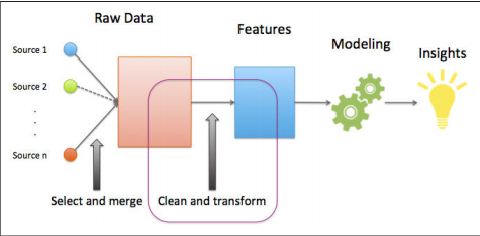
### What is Feature Engineering?

Feature engineering is about creating new input features from your existing ones.In general, you can think of data cleaning as a process of subtraction and feature engineering as a process of addition.  
Feature engineering sits right between ‘’data’’ and “modeling” in the machine learning pipeline for making sense of data.  
It is a crucial step, because the right features can make the job of modeling much easier, and therefore the whole process has a higher chance of success. Some people estimate that 80% of their effort in a machine learning application is spent on feature engineering and data cleaning.

**The Machine Learning Pipeline:**





**Data:**

What we call "data" are observations of real world phenomena. For instance, stock market data might involve observations of daily stock prices, announcements of earnings from individual companies, and even opinion articles from pundits. Personal biometric data can include measurements of our minute-by-minute heart rate, blood sugar level, blood pressure, etc. Customer intelligence data. We can come up with endless examples of data across different domains. Each piece of data provides a small window into one aspect of reality. The collection of all of these observations gives us a picture of the whole.

**Tasks:**

Why do we collect data? Usually, there are tasks we’d like to accomplish using data. These tasks might be: “Decide which stocks I should invest in," “Understand how to have a healthier lifestyle,” or “Understand my customers’ changing tastes, so that my business can serve them better.” The path from data to answers is usually a giant ball of mess.

This is because the workflow probably has to pass through multiple steps before resulting in a reasonably useful answer. For instance, the stock prices are observed on the trading floors, aggregated by an intermediary like Thompson Reuters, stored in a database, bought by your company, converted into a Hive store on a Hadoop cluster, pulled out of the store by a script, subsampled, massaged and cleaned by another script, dumped to a file on your desktop, converted to a format that you can try out in your favorite modeling library in R, Python or Scala, predictions dumped back out to a csv file, parsed by an evaluator, iterated multiple times, finally rewritten in C++ or Java by your production team, run on all of the data, and final predictions pumped out to another database.

**Models**:

Trying to understand the world through data is like trying to piece together reality using a noisy, incomplete jigsaw puzzle with a bunch of extra pieces. This is where mathematical modeling in particular statistical modeling comes in. The language of statistics contains concepts for many frequent characteristics of data: missing, redundant, or wrong. As such, it is good raw material out of which to build models.

A mathematical model of data describes the relationship between different aspects of data. For instance, a model that predicts stock prices might be a formula that maps the company’s earning history, past stock prices, and industry to the predicted stock price. A model that recommends music might measure the similarity between users, and recommend the same artists for users who have listened to a lot of the same songs. Mathematical formulas relate numeric quantities to each other. But raw data is often not numeric. So there must be a piece that connects the two together. This is where features come in.

**Features:**

A feature is a numeric representation of raw data. There are many ways to turn raw data into numeric measurements. So features could end up looking like a lot of things. The choice of features is tightly coupled with the characteristics of raw data and the choice of the model. Naturally, features must derive from the type of data that is available. Perhaps less obvious is the fact that they are also tied to the model; some models are more appropriate for some type of features, and vice versa. Feature engineering is the process of formulating the most appropriate features given the data and the model.  
Features and models sit between raw data and the desired insight. In a machine learning workflow, we pick not only the model, but also the features. This is a double jointed lever, and the choice of one affects the other. Good features make the subsequent modeling step easy and the resulting model more capable of achieving the desired task. Bad features may require a much more complicated model to achieve the same level of performance.

**Feature engineering involves:**

a) Handling numerical missing value

b) Handling categorical missing value

c) Encoding

d) Feature Scaling

e) Handling imbalance data

### Handling Missing value:

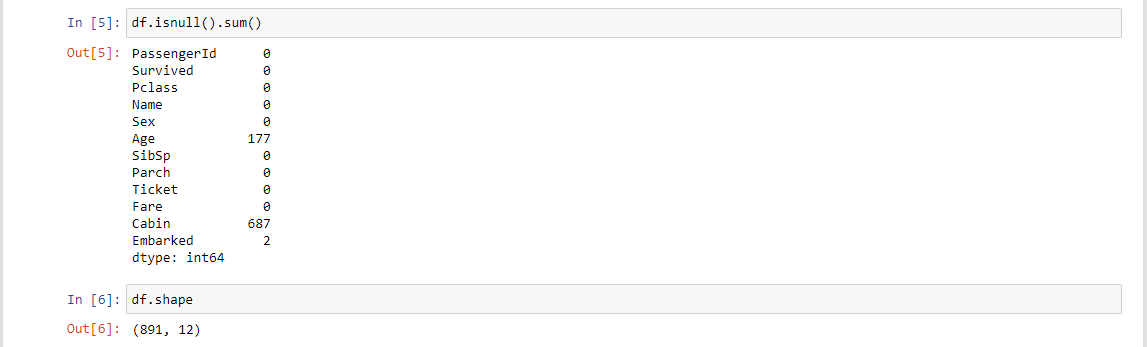
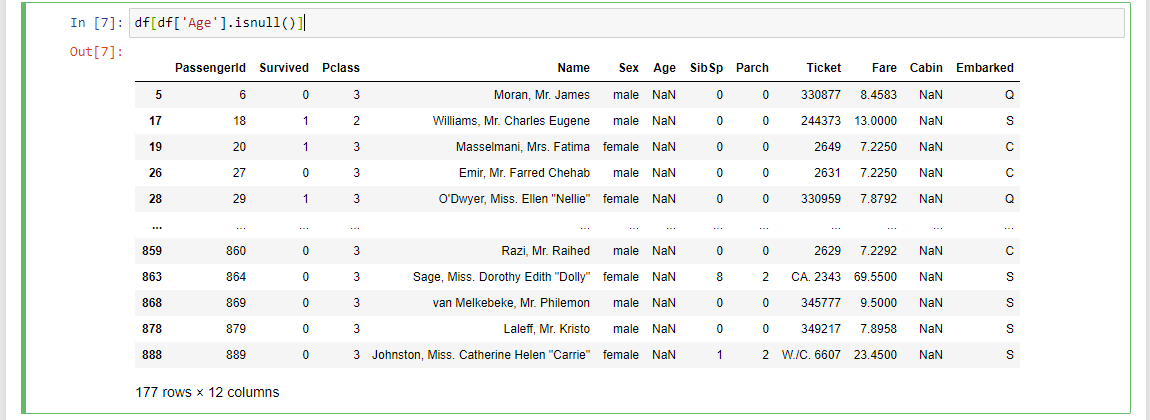
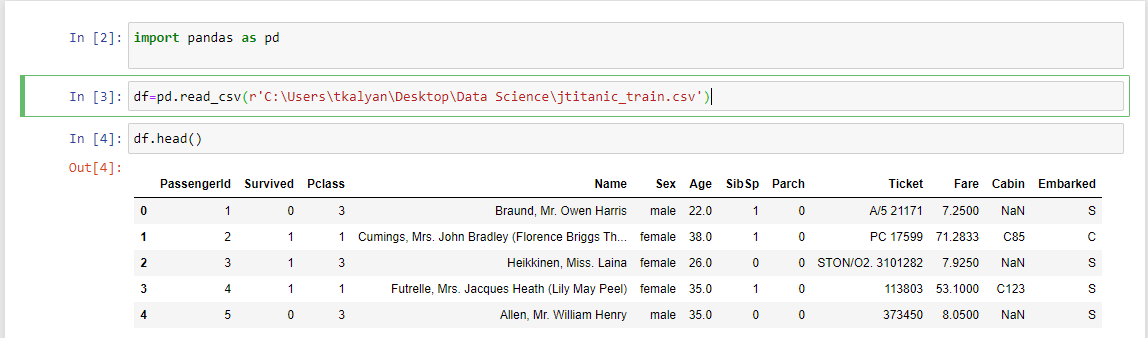
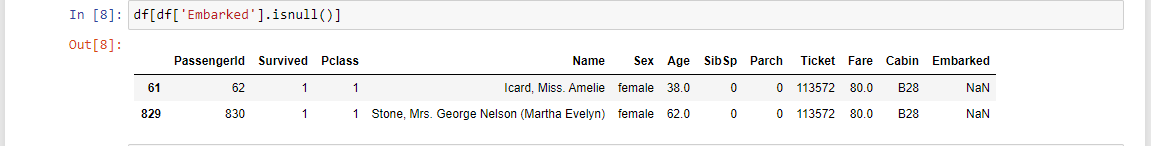
### There are some instances where a particular element is absent because of various reasons, such as, corrupt data, failure to load the information, or incomplete extraction. This is called missing values.

