

COMP30027 Machine Learning Project 2 Report

Anonymous

Introduction

The aim of the project is to predict the cooking time based on the given dataset. The dataset consists of 5 features: name (of the recipe), number of steps and ingredients needed, and the detailed description of each step and ingredient. Before I started analysing the data, I read the report *Generating Personalized Recipes from Historical User Preferences (2019)* produced by Bodhisattwa Prasad Majumder and colleagues, and I learned the background and some math behind the project which could be helpful.

To summarize, I used several types of models for learning: **ZeroR**, **RandomForest**, **K-Nearest Neighbor** and **logistic regression**, and mainly focus on randomforest. I selected the learners based on the theoretical explanation and their compatibility with the dataset, trained the model that produces predictions using Python, and finally, evaluated the performance of each model. I used **feature selection** in logistic regression by measuring mutual information. I also tried techniques such as **Principal Component Analysis** during the process for the attempts of better performance which will be explained more in detail in the report.

Approach

ZeroR

ZeroR is a method that produces predictions without considering the features but only the labels. The only advantage of using ZeroR over other models is its **simplicity**. However, the predictions made by ZeroR are much less informative than all the other models because it is just basically guessing the result by the most **frequent** label appearing in the dataset. I got a score of 50.2% **true positive** in the **confusion matrix** using ZeroR, which from it only predicted the label 2 (see in Figure 1). This shows the majority of the cooking time is median in the training dataset. As for the error rate, because it is a dumb learner in the first place, analysing error rate is less meaningful. However, it can be used as the **baseline** for the analysis.

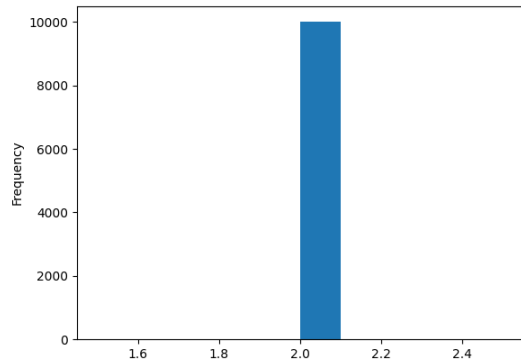


Figure 1- In the total of 10,000 test data, all of the predictions are 2

Random Forest

Random forest is an supervised ensemble learning method by constructing multiple decision trees and outputting the class that is the mode prediction of the individual trees. I selected random forest over decision tree because random forest can correct the overfitting of decision trees with enough samples (and we do), which makes the prediction more informative and accurate.

I used both datasets which convert the text feature to 50 and 100 vectors. However, I got a lower accuracy (true positive) in the competition with more features as input (67.4% on 100 vs 69.3% on 50), which is out of my expectation, I think this is because of the noise created during text feature conversion. Since there are 52 features in the dataset, I also attempt PCA to reduce some dimensions to see if I can boost the accuracy by reducing the dimension for less running time, the accuracy drops to 57.7%, which shows some of the useful features might be eliminated by PCA which could be the cause of accuracy dropping. The prediction result is shown in Figure 2.

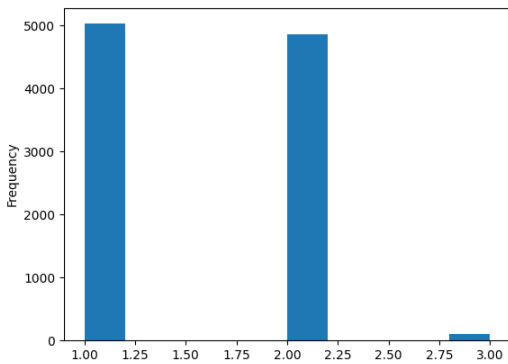


Figure 2- the random forest predicts “short” and “ median” labels about the same frequency, and a few of “long” duration is predicted.

There are limitations for using random forest as well. First, it takes a huge amount of time for me to run the program because of its algorithm; second, random forest does **not** assume **variable dependency**. In this project, the features in the dataset are **correlated** (number of steps related to the complexity dealing with ingredients), thus, the assumption should not be made.

