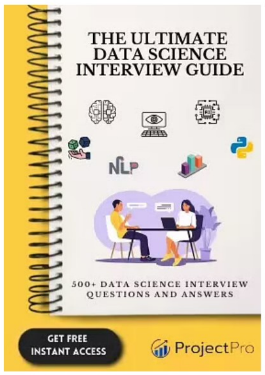
8 Feature Engineering Techniques for Machine Learning

Last Updated: 28 Feb 2022


THE ULTIMATE DATA SCIENCE INTERVIEW GUIDE
NLP
500+ DATA SCIENCE INTERVIEW QUESTIONS AND ANSWERS
GET FREE INSTANT ACCESS
ProjectPro

"Coming up with features is difficult, time-consuming, requires expert knowledge. 'Applied machine learning is basically feature engineering." — ***Prof. Andrew Ng.***

Data Scientists spend 80% of their time doing feature engineering because it's a time-consuming and difficult process. Understanding features and the various techniques involved to deconstruct this art can ease the complex process of feature engineering. So, let's get started.

Relevant Projects

Machine Learning Projects

Data Science Projects

Python Projects for Data Science

Data Science Projects in R

Machine Learning Projects for Beginners

Deep Learning Projects

Neural Network Projects

Tensorflow Projects

NLP Projects

Kaggle Projects

IoT Projects

Big Data Projects

Hadoop Real-Time Projects Examples

Spark Projects

Data Analytics Projects for Students

Table of contents

What is Feature Engineering for Machine Learning?

Feature engineering is the 'art' of formulating useful features from existing data following the target to be learned and the machine learning model used. It involves transforming data to forms that better relate to the underlying target to be learned. When done right, feature engineering can augment the value of your existing data and improve the performance of your machine learning models. On the other hand, using bad features may require you to build much more complex models to achieve the same level of performance.

This is the reason feature Engineering has found its place as an indispensable step in the machine learning pipeline. Yet, when it comes to applying this magical concept of Feature Engineering, there is no hard and fast method or theoretical framework, which is why it has maintained its status as a concept that eludes many.

This article will try to demystify this subtle art while establishing the significance it bears despite its nuances and finally gets started on our journey with a fun feature engineering Python example you can follow along!

You might also like

Data Scientist Salary
How to Become a Data Scientist
Data Analyst vs Data Scientist
Data Scientist Resume
Data Science Projects for Beginners
Machine Learning Engineer
Machine Learning Projects for Beginners
Datasets
Pandas Dataframe
Machine Learning Algorithms
Regression Analysis
MNIST Dataset
Data Science Interview Questions
Python Data Science Interview Questions
Spark Interview Questions
Hadoop Interview Questions
Data Analyst Interview Questions
Machine Learning Interview Questions
AWS vs Azure
Hadoop Architecture
Spark Architecture

Learn PySpark SQL Functions

View Project

Models Using PyTorch Autoencoders

View Project

Classification Models with RNN and LSTM

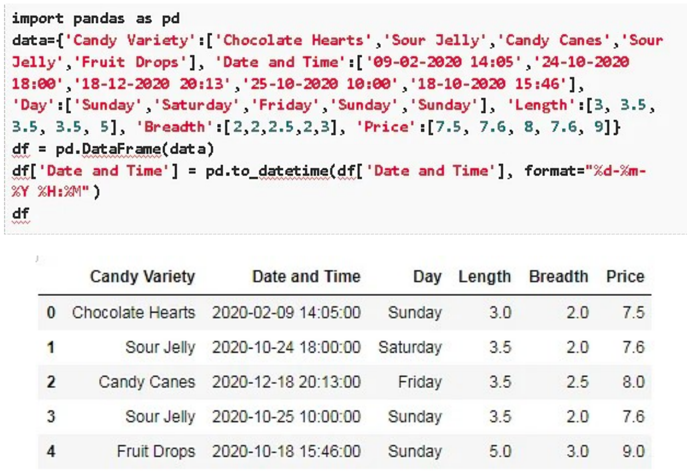
View Project

View all New Projects

Why is Feature Engineering important for Machine Learning?

To understand what feature engineering is at an intuitive level and why it is indispensable it might be useful to decipher how humans comprehend data. Humans have an ability, leaps ahead of that of a machine, to find complex patterns or relations, so much so that we can see them even when they don't actually exist. Yet even to us, data presented efficiently could mean a lot more than that which is presented randomly. If you haven't experienced this already, let's try to drive this home with a 'sweet' feature engineering example!

