORS': [100, 200, 300, 500, 1000]}

FITTING 3 FOLDS FOR EACH OF 5 CANDIDATES, TOTALLING 15 FITS

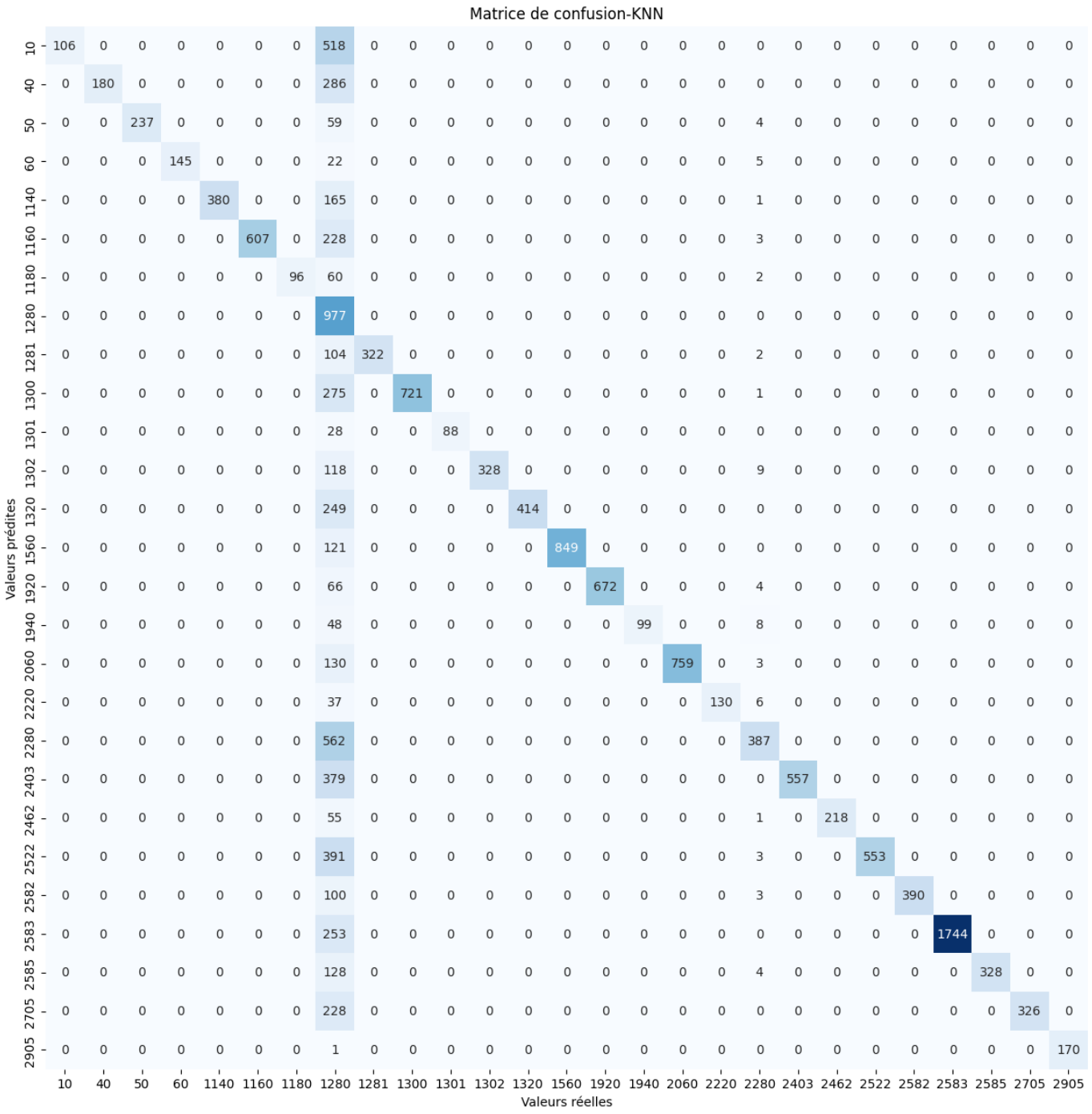
TRAIN_R2_SCORE = 0.7166170303288154

TEST_R2_SCORE = 0.7161611864097733

==========CONFUSION MATRIX==========

3. Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :



train_f1_score = [array([0.2868937 , 0.58707865, 0.90558292, 0.94570928, 0.82228999,

0.84216397, 0.7988107 , 0.79822897, 0.8501845 , 0.84218161,

0.88344988, 0.84454176, 0.80547041, 0.93720586, 0.95649241,

0.85385297, 0.92784717, 0.9112426 , 0.30963331, 0.74173927,

0.92497626, 0.74491886, 0.90461875, 0.93111803, 0.86209887,

0.74950242, 0.99928622]), array([0.28098032, 0.5795976 , 0.88777639, 0.92307692, 0.82 ,

0.84108671, 0.77575758, 0.29154519, 0.83470456, 0.83898182,

0.85406699, 0.82804569, 0.80037888, 0.93376501, 0.94726097,

0.78611632, 0.92516205, 0.8591674 , 0.55028187, 0.74153239,

0.90310078, 0.74313664, 0.88294314, 0.93067387, 0.84811238,

0.73802009, 0.99928622])]

test_f1_score = [array([0.29972752, 0.55950541, 0.89502762, 0.94478528, 0.82327586,

0.84254144, 0.78927203, 0.81287971, 0.87368421, 0.84405797,

0.90566038, 0.85821832, 0.77449168, 0.9369863 , 0.95859649,

0.85185185, 0.92205438, 0.90851735, 0.30957429, 0.74614869,

0.90618762, 0.74152824, 0.90566038, 0.93265633, 0.85 ,

0.75505618, 1.]), array([0.29041096, 0.55727554, 0.88268156, 0.9148265 , 0.82073434,

0.84013841, 0.75590551, 0.29763899, 0.85866667, 0.83934808,

0.8627451 , 0.83780332, 0.76880223, 0.93347993, 0.95049505,

0.77952756, 0.91944276, 0.85808581, 0.55483871, 0.74614869,

0.88617886, 0.73733333, 0.88335221, 0.93237102, 0.83248731,

0.74090909, 0.99706745])]

train_mse_result = 284982.4905184465

test_mse_result = 282187.1504892725

best_params: [{'max_features': 'sqrt', 'min_samples_split': 10}, {'n_neighbors': 100}]

KNN (150 WORDS BY CODE)

TRAIN_R2_SCORE = 0.8638242265848174

TEST_R2_SCORE = 0.8516379991490913

BEST_PARAMS: [{'N_NEIGHBORS': 10}]

estimator KNeighborsClassifier()

params {'n_neighbors': [10]}

---

df.shape : (82265, 4052)

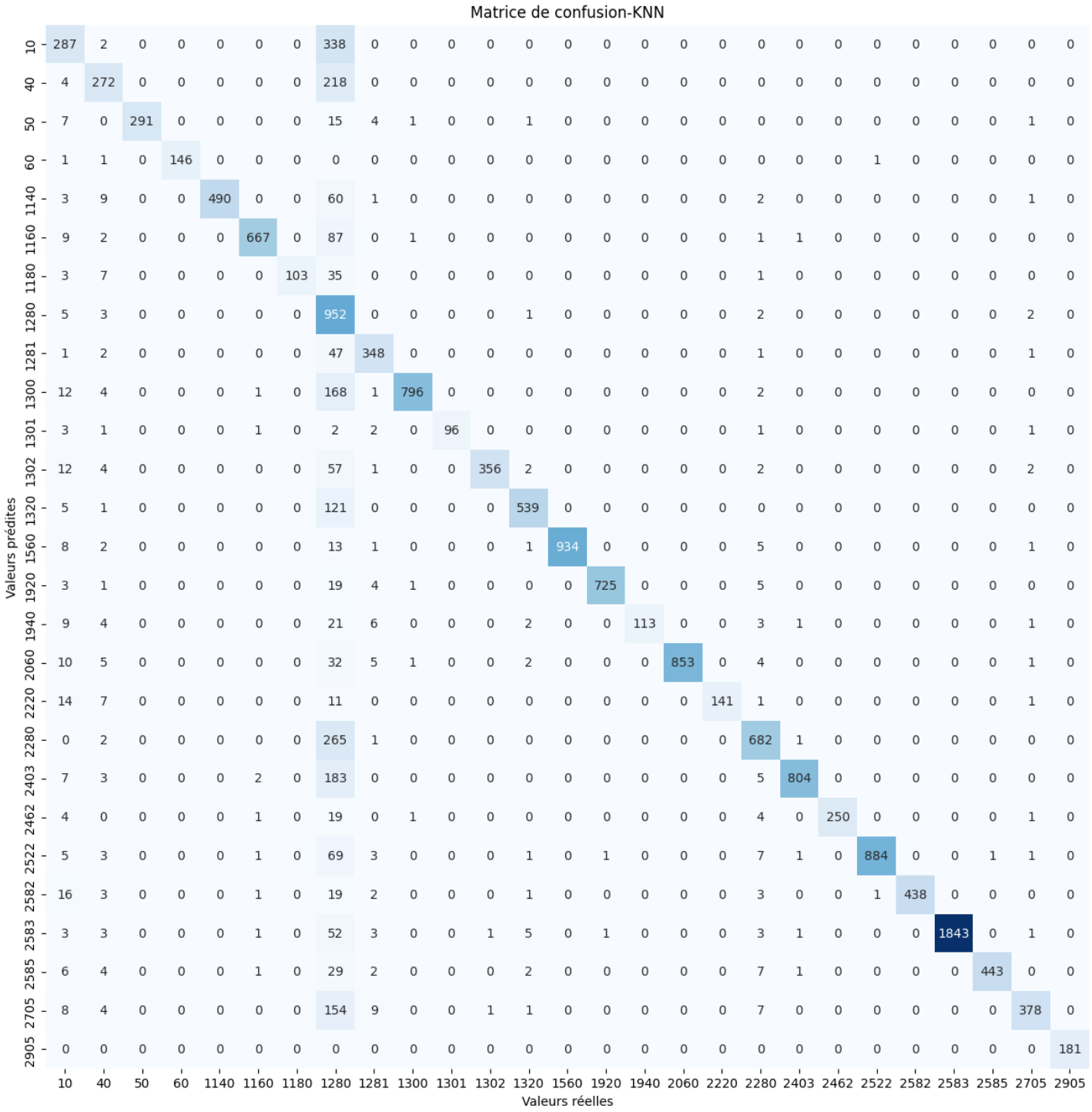
X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 4050) - (16453, 4050) - 65812 - 16453

==========CONFUSION MATRIX==========

3. Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :



train_f1_score = [array([0.57901204, 0.68607825, 0.95127796, 0.98703404, 0.93035079,

0.93022476, 0.87568556, 0.48681333, 0.88793103, 0.90987821,

0.93099671, 0.92581944, 0.88107058, 0.98251479, 0.98434668,

0.89974293, 0.97252903, 0.94146744, 0.816935 , 0.87853233,

0.95158287, 0.95363889, 0.95509992, 0.98576165, 0.95005429,

0.82464956, 0.99855072])]

test_f1_score = [array([0.53544776, 0.64531435, 0.95253682, 0.98983051, 0.9280303 ,

