# 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 :

Une image contenant texte, capture d’écran, ligne, Rectangle

Description générée automatiquement

# 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 :

Une image contenant texte, capture d’écran, ligne, Rectangle

Description générée automatiquement

# 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 :

Une image contenant texte, capture d’écran, ligne, Rectangle

Description générée automatiquement

# 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 :

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

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}]

0

# 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 :

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

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}]

# SVC (20 words by code)

estimator SVC()

params {'C': [10, 20, 30, 50, 100], 'kernel': ['linea...

-----------------------------------------------------------------------

Fitting 3 folds for each of 15 candidates, totalling 45 fits…