12/08/2020

MODELING

Team 6:
Oguzhan Akan
Darius Hooks
Jaymish Raju Patel
Jason Sabal
Bibinur Zhursinbek

California State University, Northridge
COMP 541 – Data Mining – F2020
Assignment 6

Contents

1. Selected Modeling Algorithms & Performance Metrics

Tools/Language

- Python

Algorithms

- K-Nearest Neighbors: Fast to implement and interpret
- Random Forest: Ensemble decision trees that decides according to a voting algorithm
- Gradient Boosting: Ensemble decision trees that reduces its error in every other iteration

Performance Metrics

- Classification: Accuracy, Sensitivity, Precision - F1 Score
- Regression - MedianAE, MeanAE, RMSE

2. Modeling Strategy

- Train-Test Data Split:
  - We decided to split our training and test data by 80%-20% as we wanted to have enough data for objective evaluation on both train and test sets.
  - The X_train and X_test is the same for both classification and regression models. We achieved this by setting a random state for each split.
  - Splitting made by using Sklearn's train_test_split function.

```
X_train, X_test, y1_train, y1_test = train_test_split(X, y1, test_size=0.2, random_state=17)
X_train, X_test, y2_train, y2_test = train_test_split(X, y2, test_size=0.2, random_state=17)
```

- K-Fold Cross Validation
  - We applied 3-fold stratified split to our training dataset in order to overcome possible overfitting issue:

```
# This is where we create our folds
kf = KFold(n_splits=3, random_state=17, shuffle=True)
kf.get_n_splits(X_train)
fold_indexes = {}
i = 1
for train_index, valid_index in kf.split(X_train):
    fold_indexes[i]={}
    fold_indexes[i]['train']=train_index
    fold_indexes[i]['valid']=valid_index
    i+=1
```

3. MODEL PARAMETERS & RESULTS

CLASSIFICATION MODELS

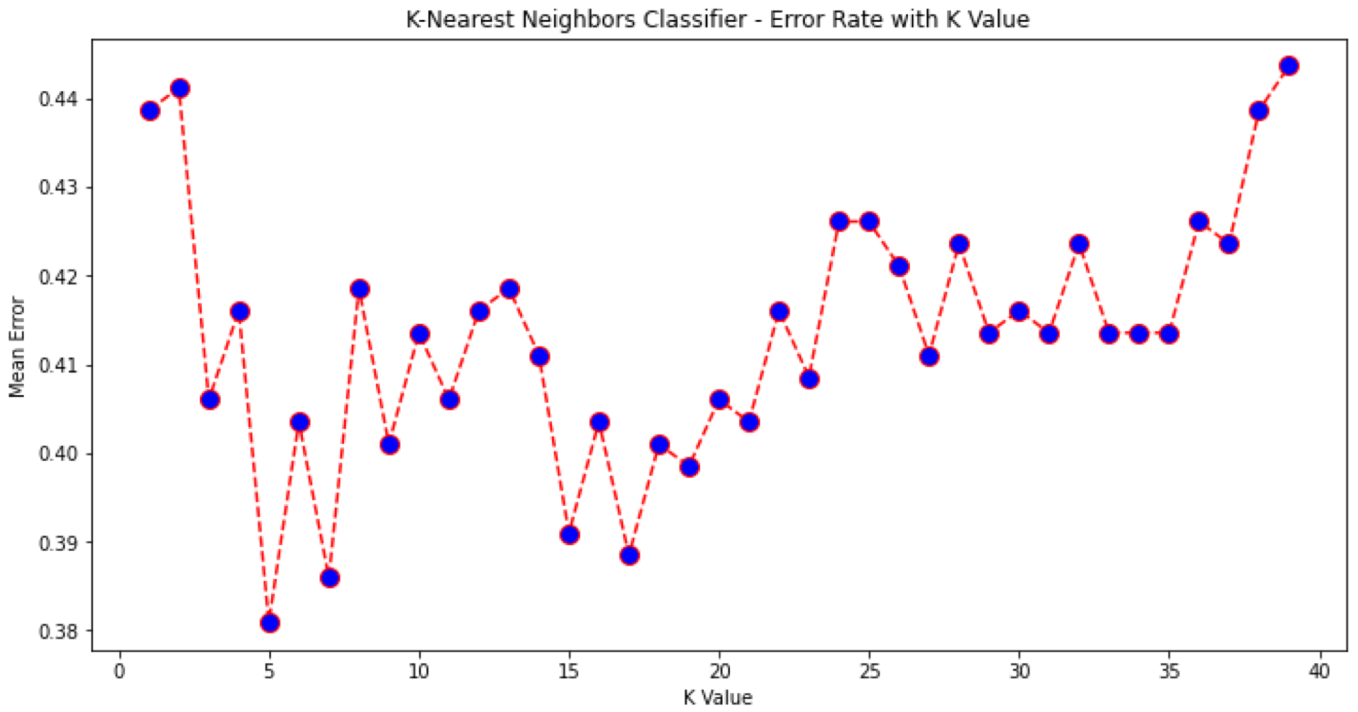
Modeling - Classification Model - **KNN**

