Annotated follow-along guide_Feature engineering with Python

January 7, 2024

1 Annotated follow-along guide: Feature engineering with Python

This notebook contains the code used in the following instructional video: Feature engineering and class balancing

1.1 Introduction

Throughout this notebook, we will learn to use Python to perform feature engineering on a dataset to prepare it for modeling using a supervised classification model. Before getting started, watch the associated instructional video and complete the in-video question. All of the code we will be implementing and related instructions are contained in this notebook.

1.2 Overview

The data we will use in this notebook is customer data from a European bank. We will return to this data often throughout this course. Later, we will compare the performance of different models on this data. Be sure to review the data dictionary to better acquaint yourself with it.

The data will be used to predict whether a customer of the bank will churn. If a customer churns, it means they left the bank and took their business elsewhere. If you can predict which customers are likely to churn, you can take measures to retain them before they do.

Topics of focus in this activity include:

- **Feature selection**
  - Removing uninformative features
- **Feature extraction**
  - Creating new features from existing features
- **Feature transformation**
  - Modifying existing features to better suit our objectives
  - Encoding of categorical features as dummies

1.3 Import packages and libraries

First, we will need to import all the required libraries and extensions. Throughout the course, we will be using numpy and pandas for operations.

[1]:
```
import numpy as np
import pandas as pd
```

1.4 Target variable

The data dictionary shows that there is a column called `Exited`. This is a Boolean value that indicates whether or not a customer left the bank (0 = did not leave, 1 = did leave). This will be our target variable. In other words, for each customer, our model should predict whether they should have a 0 or a 1 in the `Exited` column.

This is a supervised learning classification task because we will predict on a binary class. Therefore, this notebook will prepare the data for a classification model.

To begin, we will read in the data from a .csv file. Then, we will briefly examine it to better understand what it is telling us.

[2]:
```
# Read in data
df_original = pd.read_csv('Churn_Modelling.csv')
```

[3]:
```
df_original.head()
```

[3]:
```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

When modeling, a best practice is to perform a rigorous examination of the data before beginning feature engineering and feature selection. Not only does this process help you understand your data, what it is telling you, and what it is *not* telling you, but it also can give you clues that help you create new features.

You have already learned the fundamentals of exploratory data analysis (EDA), so this notebook will skip that essential part of the modeling process. Just remember that a good data science project will always include EDA.

The following table provides a quick overview of the data:

[4]:
```python
# Print high-level info about data
df_original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

From this table, we can confirm that the data has 14 features and 10,000 observations. We also know that nine features are integers, two are floats, and three are strings. Finally, we can tell that there are no null values because there are 10,000 observations, and each column has 10,000 non-null values.

1.5 Feature engineering

1.5.1 Feature selection

Feature selection is the process of choosing features to be used for modeling. In practice, feature selection takes place at multiple points in the PACE process. Although sometimes you will be given a dataset and a defined target variable, most often in practice you will begin with only a question or a problem that you are tasked with solving. In these cases, if you decide that the problem requires a model, you will then have to:

- Consider what data is available to you
- Decide on what kind of model you need
- Decide on a target variable
- Assemble a collection of features that you think might help predict on your chosen target

This all takes place during the **Plan** phase.

Then, during the **Analyze** phase, you perform EDA on the data and reevaluate your variables for appropriateness. For example, can your model handle null values? If not, what do you do with features with a lot of nulls? Perhaps you drop them. This too is feature selection.

Feature selection also occurs during the **Construct** phase. This usually involves building a model, examining which features are most predictive, and then removing the unpredictive features.

There's a lot of work involved in feature selection. In this case, we already have a dataset, and we are not performing thorough EDA on it. However, we can still examine the data to ensure that all the features can reasonably be expected to have predictive potential.

Returning to the bank data, notice that the first column is called `RowNumber`, and it just enumerates the rows. We should drop this feature, because row number shouldn't have any correlation with whether or not a customer churned.

The same is true for `CustomerID`, which appears to be a number assigned to the customer for administrative purposes, and `Surname`, which is the customer's last name. Since these cannot be expected to have any influence over the target variable, we can remove them from the modeling dataset.

Finally, for ethical reasons, we should remove the `Gender` column. The reason for doing this is that we don't want our model-making predictions (and therefore, offering promotions/financial incentives) based on a person's gender.

