(https://cognitiveclass.ai)

From Modeling to Evaluation

Introduction

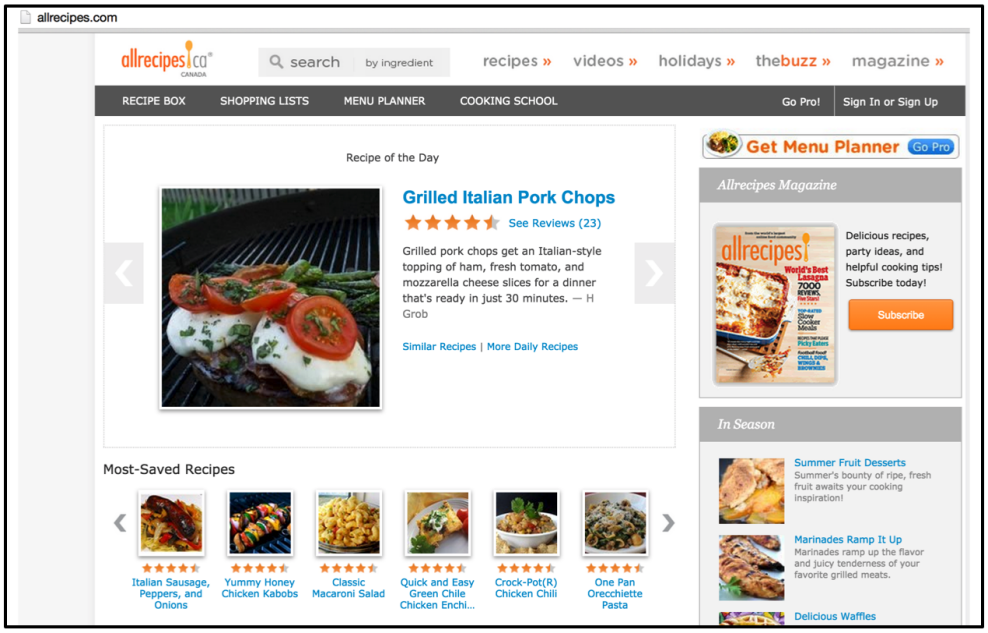
In this lab, we will continue learning about the data science methodology, and focus on the **Modeling** and **Evaluation** stages.

