CPSC 483

Data Mining & Pattern Recognition

Graduate Project

Topic: Type of Cuisine

Vidhi Patel
893375105
vidhi.patel38@csu.fullerton.edu

Table of Contents

INTRODUCTION

The aim of the project is to predict different types of cuisines from the given set of ingredients. In this project, the dataset used was available in **Kaggle.** The training dataset is approximately 38,000 which is categorized in 20 different types of cuisines. The test dataset is approximately 10,000. Therefore, the aim of the project is to run several types of algorithms in python as well as in RapidMiner tool to predict the cuisine of the test dataset based on the given set of ingredients.

Environment

1. **Operating System**: OS X EI CAPITAN
2. **Tools**: PYTHON, RAPIDMINER
3. **Libraries**: pandas, sklearn, nltk
4. **Algorithms**: Naïve Bayes Classifier, K- Nearest Neighbor Classifier, Logistic Regression
5. **Data Set**: . https://www.kaggle.com/c/whats-cooking/data

DATA PREPARATION

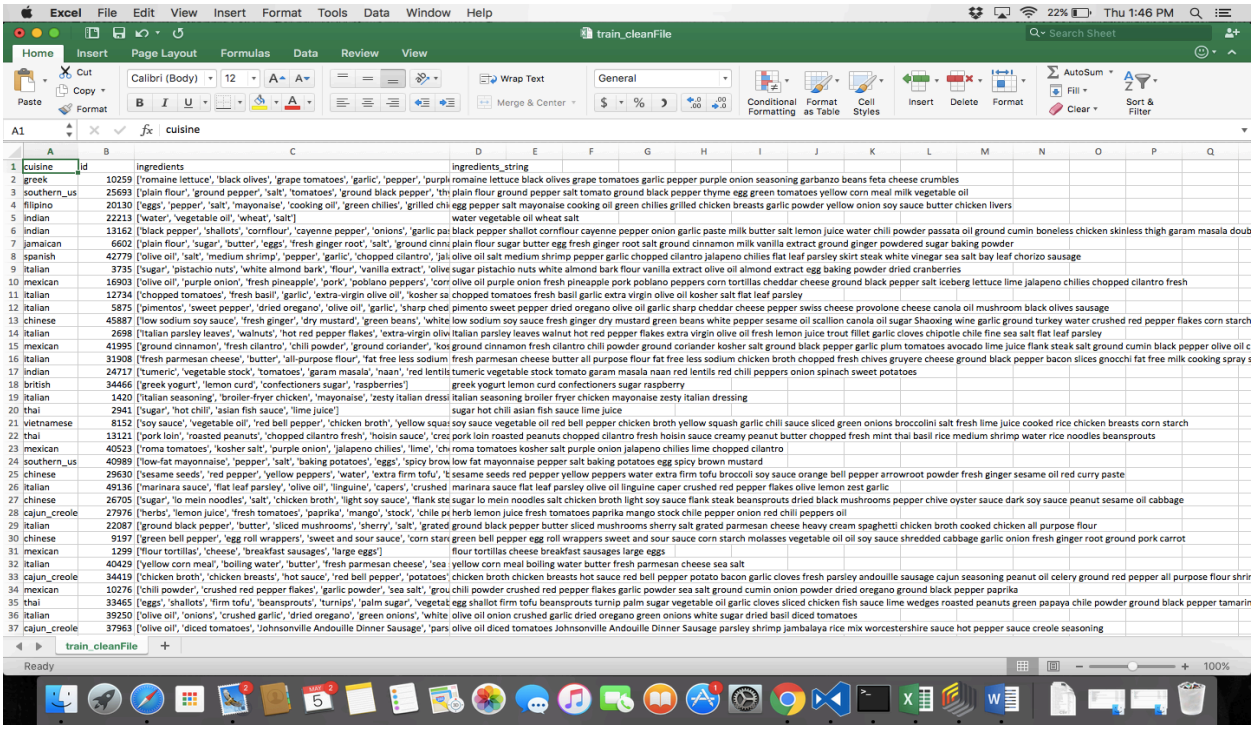
So the first task in Data Mining is to prepare the data before building the mining model to carry out predictions. Preparing the dataset for predicton means cleaning the data. Data can be scattered and stored in different formats or may also contain inconsistencies such as incorrect or missing values. The dataset available from **Kaggle** was in JSON format. So, a python script was written to convert and clean the data into CSV format.
The JSON file was imported using the PANDAS library in python. After importing the JSON file, the list of ingredients was converted into a single string by using join() and strip() functions on the ingredients in the training set. But it was not properly cleaned as it still contained inverted quotes and the square brackets. Therefore, the second option was to use the library available in python called WordNetLemmatizer to clean the data and was converted into single string. So now the training data contains three main attributes id, attributes and ingredients. And the test data contains id and ingredients. The data is then vectorized by importing the TfidfVectorizer library from the feature_extraction.text package which combines all the options of CountVectorizer and TfidTransformer in a single model. Once the data is vectored, the data is then transformed accordingly and the target that is what is to be predicted for the test data from the training data set is performed.


```
{
      "id": 10259,
  "cuisine": "greek",
    "ingredients": [
  "romaine lettuce",
    "black olives",
  "grape tomatoes",
        "garlic",
]
}
```


