

TEAM ID: PNT2022TMID25899

PROJECT NAME: DemandEst - AI powered Food Demand Forecaster

Team Leader

The screenshot displays a Jupyter Notebook interface with two visible code cells. The first cell, labeled 'In [115]:', contains Python code for importing LabelEncoder and applying it to 'center_type', 'category', and 'cuisine' columns of a dataset named 'trainfinal'. The second cell, labeled 'In [116]:', shows the output of 'trainfinal.head()', displaying the first two rows of the dataset. The output is a table with columns: id, week, city_code, region_code, center_type, op_area, category, cuisine, checkout_price, base_price, emailer_for_promotion, homepage_featured, and nur.

Label Encoding

Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with Machine Learning algorithms too.

We need to convert each text category to numbers in order for the machine to process those using mathematical equations. Label Encoding is a popular encoding technique for handling categorical variables implemented using the scikit-learn library in python. In this technique, each label is assigned a unique integer based on alphabetical ordering.

```
In [115]: from sklearn.preprocessing import LabelEncoder

lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb2.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb3.fit_transform(trainfinal['cuisine'])
```

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	

The second code cell shows the same code as the first, but the output of 'trainfinal.head()' is not visible in the screenshot.

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	
2	1196273	3	647	56	2	2.0	0	3	132.92	133.92	0	0	
3	1116527	4	647	56	2	2.0	0	3	135.86	134.86	0	0	
4	1343872	5	647	56	2	2.0	0	3	146.50	147.50	0	0	

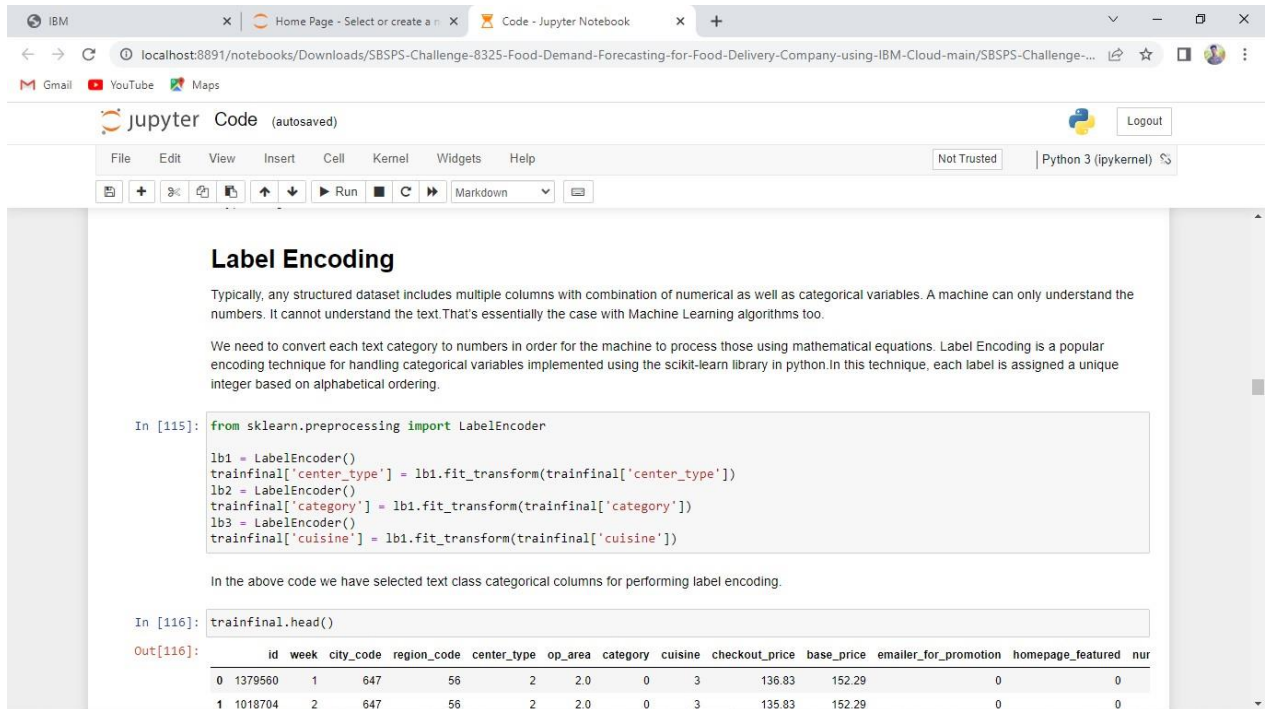
After performing label encoding, alphabetical classes- 'Center type, Category and City code are converted to numeric values.