0.92382271, 0.81746032, 0.48190332, 0.8776797 , 0.89187675,

0.94581281, 0.89672544, 0.88071895, 0.98367562, 0.97643098,

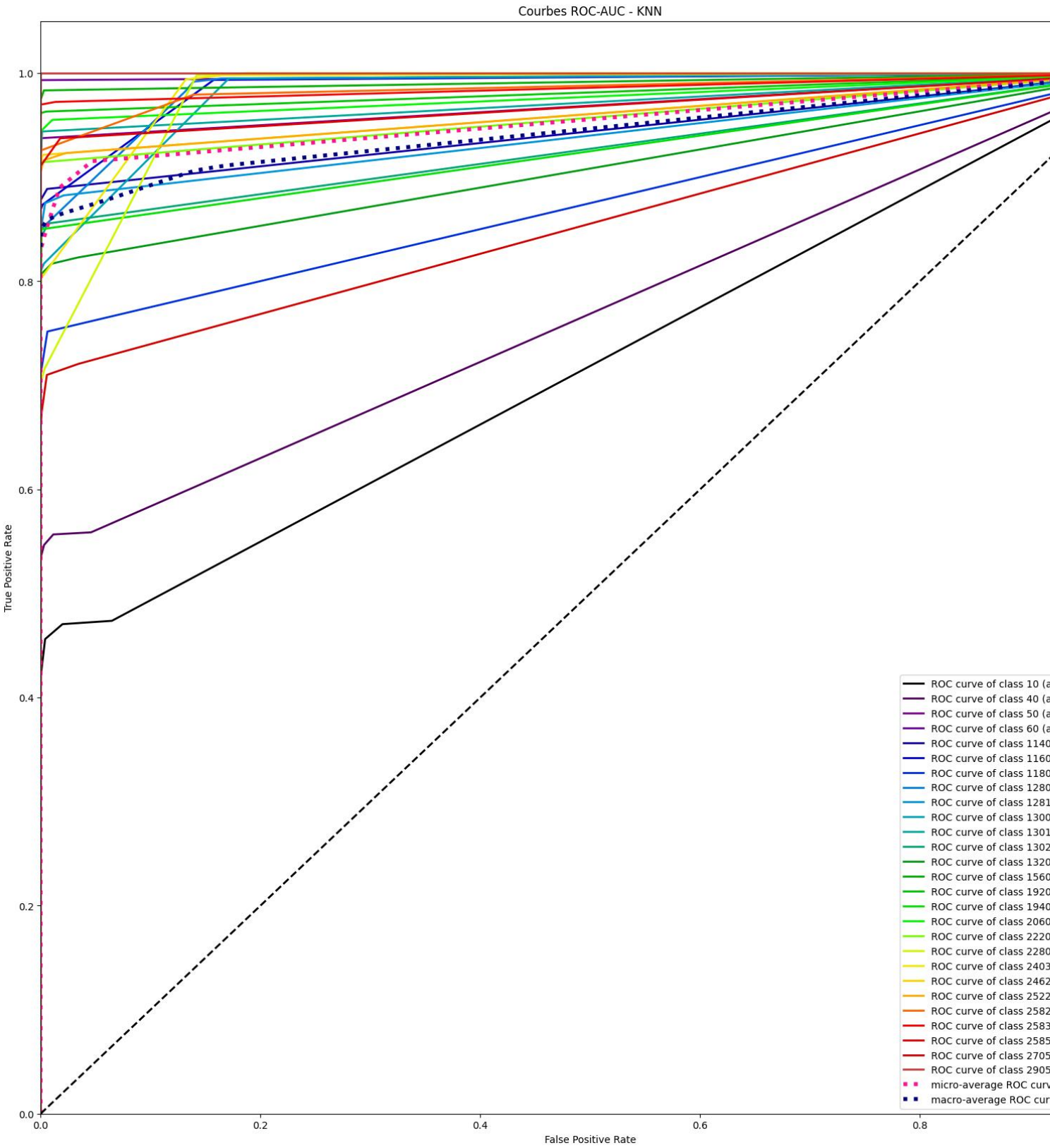
0.82783883, 0.96602492, 0.89240506, 0.80282519, 0.88643881,

0.94339623, 0.94900698, 0.95010846, 0.98031915, 0.94355698,

0.79162304, 1.])]

train_mse_result = 160215.49750805323

test_mse_result = 184986.59539293745



KNN (150 WORDS BY CODE)

ESTIMATOR KNEIGHBORSCLASSIFIER()

PARAMS {'N_NEIGHBORS': [10]}

TRAIN_R2_SCORE = 0.8857199294961405

TEST_R2_SCORE = 0.8786847383455905

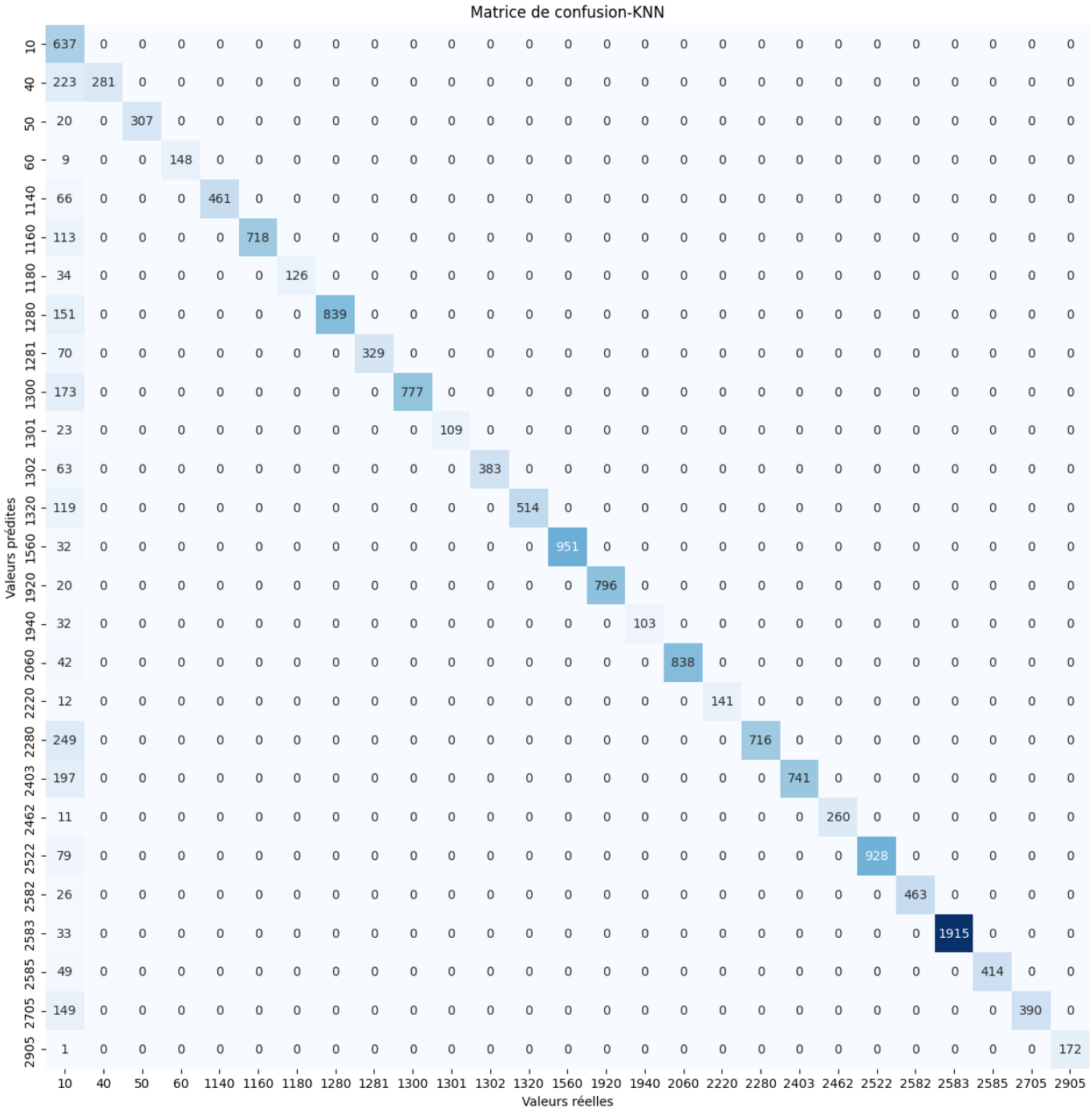
X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 4050) - (16453, 4050) - 65812 - 16453

Fitting 3 folds for each of 1 candidates, totalling 3 fits

==========CONFUSION MATRIX==========

3. Use SEABORN to draw confusion_matrix----------



train_f1_score = [array([0.39711423, 0.72972973, 0.96431404, 0.99018003, 0.94329389,

0.93843537, 0.87940631, 0.9235361 , 0.92734032, 0.91129685,

0.94292237, 0.92778741, 0.89739729, 0.99012947, 0.98742666,

0.92679002, 0.9777964 , 0.95019763, 0.83607313, 0.89386929,

0.97977528, 0.96024384, 0.97210136, 0.98762054, 0.96360759,

0.83718487, 0.99928418])]

test_f1_score = [array([0.38960245, 0.71592357, 0.96845426, 0.9704918 , 0.93319838,