eg: of the JSON file.

```
trainfile = pd.read_json("/Users/nikunjpatel/Desktop/Data Mining/DataMining Grad Project/train.json")

trainfile['ingredients_clean_string'] = [' '.join(z).strip() for z in trainfile['ingredients']]
trainfile['ingredients_string'] = [' '.join([WordNetLemmatizer().lemmatize(re.sub('[^A-Za-z]', ' ', line)) for line in
lists]).strip() for lists in trainfile['ingredients']]

testfile = pd.read_json("/Users/nikunjpatel/Desktop/Data Mining/DataMining Grad Project/test.json")

testfile['ingredients_clean_string'] = [' '.join(z).strip() for z in testfile['ingredients']]
testfile['ingredients_string'] = [' '.join([WordNetLemmatizer().lemmatize(re.sub('[^A-Za-z]', ' ', line)) for line in
lists]).strip() for lists in testfile['ingredients']]

corpustr = trainfile['ingredients_string']
vectorizertr = TfidfVectorizer(stop_words='english',
                    ngram_range = ( 1 , 1 ),analyzer="word",
                    max_df = .57 , binary=False , token_pattern=r'\w+' , sublinear_tf=False)
tfidftr = vectorizertr.fit_transform(corpustr).todense()
corpusts = testfile['ingredients_string']
vectorizerts = TfidfVectorizer(stop_words='english')
tfidfts = vectorizertr.transform(corpusts)

predictor_train = tfidftr

target_data = trainfile['cuisine']

predictor_test = tfidfts

trainfile.to_csv("train_cleanFile.csv", index=False)
testfile.to_csv("test_cleanFile.csv", index=False)
```

Code for Data Preparation



A file obtained after Data cleaning.

BUILDING THE MODEL

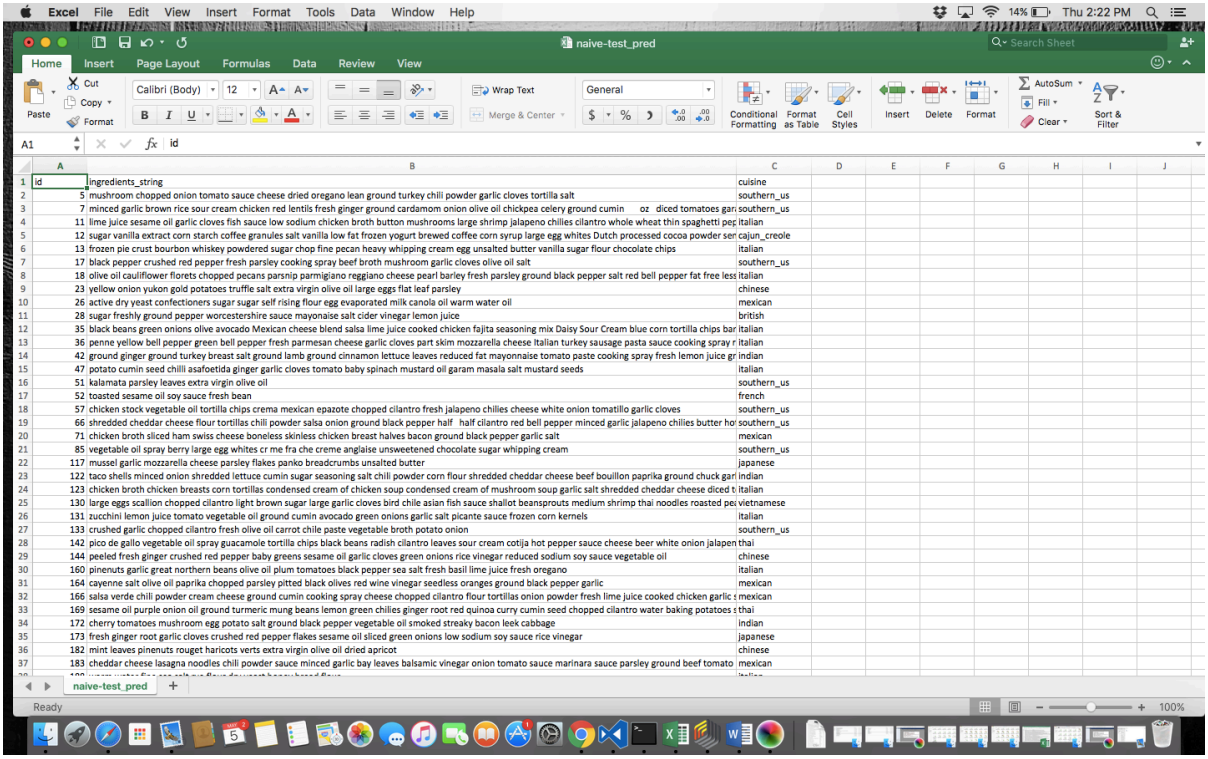
After data preparation, the next step is to build a model from the training data set so as to classify and predict the cuisine from the set of ingredients. 2-3 three models are used in order to compare the results of the predictions.

Naïve Bayes in Python

The cuisine of the test data was predicted importing the library of naïve bayes from the package called sklearn. The naïve bayes model was trained using the training dataset which was prepared from the data preparation and then prediction was done for the test dataset.

