Data Science Practicum Report : Seafood Appearances on Historical Menus

Alan Nurcahyo[*]

2020-12-01

Abstract

The New York Public Library has compiled a database of menu items dating back to the 1840s. The goal of this project is to develop an automated approach for identifying and categorizing seafood dishes The model that we use to predict unlabeled dataset is Random forest with ntree= 500 and m=500. The model has 98.21 percent accuracy, 98.38 percent sensitivity, and 97.14 specificity on test set. we find that Oyster was a popular food for almost 100 years, Lobster isn't popular seafood dishes until mid 19th century, and Shrimp steadily increased in popularity.

1. Project Overview

1.1 Project Name

Seafood Appearances on Historical Menus

1.2 Client Information

Jessica Gephart, Seafood Globalization Lab. Email: jgephart@american.edu

1.3 Project start-end

Monday, August 24, 2020 - Friday, December 4, 2020

1.4 Project description and Prerequisities

The New York Public Library has compiled a database of menu items dating back to the 1840s. The goal of this project is to develop an automated approach for identifying and categorizing seafood dishes. The resulting categorization will be used to understand the change in seafood diversity and sourcing over time within this menu collection. Seafood Globalization Lab has labeled 38,491 dishes from the New York Public Library database as seafood and nonseafood, which might be used as training and testing datasets for machine learning techniques.

Student need to understand basic R and able to implement a machine learning method.

[*]American University; an4234a@student.american.edu

1.5 Student Information

Alan Nurcahyo, Graduate Student in Master Of Data Science Program, American University.

1.6 Student's Objective

The student is required to build off an initial training dataset and apply machine learning techniques to identify and categorize menu items into seafood or nonseafood dishes. The student will use either the labeled dataset for analysis. The result is a machine learning model capable of identifying and classifying dishes into seafood dishes and nonseafood dishes from New York Public Library database. The student is expected to present and validate different models using general classifier methods, including (but not limited to) Logistic Regression, Support Vector Machine, K-nearest neighbour etc. The choosen model is expected to be robust, capable of predicting new observations when the database is updated. All work will be written in R.

1.7 Student's Task

1. Preprocessing data for training and testing data, using a labeled dataset provided by the client.
2. Analyze the power of a sample from the labeled dataset.
3. Apply machine learning algorithm for identifying and categorizing menu items.
4. Create a machine learning model from the algorithm.
5. Tune model and validate results using test dataset.
6. Summarize and visualize findings.
7. Create a report based on findings.

2 NYPL Whats on the menu? datasets

The New York Public Library's menu collection, housed in the Rare Book Division, originated through the energetic efforts of Miss Frank E. Buttolph (1850-1924), who, in 1900, began to collect menus on the Library's behalf. The collection has continued to grow through additional gifts of graphic, gastronomic, topical, or sociological interest, especially but not exclusively New York-related. The collection now contains approximately 45,000 menus dating from the 1840s to the present, about quarter of which have so far been digitized and made available in the NYPL Digital Gallery.

New York Public Library's What's on the Menu? project transcribes menu data using crowd-sourced system. Volunteers joins the project and transcribe menu from scanned or photographed menu from the collection. The trancibing process is still continuing and the data were updated regularly. Therefore, it is possible that we will have more dish from older years than later years.

2.1 Load and review dataset

There are 423165 obs and 10 column in dataset.
[1] "id": Unique Id for dish
[2] "name": Name of dish
[3] "description":
[4] "menus_appeared": indicate how many menu list the dish in it
[5] "times_appeared": indicate how many times dish appeared during first time appeared and last time appeared
[6] "first_appeared": indicate year in which the dish first time seen
[7] "last_appeared": indicate year in which the dish last time seen

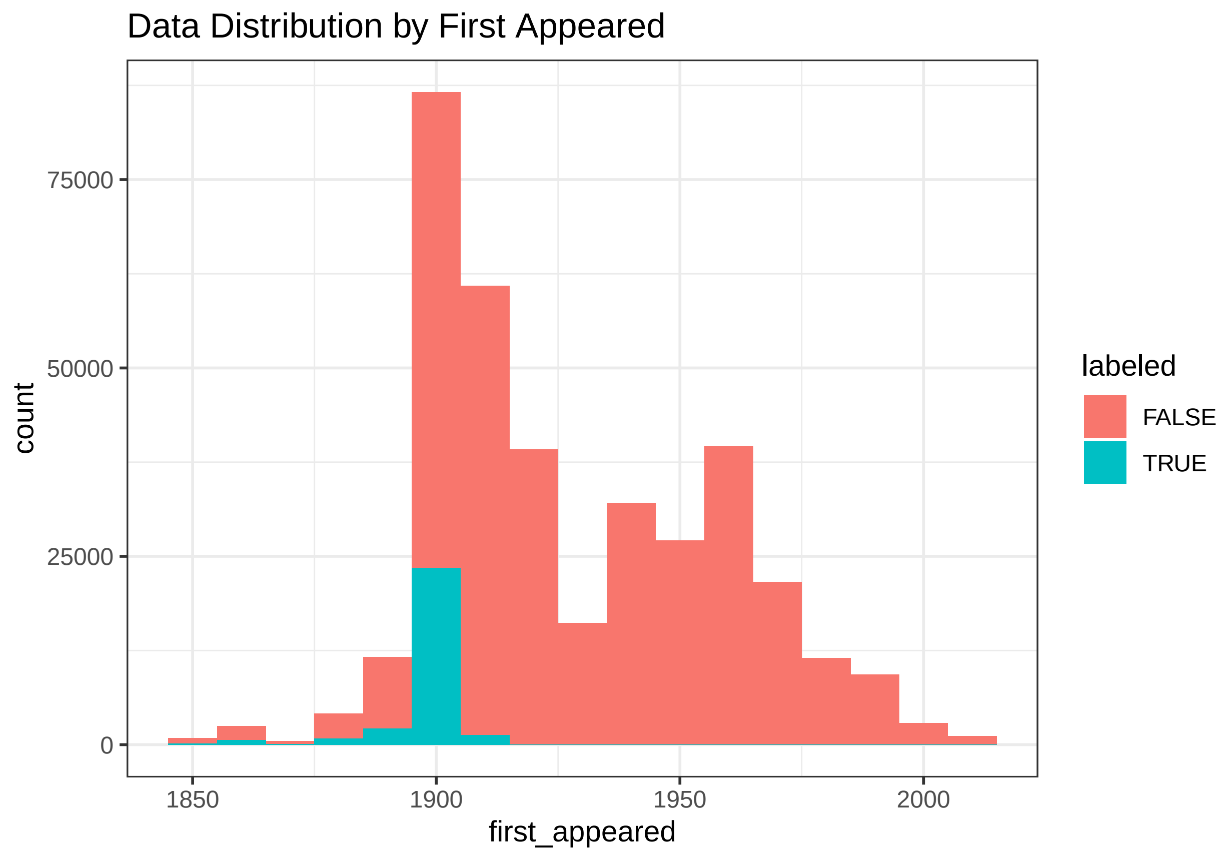
[8] “lowest_price”: indicate the lowest price of dish ever recorded
[9] “highest_price”: indicate the highest price of dish ever recorded
[10] “seafood_yn”: indicate whether the dish is seafood (coded 1) or not a seafood (coded 0). NA indicate that the dish are not labeled.
There are 3,492 observation that are labeled as seafood dishes and 25,182 labeled as non-seafood dishes.

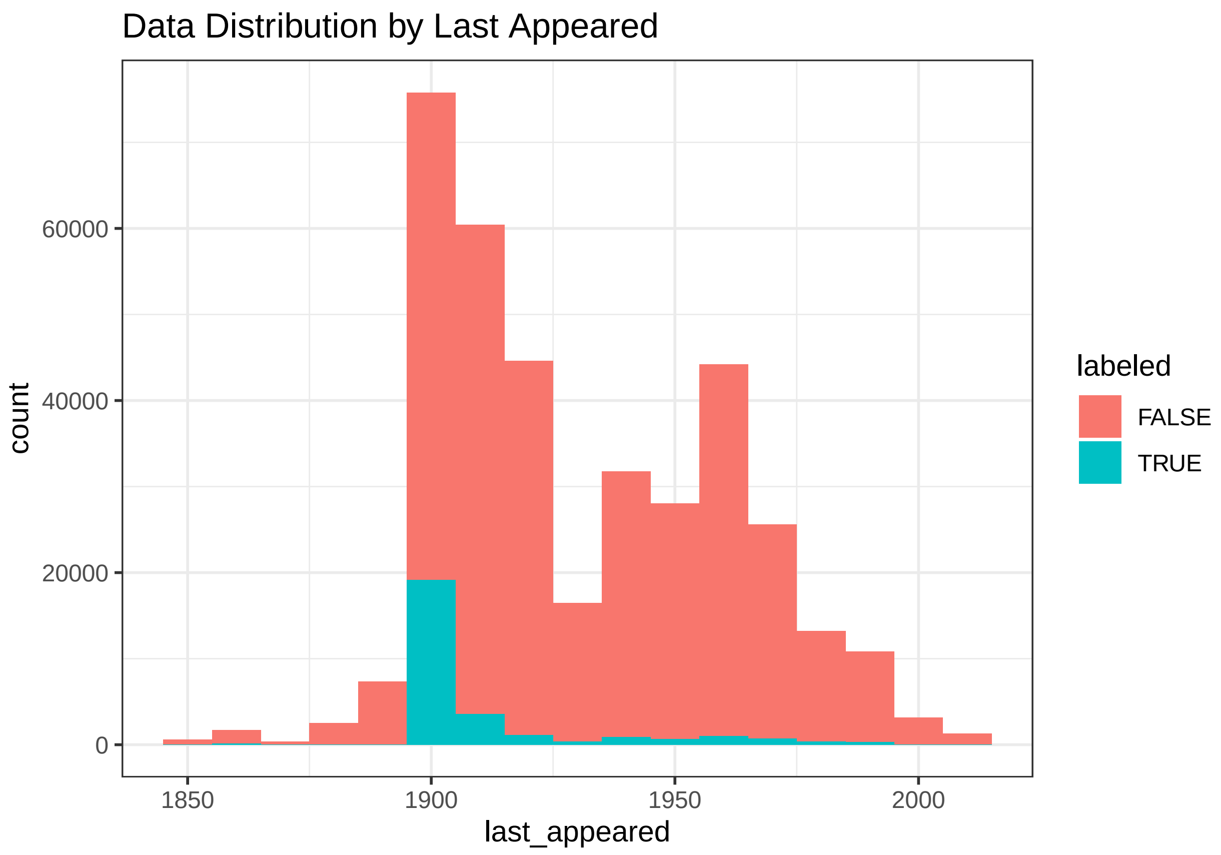
2.2 Sample evaluation

2.2.1 Distribution of labeled and non labeled by Year

Most of observtions dated from 1900’s to 1920’s, this happen due to the fact that the New York Public Library’s menu collection is started by Miss Frank E. Buttolph (1850-1924), who, in 1900, began to collect menus on the Library’s behalf. Miss Buttolph added more than 25,000 menus to the collection, before leaving the Library in 1924.