### To handle the missing data below are the methods:

### Deleting Rows

### Replacing With Mean/Median/Mode

* Random Sample imputation
* Capturing NAN values with a new feature
* End of Distribution Imputation
* Arbitrary imputation

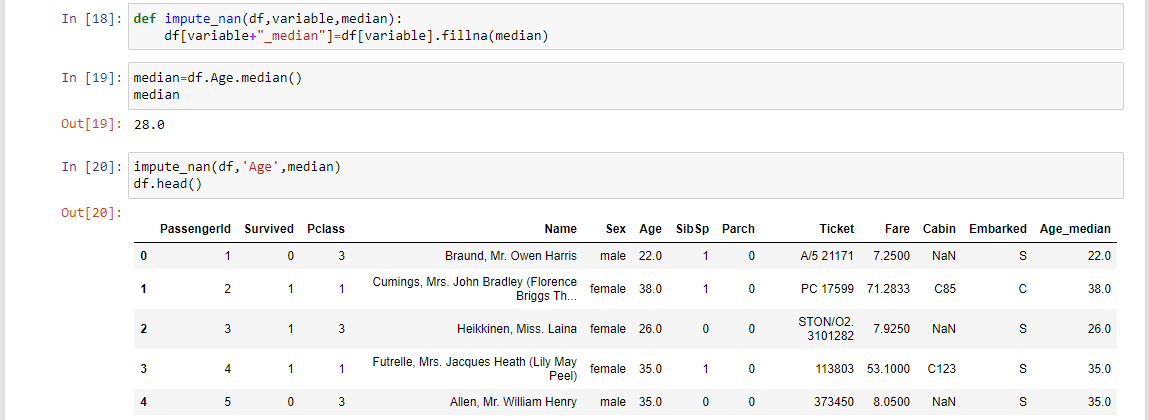
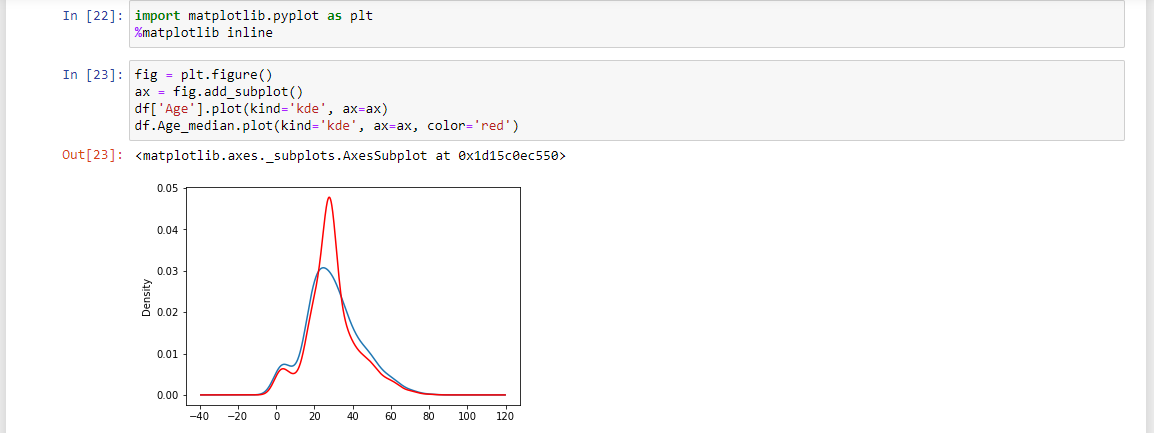
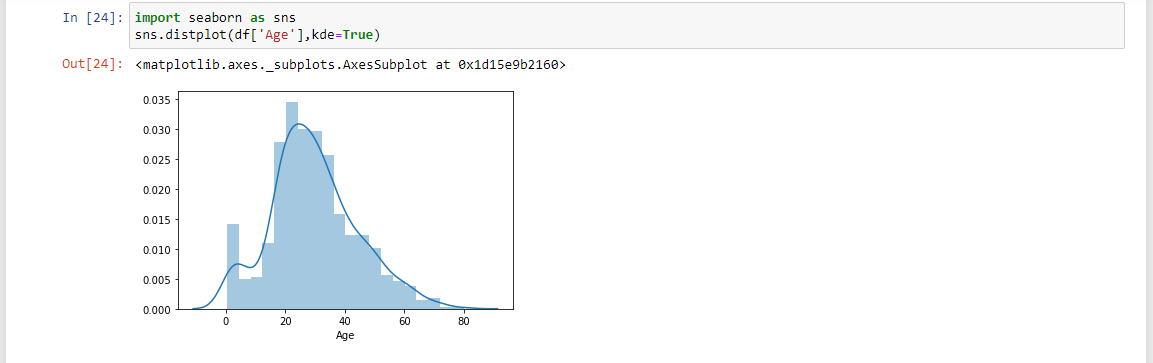
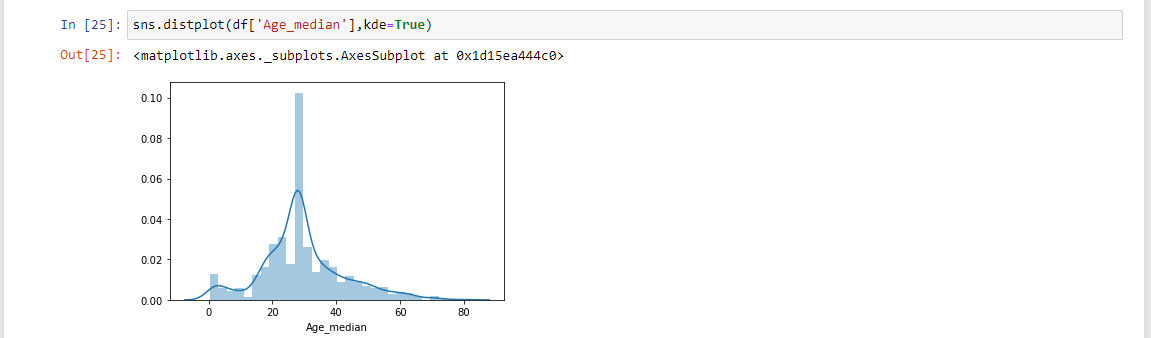
  