Say you have been provided the following data about candy orders:



```
import pandas as pd
data={'Candy Variety':['Chocolate Hearts','Sour Jelly','Candy Canes','Sour
Jelly','Fruit Drops'], 'Date and Time':['09-02-2020 14:05','24-10-2020
18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
'Day':['Sunday','Saturday','Friday','Sunday','Sunday'], 'Length':[3, 3.5,
3.5, 3.5, 5], 'Breadth':[2,2,2.5,2,3], 'Price':[7.5, 7.6, 8, 7.6, 9]}
df = pd.DataFrame(data)
df['Date and Time'] = pd.to_datetime(df['Date and Time'], format="%d-%m-
%Y %H:%M")
df
```

```
      Candy Variety        Date and Time       Day  Length  Breadth  Price
0  Chocolate Hearts  2020-02-09 14:05:00    Sunday     3.0      2.0    7.5
1        Sour Jelly  2020-10-24 18:00:00  Saturday     3.5      2.0    7.6
2       Candy Canes  2020-12-18 20:13:00    Friday     3.5      2.5    8.0
3        Sour Jelly  2020-10-25 10:00:00    Sunday     3.5      2.0    7.6
4       Fruit Drops  2020-10-18 15:46:00    Sunday     5.0      3.0    9.0
```

You have also been informed that the customers are uncompromising candy-lovers who consider their candy preference far more important than the price or even dimensions (essentially uncorrelated price, dimensions, and candy sales). What would you do when you are asked to predict which kind of candy is most likely to sell the most on a particular day?

Download Feature Engineering for Machine Learning Principles and Techniques for Data Scientists iPython Notebook

Then, I think you'd agree that the variety of candy ordered would depend more on the date than on the time of the day it was ordered and also that the sales for a particular variety of candy would vary according to the season.

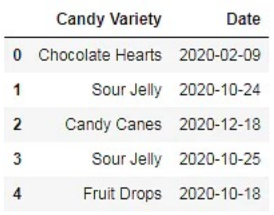
Tutorials

- Data Science Tutorial
- Jupyter Notebook Tutorial - A Complete Beginners Guide
- Best Python NumPy Tutorial for Beginners
- Tableau Tutorial for Beginners -Step by Step Guide
- MLOps Python Tutorial for Beginners -Get Started with MLOps
- Alteryx Tutorial for Beginners to Master Alteryx in 2021
- Free Microsoft Power BI Tutorial for Beginners with Examples
- Theano Deep Learning Tutorial for Beginners
- Computer Vision Tutorial for Beginners | Learn Computer Vision
- Python Pandas Tutorial for Beginners - The A-Z Guide
- NumPy Python Tutorial for Beginners
- Hadoop Online Tutorial – Hadoop HDFS Commands Guide
- MapReduce Tutorial–Learn to implement Hadoop WordCount Example
- Hadoop Hive Tutorial-Usage of Hive Commands in HQL
- Hive Tutorial-Getting Started with Hive Installation on Ubuntu
- Learn Java for Hadoop Tutorial: Inheritance and Interfaces
- Learn Java for Hadoop Tutorial: Classes and Objects
- Learn Java for Hadoop Tutorial: Arrays
- Apache Spark Tutorial - Run your

Get FREE Access to [Machine Learning and Data Science Example Codes]

Now that you instinctively know what features would most likely contribute to your predictions, let's go ahead and present our data better by simply creating a new feature Date from the existing feature Date and Time.

```
df['Date']=df['Date and Time'].dt.date
df[['Candy Variety','Date']]
```



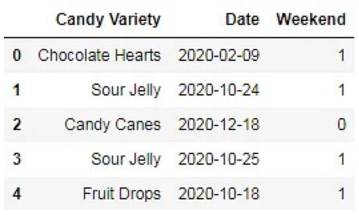
	Candy Variety	Date
0	Chocolate Hearts	2020-02-09
1	Sour Jelly	2020-10-24
2	Candy Canes	2020-12-18
3	Sour Jelly	2020-10-25
4	Fruit Drops	2020-10-18

The table you have obtained as a result should definitely make it at least a tad bit simpler for you to predict that Sour Jellies are most likely to sell, especially around the end of October (Halloween!) given the very same input data...