0.92704971, 0.88111888, 0.91744122, 0.90384615, 0.89982629,

0.90456432, 0.92400483, 0.89625109, 0.98345398, 0.98759305,

0.86554622, 0.97555297, 0.95918367, 0.85187388, 0.88266825,

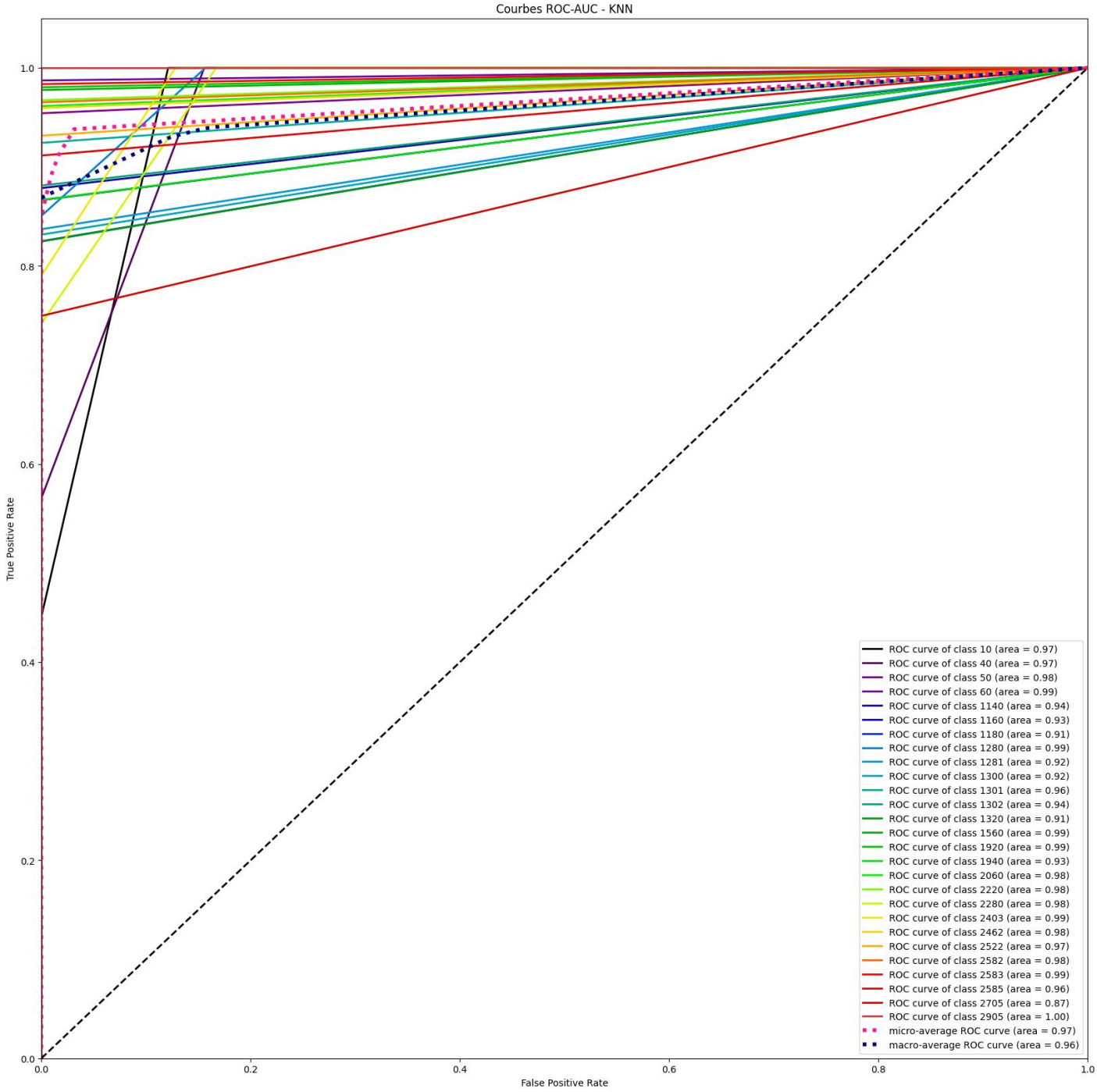
0.97928437, 0.95917313, 0.97268908, 0.99145742, 0.94412771,

0.83961249, 0.99710145])]

train_mse_result = 389357.61490305717

test_mse_result = 398629.87017565186

best_params: [{'n_neighbors': 10}]



KNN (150 WORDS CODE) AVEC SCALING

ESTIMATOR KNEIGHBORSCLASSIFIER()

PARAMS {'N_NEIGHBORS': [10, 12, 30]}

TRAIN_R2_SCORE = 0.8887436941591199

TEST_R2_SCORE = 0.88160213942746

---

df.shape : (82265, 4052)

Fitting 3 folds for each of 3 candidates, totalling 9 fits

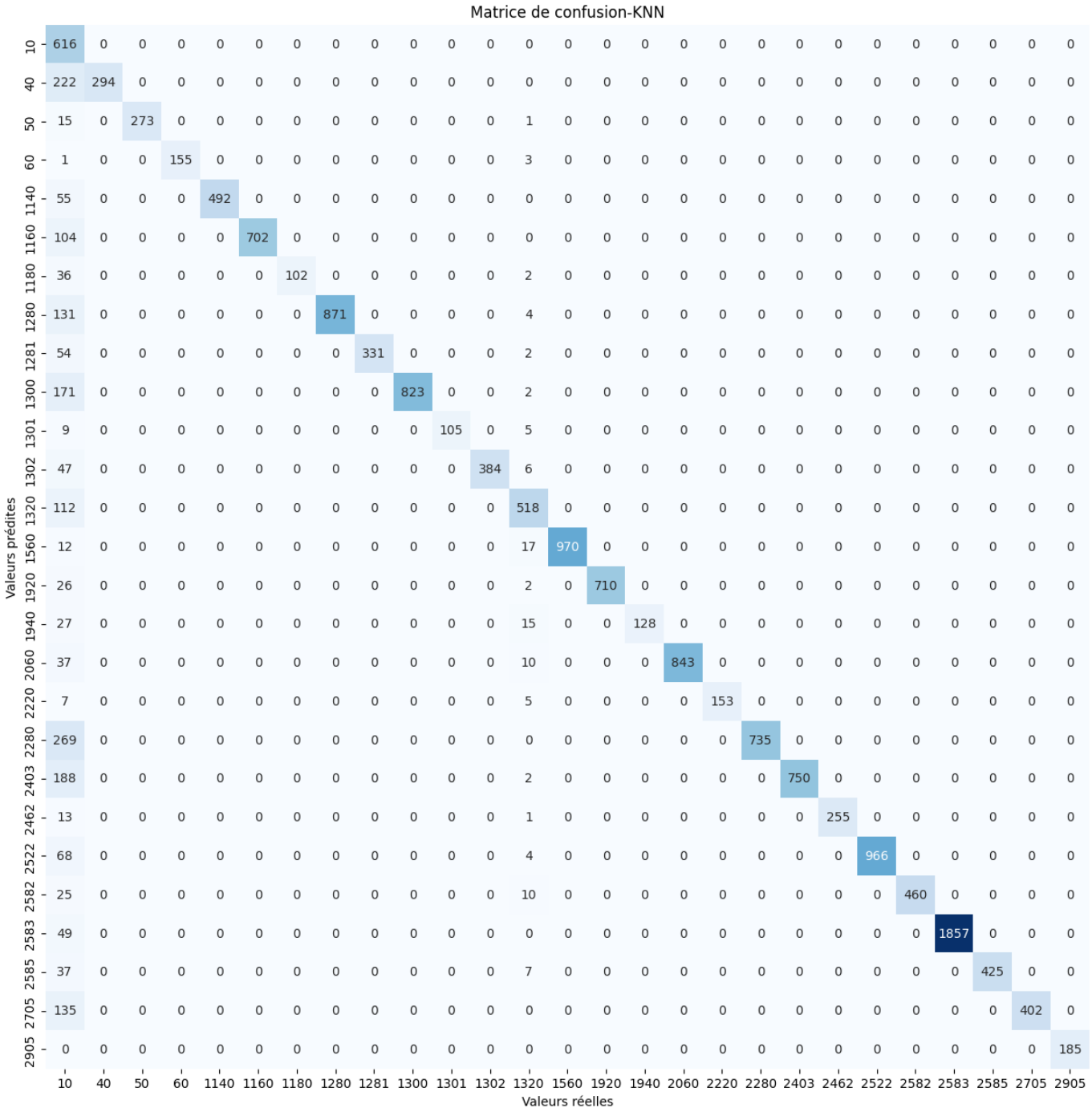
X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 4050) - (16453, 4050) - 65812 - 16453

==========CONFUSION MATRIX==========

Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :



train_f1_score = [array([0.41769083, 0.73500967, 0.9689298 , 0.99673736, 0.9419387 ,

0.93805907, 0.90718039, 0.92072124, 0.92792491, 0.91580663,

0.95499451, 0.93367639, 0.83938852, 0.98852649, 0.9897277 ,

0.92972058, 0.97787735, 0.9542903 , 0.83787973, 0.89236564,

0.98163905, 0.95949739, 0.97016461, 0.98893276, 0.96046697,

0.85121825, 0.99854227])]

test_f1_score = [array([0.39974043, 0.72592593, 0.97153025, 0.98726115, 0.94706449,

0.93103448, 0.84297521, 0.92807672, 0.92200557, 0.9048928 ,

0.9375 , 0.93544458, 0.83146067, 0.98527171, 0.98066298,

0.8590604 , 0.9728794 , 0.96226415, 0.8453134 , 0.88757396,

0.97328244, 0.96407186, 0.96335079, 0.98697847, 0.950783 ,

0.85623003, 1.])]

train_mse_result = 361571.8485230657

test_mse_result = 382027.62973317935

best_params: [{'n_neighbors': 10}]

KNN (100 WORDS BY CODE)

TRAIN_R2_SCORE = 0.8861802979450039

TEST_R2_SCORE = 0.8843028732925106

BEST_PARAMS: [{'ALGORITHM': 'AUTO', 'N_JOBS': -1, 'N_NEIGHBORS': 10, 'WEIGHTS': 'DISTANCE'}]

X_train.shape - X_test.shape - len(y_train) - len(y_test)

(67932, 2700) - (16984, 2700) - 67932 - 16984

estimator KNeighborsClassifier()

params {'n_neighbors': [10], 'weights': ['uniform', '...

Fitting 3 folds for each of 2 candidates, totalling 6 fits

train_f1_score = [array([0.39089334, 0.76324655, 0.95494071, 0.99925981, 0.92794814,

0.93105779, 0.8762421 , 0.90616622, 0.91707317, 0.91470786,

0.95813953, 0.94 , 0.90372272, 0.98390572, 0.98402839,

0.95230126, 0.97602475, 0.97179694, 0.81697044, 0.91878173,

0.97751799, 0.95299539, 0.98472906, 0.98477977, 0.959442 ,

0.85405961, 1.])]

test_f1_score = [array([0.39637953, 0.73316062, 0.94256259, 0.99678457, 0.93346981,

0.94455578, 0.92830189, 0.9010503 , 0.87483871, 0.92225201,

0.9453125 , 0.9376392 , 0.90306947, 0.98521698, 0.96850862,

0.94642857, 0.98052921, 0.93103448, 0.81997372, 0.92016083,

0.97472924, 0.94807892, 0.97773475, 0.98227216, 0.9600863 ,

0.8380744 , 0.996997])]