  
  
  


### Deleting rows:

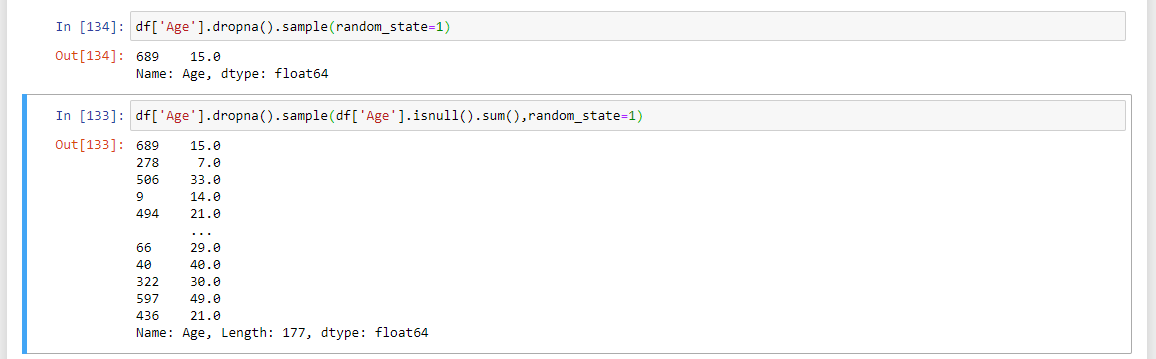
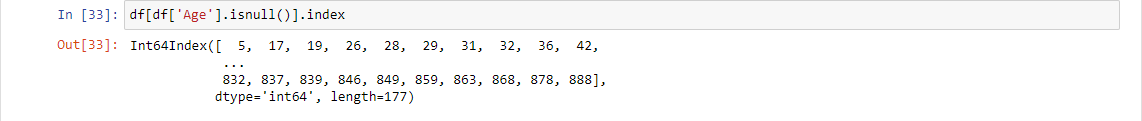
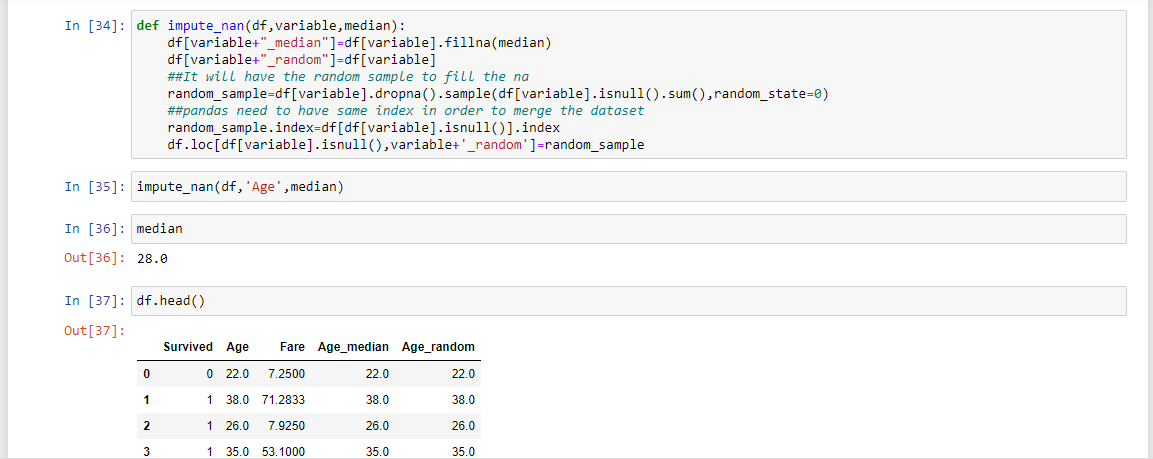
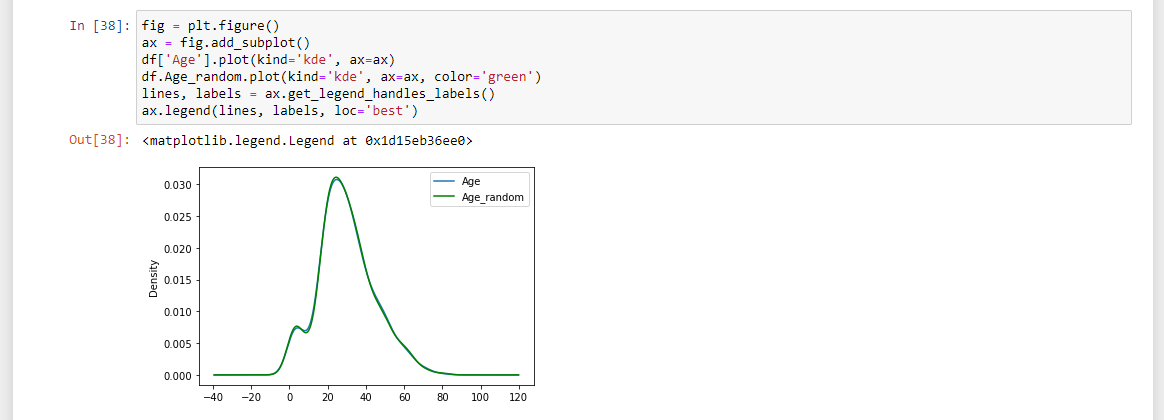
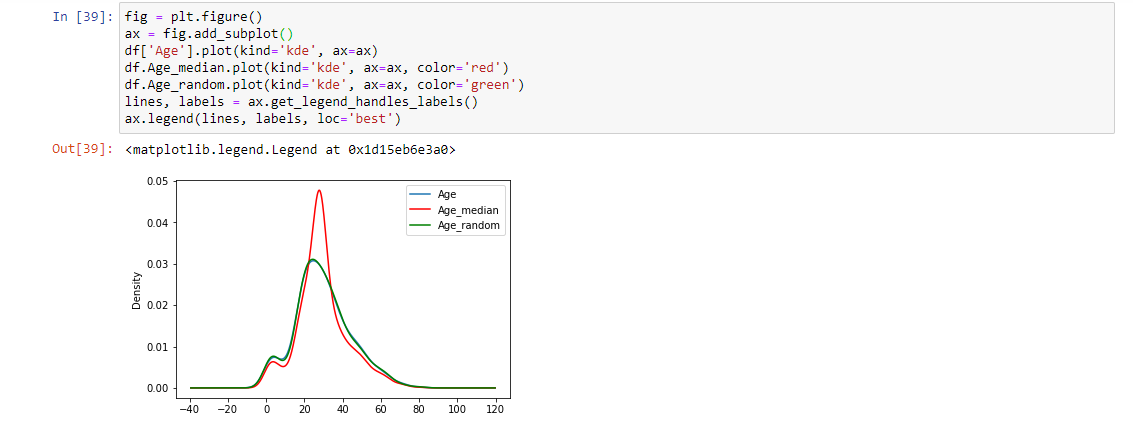
* Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values. This method is advised only when there are enough samples in the data set. One has to make sure that after we have deleted the data, there is no addition of bias. Removing the data will lead to loss of information which will not give the expected results while predicting the output.
* Dropna (in place=True)

