CUSTOMER ANALYTICS: PREPARING DATA FOR MODELLING

Omotola Ayodele Lawal

2024-09-16

Table of contents

### Project Overview

### Task

### Data Source

### Tools

### Steps/Explanations

### Data Analysis

### Results

### Recommendations

### Limitations

### References

## Project Overview



Two data scientists working on a dashboard.

A common problem when creating models to generate business value from data is that the datasets can be so large that it can take days for the model to generate predictions. Ensuring that your dataset is stored as efficiently as possible is crucial for allowing these models to run on a more reasonable timescale without having to reduce the size of the dataset.

You’ve been hired by a major online data science training provider called *Training Data Ltd.* to clean up one of their largest customer datasets. This dataset will eventually be used to predict whether their students are looking for a new job or not, information that they will then use to direct them to prospective recruiters.

You’ve been given access to customer\_train.csv, which is a subset of their entire customer dataset, so you can create a proof-of-concept of a much more efficient storage solution. The dataset contains anonymized student information, and whether they were looking for a new job or not during training:

| Column | Description |
| --- | --- |
| student\_id | A unique ID for each student. |
| city | A code for the city the student lives in. |
| city\_development\_index | A scaled development index for the city. |
| gender | The student’s gender. |
| relevant\_experience | An indicator of the student’s work relevant experience. |
| enrolled\_university | The type of university course enrolled in (if any). |
| education\_level | The student’s education level. |
| major\_discipline | The educational discipline of the student. |
| experience | The student’s total work experience (in years). |
| company\_size | The number of employees at the student’s current employer. |
| company\_type | The type of company employing the student. |
| last\_new\_job | The number of years between the student’s current and previous jobs. |
| training\_hours | The number of hours of training completed. |
| job\_change | An indicator of whether the student is looking for a new job (1) or not (0). |

## Task

The Head Data Scientist at Training Data Ltd. has asked you to create a DataFrame called ds\_jobs\_transformed that stores the data in customer\_train.csv much more efficiently. Specifically, they have set the following requirements:

* Columns containing categories with only two factors must be stored as Booleans (bool).
* Columns containing integers only must be stored as 32-bit integers (int32).
* Columns containing floats must be stored as 16-bit floats (float16).
* Columns containing nominal categorical data must be stored as the category data type.
* Columns containing ordinal categorical data must be stored as ordered categories, and not mapped to numerical values, with an order that reflects the natural order of the column.
* The DataFrame should be filtered to only contain students with 10 or more years of experience at companies with at least 1000 employees, as their recruiter base is suited to more experienced professionals at enterprise companies.

|  |
| --- |
| Important |
| If you call .info() or .memory\_usage() methods on ds\_jobs and ds\_jobs\_transformed after you’ve preprocessed it, you should notice a substantial decrease in memory usage. |

## Data Source

Data: The primary data used for this analysis is the customer\_train.csv, which is a subset of the entire customer dataset

## Tools

Jupyter lab

## Steps/Explanations

* The necessary library was imported, which is Pandas
* The Original dataset was loaded, named ds\_jobs and a copy was made, called ds\_jobs\_transformed
* Exploratory Data Analysis was performed which help identify ordinal, nominal, and two-factor categories. This was done by written codes, which iterate over all columns of the DataFrame ds\_jobs that have a data type of object (typically representing strings or categorical data) and print the value counts of each column.
* A dictionary of columns containing ordered categorical data was created. The code defines the dictionary called ordered\_cats, which contains lists of ordered categories for various features. These features represent specific categorical data in a dataset (e.g., levels of education, size of the company, work experience, etc.). This dictionary can later be used to create ordered categorical columns, for example, when transforming or encoding data in a pandas DataFrame.
* A mapping dictionary of columns containing two-factor categories to convert to Booleans was created. The code defines a Python dictionary called two\_factor\_cats. This dictionary is used to map certain categorical values into Boolean (True or False) values.
* This for col in ds\_jobs\_transformed: code iterates through each column in the ds\_jobs\_transformed DataFrame, performing different transformations based on the column name.
* For the columns 'relevant\_experience' and 'job\_change', if col in ['relevant\_experience', 'job\_change']: code uses the two\_factor\_cats dictionary to convert their categorical values into boolean values (True/False).
* The .map() function applies the mapping from two\_factor\_cats (defined previously) to the column. For example:
  + 'No relevant experience' becomes False.
  + 'Has relevant experience' becomes True.
  + 0.0 becomes False.
  + 1.0 becomes True.
* For the columns 'student\_id' and 'training\_hours', elif col in ['student\_id', 'training\_hours']: code changes their data types to int32 using .astype('int32'). This helps reduce memory usage, especially if the original data type was int64. The smaller int32 uses less memory and is sufficient for storing integers that fit within the 32-bit range.
* The column 'city\_development\_index' is converted to float16 (16-bit floating-point format) by the following code:

elif col in ['student\_id', 'training\_hours']:  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].astype('int32')