Parameter setting:

1. n_neighbors (k) = 2, …, 20
2. weights = ‘uniform’

Best value of k → 5

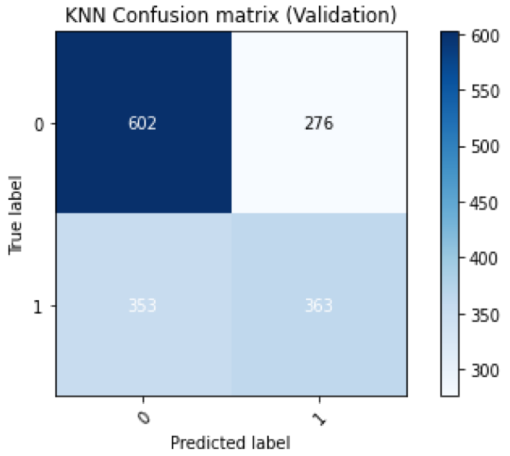
Note that both very-low and very-high number of neighbors result in high error rate. Very-low neighbors cause underfitting and very-high neighbors cause overfitting.



Modeling - Classification Model - **KNN**

Once we decided that n_neighbors = 5, we ran the 3-fold CV and built confusion matrices for validation and test data separately.

As can be seen from the figures on the right, KNN model reached ~0.6 accuracy and ~0.5 F-1 score in validation and test data.