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nbClassifier = nb.fit(predictor_train,target_data);
predictions = nbClassifier.predict(predictor_test)
testfile[['id' , 'ingredients_string']].to_csv("Test_submission-lr11.csv")
testfile['cuisine'] = predictions
testfile = testfile.sort('id' , ascending=True)
testfile[['id', 'cuisine' ]].to_csv("naive-test_pred.csv", index=False)
```

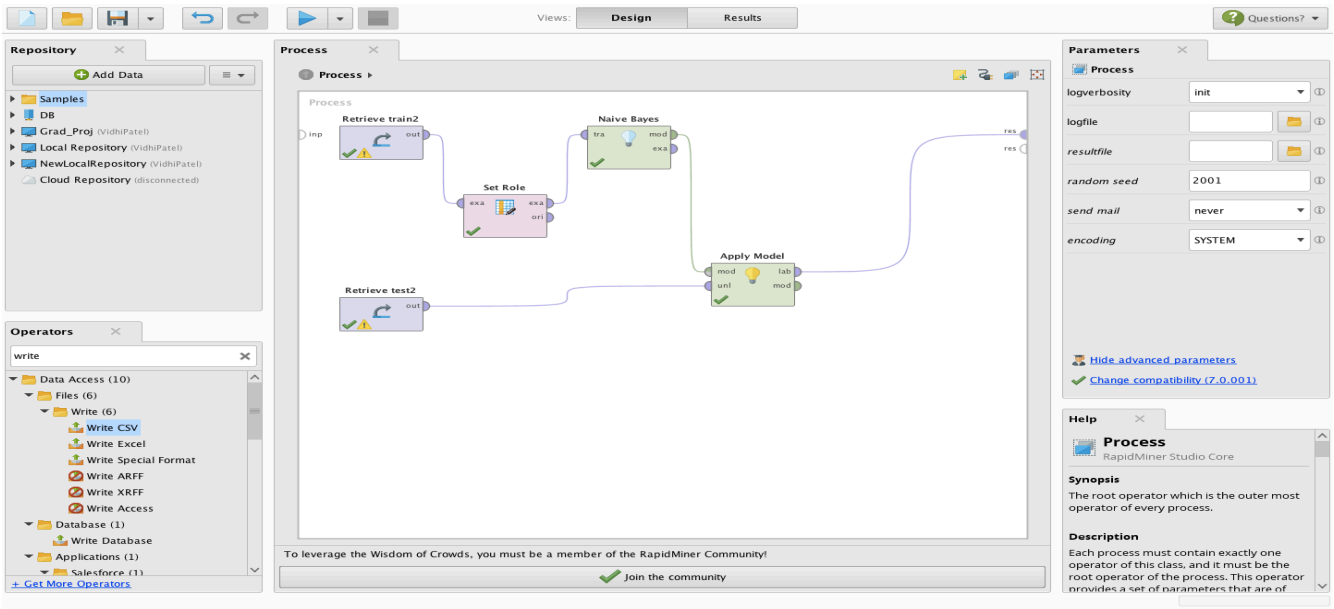
Code for performing Naïve Bayes.



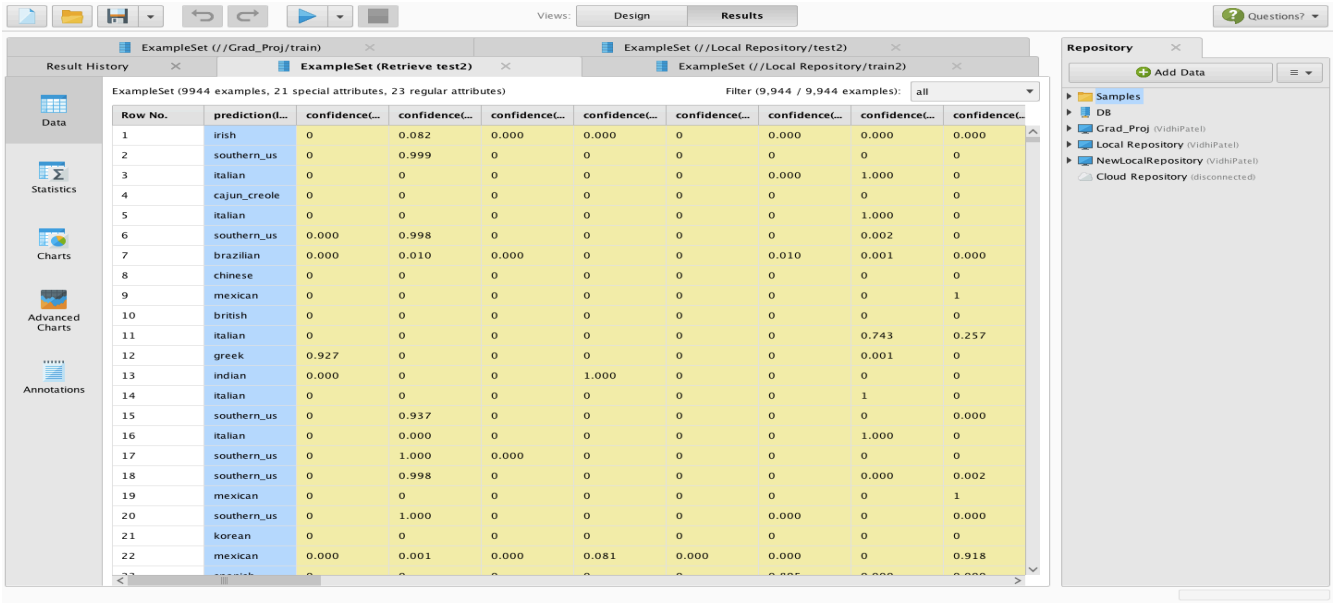
Result Predicted using Naïve Bayes in python

Naïve Bayes in RapidMiner

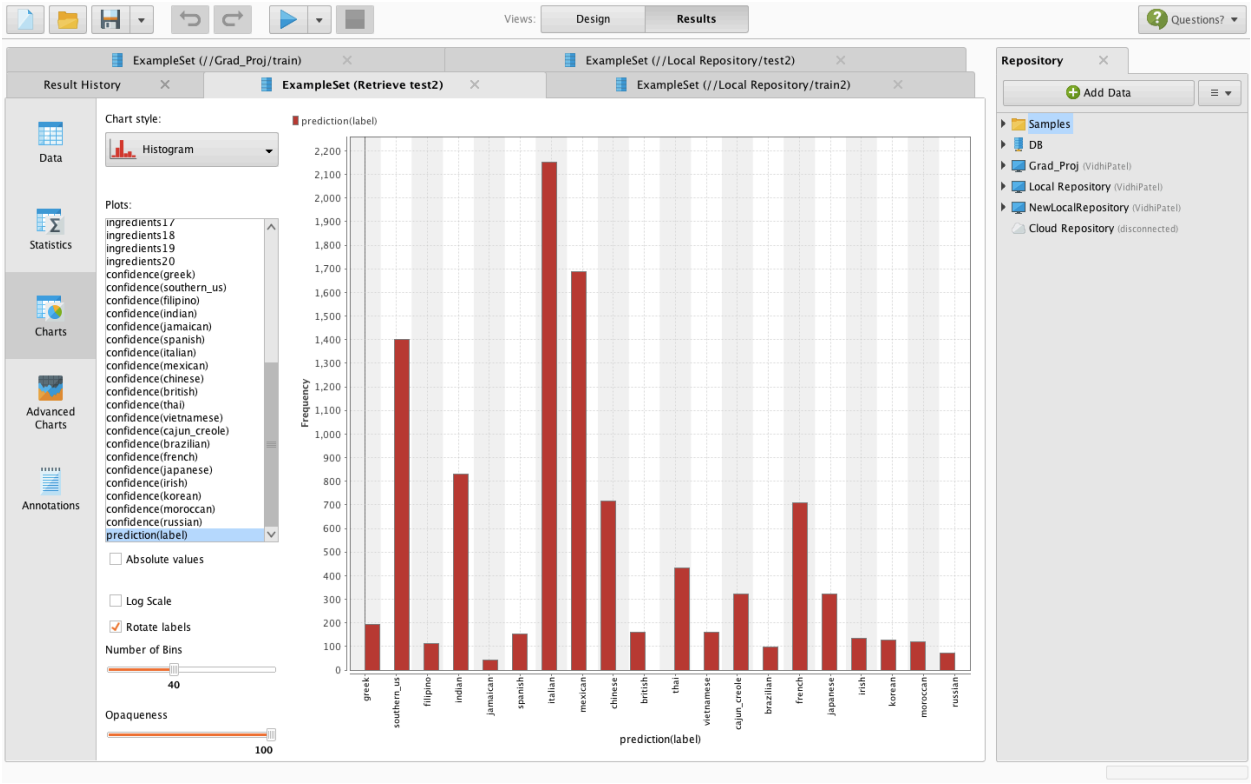
Steps to perform Naïve Bayes in Rapid Miner:-
1. Select the Training Set and then set a label to the training set by selecting set role from the operators.