In addition, if you wanted to know more about the weekend and weekday sale trends, in particular, you could categorize the days of the week in a feature called Weekend with 1=True and 0=False

With this, you could predict that it would be best to have your shelves stocked on the weekends!

```
import numpy as np
df['Weekend'] = np.where(df['Day'].isin(['Saturday', 'Sunday']), 1, 0)
df[['Candy Variety','Date','Weekend']]
```



	Candy Variety	Date	Weekend
0	Chocolate Hearts	2020-02-09	1
1	Sour Jelly	2020-10-24	1
2	Candy Canes	2020-12-18	0
3	Sour Jelly	2020-10-25	1
4	Fruit Drops	2020-10-18	1

This short example should have emphasized how a little bit of Feature Engineering could transform the way *you* understand your data. For a machine, however, such linear and straightforward relationships could do wonders.

Now that you have wrapped your head around why Feature Engineering is so important, how it could work, and also why it can't be simply done mechanically, let's explore a few feature engineering techniques that could help!

Feature Engineering Techniques for Machine Learning -Deconstructing the 'art'

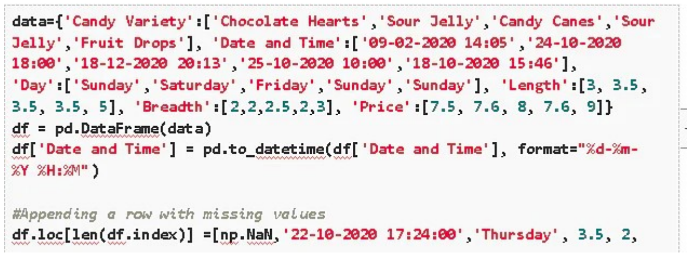
While understanding the data and the targeted problem is an indispensable part of [Feature Engineering in machine learning], and there are indeed no hard and fast rules as to how it is to be achieved, the following feature engineering techniques are a must know:

- First Spark Program
- Best PySpark Tutorial for Beginners-Learn Spark with Python
- R Tutorial- Learn Data Visualization with R using GGVIS
- Neural Network Training Tutorial
- Python List Tutorial
- MatPlotLib Tutorial
- Decision Tree Tutorial
- Neural Network Tutorial
- Performance Metrics for Machine Learning Algorithms
- R Tutorial: Data.Table
- SciPy Tutorial
- Step-by-Step Apache Spark Installation Tutorial
- Introduction to Apache Spark Tutorial
- R Tutorial: Importing Data from Web
- R Tutorial: Importing Data from Relational Database
- R Tutorial: Importing Data from Excel
- Introduction to Machine Learning Tutorial
- Machine Learning Tutorial: Linear Regression
- Machine Learning Tutorial: Logistic Regression
- Support Vector Machine Tutorial (SVM)
- K-Means Clustering Tutorial
- dplyr Manipulation Verbs
- Introduction to dplyr package
- Importing Data from Flat Files in R
- Principal Component Analysis Tutorial

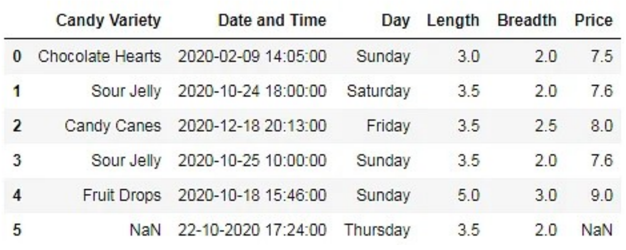
1) Imputation

Imputation deals with handling missing values in data. While deleting records that are missing certain values is one way of dealing with this issue, it could also mean losing out on a chunk of valuable data. This is where imputation can help. It can be broadly classified into two types. Namely:

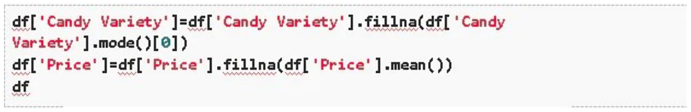
- **Categorical Imputation:** Missing categorical values are generally replaced by the most commonly occurring value in other records
- **Numerical Imputation:** Missing numerical values are generally replaced by the mean of the corresponding value in other records