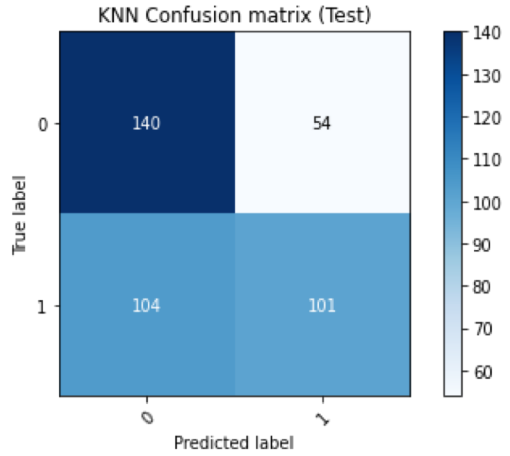


Accuracy: **0.605**

Sensitivity : **0.507**

Precision : **0.568**

F-1 Score : **0.536**



Accuracy: **0.604**

Sensitivity : **0.493**

Precision : **0.652**

F-1 Score : **0.561**

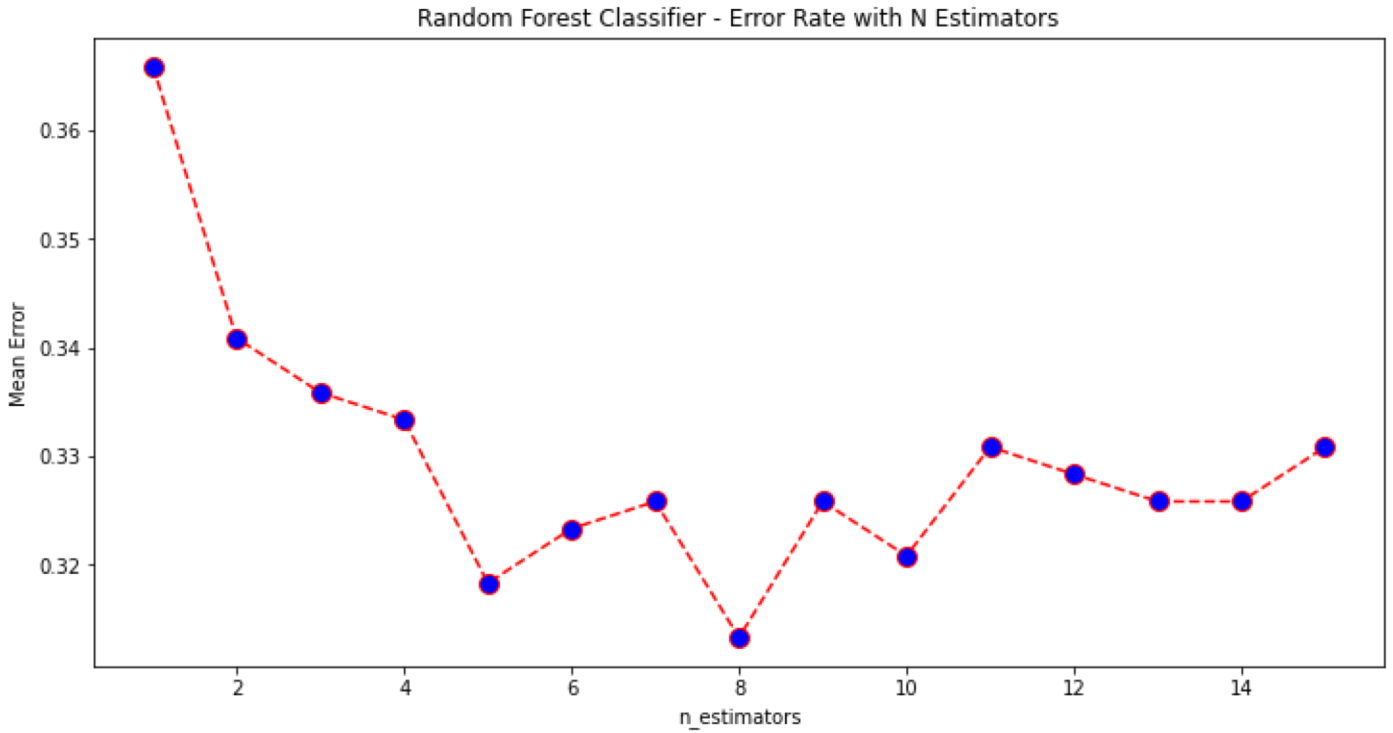
Modeling - Classification Model - **Random Forest**

Parameter setting:

1. n_estimators = 1, …, 15
2. min_samples_split = 60
3. min_samples_leaf = 20
4. max_depth = 7
5. max_leaf_nodes = 14

Best value of n_estimators → 8

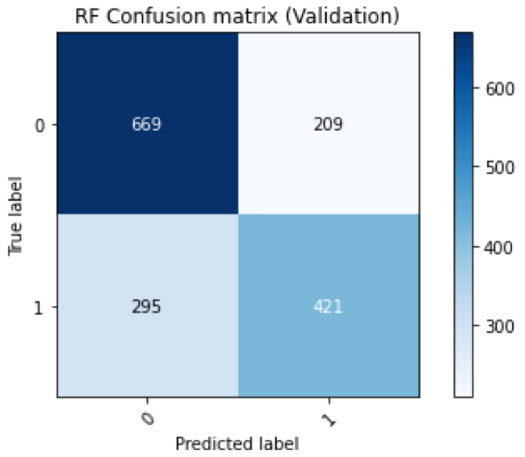
Again, using a lot of trees may increase the error rate by causing overfitting.



Modeling - Classification Model - **Random Forest**

Once we decided that n_estimators = 8, we ran the 3-fold CV and built confusion matrices for validation and test data separately.

As can be seen from the figures on the right, Random Forest model reached ~0.68 accuracy and ~0.65 F-1 score in validation and test data.

