Yap470 Final Report

1st Selin Mergen
Senior Computer Science Student
191101027
smergen@etu.edu.tr

2nd Tarık Saraç
Junior Computer Science Student
201101018
tariksarac@etu.edu.tr

***Abstract*—This report is the final report on the project. It includes the final results obtained in general terms, the models used for these results, and comparisons with other similar projects which have been mentioned before.**

***Index Terms*—vectorization, tf-idf, OneVsRest Model, SVC, Logistic Regression, Random Forest, grid search, label encoder**

I. Introduction

Have you ever been abroad? If you have, you may have noticed that all foods are different in each country. Different countries, different cultures, different people... The food also has to be different. It shapes with the all the historical events have been through, all the lands have been settled as a society. Or interactions between different nations cause change in taste which we call cuisine in general.
According to Oxford Dictionary cuisine can be defined as a style or method of cooking, especially as characteristic of a particular country, region, or establishment. The more that we think on this topic, the more we realize what ingredients are the one which makes the read difference, not the foods themselves. Can we decide cuisine of a given recipe by only using it's ingredients? Maybe... Yes?
In this project we aim to find appropriate tags which can be it's cuisine, category or nutrition label (low fat, high protein, etc.) by analysing only the ingredients. With this project we can easily label all the foods only using it's ingredients which can be useful for many recipe/food website. By labeling data we can reduce the time taken to search or find almost every relative data for that search which also can provide a filtering. (Note: On project proposal or progress report we were only aiming to predict cuisines but after we see the tags of recipes in yummly we decided to do more, generate these tags automatically.)

II. Related Work

Before we start to talk about how we achieve our goal like every people should, we did a literature review on this topic. We realize that there are many people who have worked on this subject. But surprisingly their results were quite low than expected even in a Kaggle competition the best score was 0.82. In this medium article [1] Jaime Cheng get 0.6794 at best with random forest using recipes scraped from BBC Food and Epicurious. In this paper [2] they used the same website like us(Yummly) and they got the best results with SVM model which is 0.78228. Another related paper [3] can be from this website by Rachel Lee. She algo get 0.784 from logistic regression as best. All these methods can be change on the situation but what change the score in here is how you preprocess, normalize and vectorize your dataset. On this purpose some of these articles used Word2vec, some of them used CountVectorizer, NLP or TfTdfvectorizer methodologies.

III. Dataset

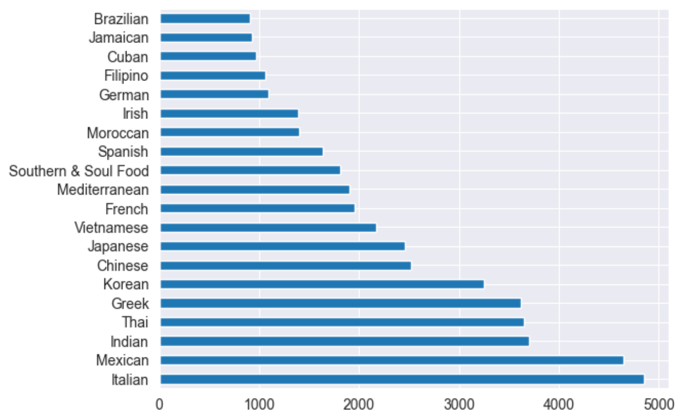
In the previous reports we mentioned to scraping our own data and on this purpose we used so many websites and so many strategies but in the end we scraped to ourselves a large and clean dataset from only one website which is yummly.com. At first we used scrapy and beautiful soup and it took days to scraped data with 160.000 recipes but only 90.000 unique. After that we changed website and used allrecipes.com but separating ingredients from their units and amounts was really hard, so we looked for other websites such as Food.com, FoodNetwork, Tasty, BBCFood.com and more until we find API of yummly.com. With the write queries we can get the links belong to recipes but still we had a problem. We had to manage infinite scrolling and this is where we used some parameters for api call. Even though we increased the recipe count per one call, yummly.com has a limit at 1000. So for each api call we only could get 1000 recipes which is a little disappointing. After dealing with this problem we finalized our dataset at 164982 unique recipes with link, name, totalTime, rating, category, cuisine, tags, Ingredients, IngredientsWithAmount and NutritionValues.
One of the biggest differences that separates us from the other studies mentioned is the dataset we have scraped. We had more labeled data than them which gives us a chance train our model in order to predict more accurate.

