BEGINNER DATA EXPLORATION DATA MINING MACHINE LEARNING

7 Feature Engineering Techniques in Machine Learning You Should Know

ANANYD36, OCTOBER 1, 2020

Article Video Book

This article was published as a part of the Data Science Blogathon.

Overview

- Feature engineering techniques are a must know concept for machine learning professionals
- Here are 7 feature engineering techniques you can start using right away

Introduction

Feature engineering is a topic every machine learning enthusiast has heard of. But the concept keeps eluding most people.

How in the world can you use feature engineering?

Why do we need the engineer features at all?

We know that machine learning algorithms use some input data to produce results. But quite often, the data you've been given might not be enough for designing a good machine learning model. That's where the power of feature engineering comes into play.

Feature engineering has two goals primarily:

- Preparing the proper input dataset, compatible with the machine learning algorithm requirements
- Improving the performance of machine learning models

In this article, we'll quickly go through 7 common feature engineering techniques that every machine learning professional should know.

List of Feature Engineering Techniques

1. Imputation
2. Handling Outliers
3. Binning
4. Log Transform
5. One-Hot Encoding
6. Grouping Operations
7. Scaling

POPULAR POSTS

- Logistic Regression- Supervised Learning Algorithm for Classification
- How to Build Word Cloud in Python?
- Build Treemaps in Python using Squarify
- How to Use Progress Bars in Python?
- Making Programming with Date and Time, less painless
- Introduction to Python Programming (Beginner's Guide)
- Seismic Analysis with Python
- 40 Questions to test a Data Scientist on Clustering Techniques (Skill test Solution)

CAREER RESOURCES

16 Key Questions You Should Answer Before Transitioning into Data Science

NOVEMBER 23, 2020

Here's What You Need to Know to Become a Data Scientist!

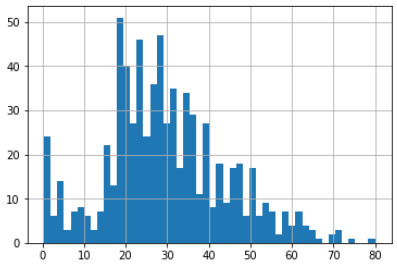
1. Imputation

Missing values are one of the most common problems you can encounter when you prepare your data for machine learning. The reason for the missing values might be human errors, interruptions in the data flow, privacy concerns, etc. Whatever the reason, missing values affect the performance of machine learning models.

Some of the imputation operations you can perform are:

- Numerical Imputation: Imputation is a more preferable option rather than dropping because it preserves the data size. However, there is an important selection of what you impute to the missing values. I suggest beginning with considering a possible default value of missing values in the column
- Categorical Imputation: Replacing the missing values with the maximum occurred value in a column is a good option for handling categorical columns
- Random sample imputation: This consists of taking random observation from the dataset and we use this observation to replace the NaN values
- End of Distribution Imputation

```
In [48]: df.Age.hist(bins=50)
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x28618d75f28>
```




```
In [53]: extreme=df.Age.mean()+3*df.Age.std()
```

Imputing done at **mean+3*std**

2. Handling Outliers

Which Machine LEarning Models Are Sensitive To Outliers?

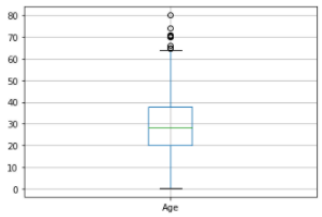
1. Naivye Bayes Classifier--- Not Sensitive To Outliers
2. SVM-------- Not Sensitive To Outliers
3. Linear Regression---------- Sensitive To Outliers
4. Logistic Regression------- Sensitive To Outliers
5. Decision Tree Regressor or Classifier---- Not Sensitive
6. Ensemble(RF,XGboost,GB)------- Not Sensitive
7. KNN------------------------- Not Sensitive
8. Kmeans---------------------- Sensitive
9. Hierarichal----------------- Sensitive
10. PCA------------------------ Sensitive
11. Neural Networks------------ Sensitive

Sensitivity to outliers for machine learning algorithms.

Before mentioning how outliers can be handled, I want to state that the best way to detect outliers is to demonstrate the data visually. All other statistical methodologies are open to making mistakes, whereas visualizing the outliers gives a chance to take a decision with high precision.