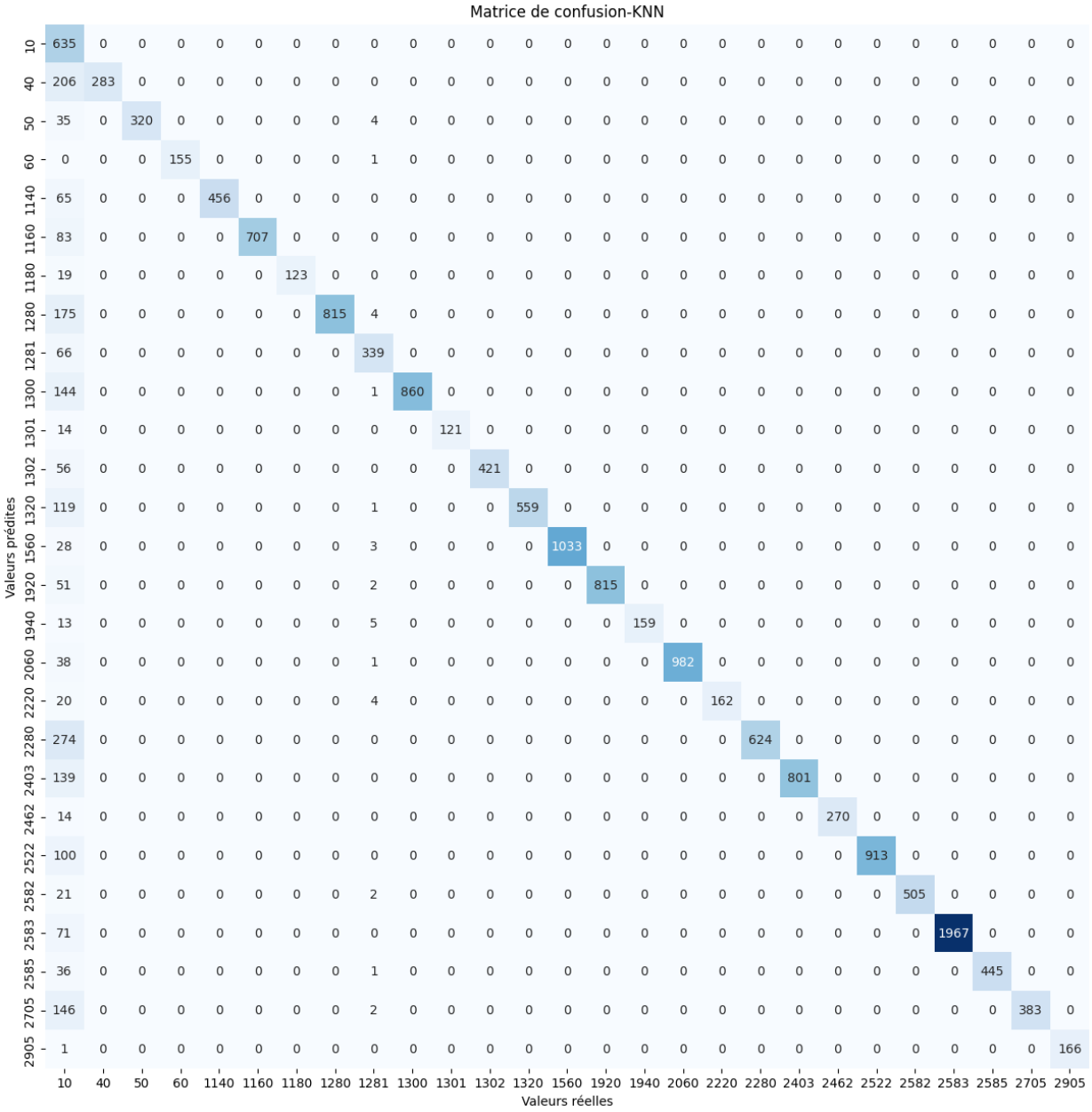
train_mse_result = 382045.8192162751

test_mse_result = 388084.74004945834

==========CONFUSION MATRIX==========

Use SEABORN to draw confusion_matrix----------

Confusion matrix as graph with Seaborn :


Matrice de confusion-KNN
Valeurs prédites
Valeurs réelles
10 40 50 60 1140 1160 1180 1280 1281 1300 1301 1302 1320 1560 1920 1940 2060 2220 2280 2403 2462 2522 2582 2583 2585 2705 2905
10 635 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
40 206 283 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
50 35 0 320 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
60 0 0 0 155 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1140 65 0 0 0 456 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1160 83 0 0 0 0 707 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1180 19 0 0 0 0 0 123 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1280 175 0 0 0 0 0 0 815 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1281 66 0 0 0 0 0 0 0 339 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1300 144 0 0 0 0 0 0 0 1 860 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1301 14 0 0 0 0 0 0 0 0 0 121 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1302 56 0 0 0 0 0 0 0 0 0 0 421 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1320 119 0 0 0 0 0 0 0 1 0 0 0 559 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1560 28 0 0 0 0 0 0 0 3 0 0 0 0 1033 0 0 0 0 0 0 0 0 0 0 0 0 0
1920 51 0 0 0 0 0 0 0 2 0 0 0 0 0 815 0 0 0 0 0 0 0 0 0 0 0 0
1940 13 0 0 0 0 0 0 0 5 0 0 0 0 0 0 159 0 0 0 0 0 0 0 0 0 0 0
2060 38 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 982 0 0 0 0 0 0 0 0 0 0
2220 20 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 162 0 0 0 0 0 0 0 0 0
2280 274 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 624 0 0 0 0 0 0 0 0
2403 139 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 801 0 0 0 0 0 0 0
2462 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 270 0 0 0 0 0 0
2522 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 913 0 0 0 0 0
2582 21 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 505 0 0 0 0
2583 71 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1967 0 0 0
2585 36 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 445 0 0
2705 146 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 383 0
2905 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 166

KNN (300 WORD BY CODE)

TRAIN_R2_SCORE = 0.9067799185558865

TEST_R2_SCORE = 0.9002613505135841

ESTIMATOR KNEIGHBORSCLASSIFIER()

PARAMS {'N_NEIGHBORS': [10]}

X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 8100) - (16453, 8100) - 65812 - 16453

Fitting 3 folds for each of 1 candidates, totalling 3 fits

train_f1_score = [array([0.75349301, 0.8144208 , 0.9837587 , 0.98947368, 0.42818645,

0.96005218, 0.90762332, 0.94754279, 0.95120364, 0.92673847,

0.97002141, 0.95146727, 0.93545683, 0.99200619, 0.99376026,

0.94339623, 0.98420685, 0.964687 , 0.89900759, 0.92226501,

0.98637602, 0.97737438, 0.98398983, 0.99484071, 0.96810207,

0.799908 , 0.97447119])]

test_f1_score = [array([0.74541752, 0.8035488 , 0.98245614, 0.97260274, 0.41079812,

0.9569378 , 0.90070922, 0.94072448, 0.95384615, 0.9218573 ,

0.95412844, 0.94033413, 0.92193919, 0.98801199, 0.99282453,

0.88732394, 0.97972973, 0.94153846, 0.91482301, 0.92876563,

0.97707231, 0.97795198, 0.96465696, 0.99503787, 0.96051227,

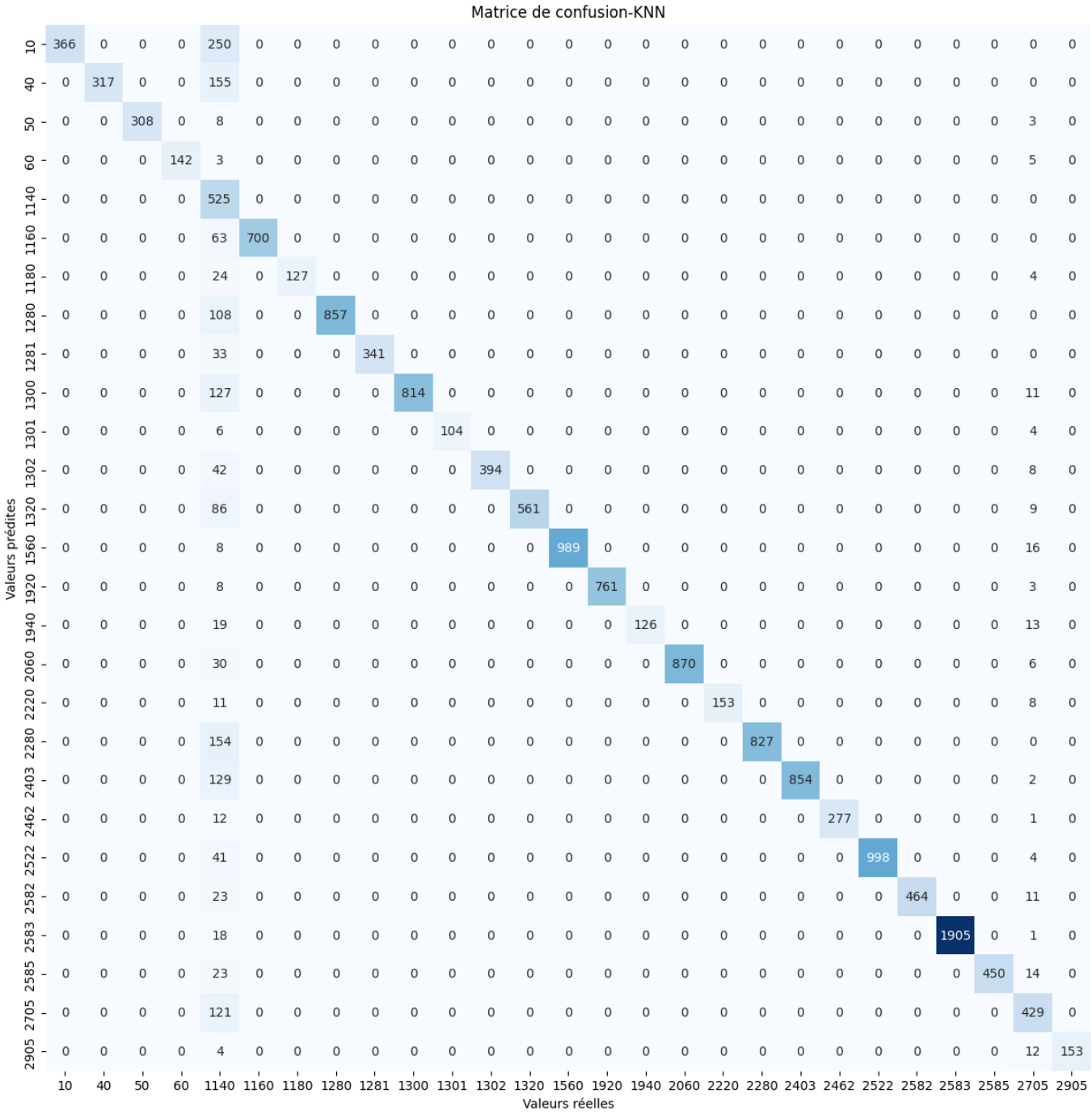
0.77019749, 0.95031056])]