Table of Contents

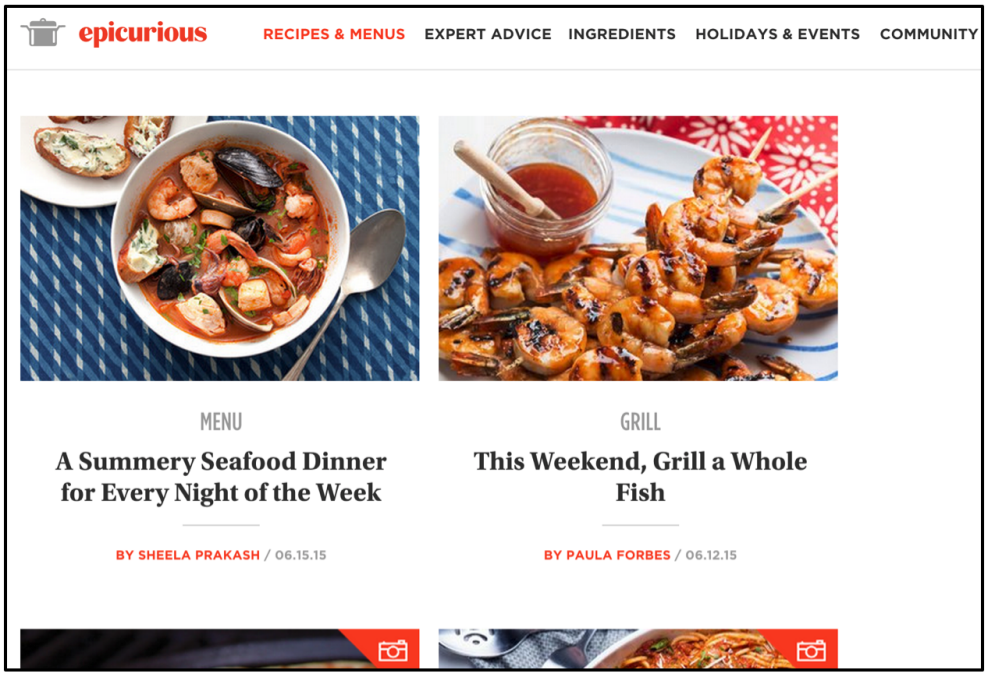
Recap

In Lab **From Understanding to Preparation**, we explored the data and prepared it for modeling.

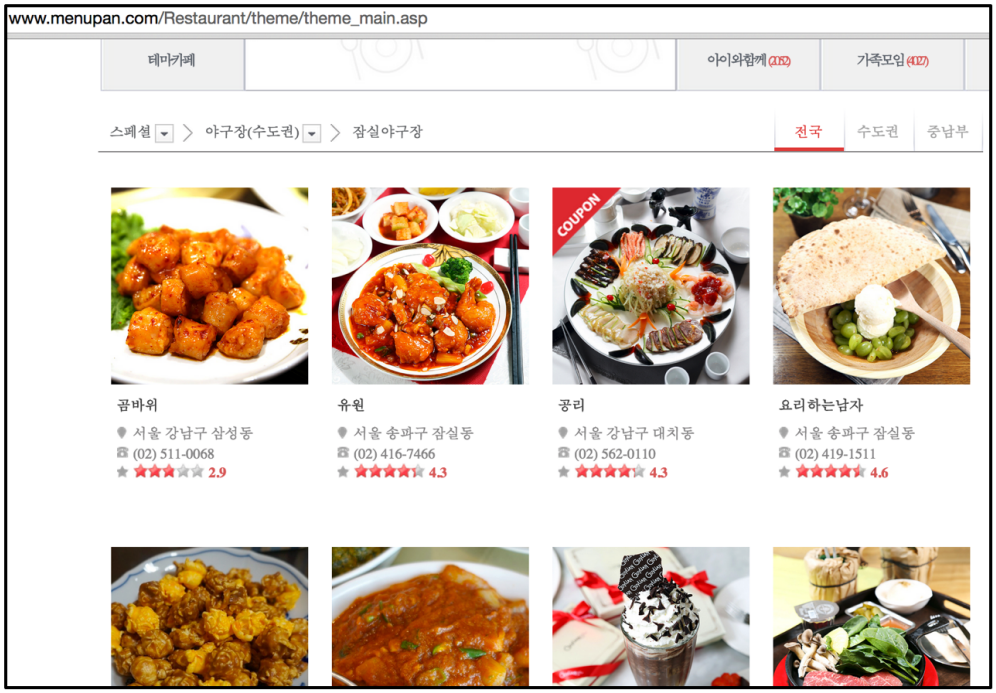
The data was compiled by a researcher named Yong-Yeol Ahn, who scraped tens of thousands of food recipes (cuisines and ingredients) from three different websites, namely:



www.allrecipes.com



www.epicurious.com



www.menupan.com

For more information on Yong-Yeol Ahn and his research, you can read his paper on [Flavor Network and the Principles of Food Pairing (http://yongyeol.com/papers/ahn-flavornet-2011.pdf)](http://yongyeol.com/papers/ahn-flavornet-2011.pdf).

Important note: Please note that you are not expected to know how to program in Python. This lab is meant to illustrate the stages of modeling and evaluation of the data science methodology, so it is totally fine if you do not understand the individual lines of code. We have a full course on programming in Python, **[Python for Data Science (http://cocl.us/PY0101EN_DS0103EN_LAB4_PYTHON_Coursera)](http://cocl.us/PY0101EN_DS0103EN_LAB4_PYTHON_Coursera)**, which is also offered on Coursera. So make sure to complete the Python course if you are interested in learning how to program in Python.