- Outlier in terms of Standard Deviation
  If a value has a distance to the average higher than x * standard deviation, it can be assumed as an outlier
- Outlier in terms of Percentiles
  Percentiles according to the range of the data. In other words, if your data ranges from 0 to 100, your top 5% is not the values between 96 and 100. Top 5% means here the values that are out of the 95th percentile of data

```
[12]: figure=df.boxplot(column="Age")
```

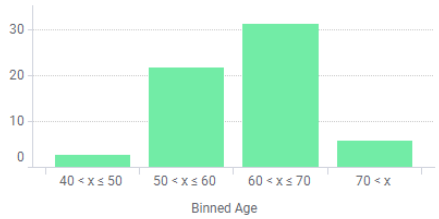


Identifying outliers through a boxplot

3. Binning

Binning can be applied on both categorical and numerical data.

The main motivation of binning is to make the model more robust and prevent overfitting. However, it has a cost on the performance. Every time you bin something, you sacrifice information and make your data more regularized.



4. Log Transform

Logarithm transformation (or log transform) is one of the most commonly used mathematical transformations in feature engineering. Here are the benefits of using log transform:

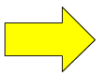
- It helps to handle skewed data and after transformation, the distribution becomes more approximate to normal
- It also decreases the effect of the outliers due to the normalization of magnitude differences and the model become more robust
- The data you apply log transform to must have only positive values, otherwise you receive an error

5. One-Hot Encoding

One-hot encoding is one of the most common encoding methods in machine learning. This method spreads the values in a column to multiple flag columns and assigns 0 or 1 to them. These binary values express the relationship between grouped and encoded column.

This method changes your categorical data, which is challenging to understand for algorithms, to a numerical format and enables you to group your categorical data without losing any information.

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

One Hot encoding Applied to Color column

6. Grouping Operations

Categorical Grouping
Using a pivot table or grouping based on aggregate functions using lambda.

```
#Pivot table Pandas Example

data.pivot_table(index='column_to_group', columns='column_to_encode',
values='aggregation_column', aggfunc=np.sum, fill_value = 0)
```

Numeric Grouping
Numerical columns are grouped using sum and mean functions in most of the cases.

```
#sum_cols: List of columns to sum
#mean_cols: List of columns to average

grouped = data.groupby('column_to_group')

sums = grouped[sum_cols].sum().add_suffix('_sum')
avgs = grouped[mean_cols].mean().add_suffix('_avg')

new_df = pd.concat([sums, avgs], axis=1)
```

7. Scaling

In most cases, the numerical features of the dataset do not have a certain range and they differ from each other. In order for a symmetric dataset, scaling is required.

- Normalization
  Normalization (or min-max normalization) scales all values in a fixed range between 0 and 1. This transformation does not change the distribution of the feature and due to the decreased standard deviations, the effects of the outliers increases. Therefore, before normalization, it is recommended to handle the outliers
- Standardization
  Standardization (or z-score normalization) scales the values while taking into account standard deviation. If the standard deviation of features is different, their range also would differ from each other. This reduces the effect of the outliers in the features.

$$x^*_{i,j} = \frac{x_{i,j} - x^{min}_j}{x^{max}_j - x^{min}_j}$$

Normalization Formulae (Min-Max Scaler)

$$z = \frac{x_i - \mu}{\sigma}$$

Standardization Formulae

Standardizations are involved majorly where there is distance involved in Gradient Descent (Linear Regression, KNN, etc.) or in ANN for faster convergence while Normalization is involved in places of classification or CNN (for scaling down the pixel values).

End Notes

This was a quick overview of the different feature engineering techniques are our disposal. This is in no way an exhaustive list but is good enough to get you started.

No more excuses next time – you can perform feature engineering that easily!

Thank you for reading and happy learning!

TAGS : BLOGATHON, FEATURE ENGINEERING, HACKATHONS, MACHINE LEARNING

PREVIOUS ARTICLE
Hypothesis Generation for Data Science Projects – A Critical Problem Solving Step

NEXT ARTICLE
HackLive – Everything You Need to Get Started with Data Science Hackathons!

Anannyd36

This article is quite old and you might not get a prompt response from the author. We request you to post this comment on Analytics Vidhya's **Discussion portal** to get your queries resolved

ONE COMMENT

ARPIT Reply
October 2, 2020 at 3:06 pm

Insightful ..