```
# Import Libraries
import numpy as np
import pandas as pd
```

```
import warnings
warnings.filterwarnings("ignore")
```

Pre-Processing Steps

```
# Creating the Dataframe
df=pd.read_csv('/content/Dataset .csv')
df.head(1)
```

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking
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

1 rows × 21 columns

```
# Droping un-necessary Columns
columns_to_drop = [
    'Restaurant ID', 'Country Code', 'City', 'Address', 'Locality',
    'Locality Verbose', 'Longitude', 'Latitude', 'Currency',
    'Has Table booking', 'Has Online delivery', 'Is delivering now',
    'Switch to order menu', 'Price range', 'Aggregate rating',
    'Rating color', 'Rating text', 'Votes'
]
df.drop(columns=columns_to_drop, inplace=True)
```

```
df
```

	Restaurant Name	Cuisines	Average Cost for two
0	Le Petit Souffle	French, Japanese, Desserts	1100
1	Izakaya Kikufuji	Japanese	1200
2	Heat - Edsa Shangri-La	Seafood, Asian, Filipino, Indian	4000
3	Ooma	Japanese, Sushi	1500
4	Sambo Kojin	Japanese, Korean	1500
...	...	...	...
9546	Naml\ Gurme	Turkish	80
9547	Ceviz A��ac\	World Cuisine, Patisserie, Cafe	105
9548	Huqqa	Italian, World Cuisine	170
9549	A���k Kahve	Restaurant Cafe	120
9550	Walter's Coffee Roastery	Cafe	55

9551 rows × 3 columns

```
# Checking for missing values
df.isnull().sum()
```

	0
Restaurant Name	0
Cuisines	9
Average Cost for two	0

dtype: int64

```
df.dropna(inplace=True)
```

```
df.shape
```

```
(9542, 3)
```

```
# Checking missing vaalues for each col
missing_values = df.isna().sum()
missing_values_column = df['Restaurant Name'].isna().sum()
missing_values_column = df['Cuisines'].isna().sum()
missing_values_column = df['Average Cost for two'].isna().sum()
```

```
df_cleaned = df.dropna()
df_cleaned = df.dropna(subset=['Restaurant Name'])
df_cleaned = df.dropna(subset=['Cuisines'])
df_cleaned = df.dropna(subset=['Average Cost for two'])
```

```
df.describe(include="all")
```

	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

```
# Converting Categorical data to NUmerical
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['Restaurant Name'] = label_encoder.fit_transform(df['Restaurant Name'])
df['Cuisines'] = label_encoder.fit_transform(df['Cuisines'])
```

```
df
```

