Pre-processing pipeline:

Even the best experienced Data Scientists are not always familiar with the best practices involved with developing a Machine Learning pipeline technique. There is a lot of confusions happen about what are steps should be involved, what should be their sequence order and in general, how to ensure that the insights happen you to create are accurate and valuable.

For building of any machine learning model, it is important to have a sufficient/suitable amount of data to train the best model. The data is often collected from various resources/repositories and might be available in different file formats. Due to this reason, data cleansing and pre-processing become a most crucial step in any machine learning projects.

Whenever new dataset/data points are added to the existing dataset/data, we need to perform the some pre-processing steps again and before we can use the machine learning models to make right predictions. This becomes a ridiculous and time-consuming process.

Whenever we have a new data point is introduced, the machine learning pipeline performs the different steps as defined and uses the machine learning model to predict the test variable.

In order to make this story intuitive, we will learn all the concepts while working on a real world data – restaurant food prediction.

As a part of this problem dataset, we are provided with the information about the restaurant (location, state, plan etc), CUISINES (different types of cuisines, category, price, etc.) and restaurant id’s data. Using this information, we have to forecast the sales of the products in the restaurant.

You can read the detailed problem dataset statement and download the dataset from [here](https://datahack.analyticsvidhya.com/contest/practice-problem-big-mart-sales-iii/). Below is the complete set of features in this data. The test/target variable here is the *cost*.

|  |  |
| --- | --- |
| TITLE | The feature of this restaurant which can help identify deal of the itemsicts. |
| RESTAURANT\_ID | A unique ID for each restaurant |
| CUISINES | The variety of cuisines that the restaurant offers |
| TIME | The open hours of the restaurant |
| CITY | The city in which the restaurant is located |
| LOCALITY | The locality of the restaurant |
| RATING | The average rating of the restaurant by customers |
| VOTES | The overall votes received by the restaurant |
| COST | The average cost of a two-person meal |

I encourage you to go through the problem statement and data description once before moving to the next section so that you have a most understanding of the features present in the data.

To build a any machine learning pipeline, the first/most requirements is to define the structure of the pipeline. In other words, we must list down the exact steps to which could go into our machine learning pipeline. In order to do so, we will build a prototype machine learning model on the existing data before we create a pipeline concept. The main idea behind building a prototype is to understand the dataset and necessary preprocessing steps that are required before fit the model building process. Based on our learning’s from the prototype models, we will design a machine learning pipeline that covers all the essential preprocessing steps to be included.

The main focus of this section will be on building a prototype that will help us in defining the actual machine learning pipeline for our restaurant prediction project.

Performing the EDA process and preprocessing techniques: What is the first thing that you have to do when you are provided with a particular dataset.. You would explore the dataset, go through the individual keys/column/attributes and clean the data to make it ready for the model building process. That is exactly what we will be doing here. We will explore the variables and find out the mandatory pre-processing steps required for the given dataset. Let us start by checking if there are any missing values in the dataset. We will use the *isnull().sum()* function here

Loading the particular libraries:

Import pandas as pd

Import matplotlibb as plt

Import seanborn as sns

Install-the-library:  
!pip3 install category\_encoders

Read the dataset:

df1 = pd.read\_excel ('DataR\_Train.xlsx')

df1

| **TITLE** | **RESTAURANT\_ID** | **CUISINES** | **TIME** | **CITY** | **LOCALITY** | **RATING** | **VOTES** | **COST** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | CASUAL DINING | 9438 | Malwani, Goan, North Indian | 11am – 4pm, 7:30pm – 11:30pm (Mon-Sun) | Thane | Dombivali East | 3.6 | 49 votes | 1200 |
| 1 | CASUAL DINING,BAR | 13198 | Asian, Modern Indian, Japanese | 6pm – 11pm (Mon-Sun) | Chennai | Ramapuram | 4.2 | 30 votes | 1500 |
| 2 | CASUAL DINING | 10915 | North Indian, Chinese, Biryani, Hyderabadi | 11am – 3:30pm, 7pm – 11pm (Mon-Sun) | Chennai | Saligramam | 3.8 | 221 votes | 800 |

Once the dataset are loaded into memory, it is key to familiarise with them. An initial exploratory data analysis should be conducted to understand the granularity of the dataset, what type of information is available and the target/final/test variable. @mean, max, min @25, @50, @75 data will be available. The values are also key to notice to find the outliers. An easy way to get the most useful statistics is using describe ():

Df.head() ---- to five column of the dataset

Df.describe() ---- finds the statistics representation @mean, max, min @25, @50, @75 data will be available.