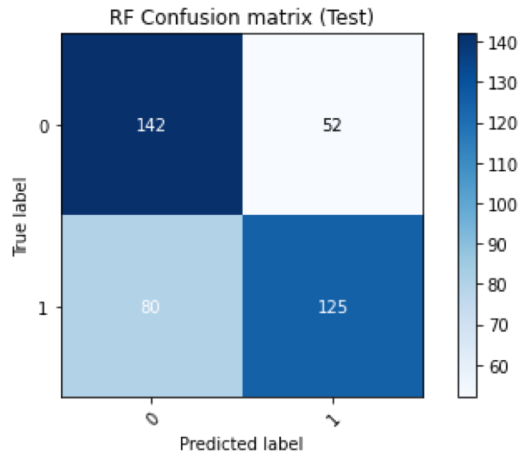


Accuracy: **0.684**

Sensitivity : **0.588**

Precision : **0.668**

F-1 Score : **0.626**



Accuracy: **0.669**

Sensitivity : **0.610**

Precision : **0.706**

F-1 Score : **0.654**

Modeling - Classification Model - **Gradient Boosting**

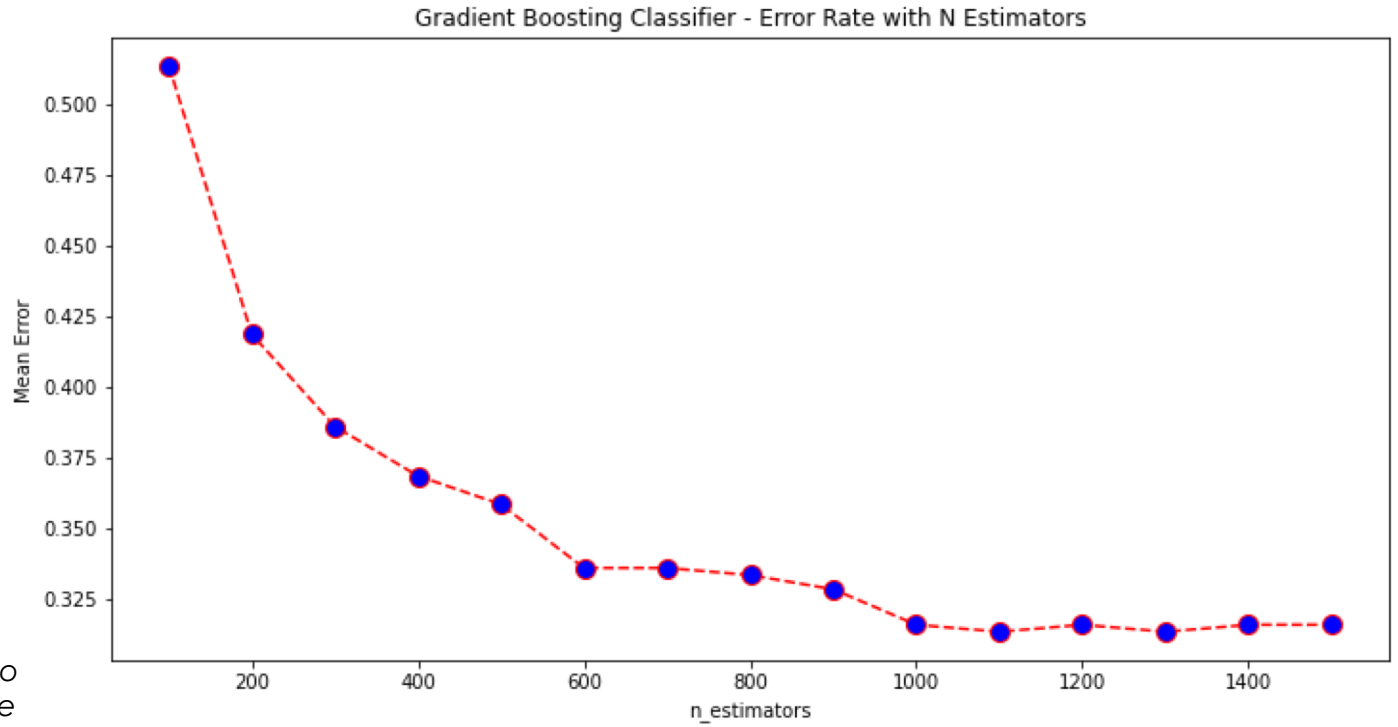
Parameter setting:

1. n_estimators=100, 200, …, 1500
2. learning_rate=0.001
3. criterion='friedman_mse'
4. min_samples_split=60
5. min_samples_leaf=20
6. max_depth=7
7. max_leaf_nodes=14

Best value of n_estimators → 1000

Note that when n_estimators >= 1000, the improvement is negligible.