### Replacing With Mean/Median/Mode:

* This strategy can be applied on a feature which has numeric data.
* We can calculate the mean, median or mode of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with the above three approximations are a statistical approach of handling the missing values. This method is also called as leaking the data while training.
  + Mean()
  + Median()
  + Mode()

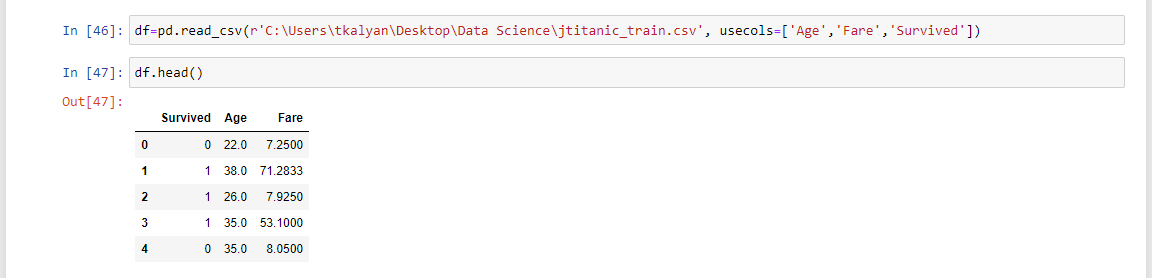
### Random Sample imputation:

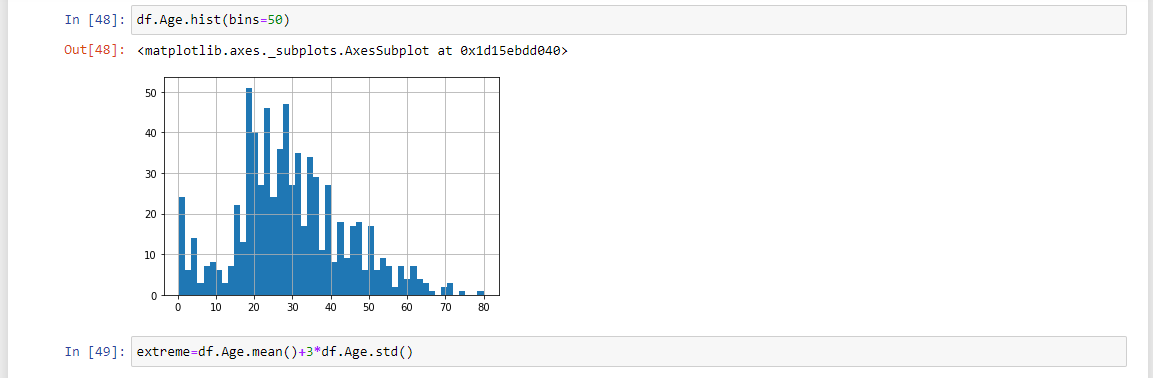
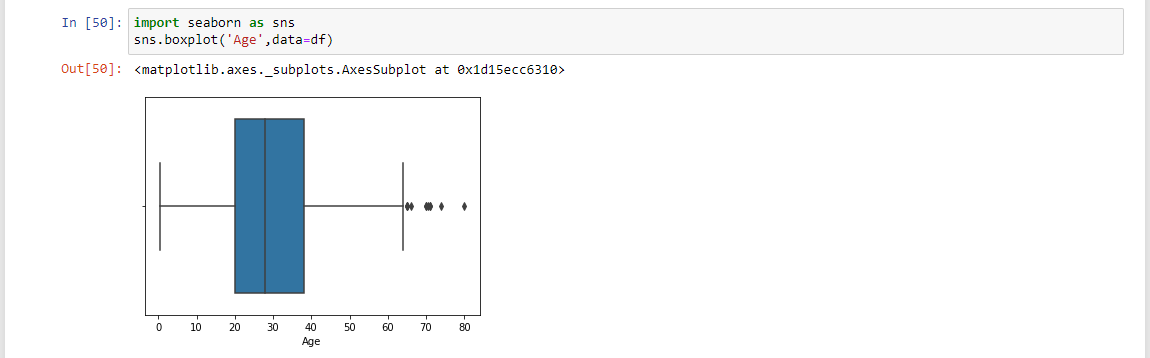
Random Sample imputation consists of taking random observations from the dataset and we use this observation to replace the NAN values  
  
  
  
  
  
  
  
  
  
  
  


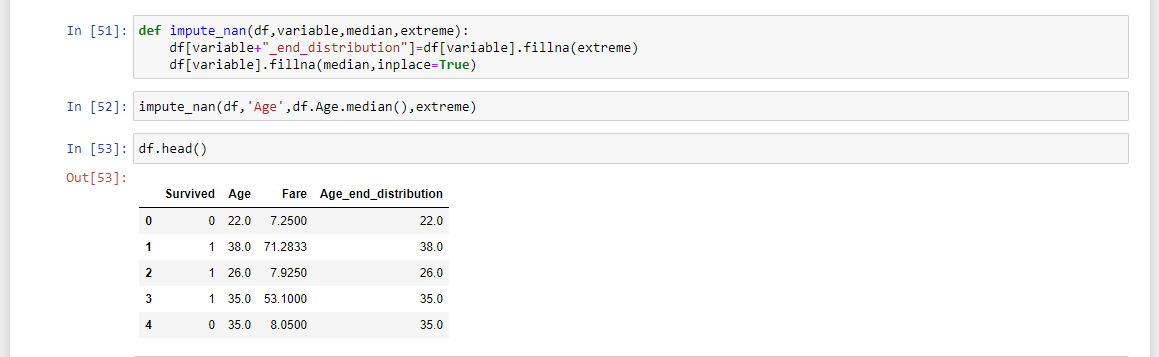
### Capturing NAN values with a new feature:

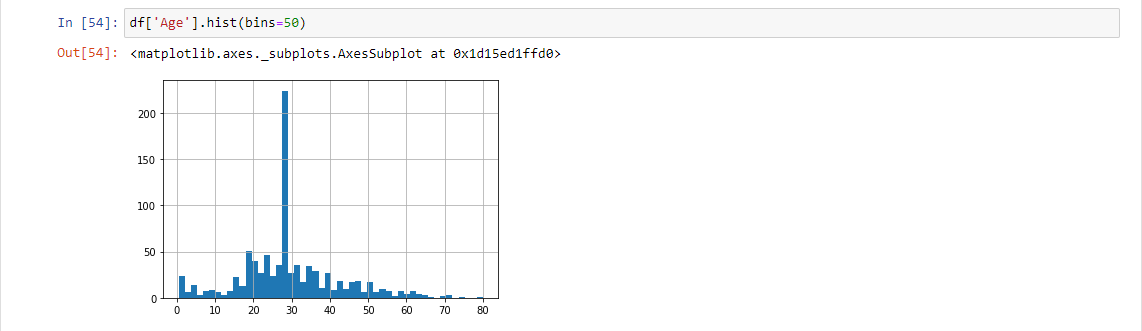
### End of Distribution imputation:

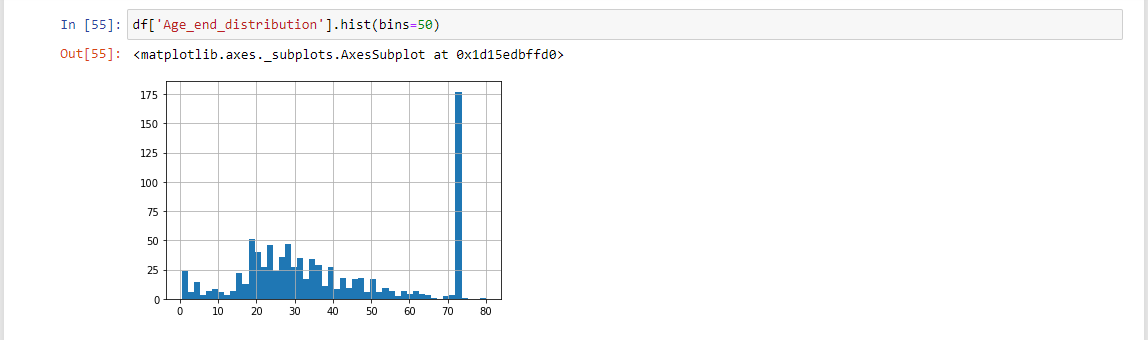
* In this scenario, one would want to replace missing data with the values that are at tails of the   
  distribution