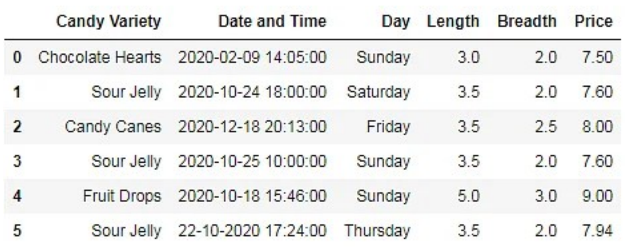
```
data={'Candy Variety':['Chocolate Hearts','Sour Jelly','Candy Canes','Sour
Jelly','Fruit Drops'], 'Date and Time':['09-02-2020 14:05','24-10-2020
18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
'Day':['Sunday','Saturday','Friday','Sunday','Sunday'], 'Length':[3, 3.5,
3.5, 3.5, 5], 'Breadth':[2,2,2.5,2,3], 'Price':[7.5, 7.6, 8, 7.6, 9]}
df = pd.DataFrame(data)
df['Date and Time'] = pd.to_datetime(df['Date and Time'], format="%d-%m-
%Y %H:%M")

#Appending a row with missing values
df.loc[len(df.index)] =[np.NaN,'22-10-2020 17:24:00','Thursday', 3.5, 2,
```



	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	2020-02-09 14:05:00	Sunday	3.0	2.0	7.5
1	Sour Jelly	2020-10-24 18:00:00	Saturday	3.5	2.0	7.6
2	Candy Canes	2020-12-18 20:13:00	Friday	3.5	2.5	8.0
3	Sour Jelly	2020-10-25 10:00:00	Sunday	3.5	2.0	7.6
4	Fruit Drops	2020-10-18 15:46:00	Sunday	5.0	3.0	9.0
5	NaN	22-10-2020 17:24:00	Thursday	3.5	2.0	NaN



```
df['Candy Variety']=df['Candy Variety'].fillna(df['Candy
Variety'].mode()[0])
df['Price']=df['Price'].fillna(df['Price'].mean())
df
```



	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	2020-02-09 14:05:00	Sunday	3.0	2.0	7.50
1	Sour Jelly	2020-10-24 18:00:00	Saturday	3.5	2.0	7.60
2	Candy Canes	2020-12-18 20:13:00	Friday	3.5	2.5	8.00
3	Sour Jelly	2020-10-25 10:00:00	Sunday	3.5	2.0	7.60
4	Fruit Drops	2020-10-18 15:46:00	Sunday	5.0	3.0	9.00
5	Sour Jelly	22-10-2020 17:24:00	Thursday	3.5	2.0	7.94

Notice how the technique of imputation given above corresponds with the principle of normal distribution (where the values in the distribution are more likely to occur closer to the mean rather than the edges) which results in a fairly good estimate of missing data. A few other ways to go about this include replacing missing values by picking the value from a normal distribution with the mean and standard deviation of the corresponding existing values or even replacing the missing value with an arbitrary value.

However, one must be reasonably cautious when using this technique because retention of data size with this technique could come at the cost of deterioration of data quality. For example, say in the above candy problem you were given 5 records instead of one with the 'Candy Variety' missing. Using the above technique you would predict the missing values as 'Sour Jelly' resulting in possibly predicting the high sales of Sour Jellies all through the year! Therefore, it is wise to filter out records that have greater than a

Pandas Tutorial Part-3

Pandas Tutorial Part-2

Pandas Tutorial Part-1

Tutorial- Hadoop Multinode Cluster Setup on Ubuntu

Data Visualizations Tools in R

R Statistical and Language tutorial

Introduction to Data Science with R

Apache Pig Tutorial: User Defined Function Example

Apache Pig Tutorial Example: Web Log Server Analytics

Impala Case Study: Web Traffic

Impala Case Study: Flight Data Analysis

Hadoop Impala Tutorial

Apache Hive Tutorial: Tables

Flume Hadoop Tutorial: Twitter Data Extraction

Flume Hadoop Tutorial: Website Log Aggregation

Hadoop Sqoop Tutorial: Example Data Export

Hadoop Sqoop Tutorial: Example of Data Aggregation

Apache Zookeeper Tutorial: Example of Watch Notification

Apache Zookeeper Tutorial: Centralized Configuration Management

Hadoop Zookeeper Tutorial for Beginners

Hadoop Sqoop Tutorial

Hadoop PIG Tutorial

Hadoop Oozie Tutorial

Hadoop NoSQL Database Tutorial

Hadoop Hive Tutorial

certain number of missing values or certain critical values missing and apply your discretion depending on the size and quality of data you are working with.