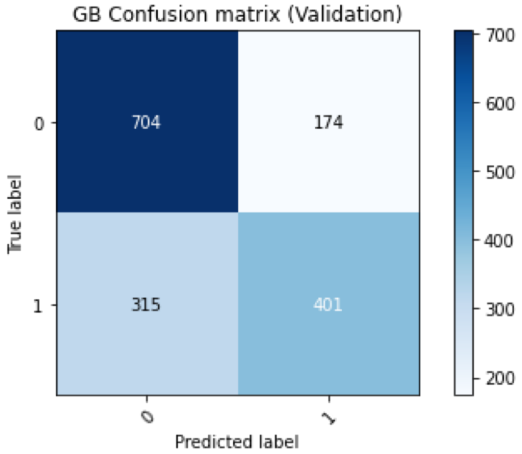
The overfitting problem observed in KNN and Random Forest is non-existent for Gradient Boosting algorithm, because Gradient Boosting works in a way that improves its errors. Thus, we do not expect the error rate to increase as we iterate over n_estimators.

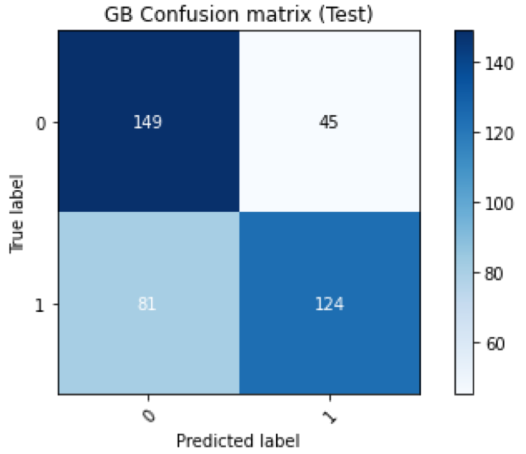


Modeling - Classification Model - **Gradient Boosting**

Once we decided that n_estimators = 1000, we ran the 3-fold CV and built confusion matrices for validation and test data separately.

As can be seen from the figures on the right, KNN model reached ~0.69 accuracy and ~0.65 F-1 score in validation and test data.