The histogram shows that the data lacks of observation from later year (after 1907). From total of 246,640 observations after 1907, only 58 of them are labeled.





We need to generate new sample so that dataset has a vocabulary from later years. To do that, we take one percent of oberservation after 1908 to be labeled. To make sure we have good additional sample we take # observation based on it's proportion to dataset. Therefore, for example, if we have more dishes in 1988 than 1990, we take more sample from 1988 than 1990. This approach also making sure that we have at least one sample from all years. We have 2396 new observations to be labeled in the next step.

Finally, we label it 1 for seafood or 0 for non-seafood, then we bind new labeled sample to dataset.

2.2.2 Translating dish name and evaluating data vocabulary

Name variable in the dataset is written in namy languages, including English, Germany, and French. This difference in language used causes several problems. firstly, it is difficult for cleaning procedure such as removing stop words and stemming. Secondly, the same dish will be seen as different dish is it'swritten in other language. Therefore, we will translate name variable in dataset using Google Translate API.

Using Google API, we make loop function that translate dish name one by one and save it into a new object. Next, we use 'cbind' to merge it with dataset as new variable translatedText.

Once we have all name in english, we can evaluate whether our our train sample has a proper vocabulary compared to the rest of dataset (unlabeled).

To do that, first we separate labeled and unlabeled dataset. Then we make train and test from labeled dataset. We will do this process again after we add more observation to enrich vocabulary in our dataset.

Next, we will create document frequency matrix from datasets so we can count each unique number of words in unlabeled dataset, dummy train dataset, and dummy test dataset.

Once we had all three document frequency, we make a table that contain words count from each matrix

There might be too much data to evaluate. To make it feasible to add more sample we decided to choose 10,000 unique words that are most frequently appear. Then, we join top term from three count table that we had before. The results is a table of most frequent data that are not available in train and/or test dataset.

Based on table above, we decide to take 760 observations of words that aren't available in labeled dataset.

Then, we labele it manually. It worth to mention that even though we already translate dish name to english, one dish name are remain unidentifiable whether it is a seafood or non seafood. We decided to leave it NA.

At last, we add our new sample to our labeled dataset. Then we remove it from unlabeled dataset. Finally we bind them together in df.all that contain all dataset.

It worth to mention that There are some human error in labeled dataset, especially when dish name is not in english. Therefore, we still continue to evaluate seafood_yn variable in labeled dataset manually. Our final labeled dataset after final evaluation is dflabeledtranslated_edited.csv.

Data Preprocessing

Overview on labeled dataset

A labeled dataset consist of 31,829 obs and 11 variables.

However, for this purpose of predicting feafood_yn based on the dish name, we only need dish name (original name and translated name) and class variable seafood_yn. 86,53 percent of obsevartion are non seafood dishes, while only 13,46 percent are seafood dishes.

Adding variable.

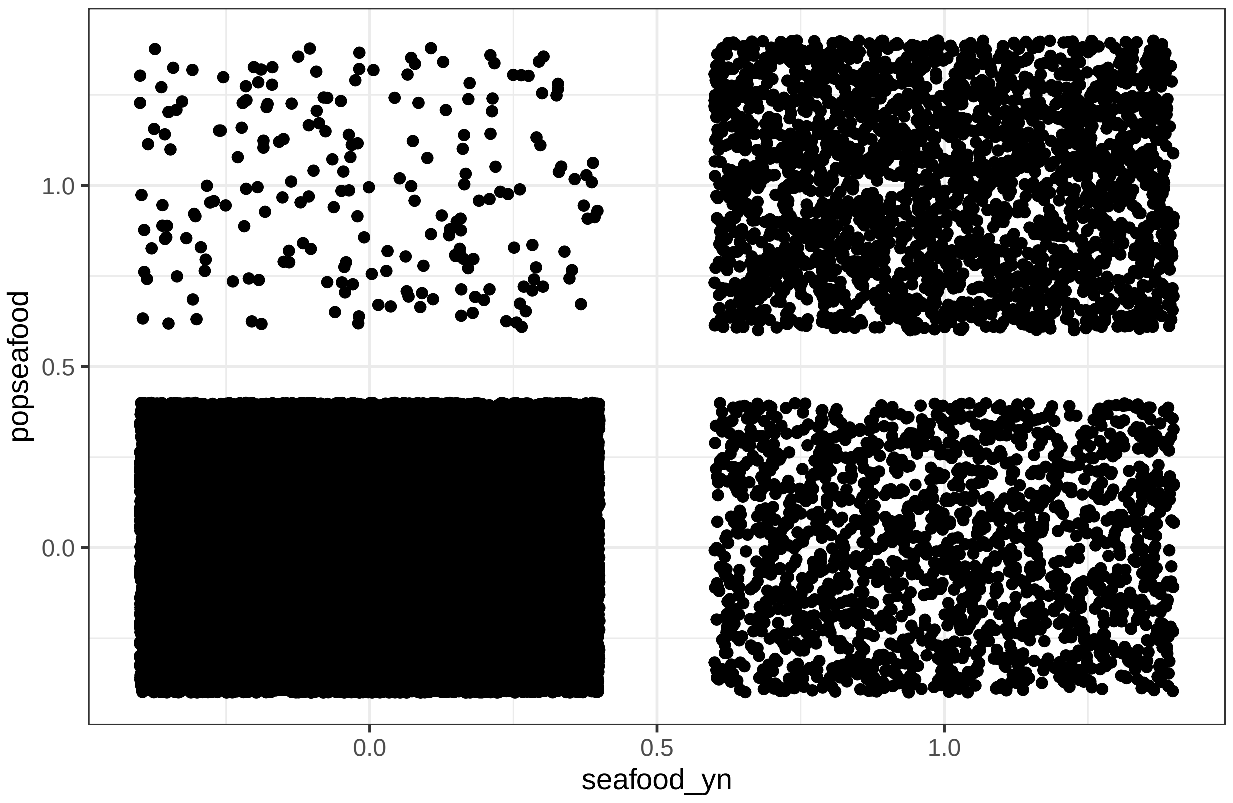
Our intuition is that there are popular seafood name that will be comprises on most seafood dishes. This keywords is something like "dead giveaway" to determine dishes is seafood or not. We will not use this variable in all of our model, but only in a model where we might want to use PCA or SVD to our variable.

We can see from graph A that most obs that do not contains popular words are indeed not a seafood and there are only few of them are seafood. In the other hand, a lot of seafood dishes do not contain these popular words, indicating that there are vast amount of vocabulary (fish/ingredient name) on a seafood dishes.

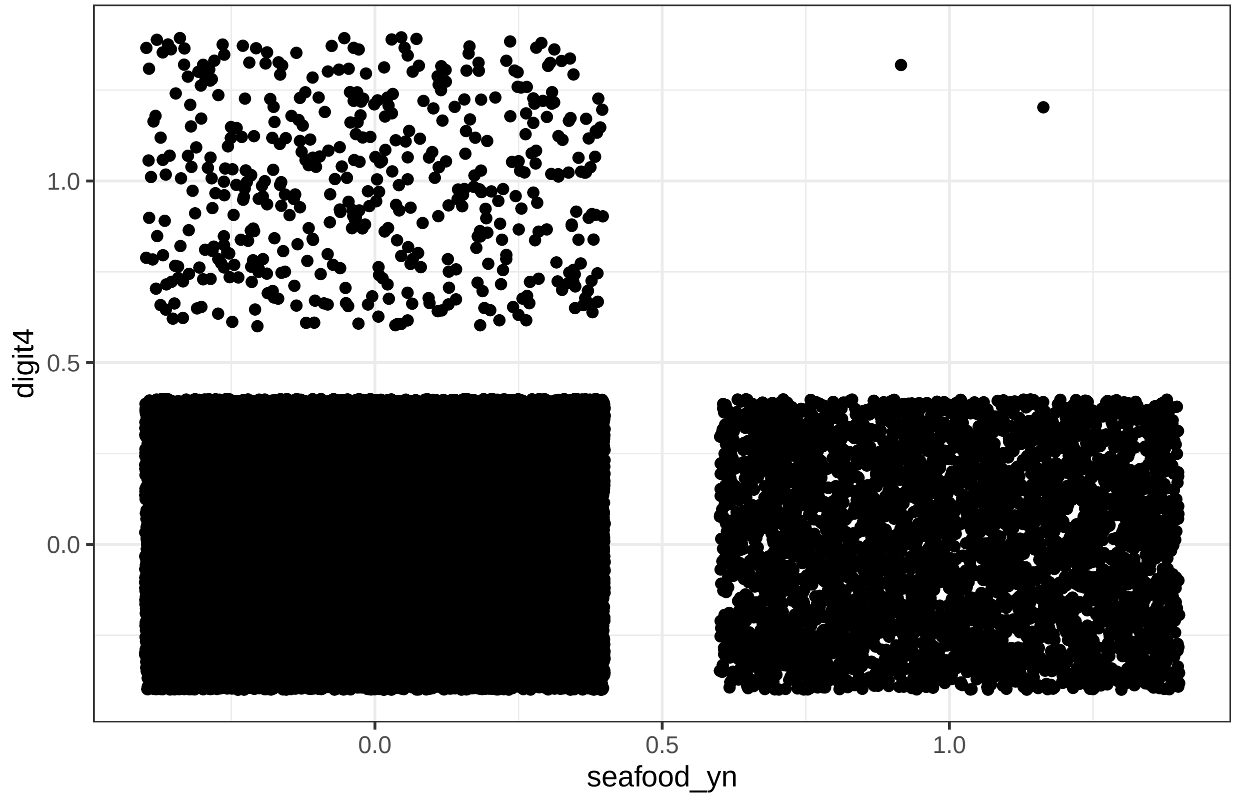
We also think that it is unusual for any dish that has a 4 digit number on it's name. Such a pattern might be popular on wine or other type of drinks. Because we intend to remove all of number in our dish name, we want to reserve is another way, by creating new variable digit4 (1 if there is 4 digit number in dish name, or 0 if there is no 4 digit number in dish name).