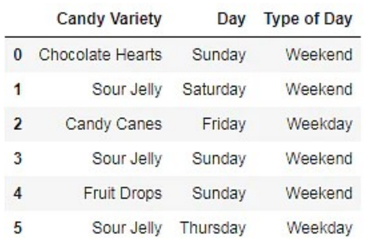
Access Solved End-to-End Data Science and Machine Learning Projects

2) Discretization

Discretization involves essentially taking a set of values of data and grouping sets of them together in some logical fashion into bins (or buckets). Binning can apply to numerical values as well as to categorical values. This could help prevent data from overfitting but comes at the cost of loss of granularity of data. The grouping of data can be done as follows:

1. Grouping of equal intervals
2. Grouping based on equal frequencies (of observations in the bin)
3. Grouping based on decision tree sorting (to establish a relationship with target)

```
df['Type of Day']=np.where(df['Day'].isin(['Saturday', 'Sunday']),
'Weekend', 'Weekday')
df[['Candy Variety','Day','Type of Day']]
```



	Candy Variety	Day	Type of Day
0	Chocolate Hearts	Sunday	Weekend
1	Sour Jelly	Saturday	Weekend
2	Candy Canes	Friday	Weekday
3	Sour Jelly	Sunday	Weekend
4	Fruit Drops	Sunday	Weekend
5	Sour Jelly	Thursday	Weekday

Recommended Projects to Learn Feature Engineering for Machine Learning

- Expedia Hotel Recommendations Data Science Project
- Ola Bike Ride Request Demand Prediction Machine Learning Project

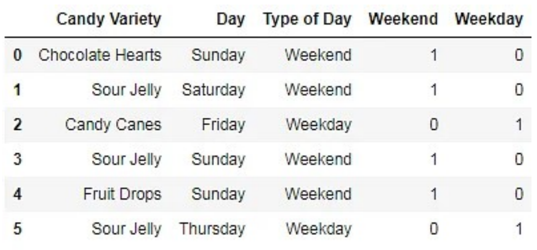
3) Categorical Encoding

Categorical encoding is the technique used to encode categorical features into numerical values which are usually simpler for an algorithm to understand. One hot encoding(OHE) is a popularly used technique of categorical encoding. Here, categorical values are converted into simple numerical 1's and 0's without the loss of information. As with other techniques, OHE has its own disadvantages and has to be used sparingly. It could result in a dramatic increase in the number of features and result in the creation of highly correlated features.

```
for x in df['Type of Day'].unique():
       df[x]=np.where(df['Type of Day']==x,1,0)
df[['Candy Variety','Day','Type of Day','Weekend','Weekday']]
```

Hadoop HDFS Tutorial
Hadoop hBase Tutorial
Hadoop Flume Tutorial
Hadoop 2.0 YARN Tutorial
Hadoop MapReduce Tutorial
Big Data Hadoop Tutorial for Beginners- Hadoop Installation

Top 15 Latest Recipes
How to create pipeline in sklearn
How to use k fold cross validation in sklearn
Find p values of regression model using sklearn
Fing p-values of regression model using sklearn
How to perform linear regression using sklearn
How to perform xgboost algorithm with sklearn
How to perform logistic regression in sklearn
How to download dataset from openml repo sklearn
What is sklearn library and how to install it
How to use Pip in Python
How to run a python file
Functions in Python
Explain the features of Amazon Relational Database System
Introduction to Amazon Relational Database System and its use cases
Explain the features of Amazon Detective



	Candy Variety	Day	Type of Day	Weekend	Weekday
0	Chocolate Hearts	Sunday	Weekend	1	0
1	Sour Jelly	Saturday	Weekend	1	0
2	Candy Canes	Friday	Weekday	0	1
3	Sour Jelly	Sunday	Weekend	1	0
4	Fruit Drops	Sunday	Weekend	1	0
5	Sour Jelly	Thursday	Weekday	0	1

Besides OHE there are other methods of categorical encodings, such as 1. Count and Frequency encoding- captures each label's representation, 2. Mean encoding -establishes the relationship with the target and 3.Ordinal encoding- number assigned to each unique label.