2. Select the Naïve Bayes model from the operators.
3. Select the Test Set from the Database.
4. Select the Apply Model from the operators.
5. Connect the Output of the Naïve Bayes and the test data set to the Apply model and run the model.



Naïve Bayes using Rapid Miner



Output of Naïve Bayes Using RapidMiner



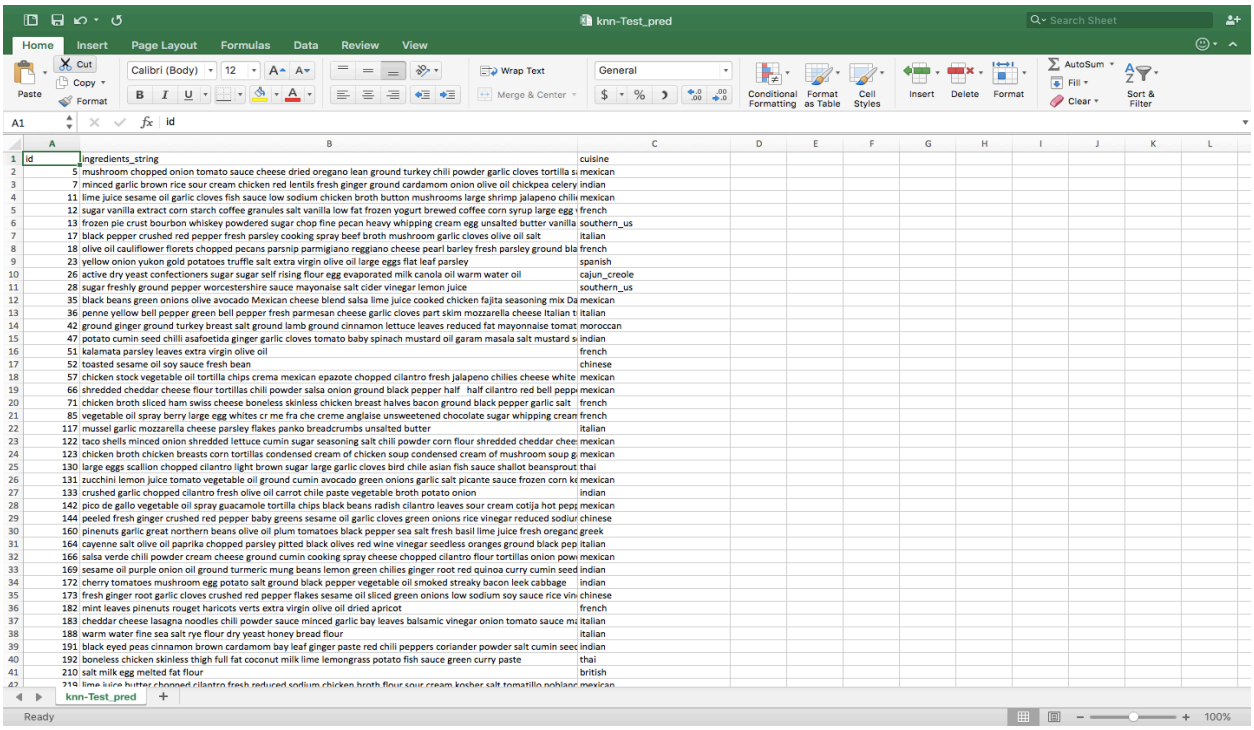
Bar Chart for the prediction of cuisines.

KNN in Python

The cuisine of the test data was predicted importing the library of KNN from the package called sklearn. The KNN model was trained using the training dataset which was prepared from the data preparation and then prediction was done for the test dataset.