We can see from graph B, most of dishes do not contain 4 digit number. We can also see that there are only two dishes that contain 4 digit number are seafood. We can see that this is a good distinction measure to determine whether the dish is seafood or not.

Graph A Seafood dishes on popular words



Graph B Seafood dishes on whether they contain 4 digit words



create training and test

We create training and test using 80:20 proportion. However, due to class imbalance on data set, we decide to make under sampling training set. This undersampling will benefit us in two ways: 1. correcting bias problem created by imbalanced class, and 2. Reducing the size of training sample, making it faster for some machine learning algorithm such as random forest.

preprocess training set

Generally we preprocess our training set with following steps:
1. Make all text lowercase
2. Remove any punctuation, numbers, symbols, stopwords, and hyphens
3. Stem the text.

After all steps completed, we generate make two version of training set: one with all words, and one with only noun. We also preprocess undersample training using the same steps.

Furthermore, due to the dimention of training data, we make "compressed" version of training set. We use Singular Value Decomposition to reduce variable to 300 variables. This reduced dimention not only will help us increasing processing speed when we run the model, but also tackle multicolleniarity problem in datasets, particularly when we want to use Logistic Regression model.

Overall we have four type of training sets:
1. Training with all vocabulary
2. Training with only noun

3. Undersampled training with all vocabulary
4. Training with SVD

preprocess training set with all vocabulary

We first create a corpus from training set. Corpus is a list consist of every words from variable translated.We then clean our corpus by following our three cleaning steps, including: make all text lowecase, removing any unused character, and stem the text, then create document frequency matrix. Document frequency matrix is a matrix that consist of every words frequency (words counts) for each observation. We then add digit4 variable and class variable (seafood_yn) to document frequency matrix and save it as data frame.

preprocess training set with only noun

We use document frequency matrix from training above then selecting for only noun. We select noun using udpipe model from udpipe package. We then add digit4 variable and class variable to document frequency matrix and save it as data frame.

preprocess training undersampling

We use the same approach to clean and create document frequency matrix. However, instead of full training dataset, we use training with under sample data frame as our corpus.

preprocess training set with SVD

As noted before we use SVD to create shorter version of training set. We take document frequency matrix from full training set then use irlba() function to create dataset with 300 variables.

preprocessing test

In order to make a prediction in our test set, we have to follow the same steps as we did for a training set. Furthermore, test set must contain the same variable as a training set, so we use training set pattern for test set.

Overall, we have 3 test set:
1. Test with all vocabulary
2. Test with only noun
3. Test with SVD.

We don't make test set for undersampling because we can use the same test as training with all words to validate the result for undersampling training.

preprocess test set all words

We process this test set using the same steps as training set. In addition we match variable for this test set with training set with all words.

preprocess test set Noun only

We process this test set using the same steps as training set. In addition we match variable for this test set with training set with nouns only

preprocess test set using all words then convert it to 300 variable using svd

We process this test set using the same steps as training set. In addition we match variable for this test set with training set with svd

Model performance evaluation

In order to evaluate models, we reate a function to determine best model. We determine best model based on accuracy the fuction will return the best accuracy, tuning with the best accuracy, and confusion matrix for the model with the best accuracy.

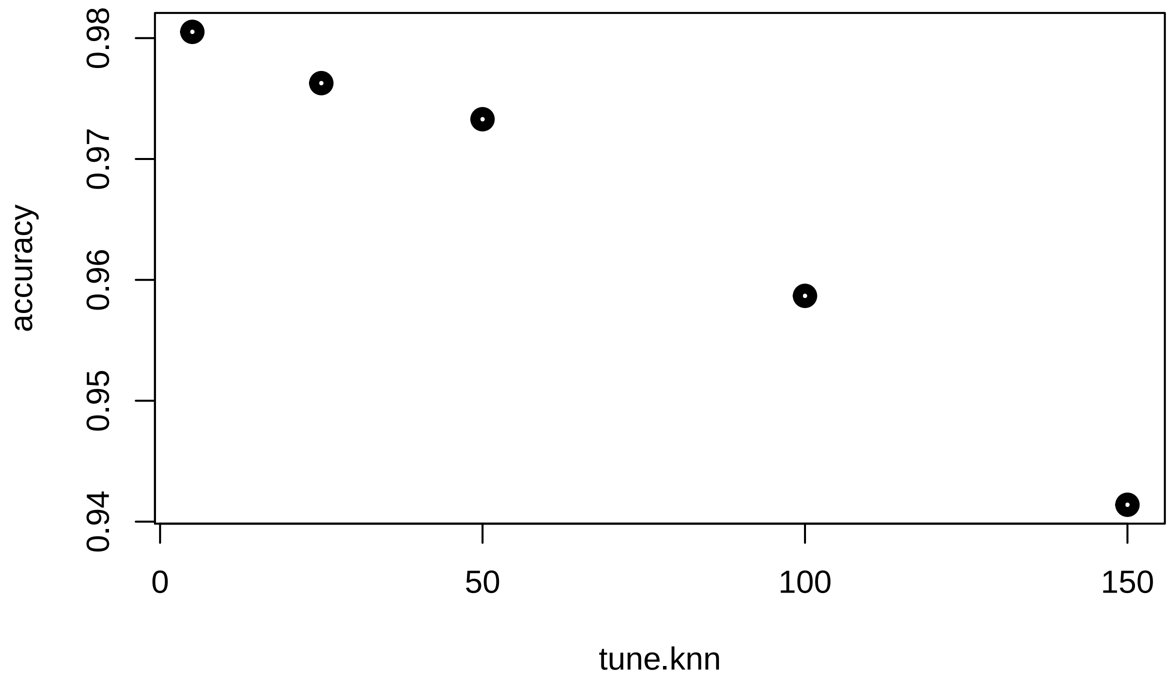
Running and Validating Machine Learning Models

K-nearest neighbour (KNN)

Our first model is K-nearest neighbour (KNN). KNN is a non-parameteric classification method where in order to predict a response class, we use the class of k-nearest neighbour in the data set. k indicate the number of neighbour we want to use in the model. For example, if we choose 3 as k, we look for 3 nearest neighbour of our predicted observation. If two or more nearest negihbour is a seafood then we predict seafood and vice versa. The distance between neighbour is determined using Euclidean Distance.

For this model we only use training with SVD. We cannot unreduced training because there is too much similarity the value between observation (most obs will have 0 value), thus there will be too many ties.

```
## $max_correct_predicted_rate
## [1] 0.9805185
##
## $best_thershold
## [1] 1
##
## $best_confusion.matric
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5435  104
##          1   20  806
##
##                Accuracy : 0.9805
##                  95% CI : (0.9768, 0.9838)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9173
##
##  Mcnemar's Test P-Value : 9.081e-14
##
##             Sensitivity : 0.9963
##             Specificity : 0.8857
##          Pos Pred Value : 0.9812
##          Neg Pred Value : 0.9758
##               Prevalence : 0.8570
##           Detection Rate : 0.8539
##     Detection Prevalence : 0.8702
##        Balanced Accuracy : 0.9410
##
##         'Positive' Class : 0
##
```



Our best model is using k = 5, with percent 98.05 accuracy. Model has a very high Sensitivity 0.9963 but relatively low Specificity : 0.8857.

Logistic Regression

Our next model is Logistic Regression. We determine our prediction by measuring probability of observation being in a class based on coefficient of our variables.

The main benefit of this model is that we can understand what words determine whether a dish is a seafood or not. However, the main disadvantages is that most words are insignificant and since most data frame has a 0 value, most independent variables are dependent with each other.

To tackle this problem we can use svd version of training set. This approach eliminate multicollinearity but at the same time deleting main advantages, that is we lose explanability of coefficient.

```
## $max_correct_predicted_rate
## [1] 0.9767478
##
## $best_thershold
## [1] 1
##
## $best_confusion.matric
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5411  104
##          1   44  806
##
##                Accuracy : 0.9767
##                  95% CI : (0.9727, 0.9803)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9024
##
##  Mcnemar's Test P-Value : 1.236e-06
##
##             Sensitivity : 0.9919
##             Specificity : 0.8857
##          Pos Pred Value : 0.9811
##          Neg Pred Value : 0.9482
##              Prevalence : 0.8570
##          Detection Rate : 0.8501
##    Detection Prevalence : 0.8665
##       Balanced Accuracy : 0.9388
##
##        'Positive' Class : 0
##
```

Other possible approach is to use variable selection. Due to high number of variables we choose to include only significant variable (>0.1 alpha) as our final model. In the code below, we train our train with only.

```
## $max_correct_predicted_rate
## [1] 0.9641791
##
## $best_thershold
## [1] 1
##
## $best_confusion.matric
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5344  117
##          1  111  793
##
##                Accuracy : 0.9642
##                  95% CI : (0.9593, 0.9686)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8534
```

```
##
##  Mcnemar's Test P-Value : 0.7405
##
##             Sensitivity : 0.9797
##             Specificity : 0.8714
##          Pos Pred Value : 0.9786
##          Neg Pred Value : 0.8772
##              Prevalence : 0.8570
##          Detection Rate : 0.8396
##    Detection Prevalence : 0.8580
##       Balanced Accuracy : 0.9255
##
##        'Positive' Class : 0
##
```

As the final approach for logistic regression, we also tried to use undersampling training with the same approach as training with only noun.

```
## $max_correct_predicted_rate
## [1] 0.8914375
##
## $best_thershold
## [1] 80
##
## $best_confusion.matric
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5448  684
##          1    7  226
##
##                Accuracy : 0.8914
##                  95% CI : (0.8835, 0.899)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : 2.473e-16
##
##                   Kappa : 0.358
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9987
##             Specificity : 0.2484
##          Pos Pred Value : 0.8885
##          Neg Pred Value : 0.9700
##              Prevalence : 0.8570
##          Detection Rate : 0.8559
##    Detection Prevalence : 0.9634
##       Balanced Accuracy : 0.6235
##
##        'Positive' Class : 0
##
```

Using Logistic regression, our best model is using training with only noun, with 97.67 percent accuracy.