IV. Exploratory Data Analysis

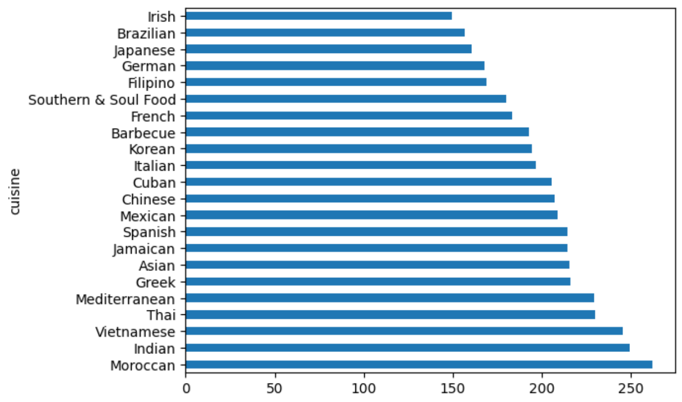
We can divide our our classification problem into 3 main parts which are 2 multilabel classification and 1 multioutput classification problem. Therefore to explore our data we need to examine our data problem specific.

A. *Cuisine Classifier*

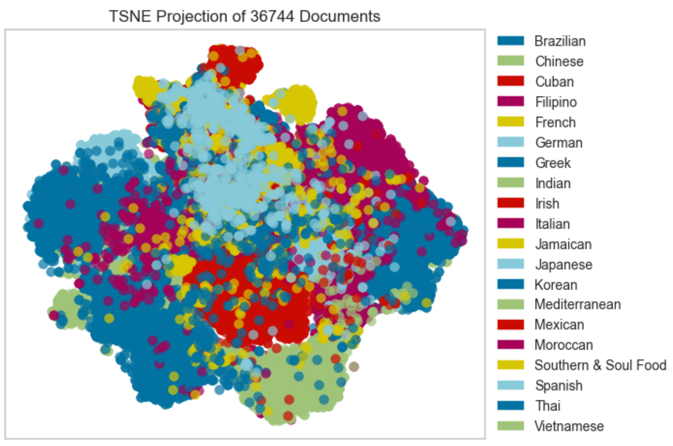
For this classification problem we have a dataset with ingredients and cuisine columns. As an output we try to classify given ingredients into one class between 20 different cuisine types.
In the figure below you can see all the cuisine types we try to label and their occurrences in the dataset.



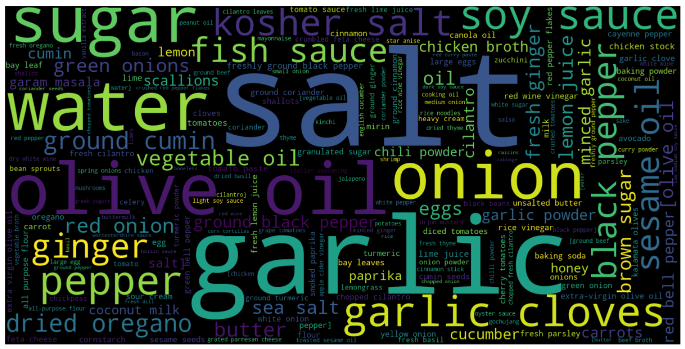
Using the columns we have we can also get the average ingredient count per cuisine statistics. The figure below represents cuisines and the average ingredients usage count.



We can also visualize cuisine similarities and relations. This figure given below shows the collisions on the same ingredients usage which also gives us the intersection between cuisines like given below:

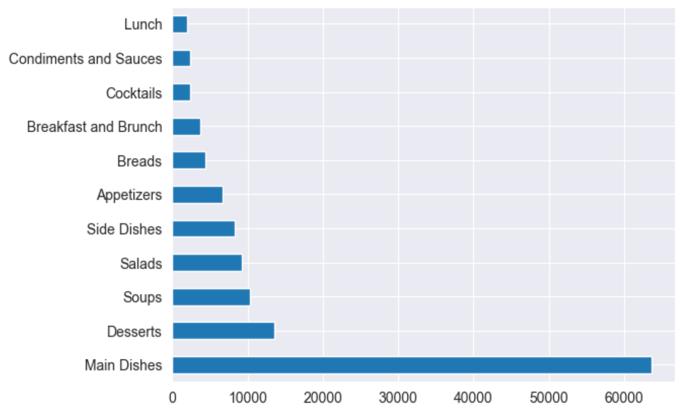


The figure below represents usage count of ingredients. The most used ingredients are bigger than the others. To visualize this we had to use CountVectorizer model and count each ingredients' usage (occurrences in whole dataset)