```
#KNN theorem
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3,algorithm='brute')
classifier=neigh.fit(predictor_train,target_data)
predictions=classifier.predict(predictor_test)
testfile['cuisine'] = predictions
testfile = testfile.sort('id', ascending=True)
testfile[['id','ingredients_string' ,'cuisine' ]].to_csv("knn-Test_pred.csv", index=False)
```

Code for performing KNN.



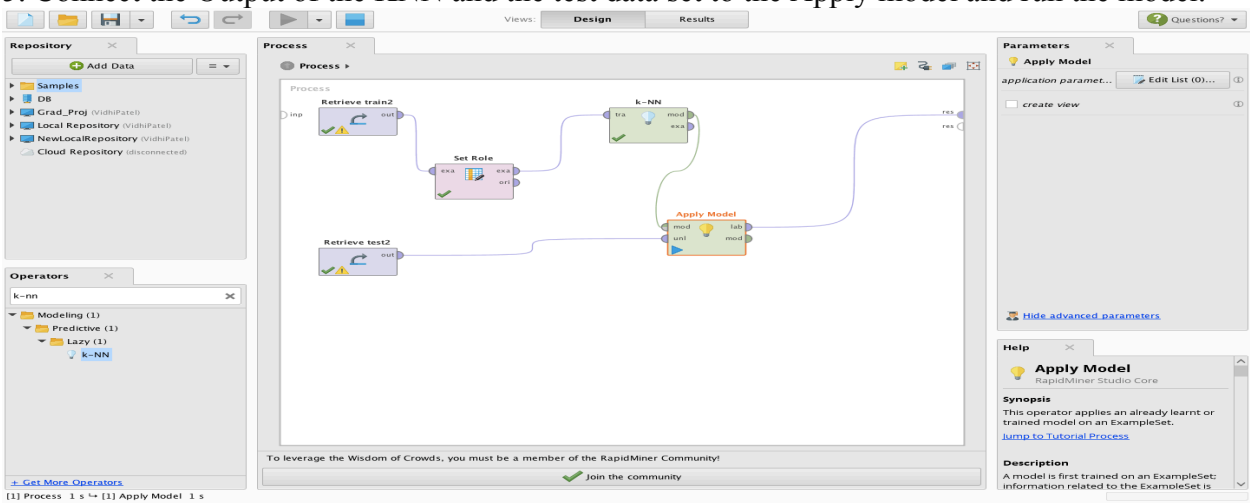
id	ingredients_string	cuisine
5	mushroom chopped onion tomato sauce cheese dried oregano lean ground turkey chili powder garlic cloves tortilla s	mexican
7	minced garlic brown rice sour cream chicken red lentils fresh ginger ground cardamom onion olive oil chickpea celery	indian
11	lime juice sesame oil garlic cloves fish sauce low sodium chicken broth button mushrooms large shrimp jalapeno chili	mexican
12	sugar vanilla extract corn starch coffee granules salt vanilla low fat frozen yogurt brewed coffee corn syrup large egg	french
13	frozen pie crust bourbon whiskey powdered sugar chop fine pecan heavy whipping cream egg unsalted butter vanilla	southern_us
17	black pepper crushed red pepper fresh parsley cooking spray beef broth mushroom garlic cloves olive oil salt	italian
18	olive oil cauliflower florets chopped pecans parsnip parmigiano reggiano cheese pearl barley fresh parsley ground bla	french
23	yellow onion yukon gold potatoes truffle salt extra virgin olive oil large eggs flat leaf parsley	spanish
26	active dry yeast confectioners sugar sugar self rising flour egg evaporated milk canola oil warm water oil	cajun_creole
28	sugar freshly ground pepper worcestershire sauce mayonaise salt cider vinegar lemon juice	southern_us
35	black beans green onions olive avocado Mexican cheese blend salsa lime juice cooked chicken fajita seasoning mix Da	mexican
36	penne yellow bell pepper green bell pepper fresh parmesan cheese garlic cloves part skim mozzarella cheese Italian t	italian
42	ground ginger ground turkey breast salt ground lamb ground cinnamon lettuce leaves reduced fat mayonnaise tomat	moroccan
47	potato cumin seed chilli asafoetida ginger garlic cloves tomato baby spinach mustard oil garam masala salt mustard s	indian
51	kalamata parsley leaves extra virgin olive oil	french
52	toasted sesame oil soy sauce fresh bean	chinese