* float16 consumes less memory than the default float64.
* It’s useful for reducing the memory footprint of large datasets when the precision provided by float16 is sufficient.
* For columns that are in the ordered\_cats dictionary created earlier, the code below converts them to ordered categorical data types.

elif col in ordered\_cats.keys():  
 category = pd.CategoricalDtype(ordered\_cats[col], ordered=True)  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].astype(category)

* ordered\_cats is a dictionary that contains the order of categories for certain columns.
* pd.CategoricalDtype creates a categorical data type with a specific order, which is useful when the categories have a meaningful order (e.g., educational levels or experience).
* ordered=True ensures that the categories are treated as ordered (e.g., “Primary School” < “High School” < “Graduate”).
* The .astype(category) applies this conversion.
* For all remaining columns (those not handled in the previous conditions), the code below converts them to the standard categorical data type without an explicit order.

else:  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].astype('category')

* Converting to category reduces memory usage, especially when the column has a limited number of distinct values (e.g., city names, job roles, etc.).
* The final DataFrame was filtered to only contain students with 10 or more years of experience at companies with at least 1000 employees, as their recruiter base is suited to more experienced professionals at enterprise companies.

# Import necessary libraries  
import pandas as pd  
  
# Load the dataset  
ds\_jobs = pd.read\_csv("customer\_train.csv")  
  
# View the dataset  
ds\_jobs.head()  
  
ds\_jobs.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 19158 entries, 0 to 19157  
Data columns (total 14 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 student\_id 19158 non-null int64   
 1 city 19158 non-null object   
 2 city\_development\_index 19158 non-null float64  
 3 gender 14650 non-null object   
 4 relevant\_experience 19158 non-null object   
 5 enrolled\_university 18772 non-null object   
 6 education\_level 18698 non-null object   
 7 major\_discipline 16345 non-null object   
 8 experience 19093 non-null object   
 9 company\_size 13220 non-null object   
 10 company\_type 13018 non-null object   
 11 last\_new\_job 18735 non-null object   
 12 training\_hours 19158 non-null int64   
 13 job\_change 19158 non-null float64  
dtypes: float64(2), int64(2), object(10)  
memory usage: 2.0+ MB

## Data Analysis

Include below are the codes used to achieve the task given

# Import necessary libraries  
import pandas as pd  
  
# Load the dataset and create a copy  
ds\_jobs = pd.read\_csv("customer\_train.csv")  
ds\_jobs\_transformed = ds\_jobs.copy()  
  
# EDA to help identify ordinal, nominal, and two-factor categories  
for col in ds\_jobs.select\_dtypes("object").columns:  
 print(ds\_jobs\_transformed[col].value\_counts(), '\n')  
  
# Create a dictionary of columns containing ordered categorical data  
ordered\_cats = {  
 'enrolled\_university': ['no\_enrollment', 'Part time course', 'Full time course'],  
 'education\_level': ['Primary School', 'High School', 'Graduate', 'Masters', 'Phd'],  
 'experience': ['<1'] + list(map(str, range(1, 21))) + ['>20'],  
 'company\_size': ['<10', '10-49', '50-99', '100-499', '500-999', '1000-4999', '5000-9999', '10000+'],  
 'last\_new\_job': ['never', '1', '2', '3', '4', '>4']  
}  
  
# Create a mapping dictionary of columns containing two-factor categories to convert to Booleans  
two\_factor\_cats = {  
 'relevant\_experience': {'No relevant experience': False, 'Has relevant experience': True},  
 'job\_change': {0.0: False, 1.0: True}  
}  
  
# Loop through DataFrame columns to efficiently change data types  
for col in ds\_jobs\_transformed:  
   
 # Convert two-factor categories to bool  
 if col in ['relevant\_experience', 'job\_change']:  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].map(two\_factor\_cats[col])  
   
 # Convert integer columns to int32  
 elif col in ['student\_id', 'training\_hours']:  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].astype('int32')  
   