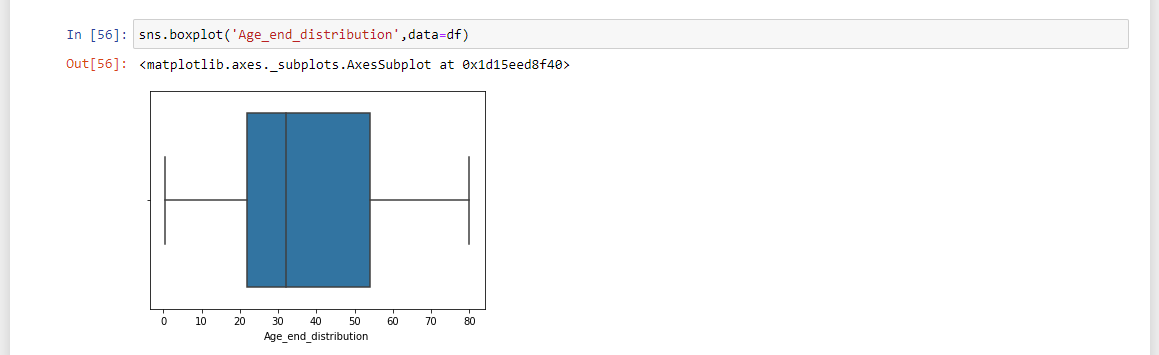




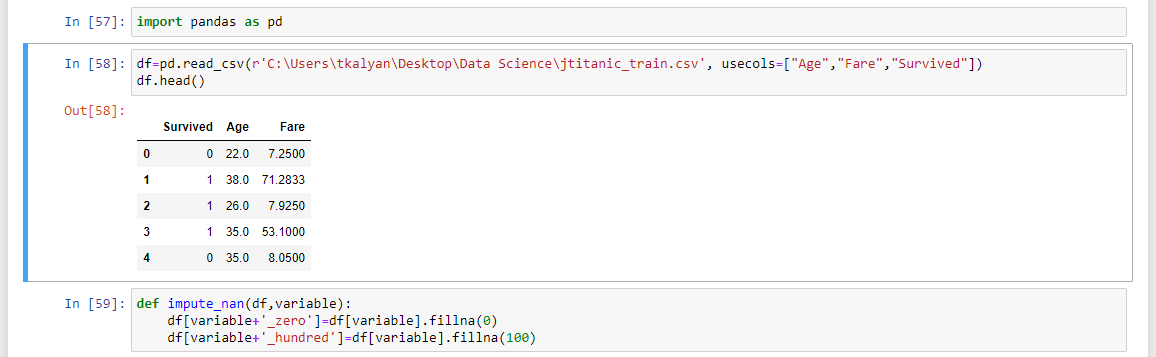


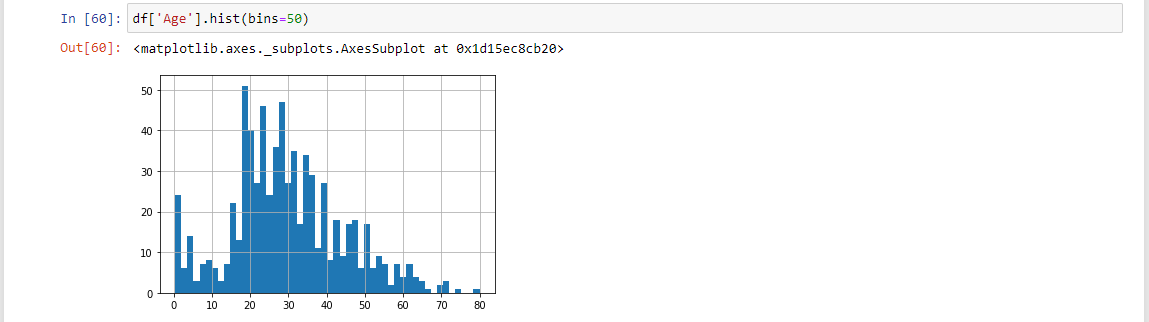




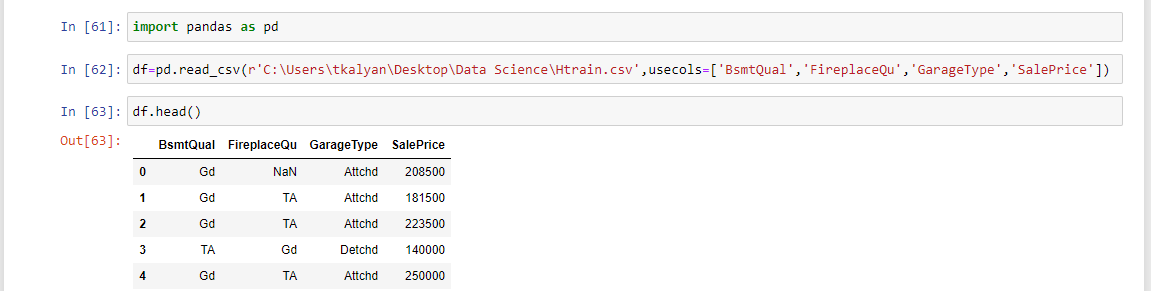
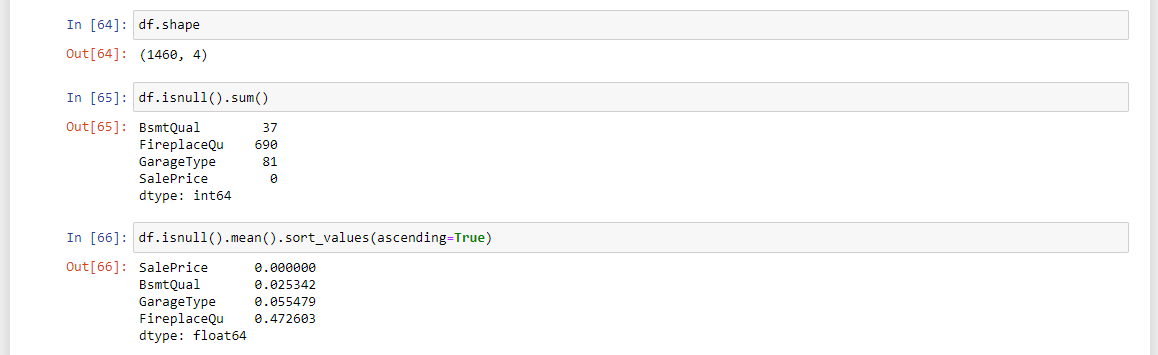
### Arbitrary Imputation:

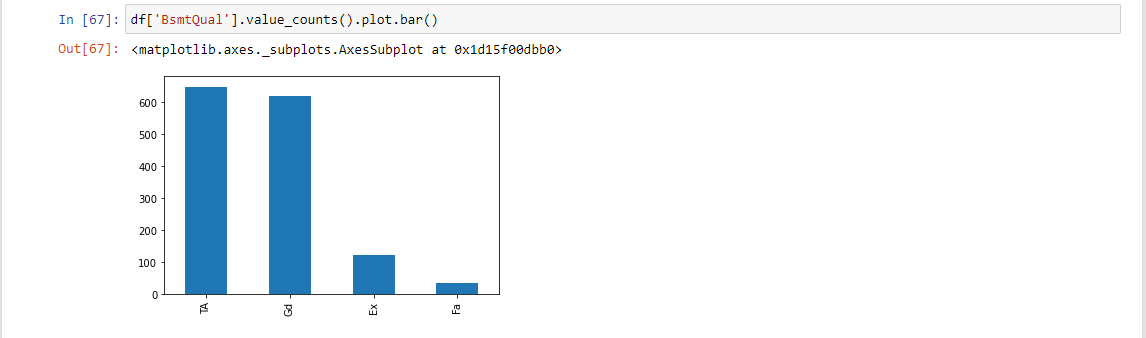
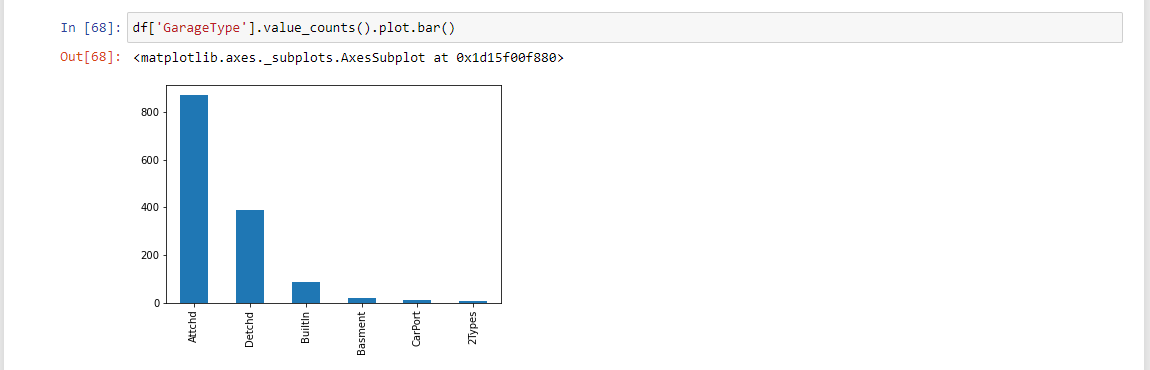
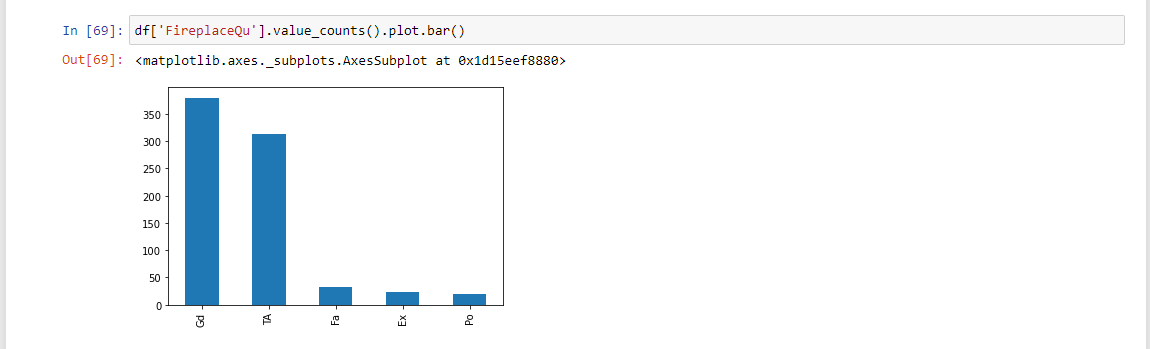
* This technique was derived from kaggle competition .It consists of replacing NAN by an arbitrary value