57	chicken stock vegetable oil tortilla chips crema mexican epazote chopped cilantro fresh jalapeno chilies cheese white	mexican
66	shredded cheddar cheese flour tortillas chili powder salsa onion ground black pepper half half cilantro red bell pepp	mexican
71	chicken broth sliced ham swiss cheese boneless skinless chicken breast halves bacon ground black pepper garlic salt	french
85	vegetable oil spray berry large egg whites cr me fra che creme anglaise unsweetened chocolate sugar whipping crean	french
117	mussel garlic mozzarella cheese parsley flakes panko breadcrumbs unsalted butter	italian
122	taco shells minced onion shredded lettuce cumin sugar seasoning salt chili powder corn flour shredded cheddar chee	mexican
123	chicken broth chicken breasts corn tortillas condensed cream of chicken soup condensed cream of mushroom soup g	mexican
130	large eggs scallion chopped cilantro light brown sugar large garlic cloves bird chile asian fish sauce shallot beansprout	thai
131	zucchini lemon juice tomato vegetable oil ground cumin avocado green onions garlic salt picante sauce frozen corn ke	mexican
133	crushed garlic chopped cilantro fresh olive oil carrot chile paste vegetable broth potato onion	indian
142	pico de gallo vegetable oil spray guacamole tortilla chips black beans radish cilantro leaves sour cream cotija hot peps	mexican
144	peeled fresh ginger crushed red pepper baby greens sesame oil garlic cloves green onions rice vinegar reduced sodiur	chinese
160	pinenuts garlic great northern beans olive oil plum tomatoes black pepper sea salt fresh basil lime juice fresh oregano	greek
164	cayenne salt olive oil paprika chopped parsley pitted black olives red wine vinegar seedless oranges ground black pep	italian
166	salsa verde chili powder cream cheese ground cumin cooking spray cheese chopped cilantro flour tortillas onion pow	mexican
169	sesame oil purple onion oil ground turmeric mung beans lemon green chilies ginger root red quinoa curry cumin seed	indian
172	cherry tomatoes mushroom egg potato salt ground black pepper vegetable oil smoked streaky bacon leek cabbage	indian
173	fresh ginger root garlic cloves crushed red pepper flakes sesame oil sliced green onions low sodium soy sauce rice vin	chinese
182	mint leaves pinenuts rouget haricots verts extra virgin olive oil dried apricot	french