Accuracy: **0.693**

Sensitivity : **0.560**

Precision : **0.697**

F-1 Score : **0.621**

Accuracy: **0.684**

Sensitivity : **0.605**

Precision : **0.734**

F-1 Score : **0.663**

3. MODEL PARAMETERS & RESULTS

REGRESSION MODELS

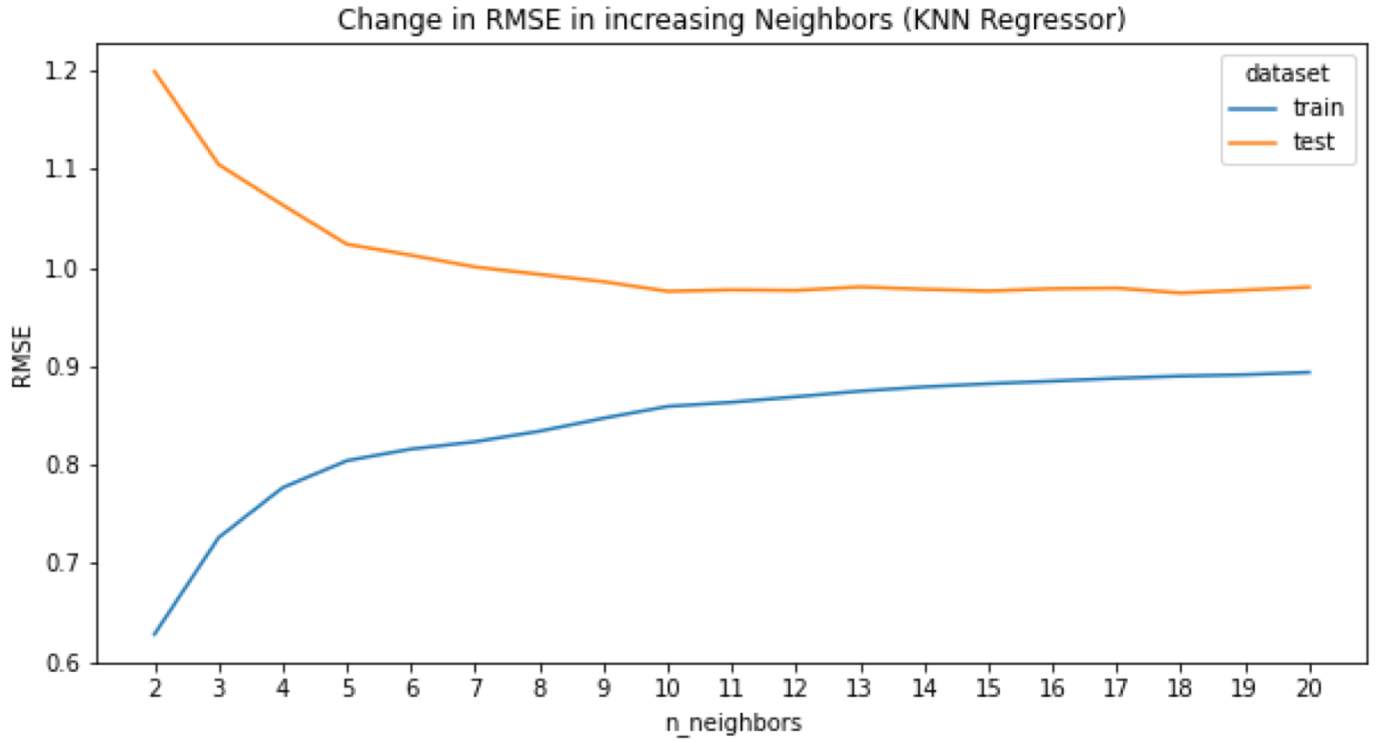
Modeling - Regression Model - **KNN Regression**

Parameter setting:

1. n_neighbors (k) = 2, …, 20

Best value of k → 18

This regression model can be considered as low-bias and average-error model. If only the train & test lines came closer, it would be almost perfect!



At k = 18;

[TRAIN] >> median_abs_err = 0.546, mean_abs_err = 0.678, **RMSE = 0.890**

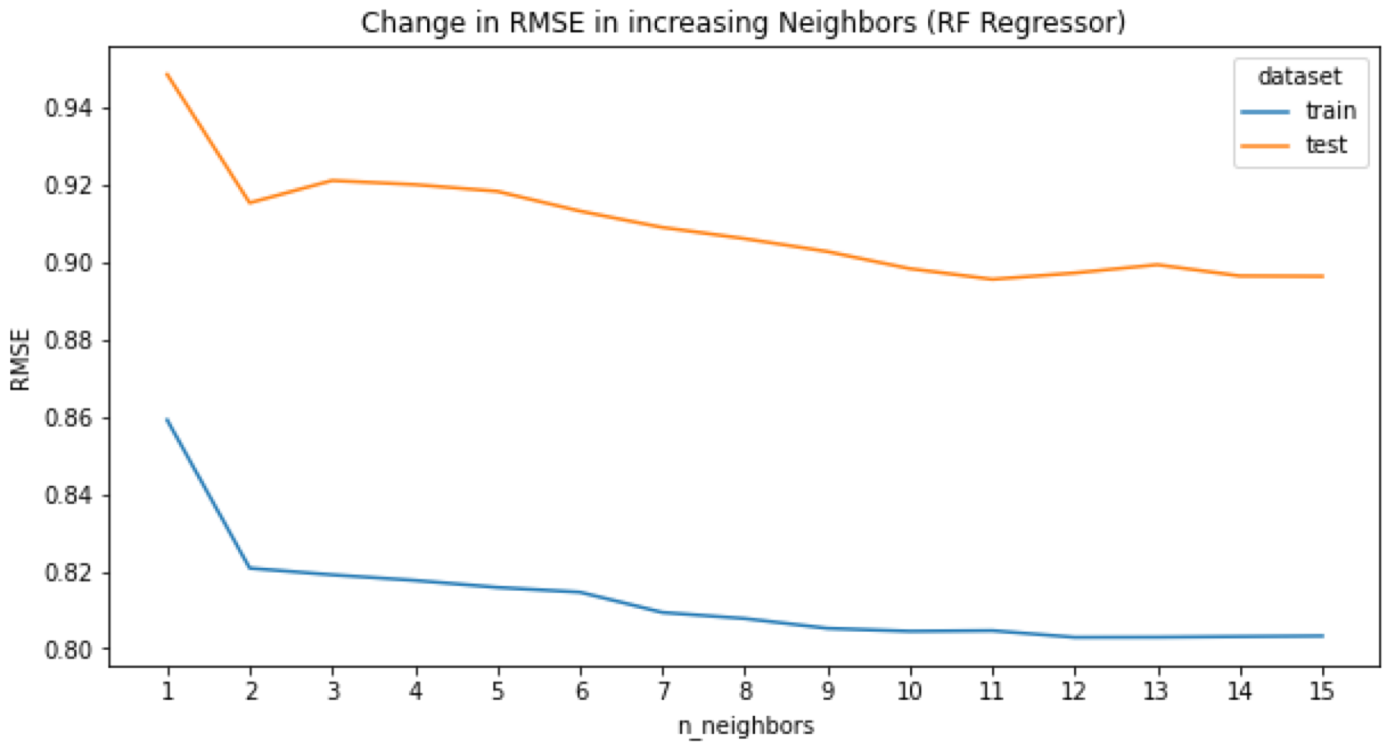
[TEST] >> median_abs_err = 0.597, mean_abs_err = 0.742, **RMSE = 0.974**