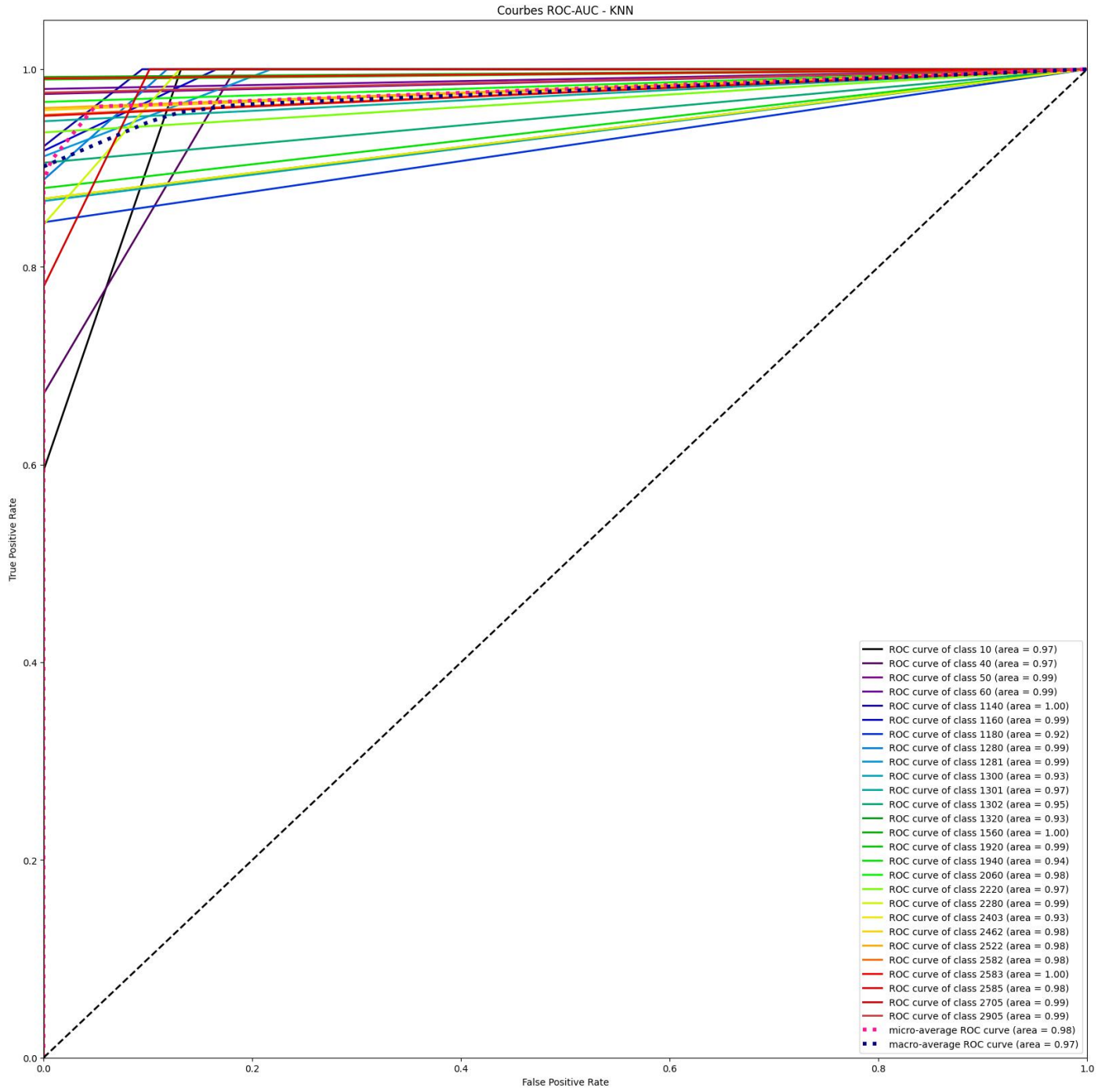
train_mse_result = 96349.62113292409

test_mse_result = 103203.23928766791

best_params: [{'n_neighbors': 10}]

==========CONFUSION MATRIX==========


Matrice de confusion-KNN
Valeurs prédites
Valeurs réelles
10 40 50 60 1140 1160 1180 1280 1281 1300 1301 1302 1320 1560 1920 1940 2060 2220 2280 2403 2462 2522 2582 2583 2585 2705 2905


Courbes ROC-AUC - KNN
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 10 (area = 0.97)
ROC curve of class 40 (area = 0.97)
ROC curve of class 50 (area = 0.99)
ROC curve of class 60 (area = 0.99)
ROC curve of class 1140 (area = 1.00)
ROC curve of class 1160 (area = 0.99)
ROC curve of class 1180 (area = 0.92)
ROC curve of class 1280 (area = 0.99)
ROC curve of class 1281 (area = 0.99)
ROC curve of class 1300 (area = 0.93)
ROC curve of class 1301 (area = 0.97)
ROC curve of class 1302 (area = 0.95)
ROC curve of class 1320 (area = 0.93)
ROC curve of class 1560 (area = 1.00)
ROC curve of class 1920 (area = 0.99)
ROC curve of class 1940 (area = 0.94)
ROC curve of class 2060 (area = 0.98)
ROC curve of class 2220 (area = 0.97)
ROC curve of class 2280 (area = 0.99)
ROC curve of class 2403 (area = 0.93)
ROC curve of class 2462 (area = 0.98)
ROC curve of class 2522 (area = 0.98)
ROC curve of class 2582 (area = 0.98)
ROC curve of class 2583 (area = 1.00)
ROC curve of class 2585 (area = 0.98)
ROC curve of class 2705 (area = 0.99)
ROC curve of class 2905 (area = 0.99)
micro-average ROC curve (area = 0.98)
macro-average ROC curve (area = 0.97)

RBF (100 WORDS BY CODE)

TRAIN_R2_SCORE = 0.8660274721935209

TEST_R2_SCORE = 0.8619704613140461

BEST_PARAMS: [{'MAX_FEATURES': 'SQRT', 'MIN_SAMPLES_SPLIT': 10}]

X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 2700) - (16453, 2700) - 65812 - 16453

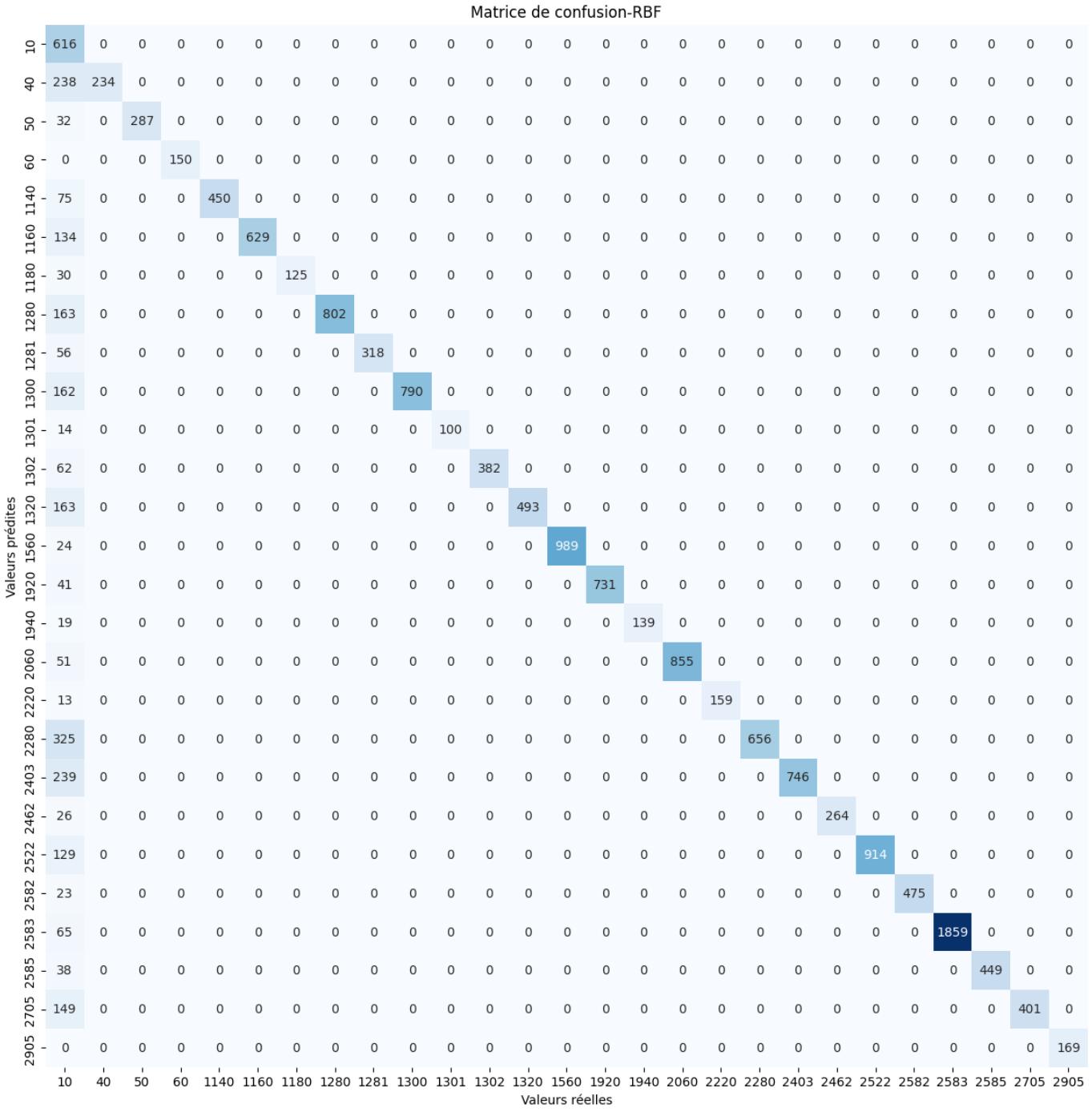
estimator RandomForestClassifier()

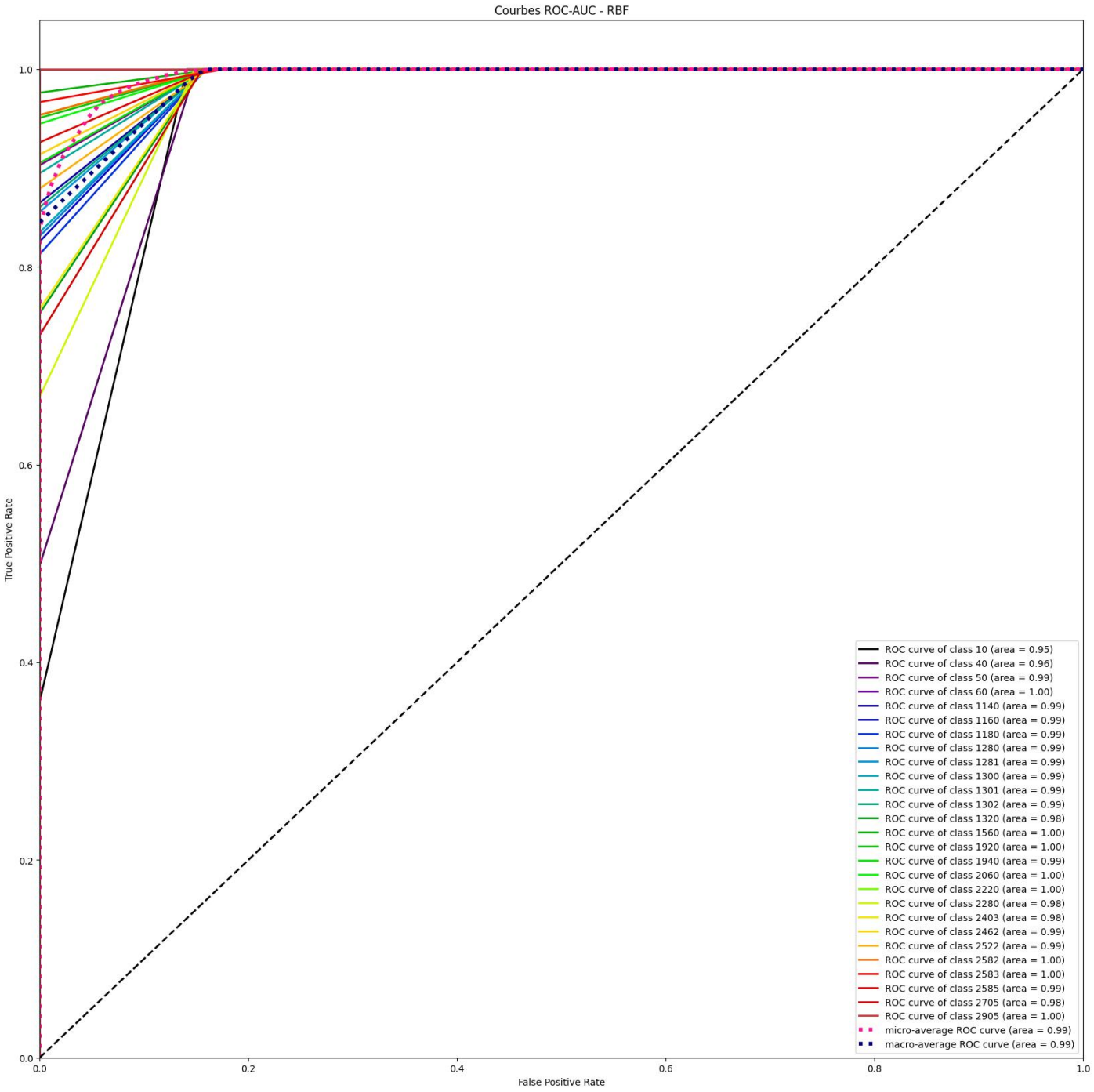
```
params    {'name': 'RBF', 'estimator': ensemble.RandomForestClassifier(), 'params': {'max_features': ["sqrt", None],
                'min_samples_split': [1, 10]}
            },
           {'name': 'SVC',  'estimator': svm.SVC(),
            'params': {'kernel':('linear', 'rbf'), 'C':[1, 10]}
            }
```

```
train_f1_score = [array([0.36168826, 0.66088117, 0.94627105, 0.99273608, 0.93363162,
       0.9073154 , 0.89071038, 0.9119452 , 0.91848373, 0.90918919,
       0.9622438 , 0.92756133, 0.87660327, 0.98651802, 0.98189068,
       0.94857143, 0.97431555, 0.96634615, 0.79063803, 0.85341426,
       0.96040987, 0.93725222, 0.98140127, 0.98680361, 0.96203209,
       0.83623877, 1.        ])]
test_f1_score = [array([0.35169854, 0.66288952, 0.94719472, 1.        , 0.92307692,
       0.90373563, 0.89285714, 0.90775325, 0.91907514, 0.90700344,
       0.93457944, 0.92493947, 0.85813751, 0.98801199, 0.97272122,
       0.93602694, 0.97103918, 0.96072508, 0.8014661 , 0.86192952,
       0.95306859, 0.93408278, 0.97636177, 0.98281787, 0.95940171,
       0.84332282, 1.        ])]
```