Decision Tree

Decision tree is a very intuitive model. Based on variables in the dataset we will create a "branch" that will separate observations whether this is a seafood dish or not. This separated group of observations became a node (leaves). We then continue to make another branch from this node until we can no longer make a new node (that is when our accuracy decline when we make new node). In prediction, the class will be determined using a mode (whether there is more seafood or non seafood in the node).

Looking at our trees, we can see that all trees create branches using popular seafood name i.e oyster, mackerel, bass, etc. Pruned trees version (trees wih smaller number of branches) prove to have less accuracy than full grown trees.

Tree 1 : Using train set with noun only

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5420  175
##          1   35  735
##
##                Accuracy : 0.967
##                  95% CI : (0.9623, 0.9713)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8561
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9936
##             Specificity : 0.8077
##          Pos Pred Value : 0.9687
##          Neg Pred Value : 0.9545
##              Prevalence : 0.8570
##          Detection Rate : 0.8515
##    Detection Prevalence : 0.8790
##       Balanced Accuracy : 0.9006
##
##        'Positive' Class : 0
##
```

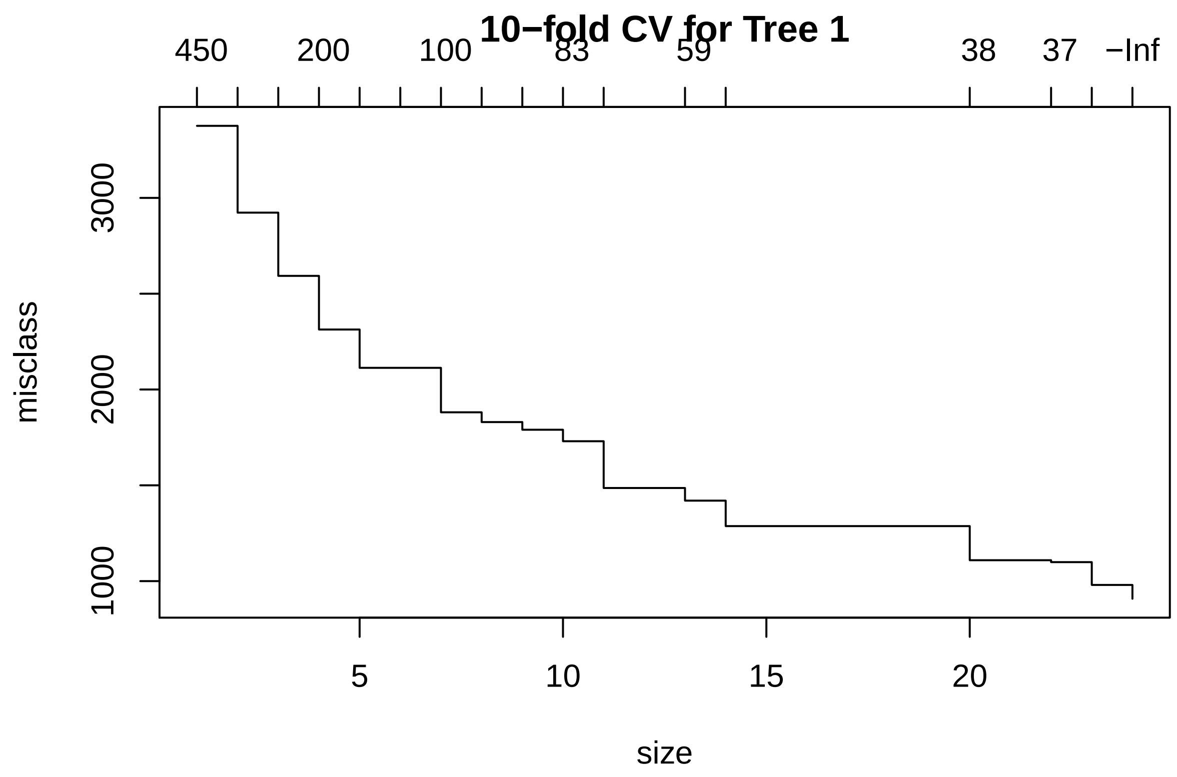
Tree 2 : Using train set with all words

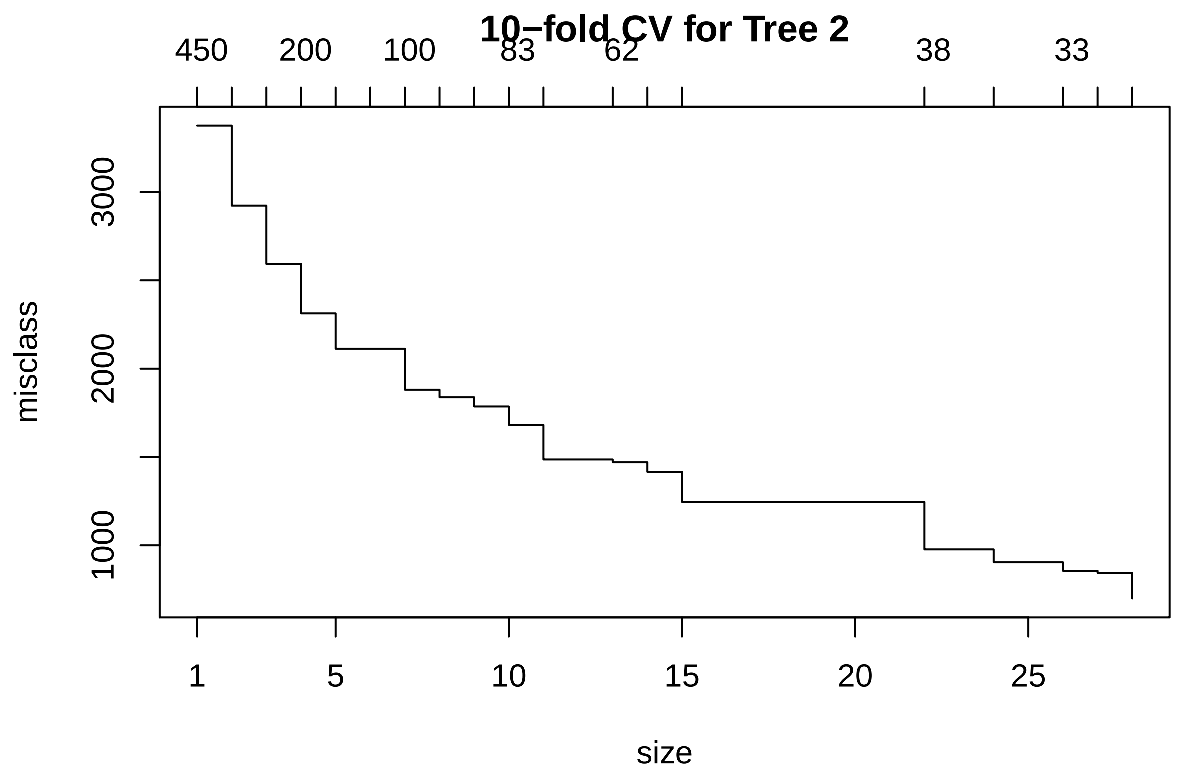
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5415  122
##          1   40  788
##
##                Accuracy : 0.9745
##                  95% CI : (0.9704, 0.9783)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8921
##
##  Mcnemar's Test P-Value : 1.966e-10
##
##             Sensitivity : 0.9927
##             Specificity : 0.8659
##          Pos Pred Value : 0.9780
##          Neg Pred Value : 0.9517
##              Prevalence : 0.8570
##          Detection Rate : 0.8507
##    Detection Prevalence : 0.8699
##       Balanced Accuracy : 0.9293
##
##        'Positive' Class : 0
##
```