 # Convert float columns to float16  
 elif col == 'city\_development\_index':  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].astype('float16')  
   
 # Convert columns containing ordered categorical data to ordered categories using dict  
 elif col in ordered\_cats.keys():  
 category = pd.CategoricalDtype(ordered\_cats[col], ordered=True)  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].astype(category)  
   
 # Convert remaining columns to standard categories  
 else:  
 ds\_jobs\_transformed[col] = ds\_jobs\_transformed[col].astype('category')  
   
# Filter students with 10 or more years experience at companies with at least 1000 employees  
ds\_jobs\_transformed = ds\_jobs\_transformed[(ds\_jobs\_transformed['experience'] >= '10') & (ds\_jobs\_transformed['company\_size'] >= '1000-4999')]  
  
ds\_jobs\_transformed.info()

city  
city\_103 4355  
city\_21 2702  
city\_16 1533  
city\_114 1336  
city\_160 845  
 ...   
city\_129 3  
city\_111 3  
city\_121 3  
city\_140 1  
city\_171 1  
Name: count, Length: 123, dtype: int64   
  
gender  
Male 13221  
Female 1238  
Other 191  
Name: count, dtype: int64   
  
relevant\_experience  
Has relevant experience 13792  
No relevant experience 5366  
Name: count, dtype: int64   
  
enrolled\_university  
no\_enrollment 13817  
Full time course 3757  
Part time course 1198  
Name: count, dtype: int64   
  
education\_level  
Graduate 11598  
Masters 4361  
High School 2017  
Phd 414  
Primary School 308  
Name: count, dtype: int64   
  
major\_discipline  
STEM 14492  
Humanities 669  
Other 381  
Business Degree 327  
Arts 253  
No Major 223  
Name: count, dtype: int64   
  
experience  
>20 3286  
5 1430  
4 1403  
3 1354  
6 1216  
2 1127  
7 1028  
10 985  
9 980  
8 802  
15 686  
11 664  
14 586  
1 549  
<1 522  
16 508  
12 494  
13 399  
17 342  
19 304  
18 280  
20 148  
Name: count, dtype: int64   
  
company\_size  
50-99 3083  
100-499 2571  
10000+ 2019  
10-49 1471  
1000-4999 1328  
<10 1308  
500-999 877  
5000-9999 563  
Name: count, dtype: int64   
  
company\_type  
Pvt Ltd 9817  
Funded Startup 1001  
Public Sector 955  
Early Stage Startup 603  
NGO 521  
Other 121  
Name: count, dtype: int64   
  
last\_new\_job  
1 8040  
>4 3290  
2 2900  
never 2452  
4 1029  
3 1024  
Name: count, dtype: int64   
  
<class 'pandas.core.frame.DataFrame'>  
Index: 2201 entries, 9 to 19143  
Data columns (total 14 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 student\_id 2201 non-null int32   
 1 city 2201 non-null category  
 2 city\_development\_index 2201 non-null float16   
 3 gender 1821 non-null category  
 4 relevant\_experience 2201 non-null bool   
 5 enrolled\_university 2185 non-null category  
 6 education\_level 2184 non-null category  
 7 major\_discipline 2097 non-null category  
 8 experience 2201 non-null category  
 9 company\_size 2201 non-null category  
 10 company\_type 2144 non-null category  
 11 last\_new\_job 2184 non-null category  
 12 training\_hours 2201 non-null int32   
 13 job\_change 2201 non-null bool   
dtypes: bool(2), category(9), float16(1), int32(2)  
memory usage: 69.5 KB

## Results

There was a marked reduction in the size of the ds\_jobs\_transformed DataFrame from 2.0+ MB to 69.5 KB, which help to save memory and prepare the data for predictive modelling.

## Recommendations

None

## Limitations

None

## References

1. Filtering DataFrames in Intermediate Python Course for Associate Data Scientist in Python Carrer Track in DataCamp Inc by Hugo Bowne-Henderson.
2. For loop in Intermediate Python Course for Associate Data Scientist in Python Carrer Track in DataCamp Inc by Hugo Bowne-Henderson.
3. Python For Data Analysis 3E (Online) by Wes Mckinney Click [here](https://wesmckinney.com/book/) to preview
4. Working with Categorical Data in Python in Intermediate Python Course for Associate Data Scientist in Python Carrer Track in DataCamp Inc by Kasey Jones.