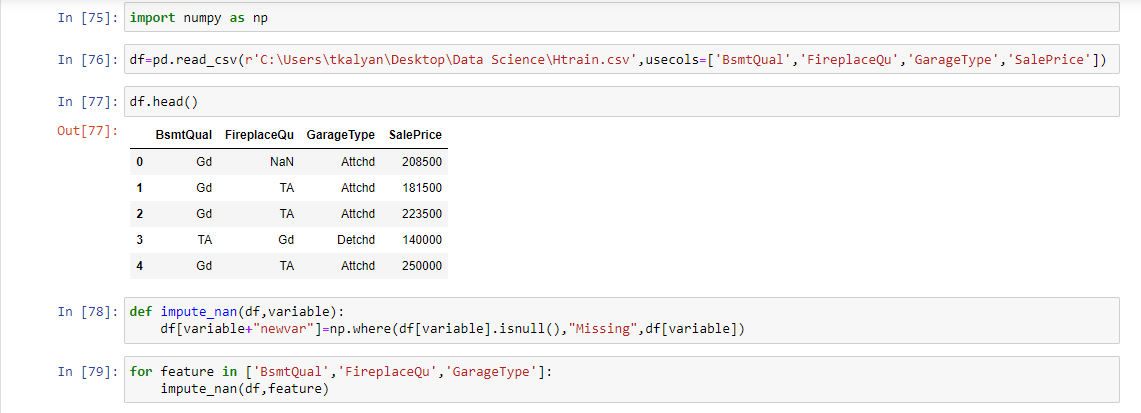
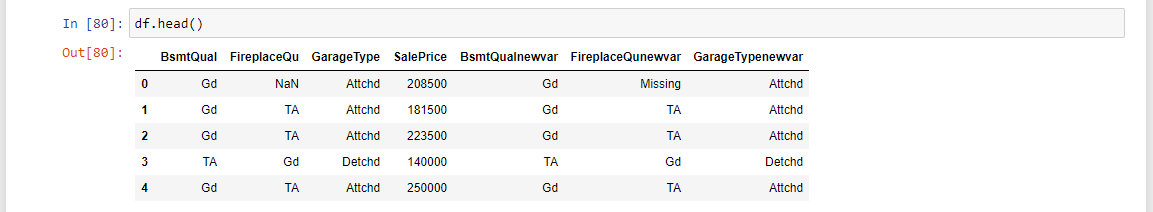




### Handling Categorical Missing Values:

  
  
  
Compute the frequency for every feature


Suppose if you have more frequent categories, we just replace NAN with a new category  
  
  


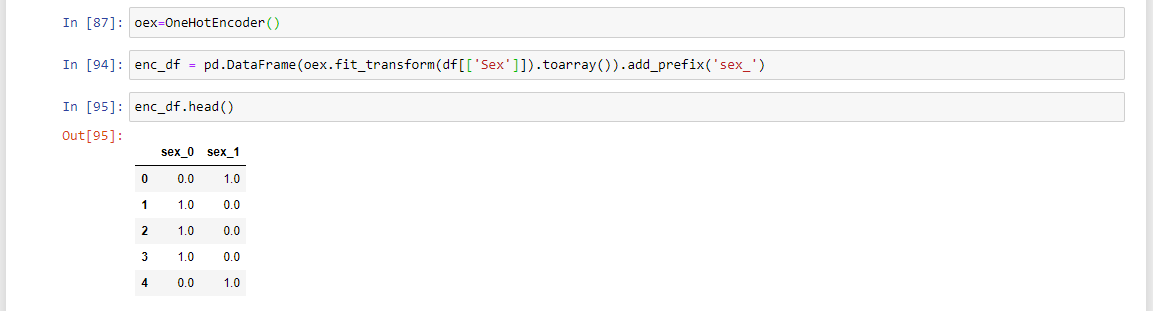
**Encoding:**

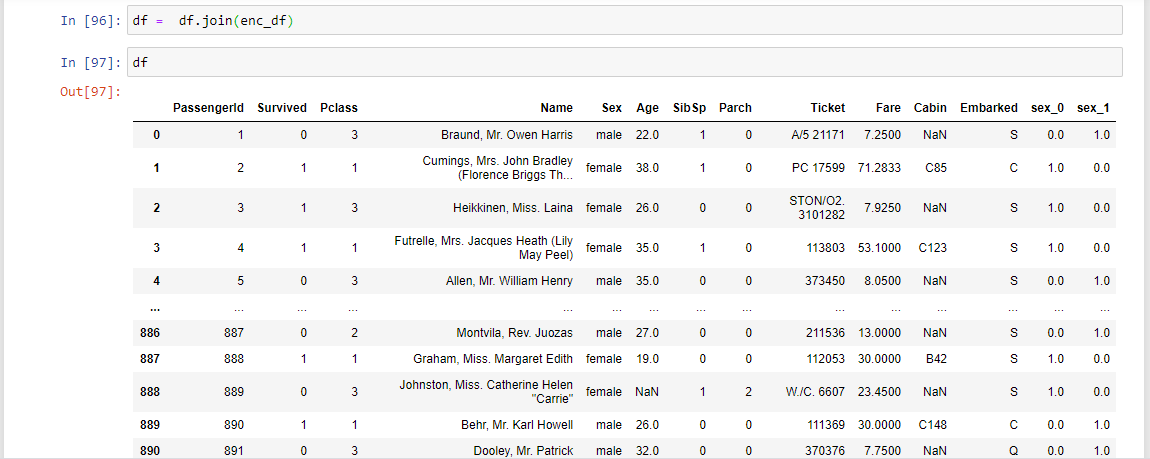
* Most machine learning algorithms and deep neural networks require numerical inputs. This means that if we have categorical data, we must first encode it to numbers in order to build models that actually work.
* Simply put, the goal of categorical encoding is to produce variables we can use to train machine learning models and build predictive features from categories.
* Traditional techniques are
* One-hot encoding
* Ordinal or label encoding

### One hot encoding:

One-hot encoding consists of encoding each categorical variable with a set of Boolean variables that take values of **0** or **1.**This value then serves to indicate if a category is present for each observation.



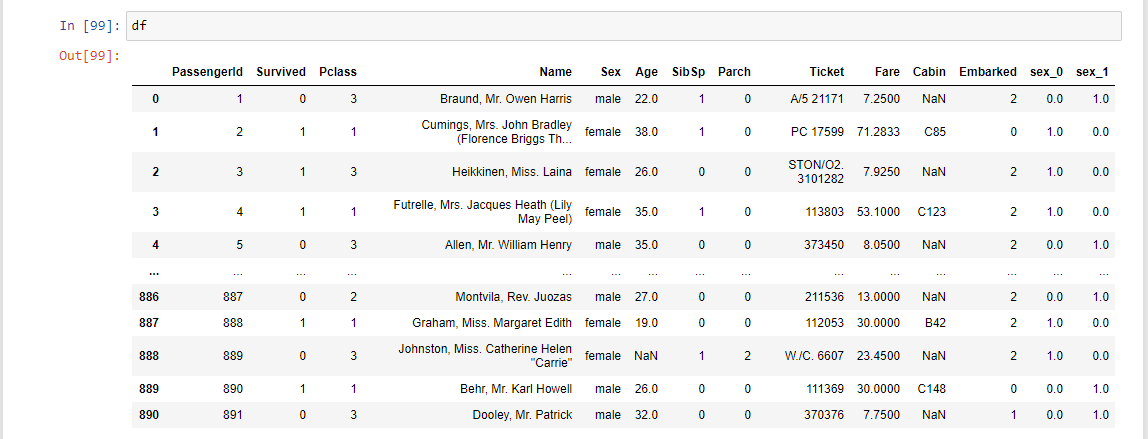




### Ordinal or label encoding:

Integer encoding (also known as label encoding) includes replacing the categories with digits from **1** to **n** (or **0** to **n-1**, depending on the implementation), where **n** is the number of the variable’s distinct categories (the **cardinality**), and these numbers are assigned arbitrarily.