	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

Building Random Forest Model

```
# Create X and Y
X = df[['Restaurant Name', 'Average Cost for two']]
Y = df['Cuisines']
```

```
from sklearn.preprocessing import StandardScaler

scaler= StandardScaler()

scaler.fit(X)
X= scaler.transform(X)
```

```
from sklearn.model_selection import train_test_split

#Split the data into test and train
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
                                                    random_state=10)
```

```
# predicting using Decision Tree Classifier.
from sklearn.tree import DecisionTreeClassifier

model_DT = DecisionTreeClassifier(random_state=10,
                                  criterion="gini")

# fit the model on data and predict the values
model_DT.fit(X_train,Y_train)      # fit is the function that is used for training the data
Y_pred = model_DT.predict(X_test) # Validation Data
#print(Y_pred)
print(list(zip(Y_test,Y_pred)))
```

```
[(1348, np.int64(1348)), (1329, np.int64(1611)), (1608, np.int64(1235)), (1650, np.int64(1655)), (1306, np.int64(1749)), (1514,
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

cfm=confusion_matrix(Y_test,Y_pred)
print(cfm)

print("Classification report: ")

print(classification_report(Y_test,Y_pred))

acc=accuracy_score(Y_test, Y_pred)
print("Accuracy of the model: ",acc)
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
```

```
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Classification report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.00      0.00      0.00         1
           6       0.25      0.14      0.18         7
           7       0.00      0.00      0.00         1
           8       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         1
          11       0.00      0.00      0.00         1
          12       0.00      0.00      0.00         1
          13       0.00      0.00      0.00         1
          15       0.00      0.00      0.00         1
          16       1.00      1.00      1.00         2
          18       0.00      0.00      0.00         1
          20       0.00      0.00      0.00         1
          21       0.00      0.00      0.00         2
          24       0.00      0.00      0.00         0
          25       0.00      0.00      0.00         0
          26       0.00      0.00      0.00         1
          28       0.00      0.00      0.00         0
          29       0.00      0.00      0.00         2
          32       0.00      0.00      0.00         0
          34       0.00      0.00      0.00         1
          35       0.00      0.00      0.00         1
          37       0.00      0.00      0.00         1
          38       0.00      0.00      0.00         1
          43       0.00      0.00      0.00         0
          44       0.00      0.00      0.00         0
          47       0.00      0.00      0.00         1
          49       0.00      0.00      0.00         1
          50       0.00      0.00      0.00         0
          53       0.00      0.00      0.00         0
          54       0.33      0.33      0.33         3
          55       0.00      0.00      0.00         1
          57       0.00      0.00      0.00         0
          58       1.00      1.00      1.00        13
          61       0.00      0.00      0.00         0
          69       0.00      0.00      0.00         1
          78       0.00      0.00      0.00         0
          79       0.00      0.00      0.00         0
          81       0.00      0.00      0.00         1
          82       0.00      0.00      0.00         0
          85       0.00      0.00      0.00         1
          89       0.00      0.00      0.00         1
          90       0.00      0.00      0.00         0
          91       0.00      0.00      0.00         0
          93       0.00      0.00      0.00         0
          98       0.00      0.00      0.00         2
         101       0.00      0.00      0.00         1
         102       0.00      0.00      0.00         0
```

Building Logistic Regression

```
from sklearn.linear_model import LogisticRegression
#create a model object
classifier = LogisticRegression(multi_class="multinomial")
#train the model object
classifier.fit(X_train,Y_train)

Y_pred = classifier.predict(X_test)
print(Y_pred)
```

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

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

cfm=confusion_matrix(Y_test,Y_pred)
print(cfm)

print("Classification report: ")

print(classification_report(Y_test,Y_pred))

acc=accuracy_score(Y_test, Y_pred)
print("Accuracy of the model: ",acc)
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
```

```
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Classification report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         7
           7       0.00      0.00      0.00         1
           8       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         1
          11       0.00      0.00      0.00         1
          12       0.00      0.00      0.00         1
          13       0.00      0.00      0.00         1
          15       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         2
          18       0.00      0.00      0.00         1
          20       0.00      0.00      0.00         1
          21       0.00      0.00      0.00         2
          26       0.00      0.00      0.00         1
          29       0.00      0.00      0.00         2
          34       0.00      0.00      0.00         1
          35       0.00      0.00      0.00         1
          37       0.00      0.00      0.00         1
          38       0.00      0.00      0.00         1
          47       0.00      0.00      0.00         1
          49       0.00      0.00      0.00         1
          54       0.00      0.00      0.00         3
          55       0.00      0.00      0.00         1
          58       0.00      0.00      0.00        13
          69       0.00      0.00      0.00         1
          81       0.00      0.00      0.00         1
          85       0.00      0.00      0.00         1
          89       0.00      0.00      0.00         1
          98       0.00      0.00      0.00         2
         101       0.00      0.00      0.00         1
         104       0.00      0.00      0.00         2
         112       0.00      0.00      0.00         1
         115       0.00      0.00      0.00         2
         121       0.00      0.00      0.00         2
         123       0.00      0.00      0.00         1
         124       0.00      0.00      0.00         1
         134       0.00      0.00      0.00         1
         143       0.00      0.00      0.00         1
         152       0.00      0.00      0.00         1
         153       0.00      0.00      0.00         1
         158       0.00      0.00      0.00         1
         159       0.00      0.00      0.00         2
         168       0.00      0.00      0.00         3
         170       0.00      0.00      0.00         1
         176       0.00      0.00      0.00         1
         177       0.00      0.00      0.00        44
         178       0.00      0.00      0.00         1
```