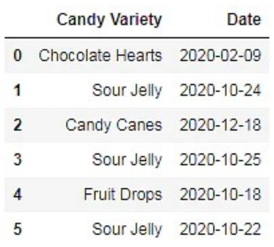
Recommended Reading:

- **The A-Z Guide to Gradient Descent Algorithm and Its Variants**
- **8 Feature Engineering Techniques for Machine Learning**
- **Exploratory Data Analysis in Python-Stop, Drop and Explore**
- **Logistic Regression vs Linear Regression in Machine Learning**
- **Correlation vs. Covariance**

4) Feature Splitting

Splitting features into parts can sometimes improve the value of the features toward the target to be learned. For instance, in this case, Date better contributes to the target function than Date and Time.

```
df['Date and Time'] = pd.to_datetime(df['Date and Time'])
df['Date']=df['Date and Time'].dt.date
df[['Candy Variety','Date']]
```



	Candy Variety	Date
0	Chocolate Hearts	2020-02-09
1	Sour Jelly	2020-10-24
2	Candy Canes	2020-12-18
3	Sour Jelly	2020-10-25
4	Fruit Drops	2020-10-18
5	Sour Jelly	2020-10-22

5) Handling Outliers

Outliers are unusually high or low values in the dataset which are unlikely to occur in normal scenarios. Since these outliers could adversely affect your prediction they must be handled appropriately. The various methods of handling outliers include:

1. Removal: The records containing outliers are removed from the distribution. However, the presence of outliers over multiple variables could result in losing out on a large portion of the datasheet with this method.
2. Replacing values: The outliers could alternatively bed treated as missing values and replaced by using appropriate imputation.
3. Capping: Capping the maximum and minimum values and replacing them with an arbitrary value or a value from a variable distribution.
4. Discretization

6) Variable Transformations

Variable transformation techniques could help with normalizing skewed data. One such popularly used transformation is the logarithmic transformation. Logarithmic transformations operate to compress the larger numbers and relatively expand the smaller numbers. This in turn results in less skewed values especially in the case of heavy-tailed distributions. Other variable transformations used include Square root transformation and Box cox transformation which is a generalization of the former two.

7) Scaling

Feature scaling is done owing to the sensitivity of some machine learning algorithms to the scale of the input values. This technique of feature scaling is sometimes referred to as feature normalization. The commonly used processes of scaling include:

1. Min-Max Scaling: This process involves the rescaling of all values in a feature in the range 0 to 1. In other words, the minimum value in the original range will take the value 0, the maximum value will take 1 and the rest of the values in between the two extremes will be appropriately scaled.
2. Standardization/Variance scaling: All the data points are subtracted by their mean and the result divided by the distribution's variance to arrive at a distribution with a 0 mean and variance of 1.

It is necessary to be cautious when scaling sparse data using the above two techniques as it could result in additional computational load.

8) Creating Features

Feature creation involves deriving new features from existing ones. This can be done by simple mathematical operations such as aggregations to obtain the mean, median, mode, sum, or difference and even product of two values. These features, although derived directly from the given data, when carefully chosen to relate to the target can have an impact on the performance(as demonstrated later!)

While the techniques listed above are by no means a comprehensive list of techniques, they are popularly used and should definitely help you get started with feature engineering in machine learning.

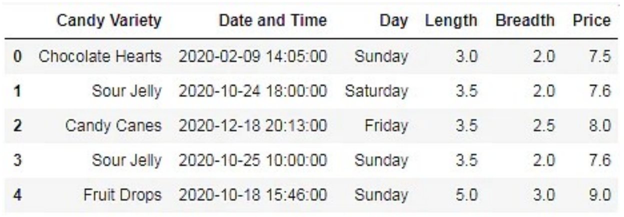
Feature Engineering Python-A Sweet

Takeaway!

We have gone over what Feature Engineering is, some commonly used feature engineering techniques, and its impact on our machine learning model's performance. But why just take someone's word for it?