Tree 3 : Using train set with undersampling

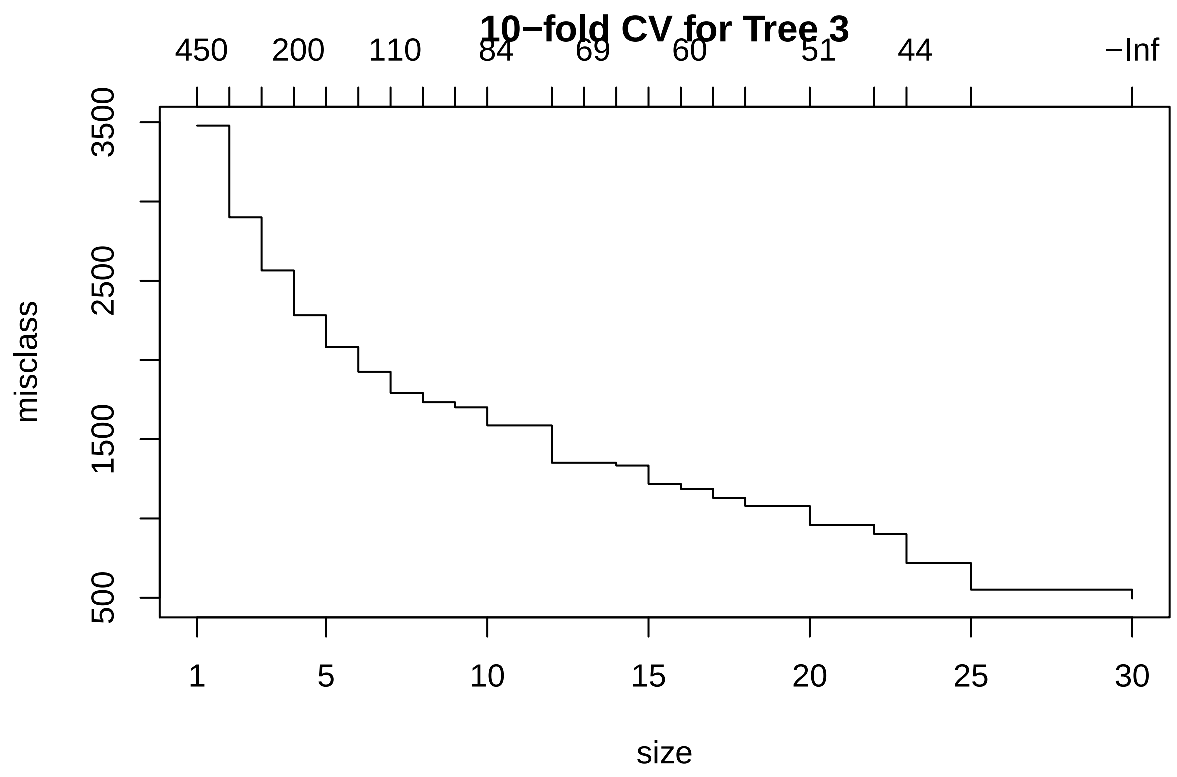
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5415  127
##          1   40  783
##
##                Accuracy : 0.9738
##                  95% CI : (0.9695, 0.9775)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8885
##
##  Mcnemar's Test P-Value : 2.835e-11
##
##             Sensitivity : 0.9927
##             Specificity : 0.8604
##          Pos Pred Value : 0.9771
##          Neg Pred Value : 0.9514
##              Prevalence : 0.8570
##          Detection Rate : 0.8507
##    Detection Prevalence : 0.8707
##       Balanced Accuracy : 0.9266
##
##        'Positive' Class : 0
##
```

We can tune the model for our training set using 10-cross fold validation to see if pruning (reducing the number of branches) will reduce error rate. We can see from graph below that pruning doesn't reducing misclassification rate, thus our model is the best decision tree for respective training set.


10−fold CV for Tree 1
450
200
100
83
59
38
37
−Inf
misclass
1000
2000
3000
size
5
10
15
20



10−fold CV for Tree 2
450
200
100
83
62
38
33
3000
2000
1000
misclass
1
5
10
15
20
25
size



Using Decision Tree, our best model is using Training set with all words, with 97.45 percent accuracy (Sensitivity : 0.9927 and Specificity : 0.8659).

Support Vector Machine

Support Vector Machine (SVM) finds a hyperplane in an N-dimentional space (in case of N-number of features) to classify observations. this hyperplane became separator between observations, in which opposite side of separator became different class. We can utilize different types of hyperplane, including linear (hyperplane is like straight line in 2 dimentional spaces), polynomial ((hyperplane is like polynomial line in 2 dimentional spaces), radial (like a circle in 2 dimentional spaces), and so on. Additionally, we can also introduce sot in our model. Cost indicate how much we want our model in allowing misclassified observation in a group. It will determinde how flexible our model is

First, try different cost and different kernel in undersample trainingand see what is the best kernel and cost

```
##   cost kernel
## 3    1 linear
```

Since the best model is linear kernel with cost = 1, we then can proceed with this model and use it with different training.

Training with undersampling

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5371   45
##          1   84  865
##
##                Accuracy : 0.9797
##                  95% CI : (0.976, 0.9831)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9187
##
##  Mcnemar's Test P-Value : 0.0008207
##
##             Sensitivity : 0.9846
##             Specificity : 0.9505
##          Pos Pred Value : 0.9917
##          Neg Pred Value : 0.9115
##              Prevalence : 0.8570
##          Detection Rate : 0.8438
##    Detection Prevalence : 0.8509
##       Balanced Accuracy : 0.9676
##
##        'Positive' Class : 0
##
```

Training with all words

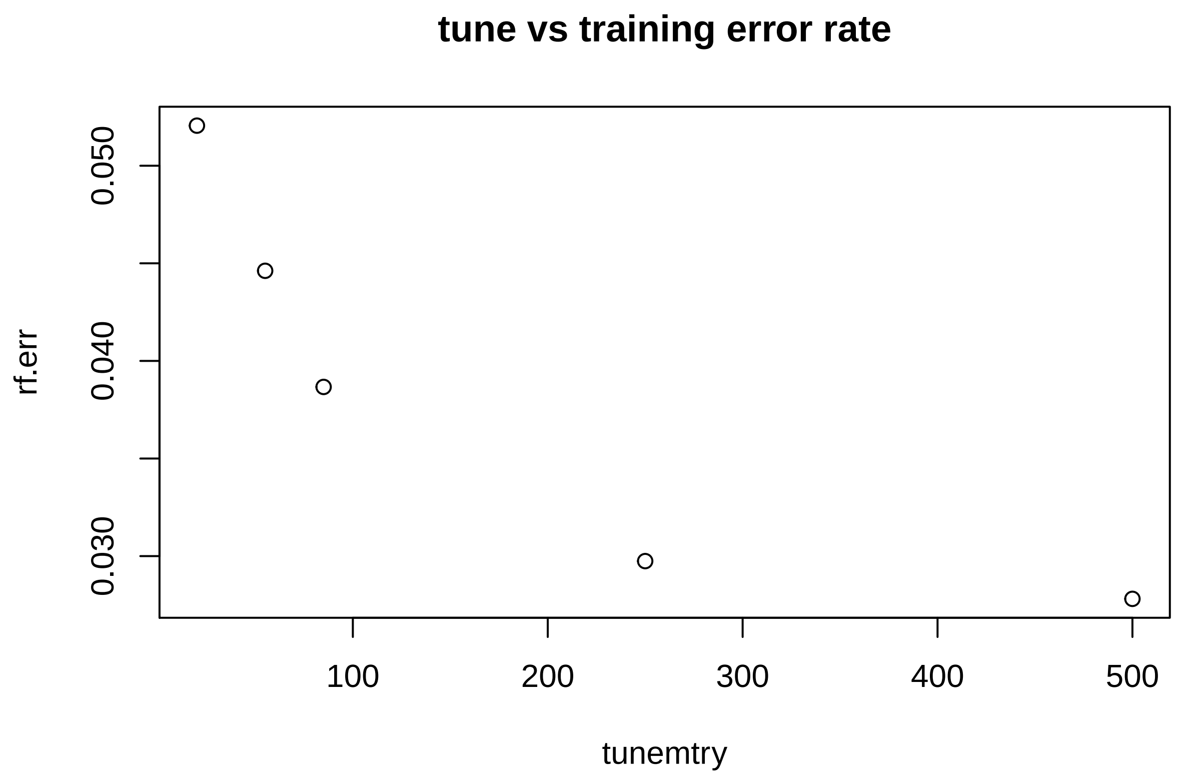
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5439  111
##          1   16  799
##
##                Accuracy : 0.98
##                  95% CI : (0.9763, 0.9833)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9149
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9971
##             Specificity : 0.8780
##          Pos Pred Value : 0.9800
##          Neg Pred Value : 0.9804
##              Prevalence : 0.8570
##          Detection Rate : 0.8545
##    Detection Prevalence : 0.8720
##       Balanced Accuracy : 0.9375
##
##        'Positive' Class : 0
##
```

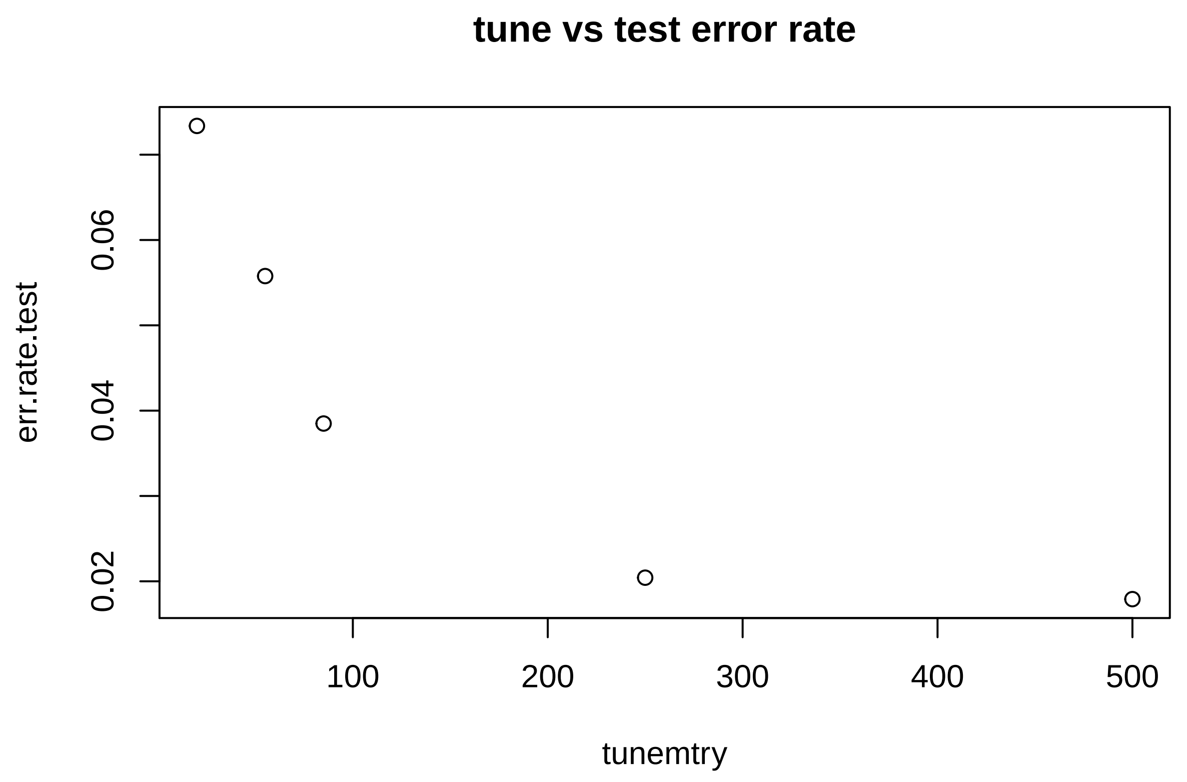
Using train with noun only gives us very high Sensitivity but much lower Specificity, while using under sampling, while giving us a little bit lower accuracy gives us a more balanced result in both Sensitivity and Specificity.

Random Forest

Random forest is a method where we use bagging on decision tree. In Random Ferest, we make n # of decision trees, with randomly selected m from p number of variable at the time (where p is number of all variable and m<p). We then make a prediction based on the average probability of all trees that we make. This model however, lose some degree of interpretability from decision tree.

Due to processing power needed to run random forest, we choose to use run random forest only on under-sampling training.





Using 500 trees, and 500 words at a time we found our forest 0.9819 in accuracy.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5367   26
##          1   88  884
##
##                Accuracy : 0.9821
##                  95% CI : (0.9785, 0.9852)
##     No Information Rate : 0.857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9289
##
##  Mcnemar's Test P-Value : 1.109e-08
##
##             Sensitivity : 0.9839
##             Specificity : 0.9714
##          Pos Pred Value : 0.9952
##          Neg Pred Value : 0.9095
##              Prevalence : 0.8570
##          Detection Rate : 0.8432
##    Detection Prevalence : 0.8473
##       Balanced Accuracy : 0.9776
```