Modeling - Regression Model - **Random Forest**

Parameter setting:

1. n_estimators = 1, …, 15
2. min_samples_split = 60
3. min_samples_leaf = 20
4. max_depth = 7
5. max_leaf_nodes = 14

Best value of n_estimators → 15



At n_estimators = 15;

[TRAIN] >> median_abs_err = 0.474, mean_abs_err = 0.605, **RMSE = 0.803**

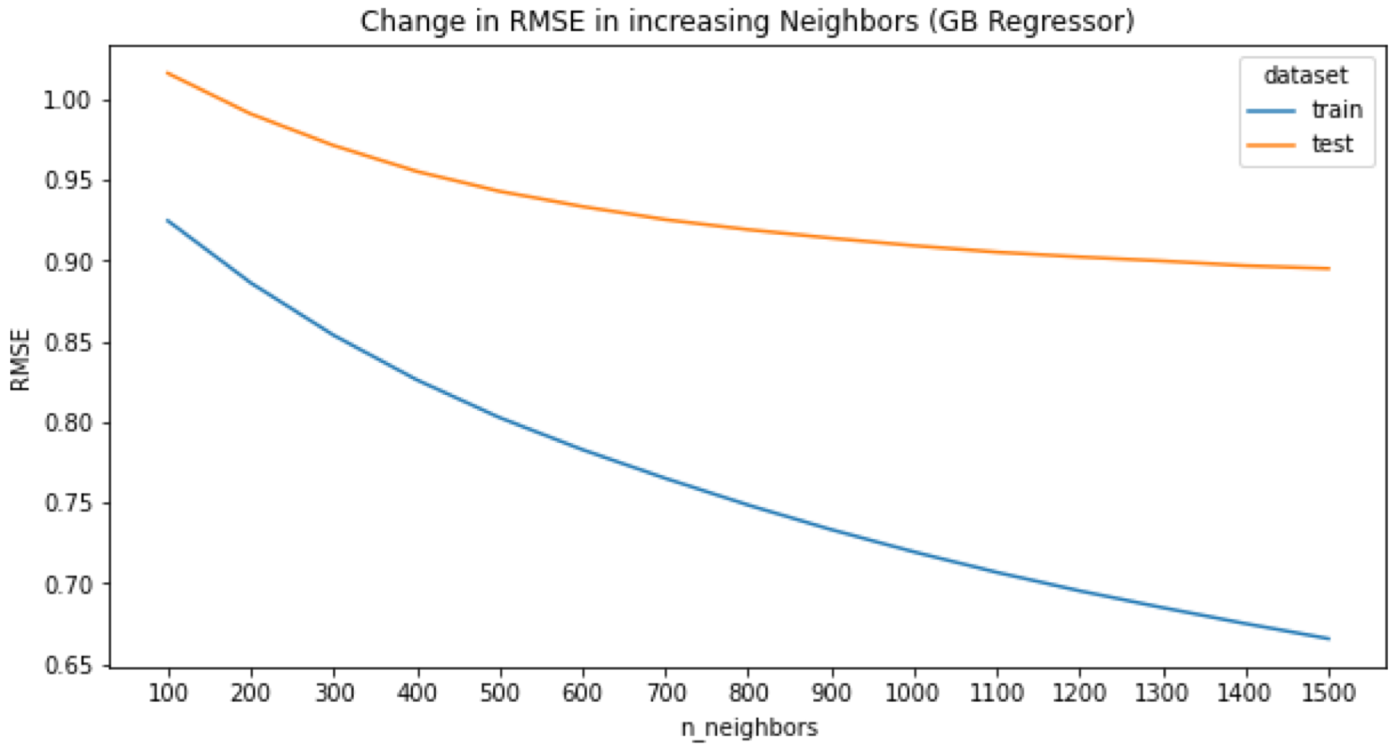
[TEST] >> median_abs_err = 0.528, mean_abs_err = 0.667, **RMSE = 0.896**

