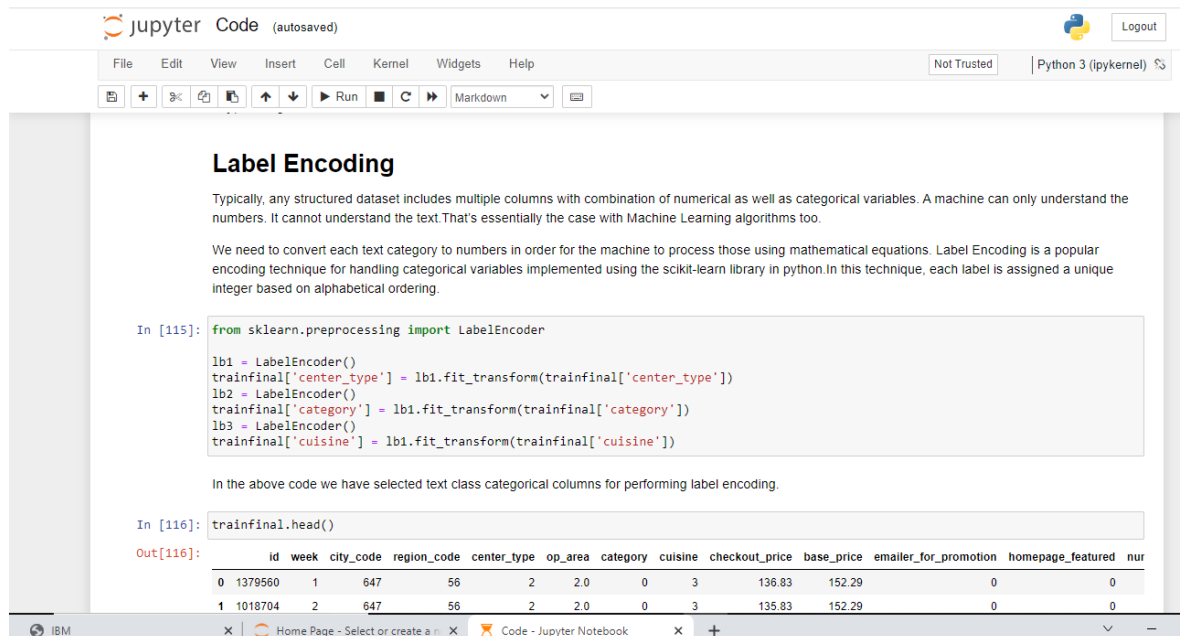


TEAM ID: PNT2022TMID52731

PROJECT NAME: DemandEst - AI powered Food Demand Forecaster

Team Leader



The screenshot shows a Jupyter Notebook titled "jupyter Code (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and markdown editing. The status bar at the top right indicates "Not Trusted" and "Python 3 (ipykernel)".

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

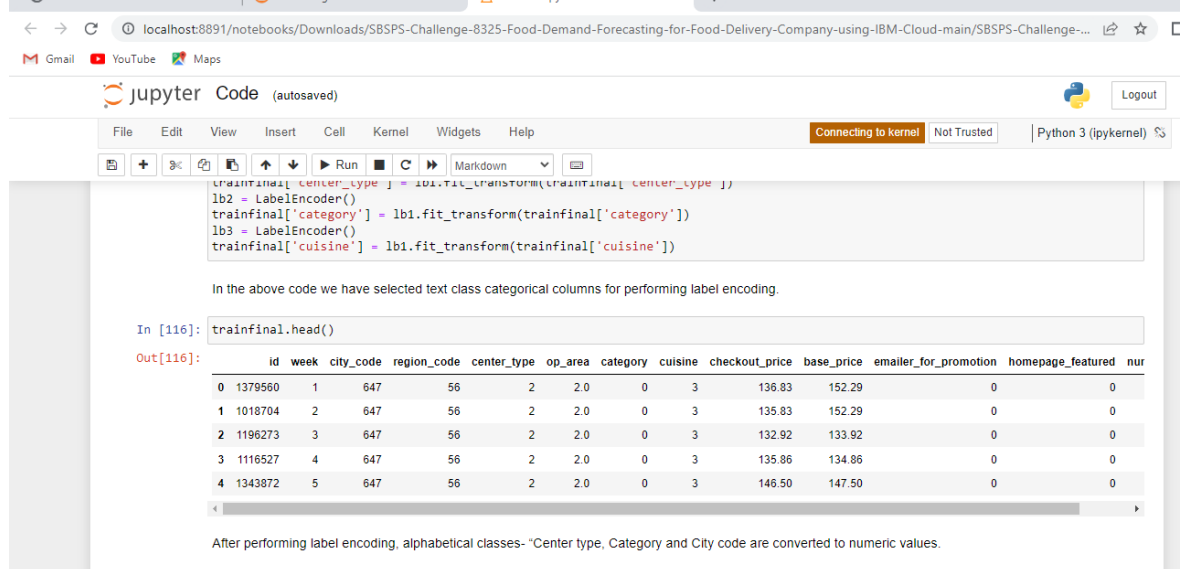
lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb1.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb1.fit_transform(trainfinal['cuisine'])
```

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 the continuation of the Jupyter Notebook. The status bar at the top right now indicates "Connecting to kernel" and "Not Trusted".

```
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.

Team Member 1

Jupyter Code (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel) Logout

In [115]:

```
from sklearn.preprocessing import LabelEncoder\n\nlb1 = LabelEncoder()\ntrainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])\n\nlb2 = LabelEncoder()\ntrainfinal['category'] = lb1.fit_transform(trainfinal['category'])\n\nlb3 = LabelEncoder()\ntrainfinal['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 the following content:

Code Cell:

```
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'])
```

Text:

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

Input:

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

Output:

	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	

Text:

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

Text:

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

Input:

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

Output:

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

Team Member 2

jupyter Code (autosaved) Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

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()
```

```
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	

jupyter Code (autosaved) Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help

Connecting to kernel Not Trusted

```
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()
```

```
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 3

jupyter Code (autosaved) Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run

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()
```

```
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	

jupyter Code (autosaved) Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help Connecting to kernel Not Trusted

Run

```
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()
```

```
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)
```