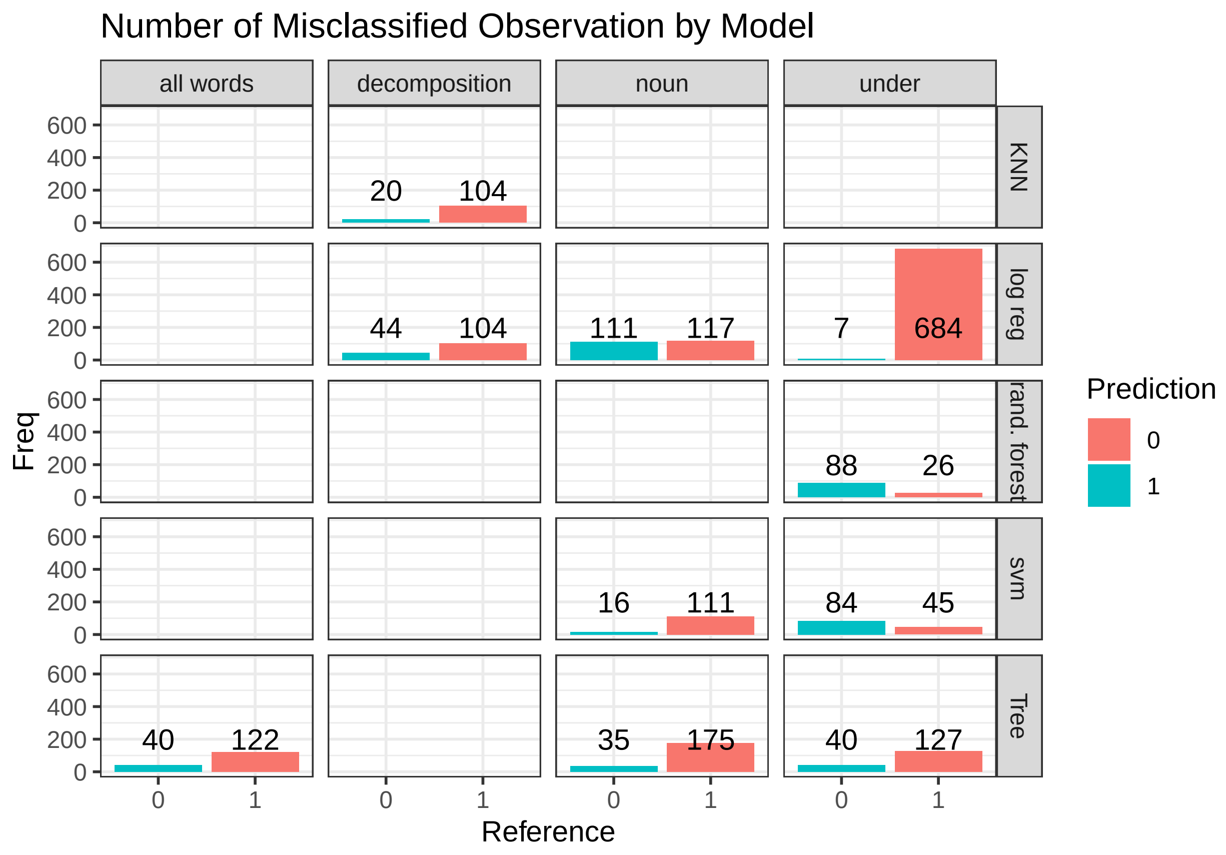
```
##
##        'Positive' Class : 0
##

##          MeanDecreaseGini
## oyster         286.00861
## lobster        218.75106
## clam           170.37005
## salmon         135.51774
## fish           102.71900
## crab            96.18526
## codfish         74.04928
## sole            73.07637
## turtl           70.81802
## sardin          67.52469
## shrimp          67.50941
## shell           67.41496
## caviar          65.76911
## broil           65.12980
## herring         64.17535
## mackerel        62.93305
## shad            60.95187
## fri             59.78187
## halibut         59.32969
## anchovi         56.88299
```

Results and Model Selection

We tried Machine learning algorithm in combination of 10 machine learning method and training set. Overall we can say that Support Vector Machine and Random Forest gives the best result in term of overall accuracy and balanced Sensitivity and Specificity. Random Forest, however, is better at predicting seafood dish, something that really important because the number of seafood dish in a labeled data is far less than non seafood dishes and we believe overall menu dataset has a similar distribution.

Additionally, Random Forest gives us major advantages in term of interpretability. While the model is not as informative as Decision Tree or logistic regression, we still can see what words are mostly a important determinant in predicting seafood or non seafood.



Predict all unlabeled data using Random Forest

We will use Random Forest with 500 trees and 500 variable as our model to predict unlabeled dataset. We use the same cleaning and preprocessing as we did with test dataset.

Due to high # of observations, we create partition with 1000 observations in each group. Using foor function, we then predict observation and add them to variable seafood_yn.

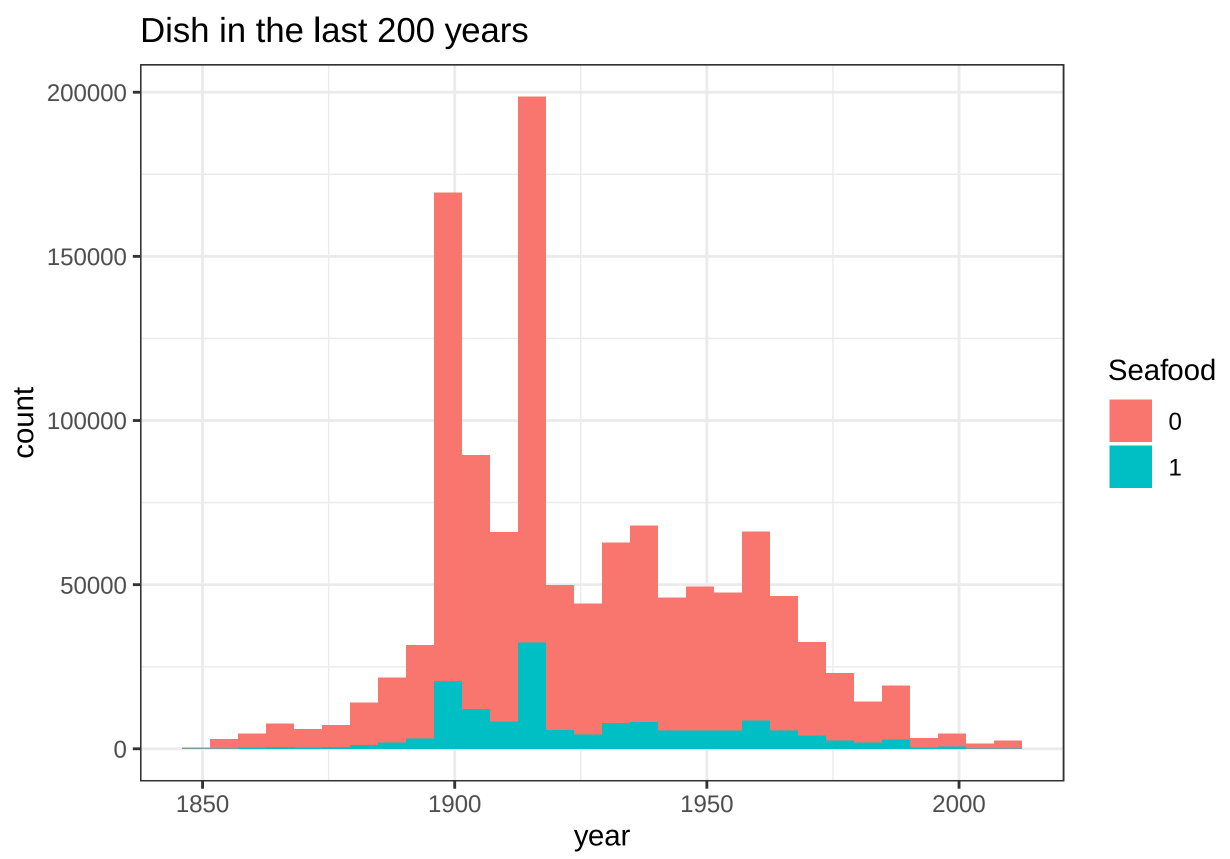
Visualisation

Once we have predicted label for all unlabeled data, we can make visualisation of all dataset. But first, we need to join all labeled data and unlabeled data that were predicted using random forest into one dataset.

Example below are graphic we can generate from dataset.