I run the error in respect to different numbers of trees splitted, and as shown in the plot below (Figure 3). The overall pattern of train and test set is the same, and as the number of tree splitted increases, the error converges to 0. Therefore, the more trees split, the less error, and that’s the reason I choose $n=100$ in the program. However, as the number of trees increases, the **time** needed for running the program **increases dramatically** as well, which is one of the limitations of this approach.

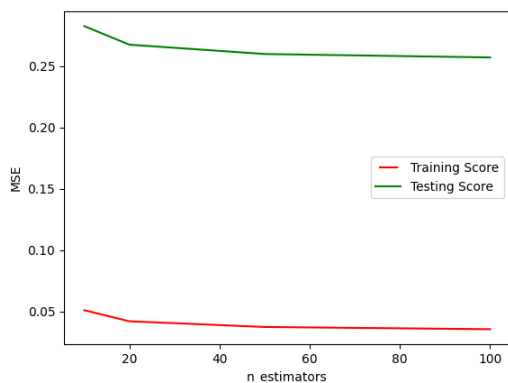
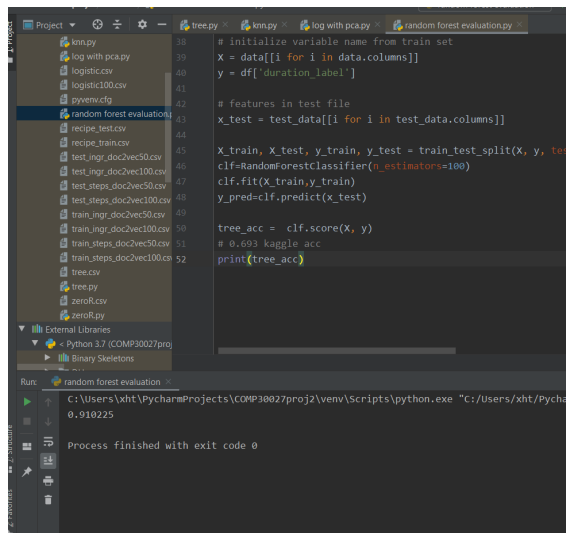


Figure 3- MSE of different numbers of trees (10, 20, 50, 100) generalized for both training and testing data. (cannot change the scale for better visualization because this plot takes half an hour to produce which is really time consuming)

The accuracy of the hold out is about 91% as shown in figure 4 which is a good predictor. There might be a possibility of overfitting here because I choose $k=100$ for minimizing the MSE.



```
38 # initialize variable name from train set
39 X = data[[i for i in data.columns]]
40 y = df['duration_label']
41
42 # features in test file
43 x_test = test_data[[i for i in test_data.columns]]
44
45 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
46 clf = RandomForestClassifier(n_estimators=100)
47 clf.fit(X_train, y_train)
48 y_pred = clf.predict(x_test)
49
50 tree_acc = clf.score(X, y)
51 # 0.603 kaggle acc
52 print(tree_acc)
```

0.910225

Process finished with exit code 0

Figure 4- accuracy score of random forest training set

K-Nearest Neighbor

The next model I tried is KNN. KNN is another supervised learning algorithm classifying data by the similarity between the data points. I chose KNN because the features in this dataset have correlation, which is an advantage of KNN over random forest. However, KNN only works better when I have a smaller dataset, not that with 40,000 dimensions of matrix, which is one of the disadvantages of using this model.

I use both converting text dataset of 50 and 100, and to reduce the number of dimensions, I tried PCA on the 50 vector dataset to see if there are any differences in accuracy (TP). The result is similar to random forest, the more features added, the lower accuracy (50vec: 67.4% VS 100vec: 66.1%), this might because of the more data added, the more noise to affect the learner, and KNN is very **sensitive to noise**. With PCA, the accuracy of 50vec drops to 63.1% compared to original data. I think this might be because the data in the original feature is already very useful, and there is no need to reduce the dimension. The result of the prediction is shown in Figure 4.