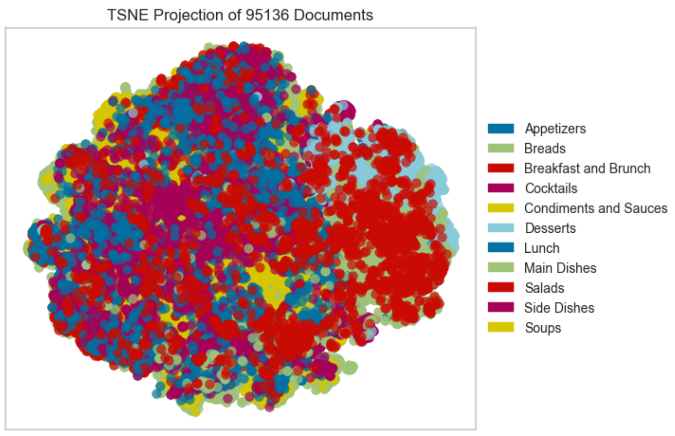


B. Category Classifier

For this classifier we have a harder problem to solve since the data is not evenly distributed. Main Dishes have +60.000 occurrences which should be dealth with.



As you can see the plot below labels are more involved. It is harder to classify recipes.



V. Method

In order to contribute to the literature, it is necessary either to train the model by following a different approach or to train a more comprehensive model by increasing the number of data. While we first examined the relative works and studies and thought about what we could improve in those models, we observed that the dataset in these studies actually formed a bottleneck, since similar data were used in most of the relative works. So that's why we determined the first step we will develop as imrpoving dataset. We found the solution on extracting our own data for this problem.
After scraping the dataset with the steps mentioned above we had a dataset with shape (346252, 9). To prepare our dataset, we trained the models in the mentioned studies with our dataset, and the results had already made a certain difference. After corrections and small changes, our dataset was ready. There was only one thing left for us to do, and that is to choose the model we will use, evaluate the preprocessing options on the data and choose the most suitable one for us.
We can divide the methodologies to be discussed hereinafter into two. Let's first consider our cuisine predictor problem. For the preprocessing step we followed the recipe written down below:

- Dropped duplicate rows in order to resolve redundency. While scraping our dataset we crawled some of the links over and over again which cause duplicated recipes. After dropping these redundent rows now our dataset is in (164982, 9) shape.
- For this problem we only need cuisine and ingredients columns. Therefore as a second step we dropped irrelevant columns such that name, link, totalTime, rating, category, tags and nutritionValues
- After previous step we had 48 different cuisine type and some of those occured only in one or two recipes which have to be dropped. We decided to drop cuisine types less than 900 occurences. After eliminating these tuples labeled with removed cuisines, we only had 20 different cuisine type which made our problem more scalable and prevent overfitting problem.
- Then we split our dataset with train_test_split method into 2 as train, test dataset
- As a final step we had to vectorize our ingredients in order to be able to train any model. So we combined all ingredients like a sentence per recipe and generated these sentences for X_train and X_test datasets. Then vectorize these sentences using tf-idf vectorizer which tells the importance of an ingredient in that document. For our target column like in many machine learning code we encode y with label encoder.

After preprocessing step we try RandomForest, SVC, LogisticRegression, AdaBoost, Decision Tree and KNN models and decide the best parameters of each using GridSearchCV. Also we used StratifiedKFold and add it to cross validation method as parameter. Finally to get accuracy scores we defined a new method like given below:

```
skfold = StratifiedKFold(n_splits=5, random_state
    =12345, shuffle=True)

def cross_validation_statistics(classifier, X, y):
    scores = cross_validate(classifier, X , y, cv=
    skfold, scoring = {'acc': 'accuracy','prec': '
    precision_weighted', 'rec': 'recall_macro'})
    print('Cross Validation Mean Fit Time:', scores[
    'fit_time'].mean())
    print('Cross Validation Mean Accuracy:', scores[
    'test_acc'].mean())
    print('Cross Validation Mean Precision:', scores
    ['test_prec'].mean())
    print('Cross Validation Mean Recall:', scores['
    test_rec'].mean())
```

Listing 1. Accuracy

Now, let's consider category prediction problem. Apart from finding cuisine, the categories of dishes can also be predicted by their ingredients. Foods can be categorized as main dishes, deserts, soups, salads, appetizers etc. For preprocessing data for category prediction we followed steps similar to cuisine prediction.
Such as:

- Removing the duplicate rows.
- Dropping all columns other than ingredient and category column.
- Removing null values.

Then we proceeded to split our dataset to train and test. After splitting our dataset, we vectorized ingredients and applied label encoder to our target variable.

After preparing our data, we trained RandomForest, SVC, LogisticRegression, AdaBoost, Decision Tree and KNN. We analyzed the fit time, predict time, accuracy, precision and recall values by applying cross validation to these models, and we determined the model that best suited our purpose.