train_mse_result = 455162.75148909015

test_mse_result = 475895.7078344375


Matrice de confusion-RBF
Valeurs prédites
Valeurs réelles
10 40 50 60 1140 1160 1180 1280 1281 1300 1301 1302 1320 1560 1920 1940 2060 2220 2280 2403 2462 2522 2582 2583 2585 2705 2905
10 616 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
40 238 234 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
50 32 0 287 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
60 0 0 0 150 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1140 75 0 0 0 450 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1160 134 0 0 0 0 629 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1180 30 0 0 0 0 0 125 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1280 163 0 0 0 0 0 0 802 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1281 56 0 0 0 0 0 0 0 318 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1300 162 0 0 0 0 0 0 0 0 790 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1301 14 0 0 0 0 0 0 0 0 0 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1302 62 0 0 0 0 0 0 0 0 0 0 382 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1320 163 0 0 0 0 0 0 0 0 0 0 0 493 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1560 24 0 0 0 0 0 0 0 0 0 0 0 0 989 0 0 0 0 0 0 0 0 0 0 0 0 0
1920 41 0 0 0 0 0 0 0 0 0 0 0 0 0 731 0 0 0 0 0 0 0 0 0 0 0 0
1940 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 139 0 0 0 0 0 0 0 0 0 0 0
2060 51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 855 0 0 0 0 0 0 0 0 0 0
2220 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 159 0 0 0 0 0 0 0 0 0
2280 325 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 656 0 0 0 0 0 0 0 0
2403 239 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 746 0 0 0 0 0 0 0
2462 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 264 0 0 0 0 0 0
2522 129 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 914 0 0 0 0 0
2582 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 475 0 0 0 0
2583 65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1859 0 0 0
2585 38 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 449 0 0
2705 149 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 401 0
2905 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169


Courbes ROC-AUC - RBF
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 10 (area = 0.95)
ROC curve of class 40 (area = 0.96)
ROC curve of class 50 (area = 0.99)
ROC curve of class 60 (area = 1.00)
ROC curve of class 1140 (area = 0.99)
ROC curve of class 1160 (area = 0.99)
ROC curve of class 1180 (area = 0.99)
ROC curve of class 1280 (area = 0.99)
ROC curve of class 1281 (area = 0.99)
ROC curve of class 1300 (area = 0.99)
ROC curve of class 1301 (area = 0.99)
ROC curve of class 1302 (area = 0.99)
ROC curve of class 1320 (area = 0.98)
ROC curve of class 1560 (area = 1.00)
ROC curve of class 1920 (area = 1.00)
ROC curve of class 1940 (area = 0.99)
ROC curve of class 2060 (area = 1.00)
ROC curve of class 2220 (area = 1.00)
ROC curve of class 2280 (area = 0.98)
ROC curve of class 2403 (area = 0.98)
ROC curve of class 2462 (area = 0.99)
ROC curve of class 2522 (area = 0.99)
ROC curve of class 2582 (area = 1.00)
ROC curve of class 2583 (area = 1.00)
ROC curve of class 2585 (area = 0.99)
ROC curve of class 2705 (area = 0.98)
ROC curve of class 2905 (area = 1.00)
micro-average ROC curve (area = 0.99)
macro-average ROC curve (area = 0.99)

SVC (100 WORDS BY CODE)

TRAIN_R2_SCORE = 0.8660274721935209

TEST_R2_SCORE = 0.8574120221236249

BEST_PARAMS: [{'C': 10, 'KERNEL': 'LINEAR'}]

params {'kernel': ('linear', 'rbf'), 'C': [10, 20]}

X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 2700) - (16453, 2700) - 65812 - 16453

---

train_f1_score = [array([0.36168826, 0.66088117, 0.94627105, 0.99273608, 0.93363162,

0.9073154 , 0.89071038, 0.9119452 , 0.91848373, 0.90918919,

0.9622438 , 0.92756133, 0.87660327, 0.98651802, 0.98189068,

0.94857143, 0.97431555, 0.96634615, 0.79063803, 0.85341426,

0.96040987, 0.93725222, 0.98140127, 0.98680361, 0.96203209,

0.83623877, 1.])]

test_f1_score = [array([0.34432644, 0.66099291, 0.94719472, 0.99328859, 0.91975309,

0.90215827, 0.88489209, 0.9052751 , 0.91751085, 0.90574713,

0.90909091, 0.92363636, 0.85614647, 0.98293173, 0.96722408,

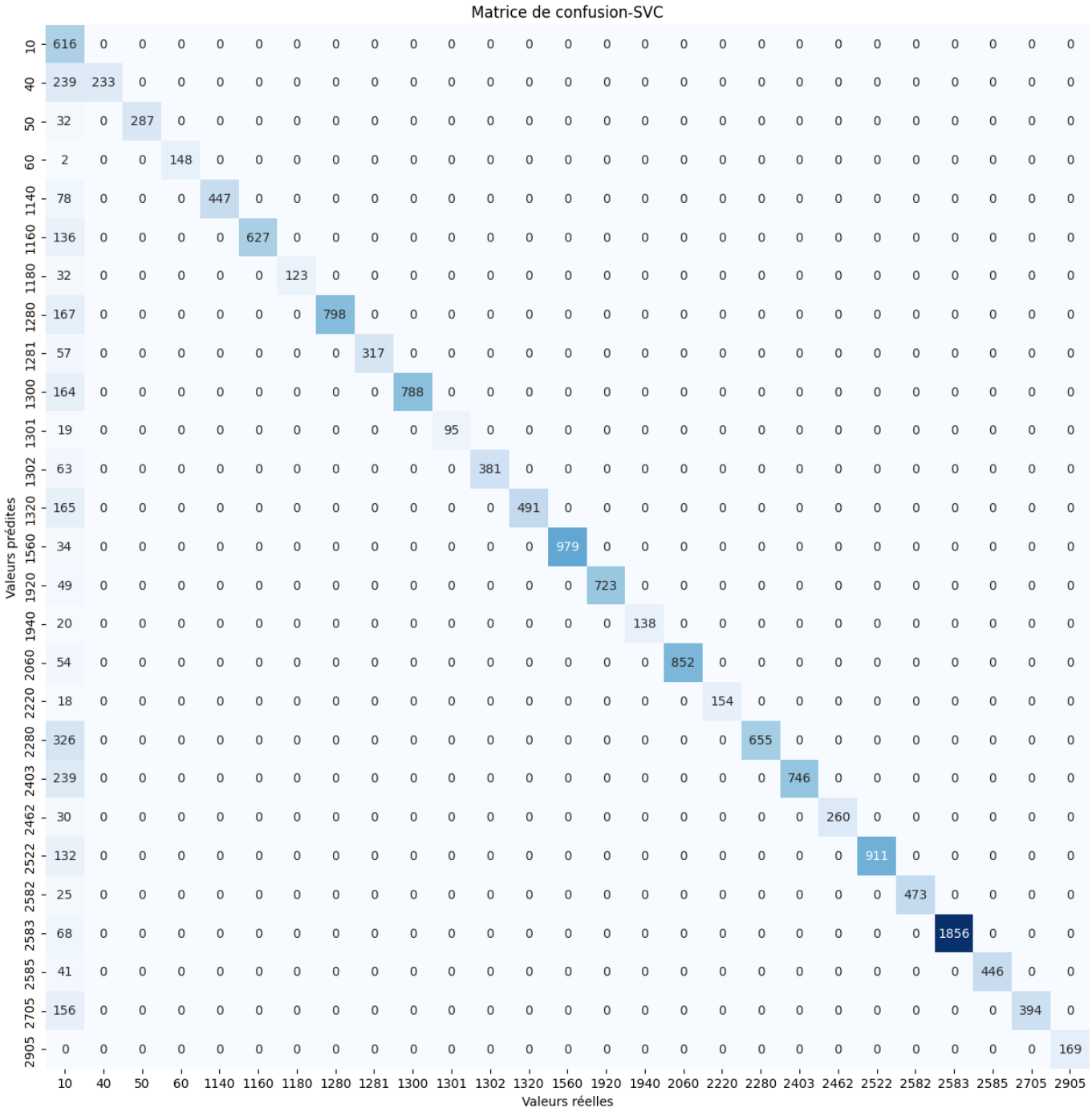
0.93243243, 0.96928328, 0.94478528, 0.8007335 , 0.86192952,

0.94545455, 0.93244626, 0.97425335, 0.98201058, 0.95605573,

0.83474576, 1.])]

train_mse_result = 455162.75148909015

test_mse_result = 492912.0065641524


Matrice de confusion-SVC
Valeurs prédites
Valeurs réelles
10 40 50 60 1140 1160 1180 1280 1281 1300 1301 1302 1320 1560 1920 1940 2060 2220 2280 2403 2462 2522 2582 2583 2585 2705 2905
10 616 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
40 239 233 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
50 32 0 287 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
60 2 0 0 148 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1140 78 0 0 0 447 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1160 136 0 0 0 0 627 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1180 32 0 0 0 0 0 123 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1280 167 0 0 0 0 0 0 798 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1281 57 0 0 0 0 0 0 0 317 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1300 164 0 0 0 0 0 0 0 0 788 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1301 19 0 0 0 0 0 0 0 0 0 95 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1302 63 0 0 0 0 0 0 0 0 0 0 381 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1320 165 0 0 0 0 0 0 0 0 0 0 0 491 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1560 34 0 0 0 0 0 0 0 0 0 0 0 0 979 0 0 0 0 0 0 0 0 0 0 0 0 0
1920 49 0 0 0 0 0 0 0 0 0 0 0 0 0 723 0 0 0 0 0 0 0 0 0 0 0 0
1940 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 138 0 0 0 0 0 0 0 0 0 0 0
2060 54 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 852 0 0 0 0 0 0 0 0 0 0
2220 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 154 0 0 0 0 0 0 0 0 0
2280 326 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 655 0 0 0 0 0 0 0 0
2403 239 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 746 0 0 0 0 0 0 0
2462 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 260 0 0 0 0 0 0
2522 132 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 911 0 0 0 0 0
2582 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 473 0 0 0 0
2583 68 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1856 0 0 0
2585 41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 446 0 0
2705 156 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 394 0
2905 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169

RFC - RANDOMFORESTCLASSIFIER (300 WORDS BY CODE) – THE BEST

TRAIN_R2_SCORE = 0.9228408193034705

TEST_R2_SCORE = 0.9139974472740534

BEST_PARAMS: [{'MAX_FEATURES': 'SQRT', 'MIN_SAMPLES_SPLIT': 10}]

X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 8100) - (16453, 8100) - 65812 - 16453

estimator RandomForestClassifier()

params {'max_features': ['sqrt'], 'min_samples_split'...