183	cheddar cheese lasagna noodles chili powder sauce minced garlic bay leaves balsamic vinegar onion tomato sauce ma	italian
188	warm water fine sea salt rye flour dry yeast honey bread flour	italian
191	black eyed peas cinnamon brown cardamom bay leaf ginger paste red chili peppers coriander powder salt cumin seed	indian
192	boneless chicken skinless thigh full fat coconut milk lime lemongrass potato fish sauce green curry paste	thai
210	salt milk egg melted fat flour	british
219	lime juice butter chopped cilantro fresh reduced sodium chicken broth flour sour cream kosher salt tomatillo poblano	mexican

Result Predicted using KNN in python

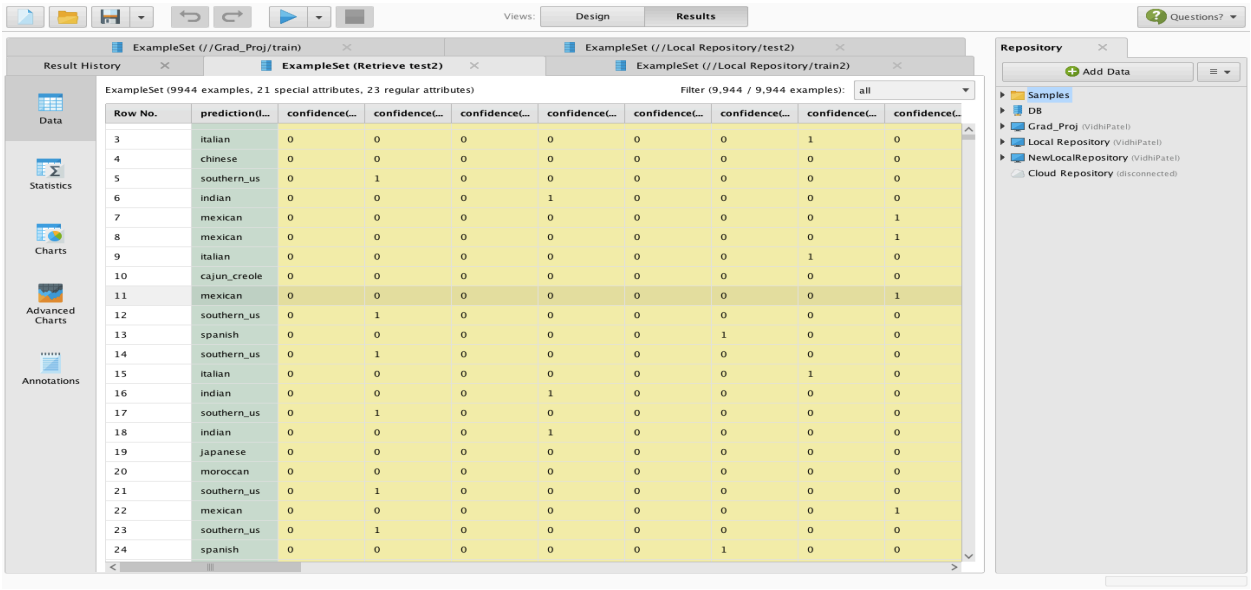
KNN in RapidMiner

Steps to perform Naïve Bayes in Rapid Miner: -

1. Select the Training Set and then set a label to the training set by selecting set role from the operators.
2. Select the KNN model from the operators.
3. Select the Test Set from the Database.
4. Select the Apply Model from the operators.
5. Connect the Output of the KNN and the test data set to the Apply model and run the model.

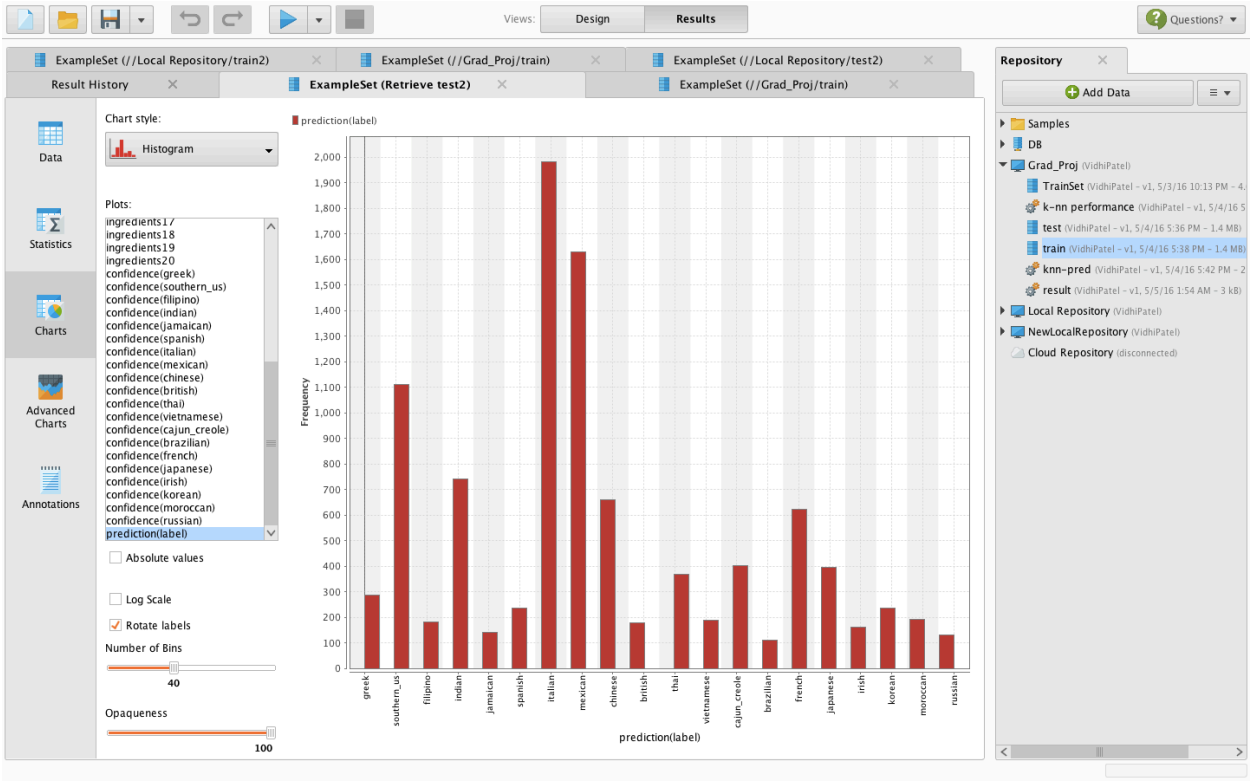


Naïve Bayes using Rapid Miner



Row No.	prediction(l...	confidence(...	confidence(...	confidence(...	confidence(...	confidence(...	confidence(...	confidence(...	confidence(...
3	italian	0	0	0	0	0	0	1	0
4	chinese	0	0	0	0	0	0	0	0
5	southern_us	0	1	0	0	0	0	0	0
6	indian	0	0	0	1	0	0	0	0
7	mexican	0	0	0	0	0	0	0	1
8	mexican	0	0	0	0	0	0	0	1
9	italian	0	0	0	0	0	0	1	0
10	cajun_creole	0	0	0	0	0	0	0	0
11	mexican	0	0	0	0	0	0	0	1
12	southern_us	0	1	0	0	0	0	0	0
13	spanish	0	0	0	0	0	1	0	0
14	southern_us	0	1	0	0	0	0	0	0
15	italian	0	0	0	0	0	0	1	0
16	indian	0	0	0	1	0	0	0	0
17	southern_us	0	1	0	0	0	0	0	0
18	indian	0	0	0	1	0	0	0	0
19	japanese	0	0	0	0	0	0	0	0
20	moroccan	0	0	0	0	0	0	0	0
21	southern_us	0	1	0	0	0	0	0	0
22	mexican	0	0	0	0	0	0	0	1
23	southern_us	0	1	0	0	0	0	0	0
24	spanish	0	0	0	0	0	1	0	0

Output of KNN Using RapidMiner



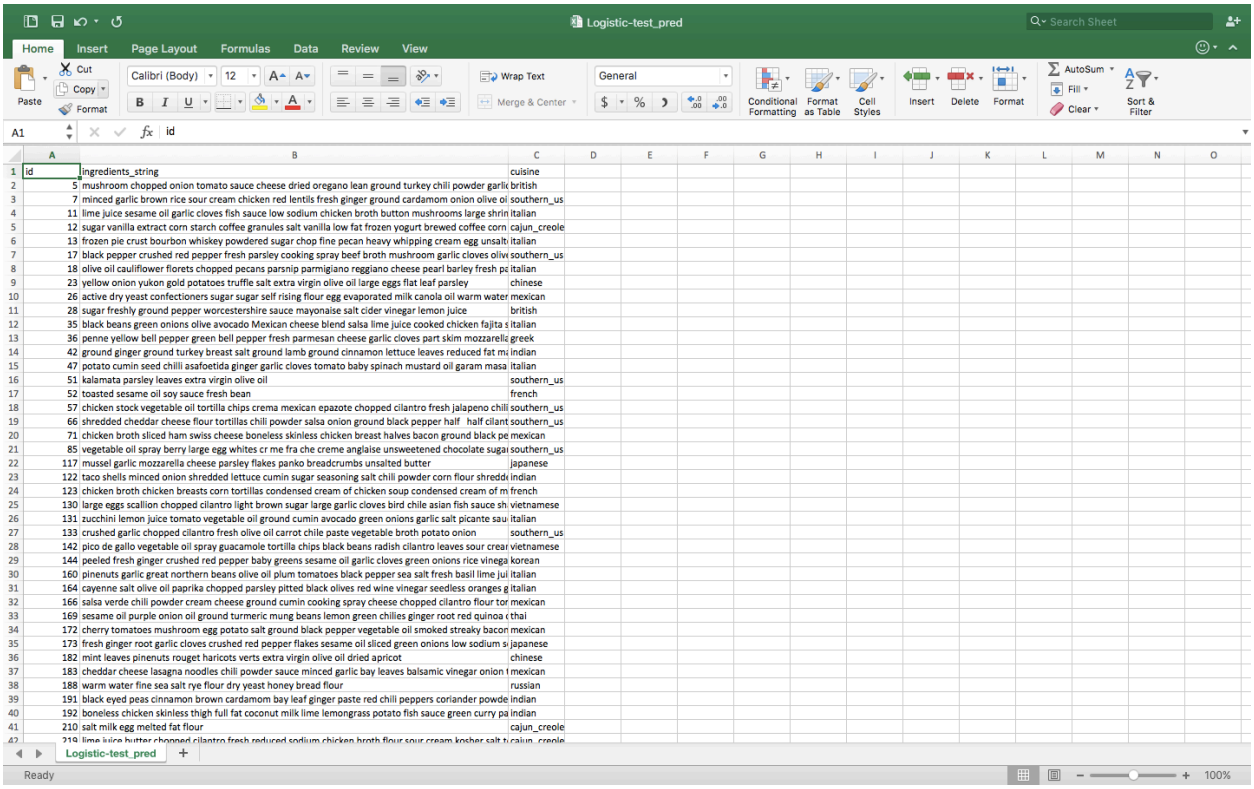
Bar Chart for the prediction of cuisines.

Logistic Regression in Python

The cuisine of the test data was predicted importing the library of Logistic Regression from the package called sklearn. The Logistic Regression model was trained using the training dataset which was prepared from the data preparation and then prediction was done for the test dataset.