VI. Experimental Results

Since here we've covered all sections, now we can look at the results of the models we've trained. Like the previous section we will be dividing results into 2 parts. Firstly, let's look at the accuracy table for cuisine classifier. For this problem we manage to trained 6 different model types with different parameters selected from grid search. The results are given below:

Model	Accuracy	Precision	Recall
SVM	0.89	0.89	0.86
Random Forest	0.85	0.85	0.79
Ada Boost	0.55	0.60	0.48
Logistic Regression	0.87	0.87	0.83
KNN	0.80	0.81	0.77
Decision Tree	0.74	0.74	0.69

As you can see we got the best results when we were using SVM but training and predicting SVM take too much time which is a downside of this model. The table below shows us fit and prediction times for these models. As you can see SVM takes the longest both fit and predict time between these models. This is a big turndown for us, we aim to give an output to the user as fast as it can be. So we decided to use Logistic regression model which is really close to the SVM model with lower fit and predict time. This is one of the fastest model for this problem.

Model	Fit Time	Predict Time
SVM	54.8	19.82
Random Forest	14.12	0.189
Ada Boost	6.15	0.068
Logistic Regression	1.99	0.007
KNN	0.005	9.08
Decision Tree	2.68	0.010

To classify cuisine firstly we used the base model for each model type and the values in the above tables you are seeing are belong to base models. After deciding the best model for our situation which is Logistic Regression we searched for the best parameters and after so many different combinations we settled on the parameters given below:

```

print(f"Best parameters: {lr_cv.best_params_}\n")

logisticRegression.set_params(**lr_cv.best_params_)
```

Listing 2. GridSearch

```
    Best parameters: {'C': 10.0, 'max_iter': 100, '
    multi_class': 'auto', 'penalty': 'l2', 'solver':
     'liblinear'}
```

Listing 3. Output

Let's see the accuracy and time values for the category predictor. In this problem we are trying to fit our data into 11 classes which are lunch, condiments and sauces, cocktails, breakfast and brunch, breads, appetizers, side dishes, salads, soups, desserts, main dishes.

Model	Accuracy	Precision	Recall
SVM	0.86	0.85	0.74
Random Forest	0.78	0.80	0.57
Ada Boost	0.53	0.49	0.33
Logistic Regression	0.83	0.82	0.70
KNN	0.76	0.76	0.66
Decision Tree	0.71	0.70	0.60

Like in the previous problem SVM has the best accuracy with the worst time complexity which made us select Logistic Regression again. Same as before we tried to apply grid search and find the best params for logistic regression.

Model	Fit Time	Predict Time
SVM	852.5	102.5
Random Forest	102.5	0.58
Ada Boost	14.98	0.14
Logistic Regression	2.86	0.012
KNN	0.012	59.43
Decision Tree	13.183	0.02

VII. Discussion

Labeling recipes by their categories and cuisines can be implemented with the methods we learnt in the class. Between all these previous studies we get the best results among them. In the study closest to us, Onevsrest model gives an output by enclosing the svmi after tf idf vectorizer. Their accuracy is 82%. The reason why we do not use a model like onevsrest is also related to the working time. Even though Onevsrest increases accuracy, we think it's not worth the extra time it brings in return.

Apart from predicting cuisine and category we implement, there have also been problems that we could hope to do but we could not manage. For example, we tried to predict nutrition labels such as high protein, low sugar, low fat, etc., based on the ingredient values that we mentioned at the beginning of the report. However, this problem is a multi-output classification problem, as we mentioned before. Although we succeeded in creating separate labels here, we could not fully implement the MultiOutputClassifier in our code. When we try to estimate each label separately, it gets 89% accuracy, but when combined, accuracy drops to 4% for a reason we couldn't understand. We decided not to implement this problem in our project. If we wanted to implement this there is still a way to predict nutrition labels which is to train different models for each label but it would also take so much time.

VIII. Conclusion

As a result, in this project, where we achieved to predict the cuisine and category of the food according to the food ingredients, we gathered a more comprehensive dataset in addition to other studies in the literature and obtained a higher accuracy than other models. In both the cuisine and category estimation part, we tried 6 different models and decided to use logistic regression as a result. We saved our the tf-idf vectorizers, label encodersand machine learning models as a pickle file and deployed our work on streamlit.

References

[1] Jaime Cheng. Predicting cuisine regionality based on recipe ingredients using nlp. February 2019. https://medium.com/@jaimejcheng/predicting-cuisine-regionality-based-on-recipe-ingredients-using-nlp-5262e83884eb.

[2] Rishikesh Ghewari and Sunil Raiyani. Predicting cuisine from ingredients. http://jmcauley.ucsd.edu/cse255/projects/fa15/029.pdf.

[3] Rachel Lee. Predicting cuisines based only on its ingredients using python. December 2019. https://www.uplevel.work/blog/feature-predicting-a-cuisine-based-only-on-its-ingredients-using-machine-learning.