Using this notebook:

To run any of the following cells of code, you can type **Shift + Enter** to excute the code in a cell.

Download the library and dependencies that we will need to run this lab.

```
In [1]:
```

We already placed the data on an IBM server for your convenience, so let's download it from server and read it into a dataframe called **recipes**.

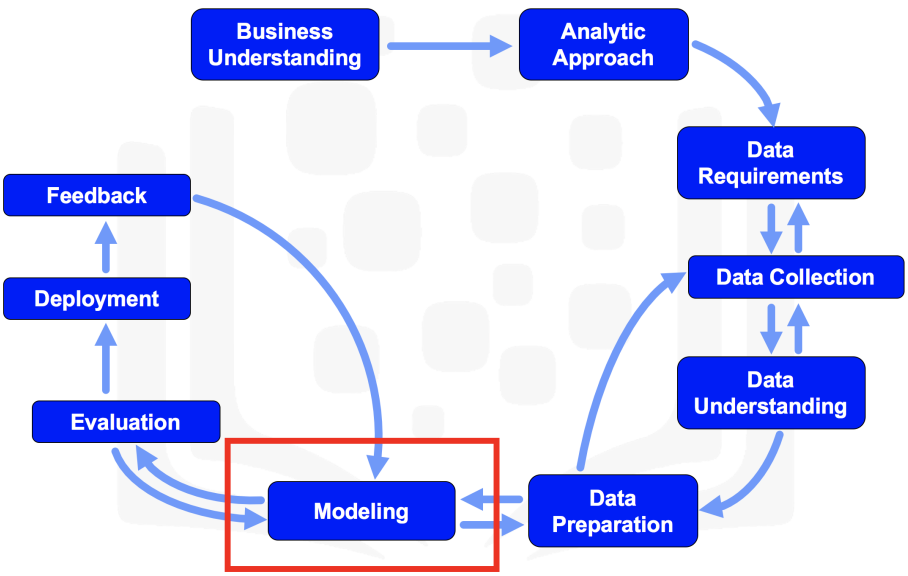
In [2]:

```
Data read into dataframe!
```

We will repeat the preprocessing steps that we implemented in Lab **From Understanding to Preparation** in order to prepare the data for modeling. For more details on preparing the data, please refer to Lab **From Understanding to Preparation**.

In [3]:

Data Modeling



Download and install more libraries and dependies to build decision trees.

