

Cuisine Classification

Task :

Cuisine Classification

Objective :

Develop a machine learning model to classify restaurants based on their cuisines.

Importing Libraries

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
```

Loading Dataset

```
In [2]: dataset = pd.read_csv("Dataset.csv")
dataset.head()
```

```
Out[2]:
```

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Price range	Aggregate rating	Rating color	Rating text	Votes
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14.565443	French, Japanese, Desserts	...	Botswana Pula(P)	Yes	No	No	No	3	4.8	Dark Green	Excellent	314
1	6304287	Izakaya Kikufuji	162	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101	14.553708	Japanese	...	Botswana Pula(P)	Yes	No	No	No	3	4.5	Dark Green	Excellent	591
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	...	Botswana Pula(P)	Yes	No	No	No	4	4.4	Green	Very Good	270
3	6318506	Ooma	162	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	...	Botswana Pula(P)	No	No	No	No	4	4.9	Dark Green	Excellent	365
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	...	Botswana Pula(P)	Yes	No	No	No	4	4.8	Dark Green	Excellent	229

5 rows × 21 columns

Data Preprocessing and Splitting

```
In [3]: # removing features that will inhibit model training
dataset.drop('Restaurant ID', axis=1, inplace=True)
dataset.drop('Country Code', axis=1, inplace=True)
dataset.drop('City', axis=1, inplace=True)
dataset.drop('Address', axis=1, inplace=True)
dataset.drop('Locality', axis=1, inplace=True)
dataset.drop('Locality Verbose', axis=1, inplace=True)
dataset.drop('Longitude', axis=1, inplace=True)
dataset.drop('Latitude', axis=1, inplace=True)
dataset.drop('Currency', axis=1, inplace=True)
dataset.drop('Has Table booking', axis=1, inplace=True)
dataset.drop('Has Online delivery', axis=1, inplace=True)
dataset.drop('Is delivering now', axis=1, inplace=True)
dataset.drop('Switch to order menu', axis=1, inplace=True)
dataset.drop('Price range', axis=1, inplace=True)
dataset.drop('Aggregate rating', axis=1, inplace=True)
dataset.drop('Rating color', axis=1, inplace=True)
dataset.drop('Rating text', axis=1, inplace=True)
dataset.drop('Votes', axis=1, inplace=True)
```

```
In [4]: #handle missing values
dataset.isna().sum()
```

```
Out[4]:
```

Restaurant Name	0
Cuisines	9
Average Cost for two	0

dtype: int64

```
In [5]: dataset.dropna(inplace=True)
```

```
In [6]: dataset.shape
```

```
Out[6]:
```

(9542, 3)

```
In [7]: dataset.describe(include="all")
```

```
Out[7]:
```

	Restaurant Name	Cuisines	Average Cost for two
count	9542	9542	9542.000000
unique	7437	1825	NaN
top	Cafe Coffee Day	North Indian	NaN
freq	83	936	NaN
mean	NaN	NaN	1200.326137
std	NaN	NaN	16128.743876
min	NaN	NaN	0.000000
25%	NaN	NaN	250.000000
50%	NaN	NaN	400.000000
75%	NaN	NaN	700.000000
max	NaN	NaN	800000.000000

```
In [8]: label_encoder = LabelEncoder()
dataset['Restaurant Name'] = label_encoder.fit_transform(dataset['Restaurant Name'])
dataset['Cuisines'] = label_encoder.fit_transform(dataset['Cuisines'])
dataset
```

```
Out[8]:
```

	Restaurant Name	Cuisines	Average Cost for two
0	3742	920	1100
1	3167	1111	1200
2	2892	1671	4000
3	4700	1126	1500
4	5515	1122	1500
...
9546	4436	1813	80
9547	1310	1824	105
9548	3063	1110	170
9549	512	1657	120
9550	7231	331	55

9542 rows × 3 columns

```
In [9]: x = dataset.drop('Cuisines',axis=1)
y = dataset['Cuisines']
```

```
In [10]: scaler = StandardScaler()
x=scaler.fit_transform(x)
```

```
In [11]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=15)
```

Random Forest Algorithm

```
In [15]: model_rfc = RandomForestClassifier(n_estimators=100, random_state=42)

model_rfc.fit(x_train, y_train)

rfc_pred = model_rfc.predict(x_test)
```

```
In [16]: accuracy = accuracy_score(y_test, rfc_pred)
print(f"Accuracy: {accuracy:.2f}")

# Precision, recall, F1-score
precision = precision_score(y_test, rfc_pred, average='micro')
recall = recall_score(y_test, rfc_pred, average='micro')
f1 = f1_score(y_test, rfc_pred, average='micro')
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-score: {f1:.2f}")

Accuracy: 0.23
Precision: 0.23
Recall: 0.23
F1-score: 0.23
```

Logistic Regression Algorithm

classifier_logreg = LogisticRegression(multi_class="multinomial", max_iter=1000)

classifier_logreg.fit(x_train, y_train)

```
In [24]: logreg_pred = classifier_logreg.predict(x_test)
print(logreg_pred)

[1306 1306 1306 ... 1306 1306 1306]
```

```
In [25]: accuracy = accuracy_score(y_test, logreg_pred)
print(f"Accuracy: {accuracy:.2f}")

# Precision, recall, F1-score
precision = precision_score(y_test, logreg_pred, average='micro')
recall = recall_score(y_test, logreg_pred, average='micro')
f1 = f1_score(y_test, logreg_pred, average='micro')
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-score: {f1:.2f}")

Accuracy: 0.10
Precision: 0.10
Recall: 0.10
F1-score: 0.10
```

Conclusion :

After comparing the results, we can see that the Random Forest model performs better than logistic regression. Even though I tried different preprocessing steps and model choices, I couldn't improve the accuracy score. This could be due to some hidden biases in either the model training or the dataset.

```
In [ ]:
```