```
#logistic Regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lrClassifier = lr.fit(predictor_train,target_data)
predictions = lrClassifier.predict(predictor_test)
testfile['cuisine'] = predictions
testfile = testfile.sort('id', ascending=True)
testfile[['id','ingredients_string', 'cuisine' ]].to_csv("Logistic-test_pred.csv", index=False)
```

Code using Logistic Regression in python



Result Predicted using KNN in python

CONCLUSION

After doing the project there were some things which came to my notice.

- Firstly, the predictions of different cuisines using **Naïve Bayes in Rapid Miner** and **Naïve Bayes in Python** are almost the same. Also predictions achieved using **Logistic Regression in Python** are quite similar to the results of Naïve Bayes in Rapid Miner and Naïve Bayes in Python.
- Secondly, the predictions achieved using **KNN in Python and RapidMiner** doesn't match. Also the predictions don't match to the results achieved above even after using different values for **weighted distance(k).**
- Lastly, tried implementing **Logistic Regression in Rapid Miner**. But the problem was that the model doesn't support for polynomial values.

REFERENCES

1. https://www.kaggle.com/c/whats-cooking/data
2. http://docs.rapidminer.com/studio/getting-started/
3. http://docs.rapidminer.com/studio/operators/modeling/predictive/bayesian/naive_bayes.html
4. https://pypi.python.org/pypi/pandas/0.16.2/
5. http://scikit-learn.org/stable/modules/naive_bayes.html