Finally display number of rows and columns of trainfinal using shape()

```
In [117]: trainfinal.shape
```

```
Out[117]: (456548, 13)
```

Team Member 1



The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook is titled "jupyter Code (autosaved)" and has a "Python 3 (ipykernel)" kernel. The main content area displays a section titled "Label Encoding".

Label Encoding

Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with Machine Learning algorithms too.

We need to convert each text category to numbers in order for the machine to process those using mathematical equations. Label Encoding is a popular encoding technique for handling categorical variables implemented using the scikit-learn library in python. In this technique, each label is assigned a unique integer based on alphabetical ordering.

```
In [115]: from sklearn.preprocessing import LabelEncoder

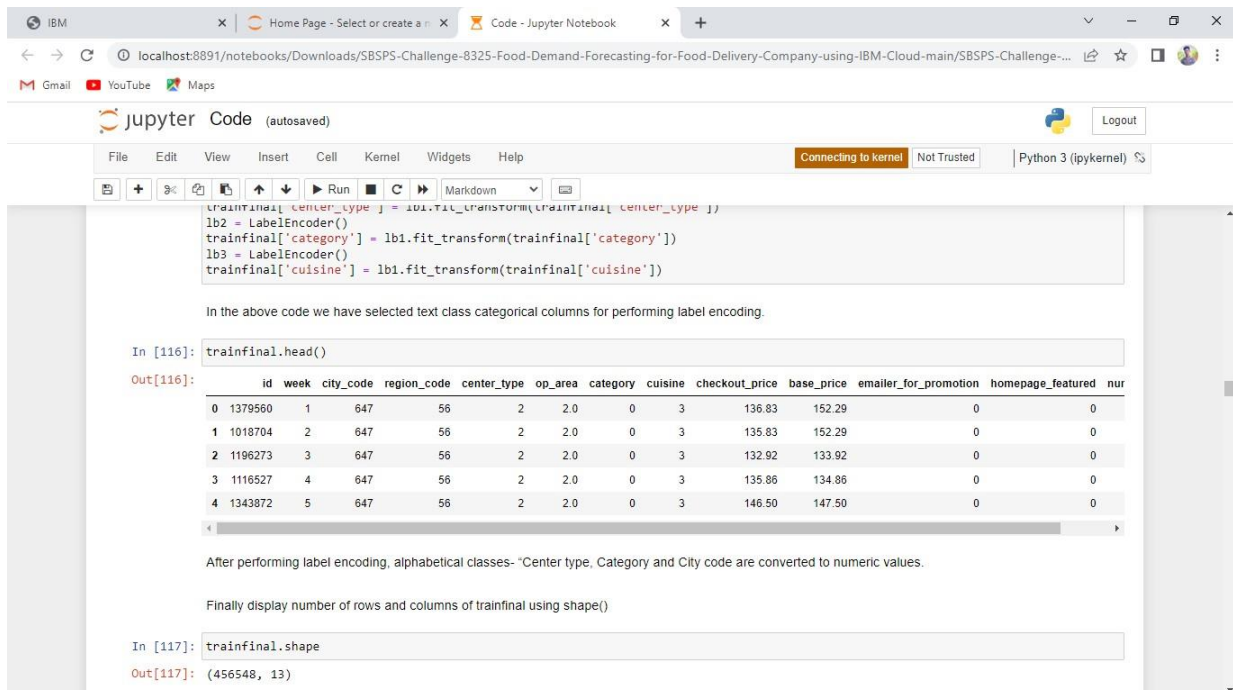
lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb1.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb1.fit_transform(trainfinal['cuisine'])
```

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

```
Out[116]:
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	



The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook is titled "jupyter Code (autosaved)" and has a "Python 3 (ipykernel)" kernel. The main content area displays a section titled "Label Encoding".

Label Encoding

Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with Machine Learning algorithms too.

We need to convert each text category to numbers in order for the machine to process those using mathematical equations. Label Encoding is a popular encoding technique for handling categorical variables implemented using the scikit-learn library in python. In this technique, each label is assigned a unique integer based on alphabetical ordering.

