**Data preprocessing:**

I used the pandas DataFrame to handle the data.

The data has 39774 recipes with 6714 different ingredients. Since each recipe has a different number of ingredients, I converted the data into a sparsely filled binary array, with the ingredients as columns for each recipe row, where the ingredients present in the recipe are marked as 1’s and the ingredients not used in the recipe are marked as 0’s.

This is an example of the data stored for a single recipe. Note cuisine and id at the end of the list.

romaine lettuce 1

black olives 1

grape tomatoes 1

garlic 1

pepper 1

purple onion 1

seasoning 1

garbanzo beans 1

feta cheese crumbles 1

plain flour 0

ground pepper 0

chicken livers 0

water 0

...

turkey giblet stock 0

tomato garlic pasta sauce 0

crushed cheese crackers 0

cuisine greek

id 10259

The training data was then split up in the ratio of 70:30 for training and testing respectively. Thus each of the models was trained on 70% of the training data and scored for accuracy with the remaining 30%.

**Machine Learning:**

The following algorithms were used on this data.

|  |  |  |
| --- | --- | --- |
| Algorithm | Training data score | Test data score |
| DecisionTreeClassifier | 0.999784475017 | 0.604775869292 |
| GaussianNB | 0.5444161069 | 0.383996648513 |
| RandomForestClassifier | 0.999784475017 | 0.71193967323 |
| LogisticRegression | 0.87603721398 | 0.780058651026 |
| MLPClassifier | 0.726211430008 | 0.70297444491 |

**Parameter Tuning:**

Since Logistic Regression and Random Forest Classifier performed the best, I then decided to check the results of those with parameter tuning.

Random Forest Classifier:

These are test scores with setting the

* ‘criterion’ parameter - ‘gini’ and ‘entropy’
* ‘n\_estimators’ parameter – 10, 100, 500

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10 | 100 | 500 |
| Gini | 0.663259321324 | 0.708001675744 | 0.715961457897 |
| Entropy | 0.627398408044 | 0.675994972769 | 0.684625052367 |

Logistic Regression:

|  |  |
| --- | --- |
| C values | Test scores |
| 0.001 | 0.483200670297 |
| 0.01 | 0.628906577294 |
| 0.1 | 0.739673229996 |
| 1 | 0.780058651026 |
| 10 | 0.771260997067 |
| 100 | 0.736908253037 |

Since Logistic Regression with parameter C=1 performed the best, I then tried out C values of 2,3,4,5,6,7.

|  |  |
| --- | --- |
| C values | Test scores |
| 2 | 0.782739840804 |
| 3 | 0.782823627985 |
| 4 | 0.780477586929 |
| 5 | 0.778801843318 |
| 6 | 0.776874738165 |
| 7 | 0.775785504818 |

Logistic Regression with C=3 was selected to run on the actual test JSON file and the results were saved in **output.csv**

**Tools Used:**

* Jupyter Notebook
* Python 2
* Libraries
  + pandas,
  + scikit-learn