In [4]:

```
Solving environment: done

## Package Plan ##

  environment location: /home/jupyterlab/conda

  added / updated specs:
    - python-graphviz


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    python-3.6.7               |       h0371630_0        34.3 MB
    cryptography-2.4.2         |   py36h1ba5d50_0         618 KB
    python-graphviz-0.8.4      |           py36_1          27 KB
    grpcio-1.16.1              |   py36hf8bcb03_1         1.1 MB
    libarchive-3.3.3           |       h5d8350f_5         1.5 MB
    certifi-2018.11.29         |           py36_0         146 KB
    ------------------------------------------------------------
                                           Total:        37.6 MB

The following NEW packages will be INSTALLED:

    python-graphviz: 0.8.4-py36_1

The following packages will be UPDATED:

    cryptography:    2.3.1-py36hdffb7b8_0    conda-forge --> 2.4.2-py3
6h1ba5d50_0
    curl:            7.63.0-h74213dd_0       conda-forge --> 7.63.0-hb
c83047_1000
    grpcio:          1.16.0-py36hd60e7a3_0   conda-forge --> 1.16.1-py
36hf8bcb03_1
    libarchive:      3.3.3-h823be47_0        conda-forge --> 3.3.3-h5d
8350f_5
    libcurl:         7.63.0-hbdb9355_0       conda-forge --> 7.63.0-h2
0c2e04_1000
    libssh2:         1.8.0-h5b517e9_3        conda-forge --> 1.8.0-h1b
a5d50_4
    openssl:         1.0.2p-h470a237_1       conda-forge --> 1.1.1a-h7
b6447c_0
    pycurl:          7.43.0.2-py36hb7f436b_0             --> 7.43.0.2-
py36h1ba5d50_0
    python:          3.6.6-h5001a0f_3        conda-forge --> 3.6.7-h03
71630_0
    qt:              5.9.6-h8703b6f_2                    --> 5.9.7-h58
67ecd_1

The following packages will be DOWNGRADED:

    ca-certificates: 2018.11.29-ha4d7672_0   conda-forge --> 2018.03.0
7-0
    certifi:         2018.11.29-py36_1000    conda-forge --> 2018.11.2
9-py36_0
    conda:           4.5.12-py36_1000        conda-forge --> 4.5.12-py
```

```
36_0
    krb5:            1.16.2-hbb41f41_0       conda-forge --> 1.16.1-h1
73b8e3_7


Downloading and Extracting Packages
python-3.6.7         | 34.3 MB   | ##################################
## | 100%
cryptography-2.4.2   | 618 KB    | ##################################
## | 100%
python-graphviz-0.8. | 27 KB     | ##################################
## | 100%
grpcio-1.16.1        | 1.1 MB    | ##################################
## | 100%
libarchive-3.3.3     | 1.5 MB    | ##################################
## | 100%
certifi-2018.11.29   | 146 KB    | ##################################
## | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Check the data again!

In [5]:

Out[5]:

	cuisine	almond	angelica	anise	anise_seed	apple	apple_brandy	apricot	armagnac	a
0	vietnamese	0	0	0	0	0	0	0	0	
1	vietnamese	0	0	0	0	0	0	0	0	
2	vietnamese	0	0	0	0	0	0	0	0	
3	vietnamese	0	0	0	0	0	0	0	0	
4	vietnamese	0	0	0	0	0	0	0	0	

[bamboo_tree] Only Asian and Indian Cuisines

Here, we are creating a decision tree for the recipes for just some of the Asian (Korean, Japanese, Chinese, Thai) and Indian cuisines. The reason for this is because the decision tree does not run well when the data is biased towards one cuisine, in this case American cuisines. One option is to exclude the American cuisines from our analysis or just build decision trees for different subsets of the data. Let's go with the latter solution.

Let's build our decision tree using the data pertaining to the Asian and Indian cuisines and name our decision tree *bamboo_tree*.

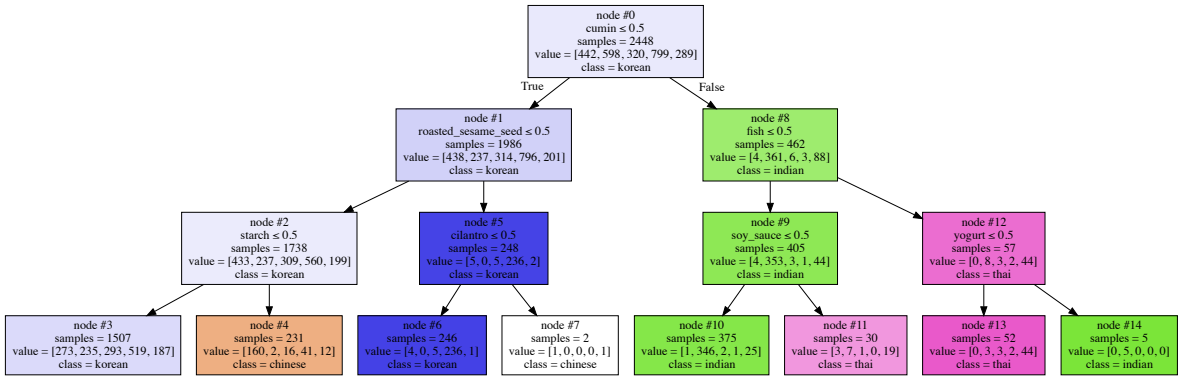
In [6]:

```
Decision tree model saved to bamboo_tree!
```

Let's plot the decision tree and examine how it looks like.

In [7]:

Out[7]:



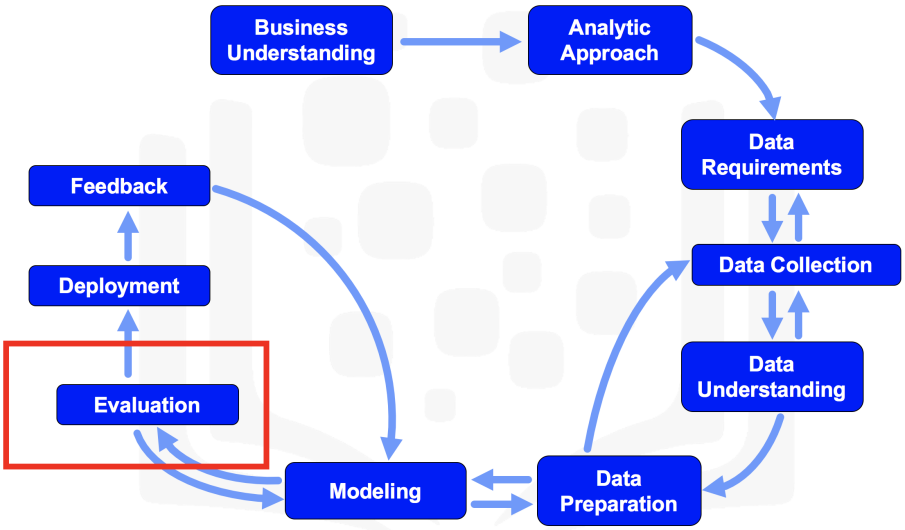
The decision tree learned:

- If a recipe contains *cumin* and *fish* and **no** *yoghurt*, then it is most likely a **Thai** recipe.
- If a recipe contains *cumin* but **no** *fish* and **no** *soy_sauce*, then it is most likely an **Indian** recipe.

You can analyze the remaining branches of the tree to come up with similar rules for determining the cuisine of different recipes.

Feel free to select another subset of cuisines and build a decision tree of their recipes. You can select some European cuisines and build a decision tree to explore the ingredients that differentiate them.

Model Evaluation



To evaluate our model of Asian and Indian cuisines, we will split our dataset into a training set and a test set. We will build the decision tree using the training set. Then, we will test the model on the test set and compare the cuisines that the model predicts to the actual cuisines.

Let's first create a new dataframe using only the data pertaining to the Asian and the Indian cuisines, and let's call the new dataframe **bamboo**.

```
In [8]:
```

Let's see how many recipes exist for each cuisine.

In [9]:

```
Out[9]: korean      799
        indian      598
        chinese     442
        japanese    320
        thai        289
        Name: cuisine, dtype: int64
```

Let's remove 30 recipes from each cuisine to use as the test set, and let's name this test set **bamboo_test**.

In [10]:

Create a dataframe containing 30 recipes from each cuisine, selected randomly.

In [11]:

Check that there are 30 recipes for each cuisine.

In [12]:

```
Out[12]: thai        30
         chinese     30
         indian      30
         japanese    30
         korean      30
         Name: cuisine, dtype: int64
```

Next, let's create the training set by removing the test set from the **bamboo** dataset, and let's call the training set **bamboo_train**.

In [13]:

Check that there are 30 *fewer* recipes now for each cuisine.

In [14]:

```
Out[14]: korean      769
         indian      568
         chinese     412
         japanese    290
         thai        259
         Name: cuisine, dtype: int64
```

Let's build the decision tree using the training set, **bamboo_train**, and name the generated tree **bamboo_train_tree** for prediction.

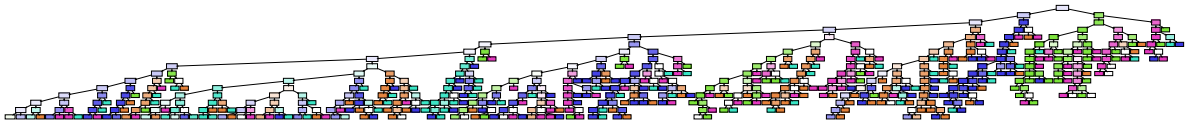
In [15]:

```
Decision tree model saved to bamboo_train_tree!
```

Let's plot the decision tree and explore it.

In [16]:

Out[16]:



Now that we defined our tree to be deeper, more decision nodes are generated.

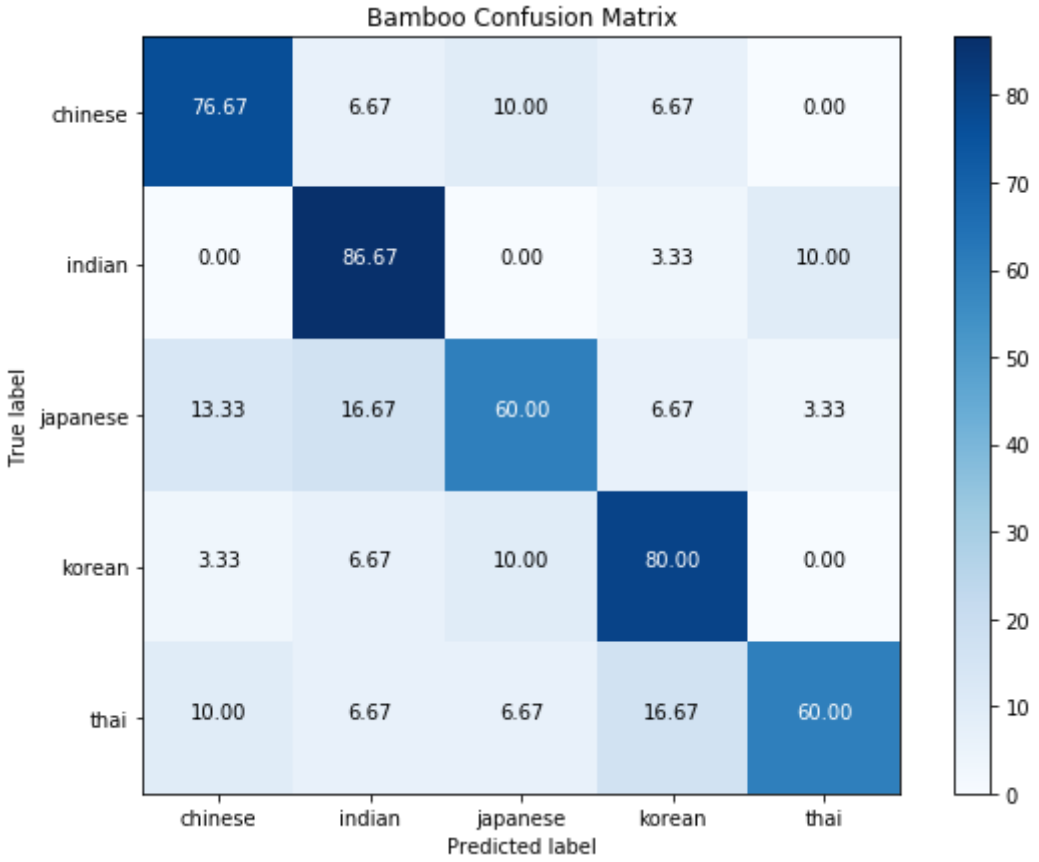
Now let's test our model on the test data.

In [17]:

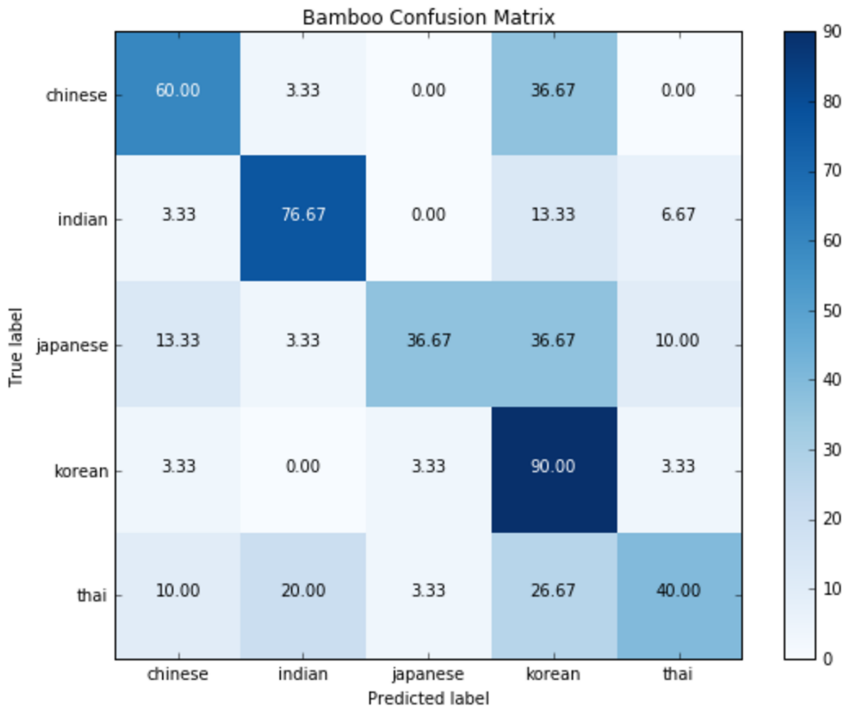
To quantify how well the decision tree is able to determine the cuisine of each recipe correctly, we will create a confusion matrix which presents a nice summary on how many recipes from each cuisine are correctly classified. It also sheds some light on what cuisines are being confused with what other cuisines.

So let's go ahead and create the confusion matrix for how well the decision tree is able to correctly classify the recipes in **bamboo_test**.

In [18]:



After running the above code, you should get a confusion matrix similar to the following:



The rows represent the actual cuisines from the dataset and the columns represent the predicted ones. Each row should sum to 100%. According to this confusion matrix, we make the following observations:

- Using the first row in the confusion matrix, 60% of the **Chinese** recipes in **bamboo_test** were correctly classified by our decision tree whereas 37% of the **Chinese** recipes were misclassified as **Korean** and 3% were misclassified as **Indian**.
- Using the Indian row, 77% of the **Indian** recipes in **bamboo_test** were correctly classified by our decision tree and 3% of the **Indian** recipes were misclassified as **Chinese** and 13% were misclassified as **Korean** and 7% were misclassified as **Thai**.