Fitting 3 folds for each of 1 candidates, totalling 3 fits

train_f1_score = [array([0.49593012, 0.79447115, 0.98415153, 0.99516908, 0.96825397,

0.95305318, 0.92307692, 0.94928335, 0.95154472, 0.93677555,

0.9894958 , 0.95852018, 0.93389297, 0.99550302, 0.99721813,

0.98020586, 0.98858892, 0.98505114, 0.89335485, 0.9163918 ,

0.98128708, 0.97629708, 0.99320071, 0.99503514, 0.97842105,

0.87859506, 1.])]

test_f1_score = [array([0.46543257, 0.76923077, 0.9775641 , 1. , 0.95626243,

0.94248094, 0.9122807 , 0.94362343, 0.94200849, 0.93452714,

0.96832579, 0.94911243, 0.92016461, 0.99454094, 0.99413681,

0.96052632, 0.98430493, 0.97005988, 0.90145577, 0.91519824,

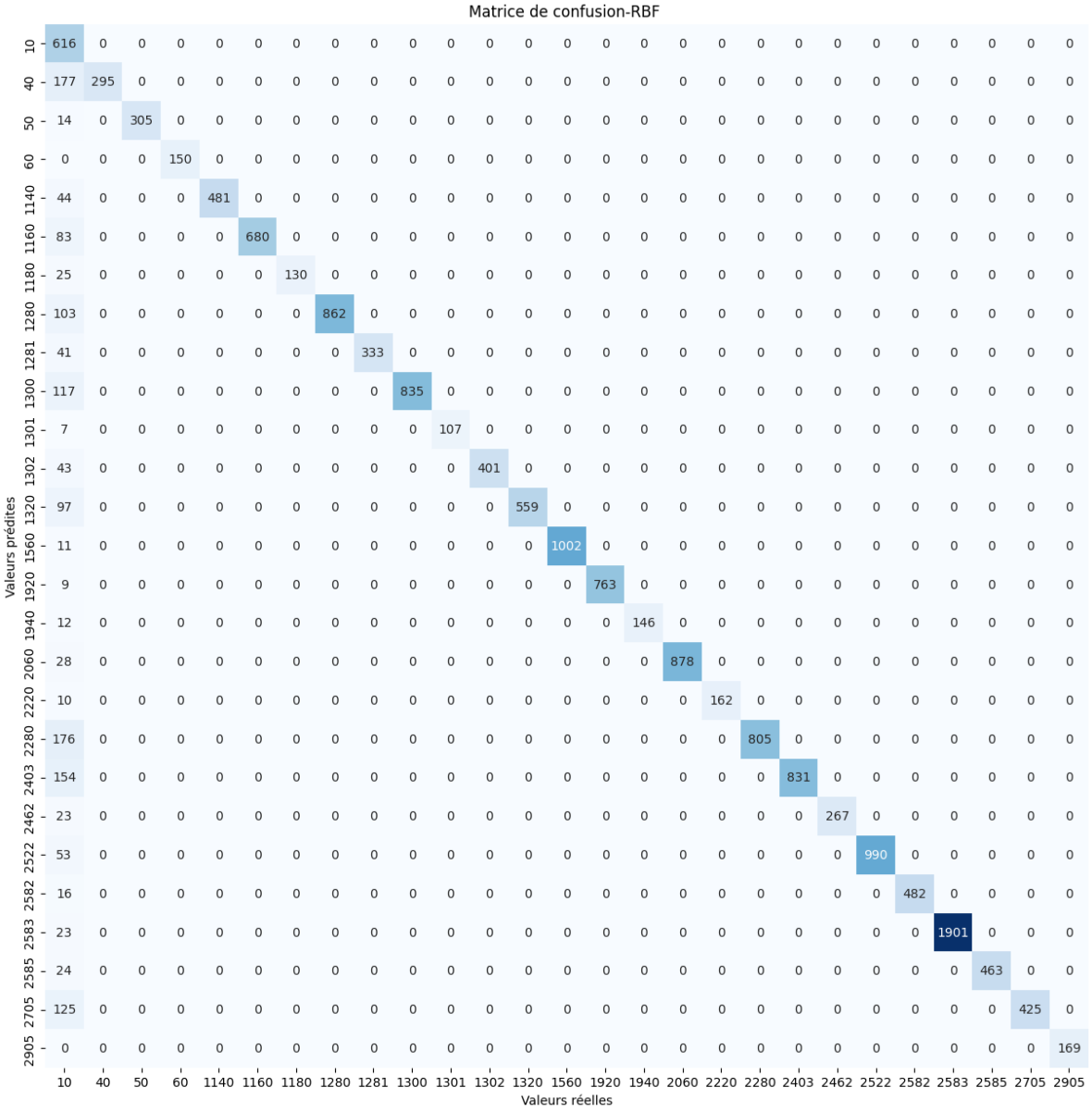
0.95870736, 0.97393015, 0.98367347, 0.99398693, 0.97473684,

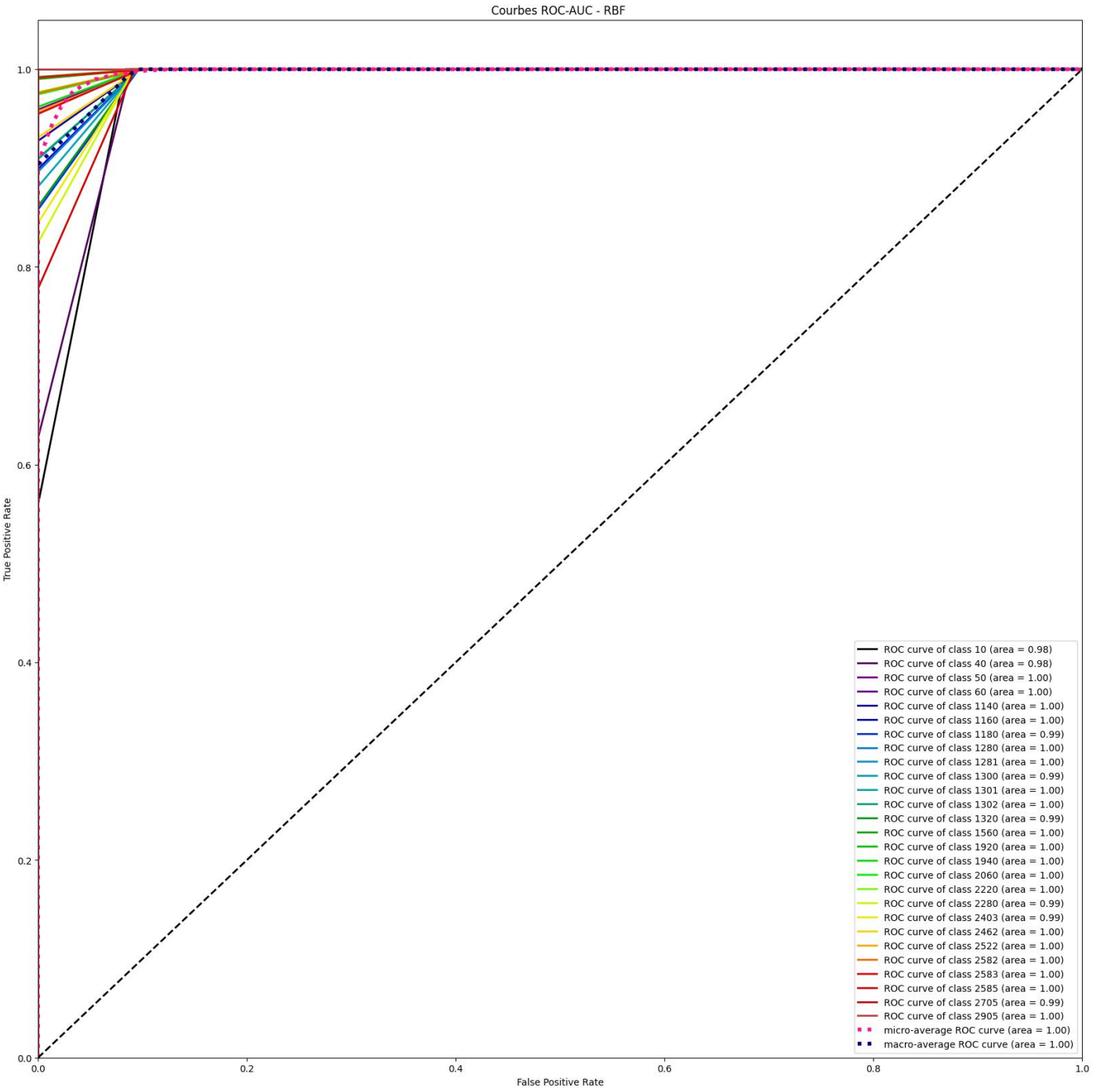
0.87179487, 1.])]

train_mse_result = 257461.71090378653

test_mse_result = 287754.30657023034

==========CONFUSION MATRIX==========


Matrice de confusion-RBF
Valeurs prédites
Valeurs réelles
10 40 50 60 1140 1160 1180 1280 1281 1300 1301 1302 1320 1560 1920 1940 2060 2220 2280 2403 2462 2522 2582 2583 2585 2705 2905


Courbes ROC-AUC - RBF
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 10 (area = 0.98)
ROC curve of class 40 (area = 0.98)
ROC curve of class 50 (area = 1.00)
ROC curve of class 60 (area = 1.00)
ROC curve of class 1140 (area = 1.00)
ROC curve of class 1160 (area = 1.00)
ROC curve of class 1180 (area = 0.99)
ROC curve of class 1280 (area = 1.00)
ROC curve of class 1281 (area = 1.00)
ROC curve of class 1300 (area = 0.99)
ROC curve of class 1301 (area = 1.00)
ROC curve of class 1302 (area = 1.00)
ROC curve of class 1320 (area = 0.99)
ROC curve of class 1560 (area = 1.00)
ROC curve of class 1920 (area = 1.00)
ROC curve of class 1940 (area = 1.00)
ROC curve of class 2060 (area = 1.00)
ROC curve of class 2220 (area = 1.00)
ROC curve of class 2280 (area = 0.99)
ROC curve of class 2403 (area = 0.99)
ROC curve of class 2462 (area = 1.00)
ROC curve of class 2522 (area = 1.00)
ROC curve of class 2582 (area = 1.00)
ROC curve of class 2583 (area = 1.00)
ROC curve of class 2585 (area = 1.00)
ROC curve of class 2705 (area = 0.99)
ROC curve of class 2905 (area = 1.00)
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)