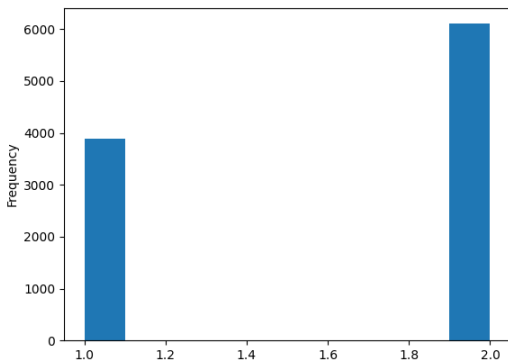


Figure 5- KNN predicts almost no label of “long” duration, and the occurrence of “median” is higher than “short”.

Unfortunately, for error analysis, when I wanted to compare how different k chosen might affect the error, and I wanted to try from 1 to 10, the program takes an hour to run the first loop ($k=1$) which is long enough for me to quit. The reason for this **running time** is obvious: there are $40,000 * 52$ numbers for the combination needed to calculate the similarity, which is a huge dataset, so it is really unrealistic to use KNN for a large dataset in real practice.

```

50 train_results = []
51 test_results = []
52
53 for k in range(10):
54     k = k+1
55     knn = KNeighborsClassifier(n_neighbors=k)
56     knn.fit(X_train, y_train)
57
58     train_results.append(mean_squared_error(y_train, knn.predict(X_train)))
59     test_results.append(mean_squared_error(y_test, knn.predict(X_test)))
60
61
62 line1 = plt.plot(k, train_results, color="r", label="Training Score")
63 line2 = plt.plot(k, test_results, color="g", label="Testing Score")
64 plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
65 plt.xlabel('Value of K for KNN')
66 plt.ylabel('MSE')
67 plt.savefig("knn vs error")
68 plt.show()

```

Figure 6- the error analysis program for KNN model which takes a long time to execute, and it is impossible to wait in practice.

Logistic Regression

I also tried **logistic regression** as a learner during the project. I used logistic regression because it can take either continuous or discrete input and produces discrete output which is the label of the dataset. As shown in figure 7, I used the mutual information as the parameter for feature selection, and the prediction accuracy (70%) is slightly better than the random forest which does not use the feature selection.

Comparing PCA and mutual information feature selection which choose the top MI in the text, PCA is more of reducing the dimension, and the second method is more of **filtering** out unnecessary attributes. Both feature selection methods result in loss of data, but PCA will depend more on the conditional dependency between attributes, while mutual information selection only focus on the stronger feature, I think both method can help the performance better.

```
# features in test file
x_test = test_data[[i for i in test_data.columns]]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
mi = SelectKBest(score_func=mutual_info_classif, k=10)
X_train = mi.fit_transform(X_train, y_train)
X_test = mi.transform(X_test)

print(X_test.shape, X_train.shape)

for feat_num in mi.get_support(indices=True):
    print(vectoriser.get_feature_names()[feat_num])

'''logisticRegr = LogisticRegression(solver = 'lbfgs')
logisticRegr.fit(X_train, y_train)
y_pred = logisticRegr.predict(X_test)
print(y_pred)'''
```

Figure 7- the use of feature selection in logistic model

Conclusion

In conclusion, random forest performs the best true positive, followed by KNN, then ZeroR. The result of the model predicts that the duration label of “short” and “median” occurs the most.

During the interpretation of the data and result analysis, I discover that depending on the characteristics of the features, the choice of model should also be carefully considered. Some of the improvements I might consider in the future would be taking the characteristics of each model into account but not only focus on the general accuracy of the prediction.

(word count: 1350)

Work Cited

(MLA format)

Majumder, Bodhisattwa Prasad, et al. “Generating Personalized Recipes from Historical User Preferences.” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on*

Natural Language Processing (EMNLP-IJCNLP), 2019,
doi:10.18653/v1/d19-1613.

“How to Plot the Error and the Tree Size of a Random Forest.” *Stack Overflow*, 10 Jan. 2020,
stackoverflow.com/questions/59678961/how-to-plot-the-error-and-the-tree-size-of-a-random-forest.

Singh, Aishwarya. “A Practical Introduction to K-Nearest Neighbors Algorithm for Regression (with Python Code).” *Analytics Vidhya*, 25 May 2020,
www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python.

“K-Nearest Neighbors (KNN) Classification Model.” *Ritchieng.Github.Io*, 2020,
www.ritchieng.com/machine-learning-k-nearest-neighbors-knn.