Finding the data set where the column have null values

Df.isnull ().sum ()

CITY 147

COST 4231

CUISINES 0

LOCALITY 128

RATING 4

RESTAURANT\_ID 0

TIME 0

TITLE 0

VOTES 1606

input 0

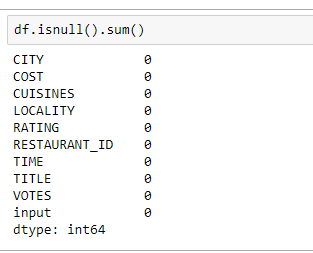
dtype: int64

There are 4 varibale with missing values.

#Filling Null Values with Mode, median

for column in ['CITY','LOCALITY','RATING','VOTES','COST']:

df[column].fillna(df[column].mode()[0], inplace=True)



Data Collection: No machine learning projects can be successful without dataset. It is necessary to get enough high-quality sample sets to ensure the success of the project. More specifically, an algorithm like Linear Regression or Logistic Regression can work well even with only a few hundreds of observations, while more complex algorithms, anything from CART to neural networks, will need at least several thousands. In terms of quality, data from automatic, inflexible systems, without human interference will be better than suited for machine learning application. For instance, free text fields should be avoided. If the data collection set-up is a Google Form that the Data Scientist creates a dropdown menu is much more preferred than a free text box. In general, however, every Data Scientist should accept that data is going to be a messy, manually typed and full of missing data in column values. It just is part of the job.

These are few techniques that are used to entering/dividing the splitting the data set into training & testing.

* Gathering the information
* Data collection
* Loading the data
* Performed that EDA process
* Data Aggregation and data cleaning technique
* Dealing with categorical data set

One of the common ways to encode categories is to map them to numbers, simply assigning the same number to the same category value. However, this approach is extremely detrimental because it creates relationships that do not really exist in reality data. For instance, if we encode [‘**CUISINES**, ‘**CITY**, ‘**TITLE**] into [1, 2, 3], we are basically stating that ‘**TITLE**is three times as ‘restaurant, whatever that means, or that twice ‘restaurant’ is equal to ‘cuisines.

Pipeline-constructor with tuples of (<a descriptive name>, a function). You can pass arguments to the first function’s init() method where it says some argus. Each method must be implement the <fit() and transform()> functions. Except the last function only implements fit().

pipeline = Pipeline([ ('<someName>', <someFunction(some args))>, <('next Name', nextFuntion())> ])

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

df = pipeline.fit\_transform(dataframe)

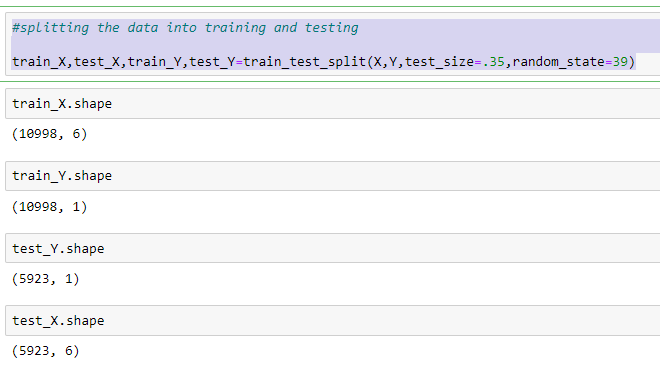
The second step calls the StandardScaler() to normalize the values in the array. We don’t have to pass it any arguments since it knows to use the data from the previous step. pipeline = Pipeline([ ('toNumbers', ToNumbers(cols)), ('scaler', StandardScaler()) ])

data = pipeline.fit\_transform(df)

**Split the dataset into Training and Test data:**

Once we have a dataset consisting of the relevant features/keys, we can proceed to split it into Training & Test sets. Usually, Training account for 70% of the dataset, while Test is the remaining 30%. If there are enough data to find, the Test set can be reduced to 25%, since the number of samples will be sufficiently high data anyway. In particular, the Training data would be the 75% of the data ordered temporally, while the Test set should be the most recent part. This prevents the model from, quite literally.

#splitting the data into training and testing

train\_X,test\_X,train\_Y,test\_Y=train\_test\_split(X,Y,test\_size=.35,random\_state=39)

Only once we have split the dataset and we can deal with missing values. If we were to impute the data with all the datasets, we would be filling the Training set with information from the Testing set. The correct approach is to fit an Imputation Learner with the Training set only, and then transform both the Training and Testing sets. There are many solutions that can be taken to impute missing data. However, these approaches are generally to avoid. The main reason is that they completely modified the distribution of the feature, reducing the variance, because we are adding multiple times a constant term values. In addition, they do not consider the relationship across the features, focusing only on a feature at a time.

Perform the model to fit the best result, the set of models we usually work.

We can use a variety of metrics, but there are two distinct sets for Classification and Regression tasks that are mainly focused.

Conclusion: Having a well-defined set of structure before performing any task often helps in efficient execution of the same data. And this is true even in case of building a machine learning models. Once you have to build a model on a dataset, you can easily break down the steps and define a structured Machine learning pipeline. In this stage I have just perform the sample dataset I have completed my project work as a restaurant prediction.