LREG (100 WORDS BY CODE) – 4MIN

BEST_PARAMS: [{'C': 30}]

TRAIN_R2_SCORE = 0.8658603294232055

TEST_R2_SCORE = 0.8622135780708685

X_train.shape - X_test.shape - len(y_train) - len(y_test)

(65812, 2700) - (16453, 2700) - 65812 - 16453

estimator LogisticRegression()

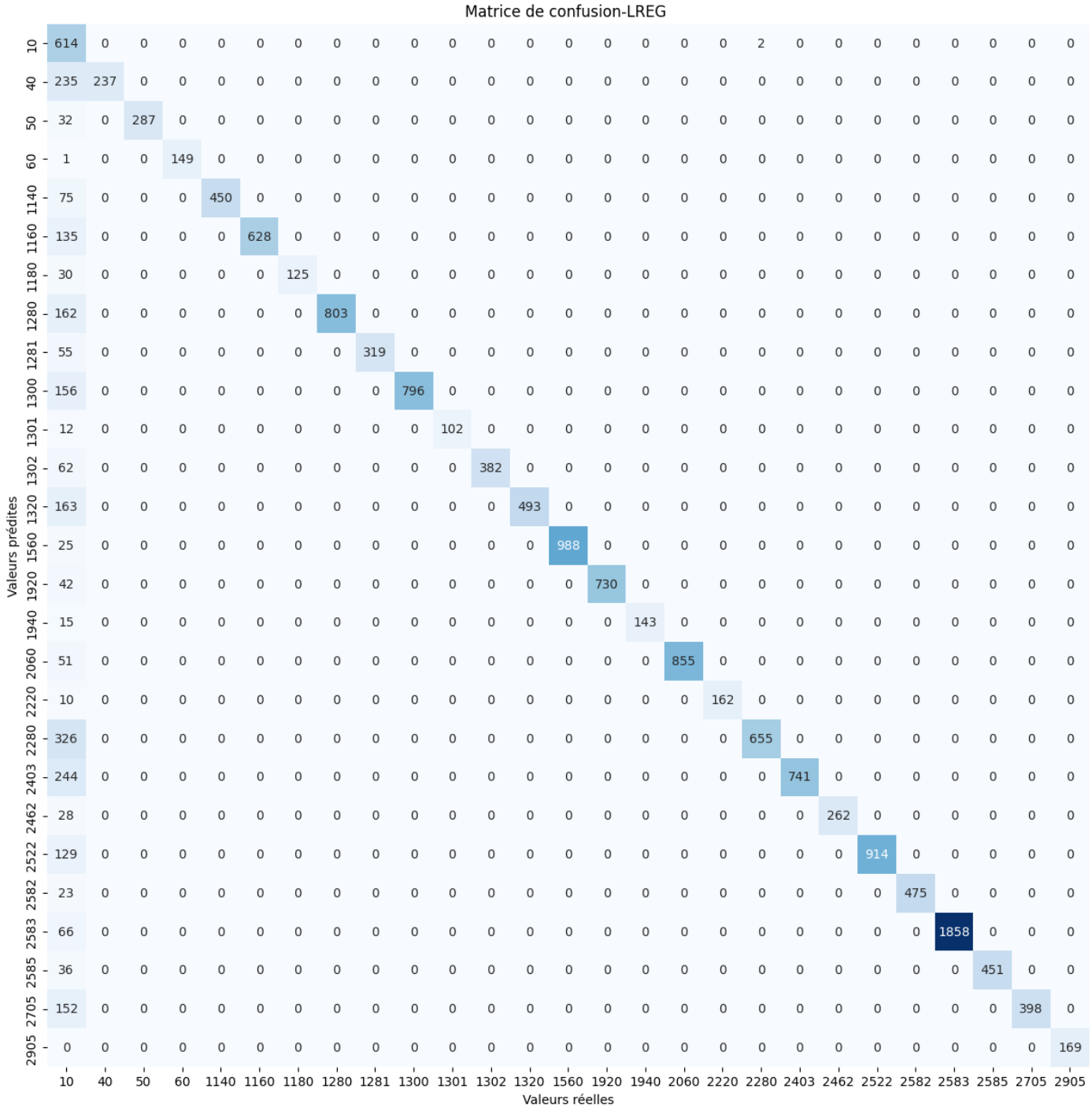
params {'C': [5, 10, 20]}

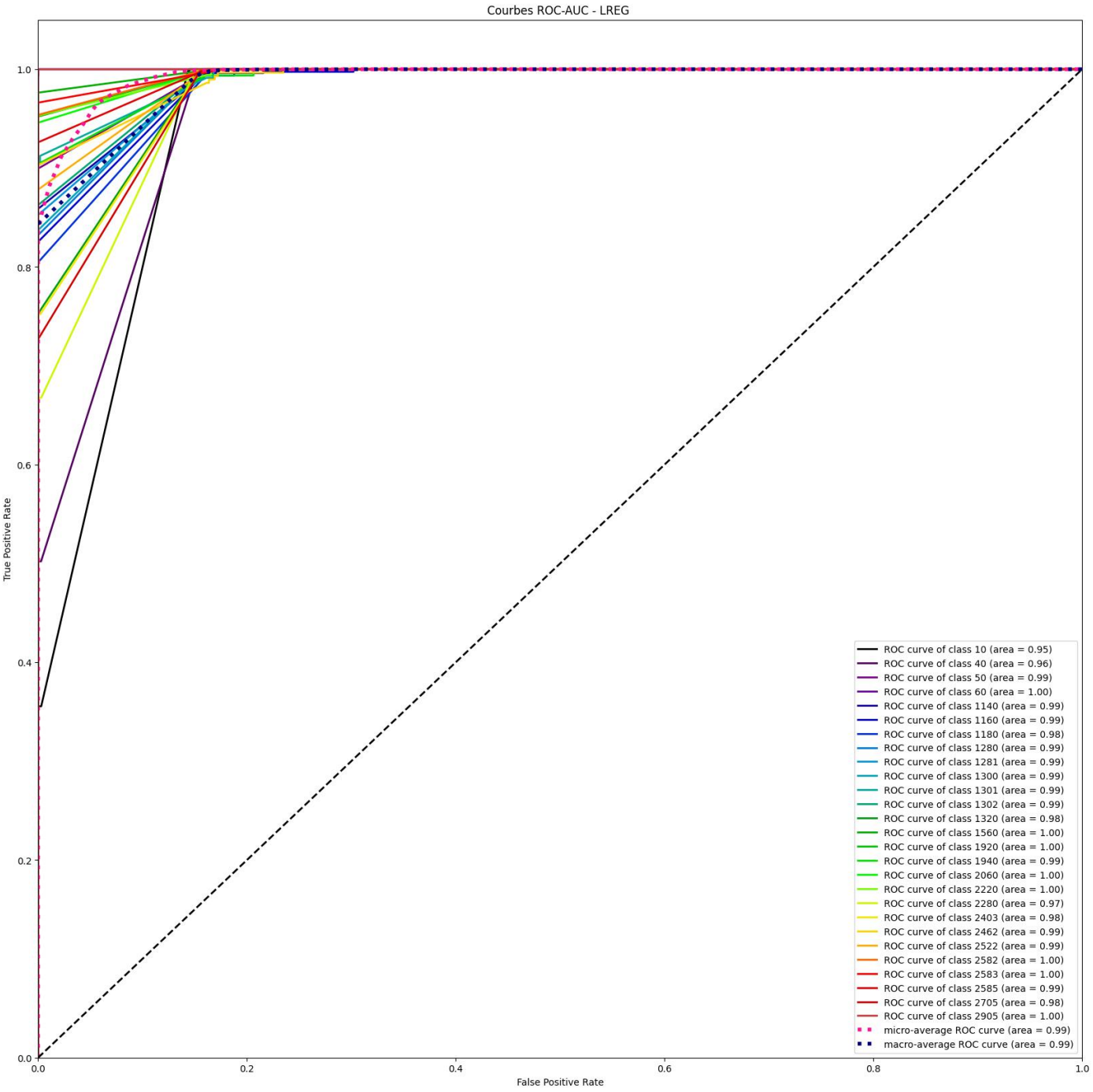
train_f1_score = [array([0.36140046, 0.66355763, 0.94627105, 0.99273608, 0.93363162,
0.9073154 , 0.89071038, 0.9119452 , 0.91848373, 0.90918919,
0.9622438 , 0.92756133, 0.87660327, 0.98651802, 0.98189068,
0.95114007, 0.97431555, 0.96634615, 0.79063803, 0.85167173,
0.96040987, 0.93652531, 0.98114169, 0.98680361, 0.96119882,
0.83593131, 1.])]

test_f1_score = [array([0.35135908, 0.66854725, 0.94719472, 0.99665552, 0.92307692,
0.90294752, 0.89285714, 0.90837104, 0.92063492, 0.91075515,
0.94444444, 0.92493947, 0.85813751, 0.98750625, 0.97203728,
0.95016611, 0.97103918, 0.97005988, 0.7997558 , 0.85863268,
0.94927536, 0.93408278, 0.97636177, 0.98254892, 0.96162047,
0.83966245, 1.])]

train_mse_result = 456855.5308302437

test_mse_result = 477873.36564760224




Courbes ROC-AUC - LREG
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 10 (area = 0.95)
ROC curve of class 40 (area = 0.96)
ROC curve of class 50 (area = 0.99)
ROC curve of class 60 (area = 1.00)
ROC curve of class 1140 (area = 0.99)
ROC curve of class 1160 (area = 0.99)
ROC curve of class 1180 (area = 0.98)
ROC curve of class 1280 (area = 0.99)
ROC curve of class 1281 (area = 0.99)
ROC curve of class 1300 (area = 0.99)
ROC curve of class 1301 (area = 0.99)
ROC curve of class 1302 (area = 0.99)
ROC curve of class 1320 (area = 0.98)
ROC curve of class 1560 (area = 1.00)
ROC curve of class 1920 (area = 1.00)
ROC curve of class 1940 (area = 0.99)
ROC curve of class 2060 (area = 1.00)
ROC curve of class 2220 (area = 1.00)
ROC curve of class 2280 (area = 0.97)
ROC curve of class 2403 (area = 0.98)
ROC curve of class 2462 (area = 0.99)
ROC curve of class 2522 (area = 0.99)
ROC curve of class 2582 (area = 1.00)
ROC curve of class 2583 (area = 1.00)
ROC curve of class 2585 (area = 0.99)
ROC curve of class 2705 (area = 0.98)
ROC curve of class 2905 (area = 1.00)
micro-average ROC curve (area = 0.99)
macro-average ROC curve (area = 0.99)