Modeling - Regression Model - **Gradient Boosting**

Parameter setting:

1. n_estimators=100, 200, …, 1500
2. learning_rate=0.001
3. criterion='friedman_mse'
4. min_samples_split=60
5. min_samples_leaf=20
6. max_depth=7
7. max_leaf_nodes=14

Best value of n_estimators → 1500



At n_estimators = 1500;

[TRAIN] >> median_abs_err = 0.406, mean_abs_err = 0.508, **RMSE = 0.666**

[TEST] >> median_abs_err = 0.488, mean_abs_err = 0.652, **RMSE = 0.895**

4. MODEL COMPARISONS

Evaluation - **Classification Models**

Model Ranking according to F-1 Score: Gradient Boosting > Random Forest > KNN

	Validation				Test			
	Accuracy	Sensitivity	Precision	**F-1 Score**	Accuracy	Sensitivity	Precision	**F-1 Score**
KNN	0.605	0.507	0.568	0.536	0.604	0.493	0.652	0.561
Random Forest	0.684	0.588	0.668	0.626	0.669	0.610	0.706	0.654
Gradient Boosting	0.693	0.560	0.697	0.621	0.684	0.605	0.734	0.663
	Overall, how often is the classifier correct?	*When it's actually yes, how often does it predict yes?*	*When it predicts yes, how often is it correct?*	*Harmonic mean of sensitivity and precision*	*Overall, how often is the classifier correct?*	*When it's actually yes, how often does it predict yes?*	*When it predicts yes, how often is it correct?*	*Harmonic mean of sensitivity and precision*

Evaluation - **Regression Models**

Model Ranking according to RMSE: Gradient Boosting ~= Random Forest > KNN

	Validation			Test		
	Median Abs. Err.	Mean Abs. Err.	**RMSE**	Median Abs. Err.	Mean Abs. Err.	**RMSE**
KNN	0.546	0.678	0.890	0.597	0.742	0.974
Random Forest	0.474	0.605	0.803	0.528	0.667	0.896
Gradient Boosting	0.406	0.508	0.666	0.488	0.652	0.895
	Midpoint of the sorted absolute errors	*Average of the absolute errors*	*Root Mean Squared Error*	*Midpoint of the sorted absolute errors*	*Average of the absolute errors*	*Root Mean Squared Error*

Conclusion

- All of our classification models reached to our initial F-1 score target which was 0.4. Actually, the Random Forest and Gradient Boosting models reached to 0.65 which is significantly good.
- Besides, the difference between Validation and Test data metrics is very low, indicating a non-overfitting conclusion.
- Our Regression models also performed really promising as they all had a RMSE of less than 1.
- Nonetheless, a movie success depends on a lot of features that are related to the movies, situation in the country, and so on.
- The movie profitability prediction helps movie production companies to invest in the movie projects.