```
In [115]: from sklearn.preprocessing import LabelEncoder

lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb1.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb1.fit_transform(trainfinal['cuisine'])
```

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

```
Out[116]:
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	
2	1196273	3	647	56	2	2.0	0	3	132.92	133.92	0	0	
3	1116527	4	647	56	2	2.0	0	3	135.86	134.86	0	0	
4	1343872	5	647	56	2	2.0	0	3	146.50	147.50	0	0	

After performing label encoding, alphabetical classes- "Center type, Category and City code are converted to numeric values.

Finally display number of rows and columns of trainfinal using shape()

```
In [117]: trainfinal.shape
```

```
Out[117]: (456548, 13)
```

Team Member 2

Label Encoding

Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with Machine Learning algorithms too.

We need to convert each text category to numbers in order for the machine to process those using mathematical equations. Label Encoding is a popular encoding technique for handling categorical variables implemented using the scikit-learn library in python. In this technique, each label is assigned a unique integer based on alphabetical ordering.

```
In [115]: from sklearn.preprocessing import LabelEncoder

lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb2.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb3.fit_transform(trainfinal['cuisine'])
```

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	

```
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb2.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb3.fit_transform(trainfinal['cuisine'])
```

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	
2	1196273	3	647	56	2	2.0	0	3	132.92	133.92	0	0	
3	1116527	4	647	56	2	2.0	0	3	135.86	134.86	0	0	
4	1343872	5	647	56	2	2.0	0	3	146.50	147.50	0	0	

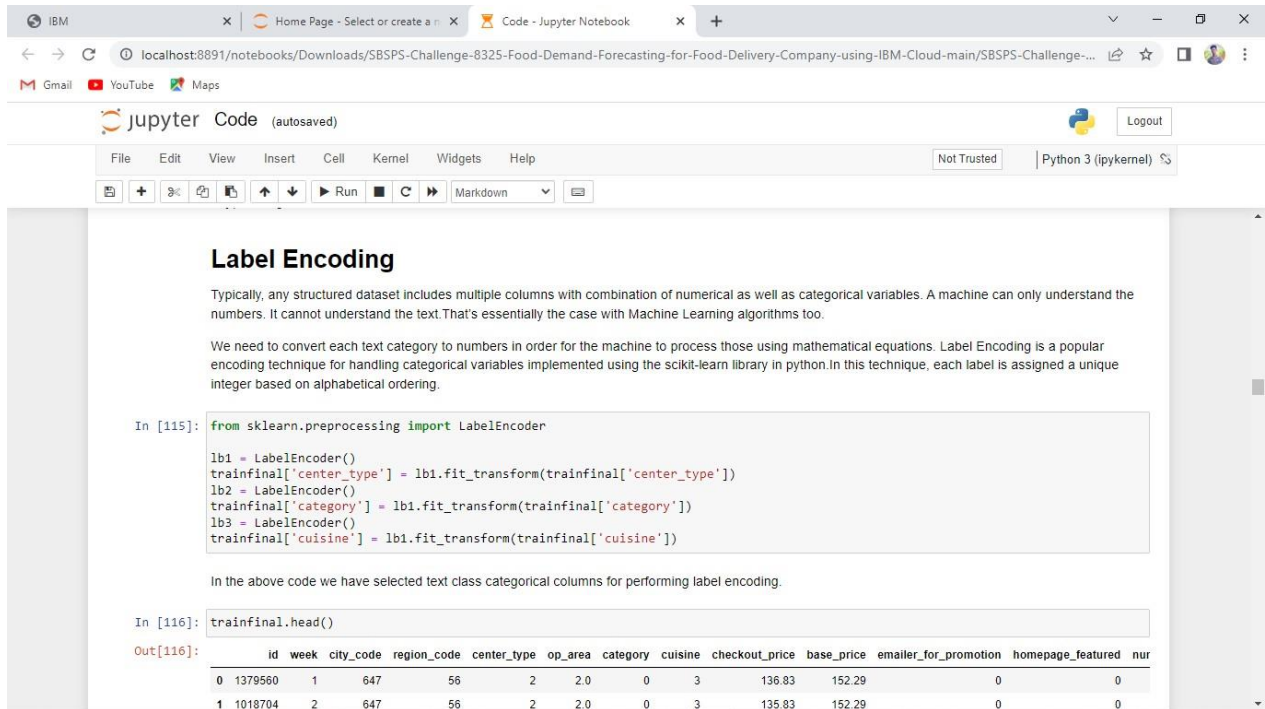
After performing label encoding, alphabetical classes- 'Center type, Category and City code are converted to numeric values.