Let's consider a simple price prediction problem for our candy sales data –

```
data={'Candy Variety':['Chocolate Hearts','Sour Jelly','Candy Canes','Sour
Jelly','Fruit Drops'], 'Date and Time':['09-02-2020 14:05','24-10-2020
18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
'Day':['Sunday','Saturday','Friday','Sunday','Sunday'], 'Length':[3, 3.5,
3.5, 3.5, 5], 'Breadth':[2,2,2.5,2,3], 'Price':[7.5, 7.6, 8, 7.6, 9]}
df = pd.DataFrame(data)
df
```



```
      Candy Variety        Date and Time       Day  Length  Breadth  Price
0  Chocolate Hearts  2020-02-09 14:05:00    Sunday     3.0      2.0    7.5
1        Sour Jelly  2020-10-24 18:00:00  Saturday     3.5      2.0    7.6
2       Candy Canes  2020-12-18 20:13:00    Friday     3.5      2.5    8.0
3        Sour Jelly  2020-10-25 10:00:00    Sunday     3.5      2.0    7.6
4       Fruit Drops  2020-10-18 15:46:00    Sunday     5.0      3.0    9.0
```

We will use a simple linear regression model to predict the price of the various types of candies and experience first-hand how to implement python feature engineering.

Let's start by building a function to calculate the coefficients using the standard formula for calculating the slope and intercept for our simple linear regression model.

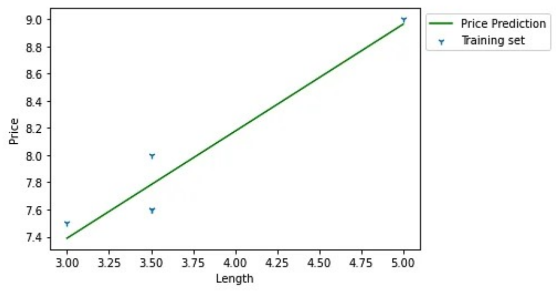
```
import matplotlib.pyplot as plt
import numpy as np

def simple_linear_regression(x, y):
    # number of observations
    n = np.size(x)

    mean_x = np.mean(x)
    mean_y = np.mean(y)


    xy = np.sum(y*x) - n*mean_y*mean_x
    xx = np.sum(x*x) - n*mean_x*mean_x

    # calculating slope
    m = xy / xx

    #calculating intercept
    c = mean_y - m*mean_x

    return m,c
```

Now we build our initial model *without* any Feature Engineering, by trying to relate one of the given features to our target. From observing the given data we know that it is most likely that the Length or the Breadth of the candy is most likely related to the price.

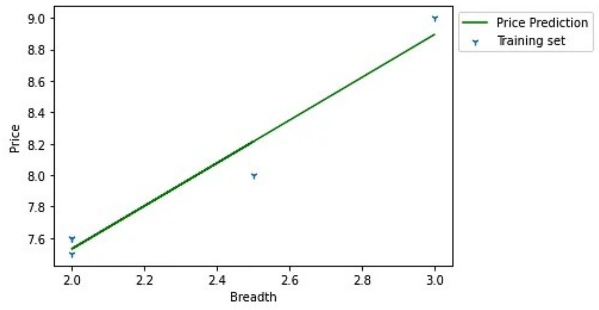
Get confident to build end-to-end projects.

Let us start by trying to relate the length of the candy with the price.

```
x=df['Length'].to_numpy()
y=df['Price'].to_numpy()

m,c = simple_linear_regression(x,y)
y_pred = c + m*x

plt.plot(x, y_pred , color = "g", label='Price Prediction')
plt.scatter(df['Length'].to_numpy() , y, marker='1', label='Training set')
plt.xlabel('Length')
plt.ylabel('Price')
plt.legend(bbox_to_anchor=(1, 1))
plt.show()
```

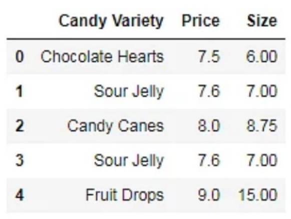


We observe from the figure that Length does not have a particularly linear relation with the price.
We attempt a similar prediction with the Breadth to get a somewhat similar outcome. (You can execute this by simply replacing ‘Length by ‘Breadth in the above code block.)



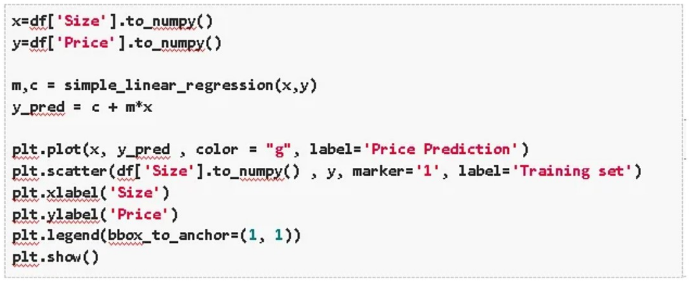
Finally, it’s time to apply our newly gained knowledge of Feature Engineering! Instead of using just the given features, we use the Length and Breadth feature to derive a new feature called Size which (you might have already guessed) should have a much more monotonic relation with the Price of candy than the two features it was derived from.

```
df['Size']=df['Breadth']*df['Length']
df[['Candy Variety','Price', 'Size']]
```