[5]:
```python
# Create a new df that drops RowNumber, CustomerId, Surname, and Gender cols
churn_df = df_original.drop(['RowNumber', 'CustomerId', 'Surname', 'Gender'],
                            axis=1)
```

[6]:
```python
churn_df.head()
```

[6]:
```
   CreditScore Geography  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619    France   42       2       0.00              1          1
1          608     Spain   41       1   83807.86              1          0
2          502    France   42       8  159660.80              3          1
3          699    France   39       1       0.00              2          0
4          850     Spain   43       2  125510.82              1          1

   IsActiveMember  EstimatedSalary  Exited
0               1        101348.88       1
1               1        112542.58       0
2               0        113931.57       1
3               0         93826.63       0
4               1         79084.10       0
```

1.5.2 Feature extraction

Depending on your data, you may be able to create brand new features from your existing features. Oftentimes, features that you create yourself are some of the most important features selected by your model. Usually this is the case when you have both domain knowledge for the problem you're solving and the right combinations of data.

For example, suppose you knew that your bank had a computer glitch that caused many credit card transactions to be mistakenly declined in October. It would be reasonable to suspect that people who experienced this might be at increased risk of leaving the bank. If you had a feature that represented each customer's number of credit card transactions each month, you could create a new feature; for example, `OctUseRatio`, where:

$$\text{OctUseRatio} = \frac{\text{num of Oct. transactions}}{\text{avg num monthly transactions}}$$

This new feature would then give you a ratio that might be indicative of whether the customer experienced declined transactions.

We don't have this kind of specific circumstantial knowledge, and we don't have many features to choose from, but we can create a new feature that might help improve the model.

Let's create a `Loyalty` feature that represents the percentage of each customer's life that they were customers. We can do this by dividing `Tenure` by `Age`:

$$\text{Loyalty} = \frac{\text{Tenure}}{\text{Age}}$$

The intuition here is that people who have been customers for a greater proportion of their lives might be less likely to churn.

[7]:
```
# Create Loyalty variable
churn_df['Loyalty'] = churn_df['Tenure'] / churn_df['Age']
```

[8]:
```
churn_df.head()
```

[8]:
```
   CreditScore Geography  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619    France   42       2       0.00              1          1
1          608     Spain   41       1   83807.86              1          0
2          502    France   42       8  159660.80              3          1
3          699    France   39       1       0.00              2          0
4          850     Spain   43       2  125510.82              1          1

   IsActiveMember  EstimatedSalary  Exited   Loyalty
0               1        101348.88       1  0.047619
1               1        112542.58       0  0.024390
2               0        113931.57       1  0.190476
3               0         93826.63       0  0.025641
4               1         79084.10       0  0.046512
```

The new variable appears as the last column in the updated dataframe.

1.5.3 Feature transformation

The next step is to transform our features to get them ready for modeling. Different models have different requirements for how the data should be prepared and also different assumptions about their distributions, independence, and so on. You learned about some of these already for linear and logistic regression, and you will continue learning about them as you encounter new modeling techniques.

The models we will be building with this data are all classification models, and classification models generally need categorical variables to be encoded. Our dataset has one categorical feature: `Geography`. Let's check how many categories appear in the data for this feature.

[9]:
```
# Print unique values of Geography col
churn_df['Geography'].unique()
```

[9]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

There are three unique values: France, Spain, and Germany. Encode this data so it can be represented using Boolean features. We will use a pandas function called `pd.get_dummies()` to do this.

When we call `pd.get_dummies()` on this feature, it will replace the `Geography` column with three new Boolean columns–one for each possible category contained in the column being dummied.

When we specify `drop_first=True` in the function call, it means that instead of replacing `Geography` with three new columns, it will instead replace it with two columns. We can do this because no information is lost from this, but the dataset is shorter and simpler.

In this case, we end up with two new columns called `Geography_Germany` and `Geography_Spain`. We don't need a `Geography_France` column, because if a customer's values in `Geography_Germany` and `Geography_Spain` are both 0, we will know they are from France!

[10]:
```
# Dummy encode categorical variables
churn_df = pd.get_dummies(churn_df, drop_first=True)
```

[11]:
```
churn_df.head()
```

[11]:
```
   CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619   42       2       0.00              1          1
1          608   41       1   83807.86              1          0
2          502   42       8  159660.80              3          1
3          699   39       1       0.00              2          0
4          850   43       2  125510.82              1          1

   IsActiveMember  EstimatedSalary  Exited   Loyalty  Geography_Germany  \
0               1        101348.88       1  0.047619                  0
1               1        112542.58       0  0.024390                  0
2               0        113931.57       1  0.190476                  0
3               0         93826.63       0  0.025641                  0
4               1         79084.10       0  0.046512                  0
```

```
   Geography_Spain
0                0
1                1
2                0
3                0
4                1
```

We can now use our new dataset to build a model.

1.6 Conclusion

You now understand how to use Python to perform feature engineering on a dataset to prepare it for modeling. Going forward, you can start using Python to perform feature engineering on your own data.