Please note that because decision trees are created using random sampling of the datapoints in the training set, then you may not get the same results every time you create the decision tree even using the same training set. The performance should still be comparable though! So don't worry if you get slightly different numbers in your confusion matrix than the ones shown above.

Using the reference confusion matrix, how many **Japanese** recipes were correctly classified by our decision tree?

Your Answer: 36.67

Double-click **here** for the solution.

Also using the reference confusion matrix, how many **Korean** recipes were misclassified as **Japanese**?

Your Answer: 3.33

Double-click **here** for the solution.

What cuisine has the least number of recipes correctly classified by the decision tree using the reference confusion matrix?

Your Answer: Japanese

Double-click **here** for the solution.

Thank you for completing this lab!

This notebook was created by Alex Aklson (https://www.linkedin.com/in/aklson/). We hope you found this lab session interesting. Feel free to contact us if you have any questions!

This notebook is part of a course on **Coursera** called *Data Science Methodology*. If you accessed this notebook outside the course, you can take this course, online by clicking here (https://cocl.us/DS0103EN_Coursera_LAB4).

Copyright © 2018 Cognitive Class (https://cognitiveclass.ai/?utm_source=bducopyrightlink&utm_medium=dswb&utm_campaign=bdu). This notebook and its source code are released under the terms of the MIT License (https://bigdatauniversity.com/mit-license/).