Number of dishes throughout years

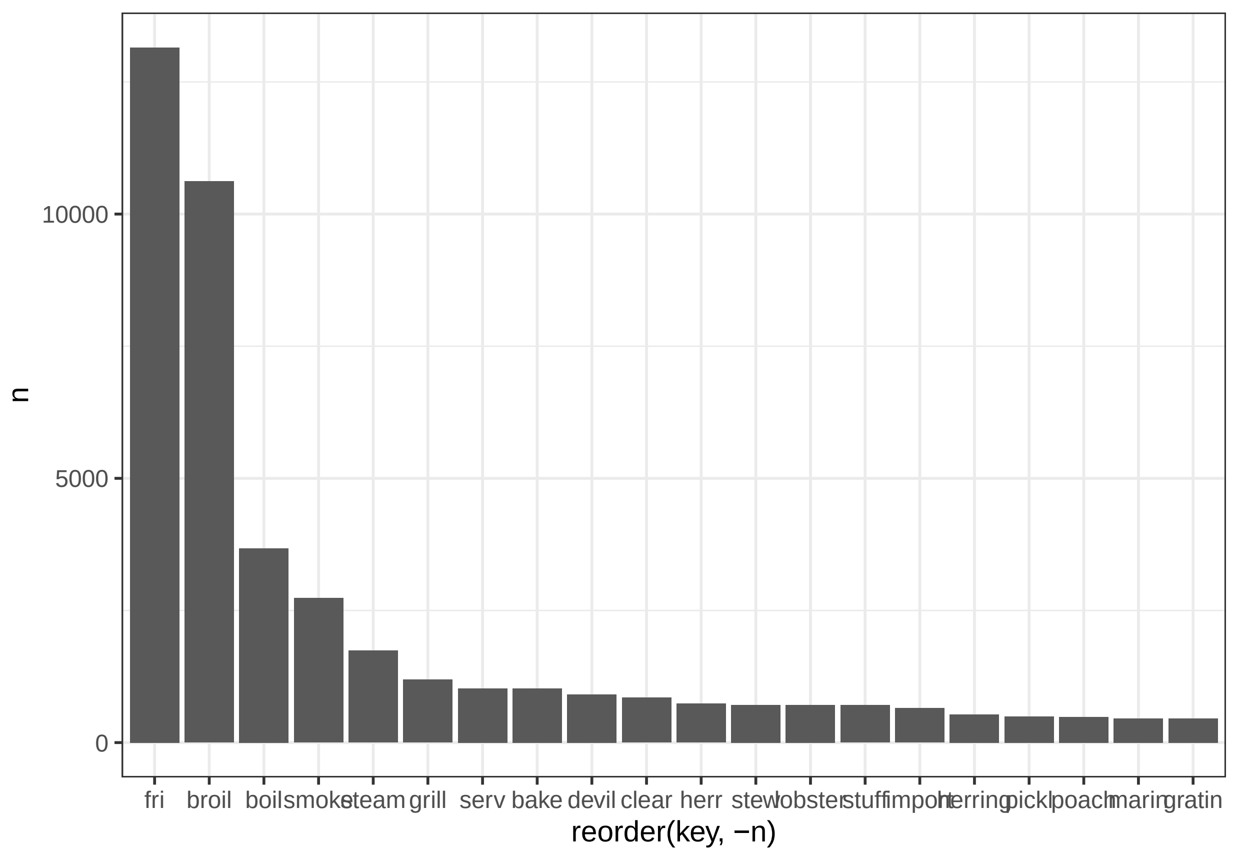
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



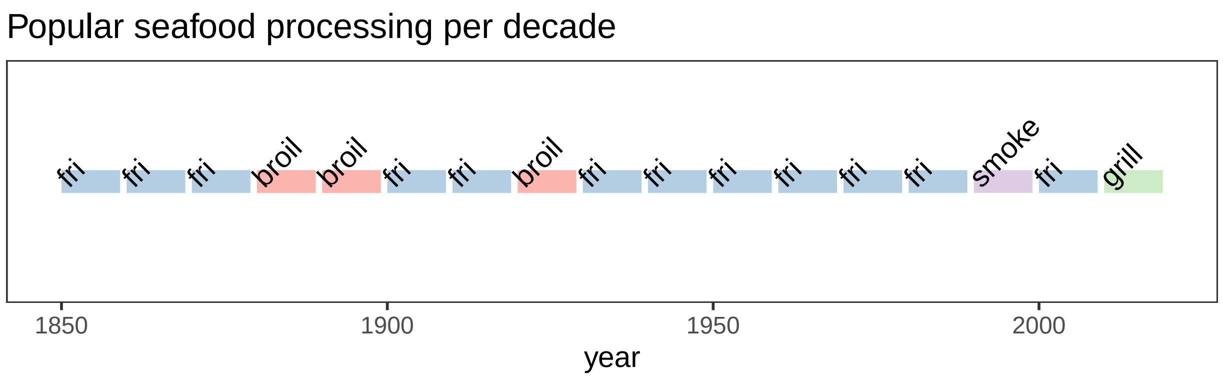
Popular seafood processing method

```
##
##     ADJ    ADP    ADV    AUX  CCONJ    DET   INTJ   NOUN    NUM   PART   PRON
##   35766  44284   2875   1666  13178   9642   2825 150726   5273   2505   2470
##   PROPN  PUNCT  SCONJ    SYM   VERB      X
## 271352  78964    986    599  55964   2948
```

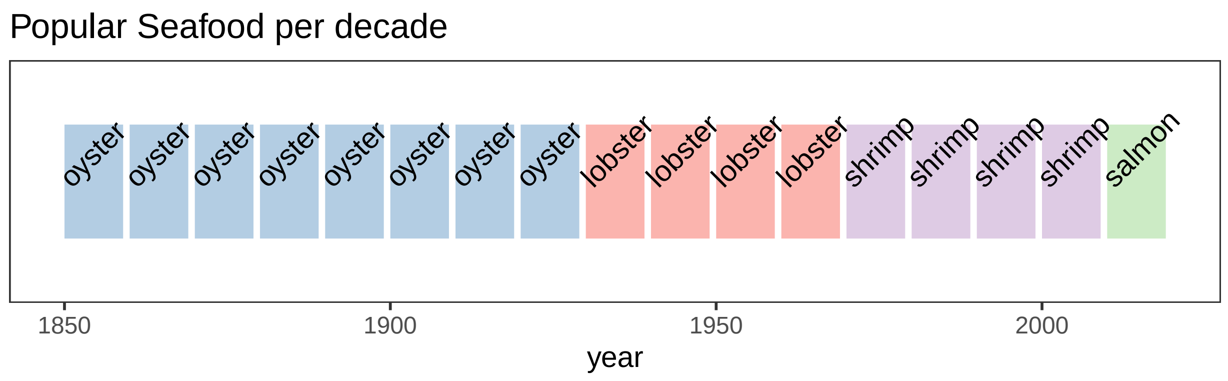
```
## Warning: 'as.data.frame.dfm' is deprecated.
## Use 'convert(x, to = "data.frame")' instead.
## See help("Deprecated")
```



```
## Warning: 'remove_hyphens' is deprecated, use 'split_hyphens' instead.
```