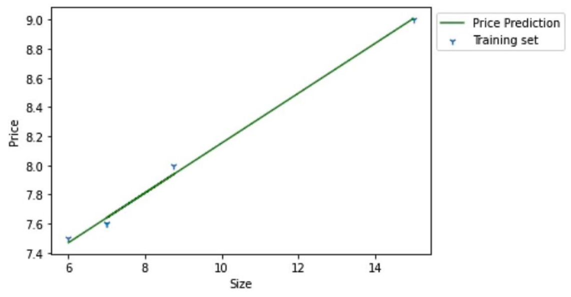


	Candy Variety	Price	Size
0	Chocolate Hearts	7.5	6.00
1	Sour Jelly	7.6	7.00
2	Candy Canes	8.0	8.75
3	Sour Jelly	7.6	7.00
4	Fruit Drops	9.0	15.00

We now use this new feature Size to build a new simple linear regression model.



```
x=df['Size'].to_numpy()
y=df['Price'].to_numpy()

m,c = simple_linear_regression(x,y)
y_pred = c + m*x

plt.plot(x, y_pred , color = "g", label='Price Prediction')
plt.scatter(df['Size'].to_numpy() , y, marker='1', label='Training set')
plt.xlabel('Size')
plt.ylabel('Price')
plt.legend(bbox_to_anchor=(1, 1))
plt.show()
```



If you thought that the previous predictions with the Length(or Breadth) feature were not too disappointing, the results with the Size feature you will agree are quite spectacular!

We have demonstrated with this example, that by simply multiplying the Length and Breadth features of a pack of candy you can achieve the Price predictions well beyond what you would with the much less efficient relationship of Prices to Length (or Breadth). However, when working with real-life data, the way you use Feature Engineering could be the difference between a simple model that works perfectly well and a complex model that doesn't.

Closing Thoughts on Machine Learning

Feature Engineering Techniques

Candies aside, the takeaway from this should be that simple but well-thought-out [Feature Engineering] could be what brings us to the tipping point between a good machine learning model and a bad one. It is important to remember that the activities involved in Feature Engineering, owing to their nature, need not always be straightforward. It could involve an iterative process of brainstorming, creating features, building models, and doing it all over again from the top. It cannot be exaggerated enough that there is no ultimate approach, just one that is right for your purpose.

But rest assured, with practice it definitely gets easier. The aspects covered in this article should definitely help you get started on your journey towards simpler models and better predictions.

And when in doubt, still choose to trust the process of Feature Engineering, for as Ronald Coase rightly said 'If you torture the data long enough, they will confess anything.'

PREVIOUS   **NEXT**

Your Data Skills Need to Get Stronger
And We Have The Projects Ready
GET ACCESS TO SOLVED PROJECTS

Tr
Ca

Ma

Dat

Deep Learning Projects

Big Data Projects

Apache Hadoop Projects

Apache Spark Projects

Show more

Trending Projects

Walmart Sales Forecasting Data Science Project

BigMart Sales Prediction ML Project

Music Recommender System Project

Credit Card Fraud Detection Using Machine Learning

Resume Parser Python Project for Data Science

Time Series Forecasting Projects

Show more

Trending Blogs

Machine Learning Projects for Beginners with Source Code

Data Science Projects for Beginners with Source Code

Big Data Projects for Beginners with Source Code

IoT Projects for Beginners with Source Code

Data Analyst vs Data Scientist

Data Science Interview Questions and Answers

Show more

Trending Recipes

Search for a Value in Pandas DataFrame

Pandas Create New Column based on Multiple Condition

LSTM vs GRU

Plot ROC Curve in Python

Python Upload File to Google Drive

Optimize Logistic Regression Hyper Parameters

Show more

Trending Tutorials

PCA in Machine Learning Tutorial

PySpark Tutorial

Hive Commands Tutorial

MapReduce in Hadoop Tutorial

Apache Hive Tutorial -Tables

Linear Regression Tutorial

Show more

ProjectPro