Finally display number of rows and columns of trainfinal using shape()

```
In [117]: trainfinal.shape
```

```
Out[117]: (456548, 13)
```

Team Member 3



The screenshot shows a Jupyter Notebook interface with the following content:

Label Encoding

Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with Machine Learning algorithms too.

We need to convert each text category to numbers in order for the machine to process those using mathematical equations. Label Encoding is a popular encoding technique for handling categorical variables implemented using the scikit-learn library in python. In this technique, each label is assigned a unique integer based on alphabetical ordering.

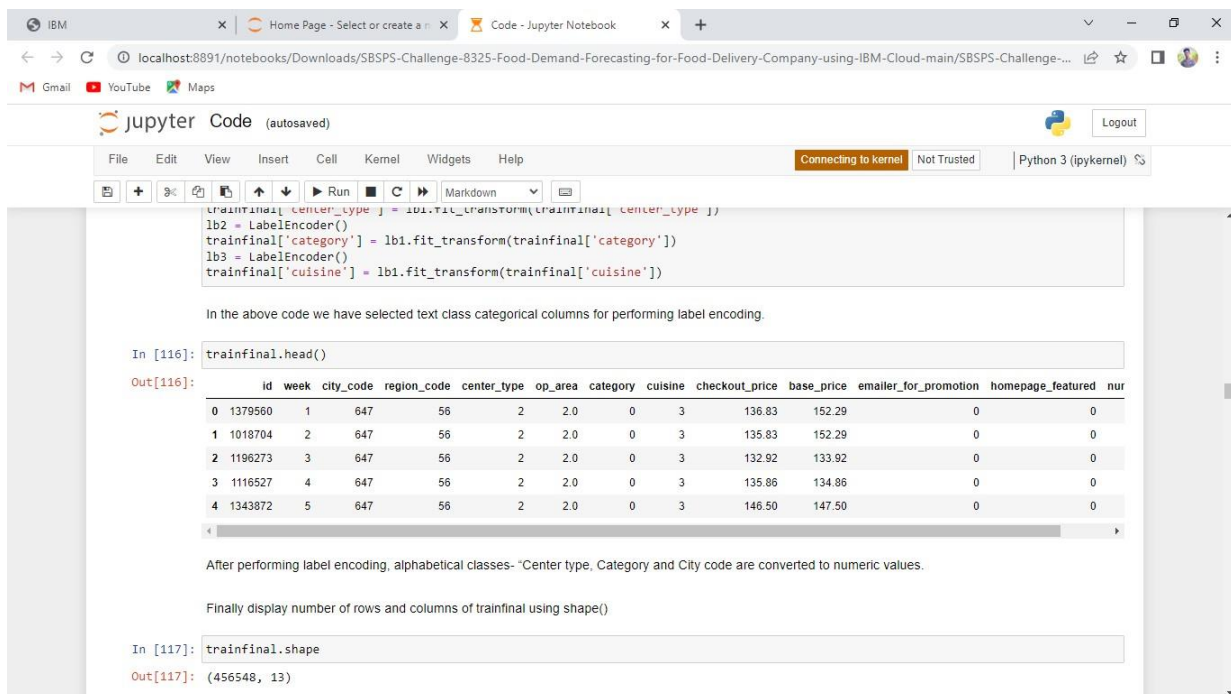
```
In [115]: from sklearn.preprocessing import LabelEncoder

lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb1.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb1.fit_transform(trainfinal['cuisine'])
```

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	



The screenshot shows the continuation of the Jupyter Notebook with the following content:

```
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb1.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb1.fit_transform(trainfinal['cuisine'])
```

In the above code we have selected text class categorical columns for performing label encoding.

```
In [116]: trainfinal.head()
```

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	nur
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	
2	1196273	3	647	56	2	2.0	0	3	132.92	133.92	0	0	
3	1116527	4	647	56	2	2.0	0	3	135.86	134.86	0	0	
4	1343872	5	647	56	2	2.0	0	3	146.50	147.50	0	0	

After performing label encoding, alphabetical classes- "Center type, Category and City code are converted to numeric values.

Finally display number of rows and columns of trainfinal using shape()

```
In [117]: trainfinal.shape
```

```
Out[117]: (456548, 13)
```