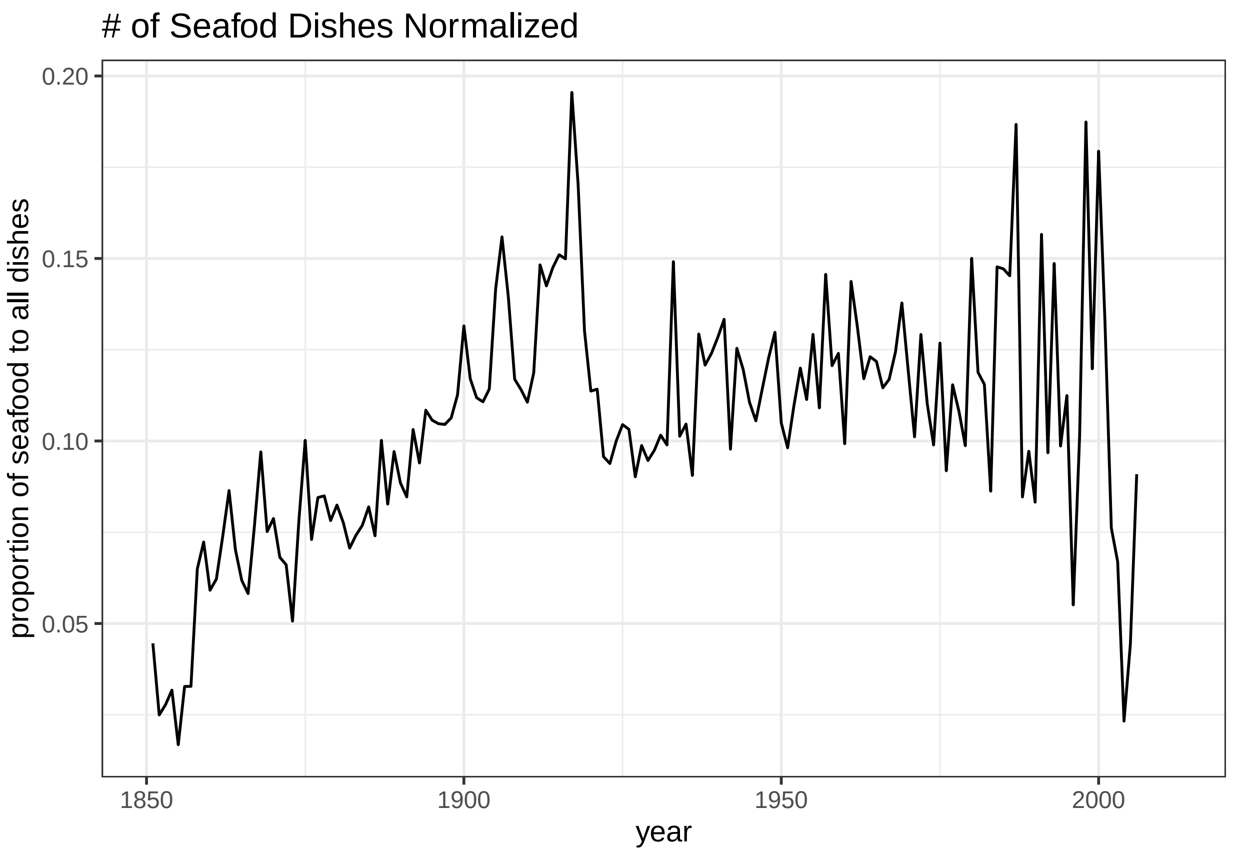


Popular seafood per decade

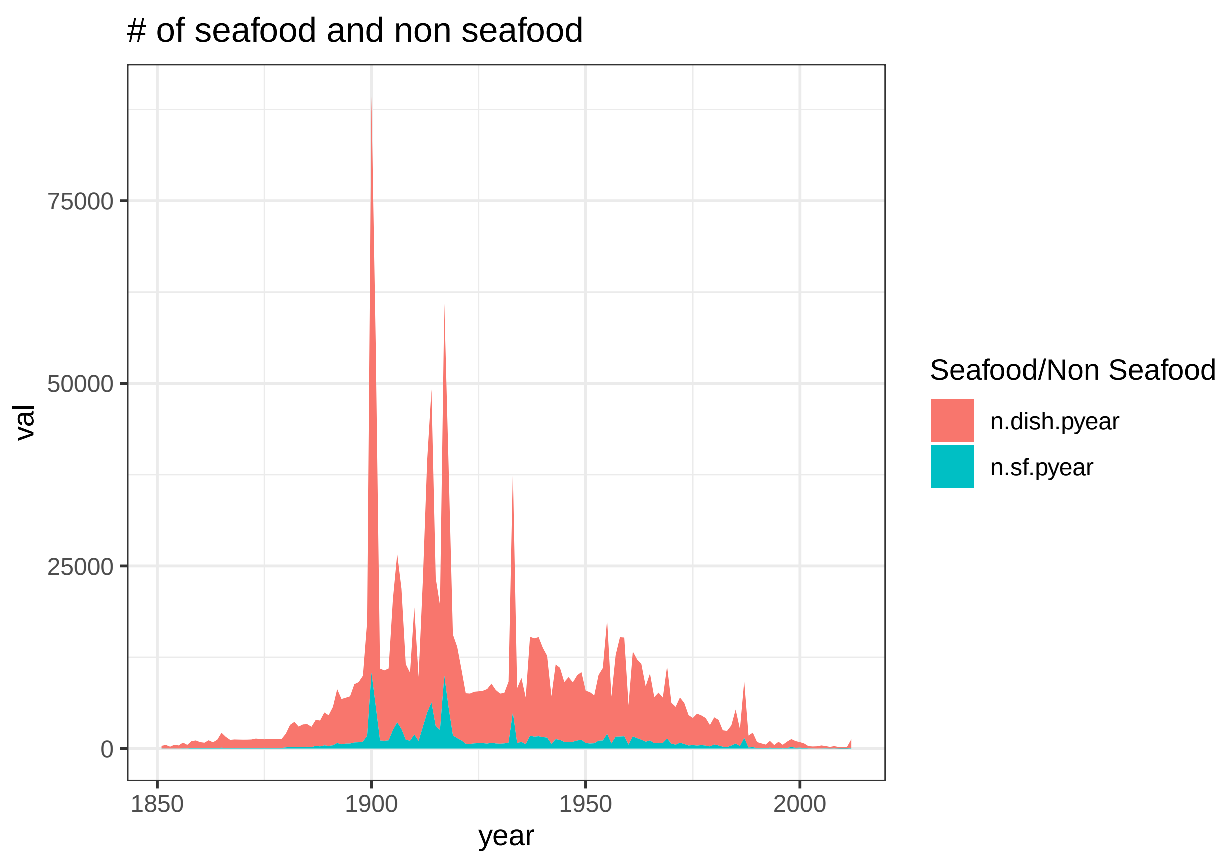


Plot time series standardise number of dishes

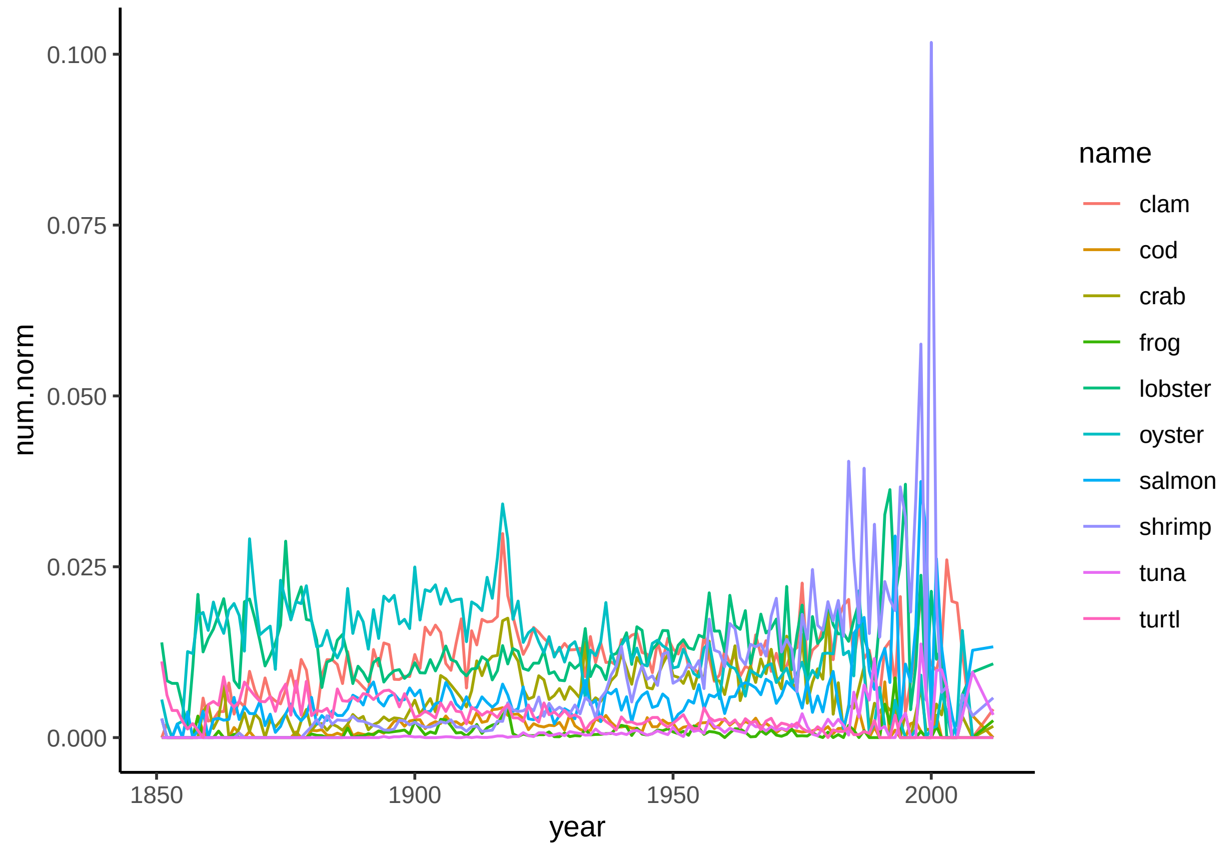
```
## Joining, by = "year"
```



```
## Warning: Removed 4 rows containing missing values (position_stack).
```

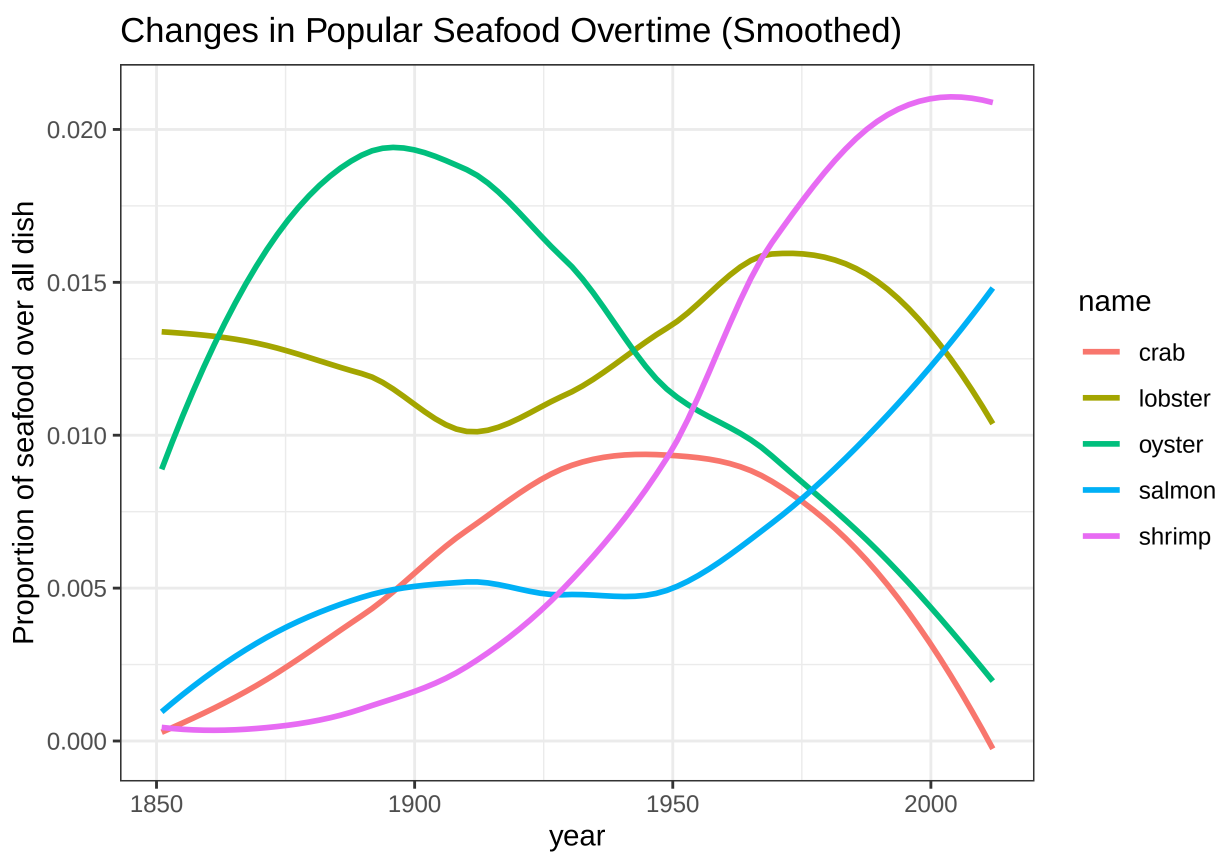


```
## Joining, by = "year"
```



Plot changes in popular dishes overtime

```
## 'geom_smooth()' using method = 'loess' and formula 'y ~ x'
```



Summary

- The model that we use to predict unlabeled dataset is Random forest with ntree= 500 and m=500.
- The model has 98.21% accuracy, 98.38% sensitivity, and 97.14 specificity on test set.
- Several pattern that we found including : Oyster was a popular food for almost 100 years! Lobster isn't popular seafood dishes until mid 19th century, and Shrimp steadily increased in popularity.
- In case of additional dataset need to be predicted using this model I would reccomend this steps of process
- Next steps, the labeled dataset can be used to explain other pattern such as fishery collapse, trading.
- All code, report and supporting file are in seafood globalization github repository.