**Feature scaling:**

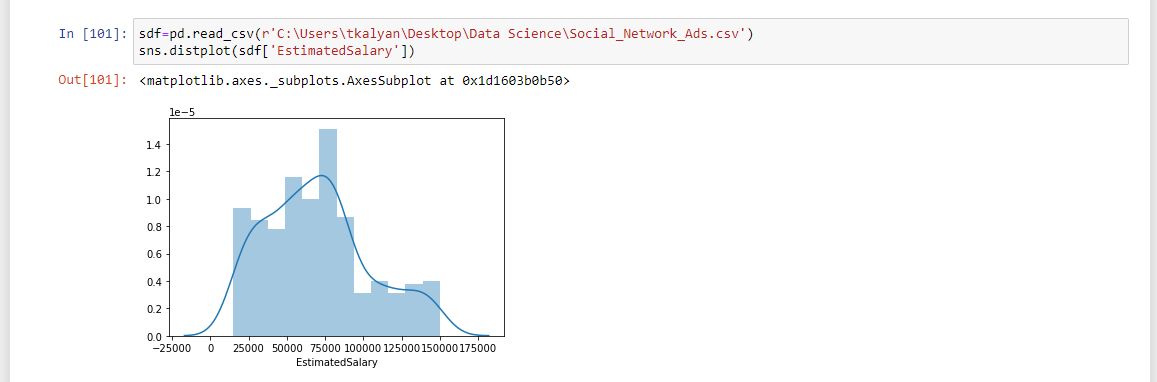
* In many machine learning algorithms, to bring all features in the same standing, we need to do scaling so that one significant number doesn’t impact the model just because of their large magnitude.
* The most common techniques of feature scaling are
* Normalization
* Standardization

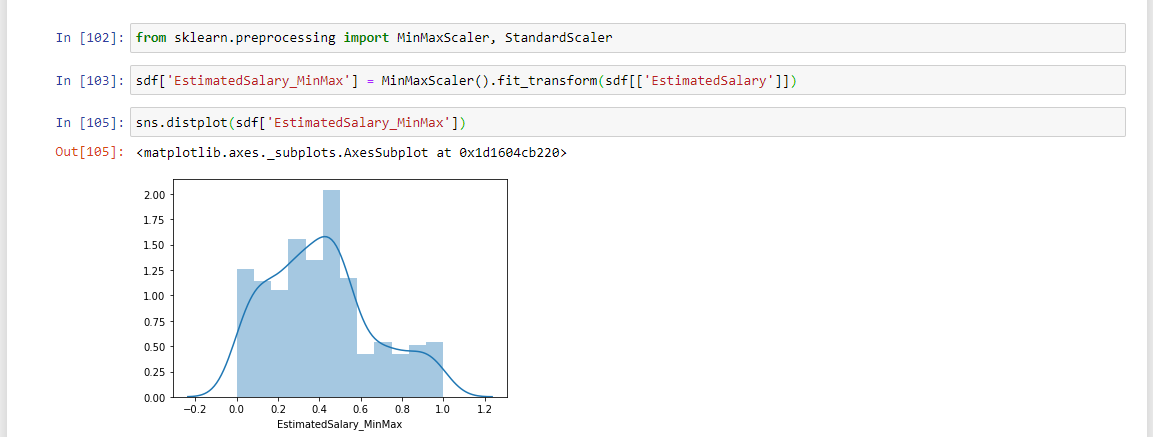
**Normalization:**

* Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.
* Formula for normalization:

x`=(x-xmin)/(xmax-xmin)

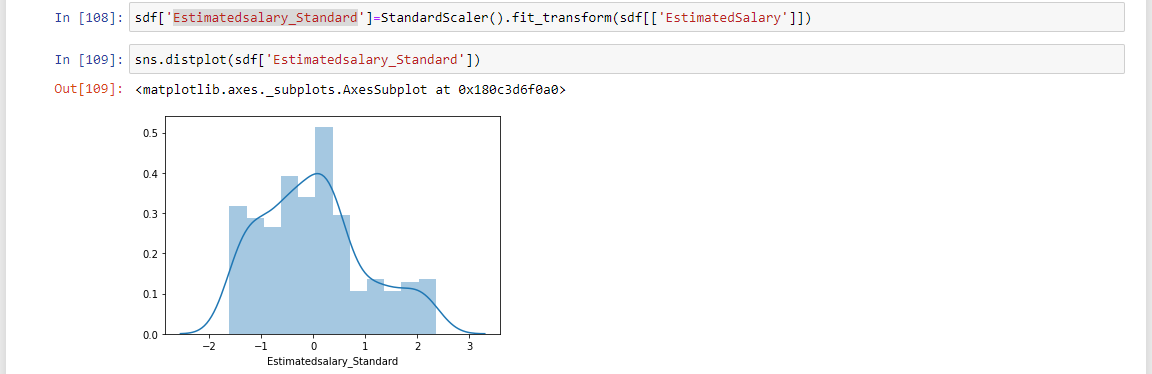
* Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.
* When the value of X is the minimum value in the column, the numerator will be 0, and hence X’ is 0
* On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X’ is 1
* If the value of X is between the minimum and the maximum value, then the value of X’ is between 0 and 1





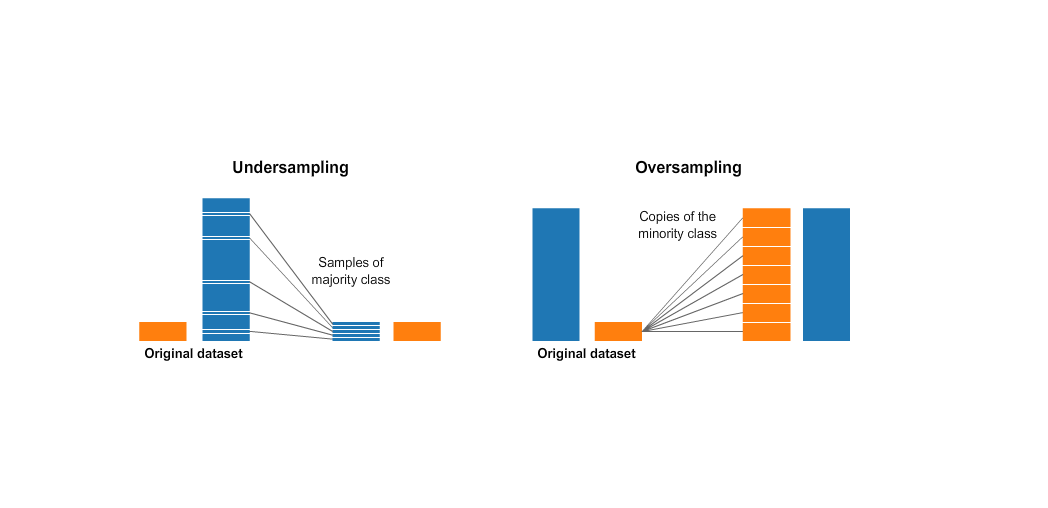
**Standardization:**

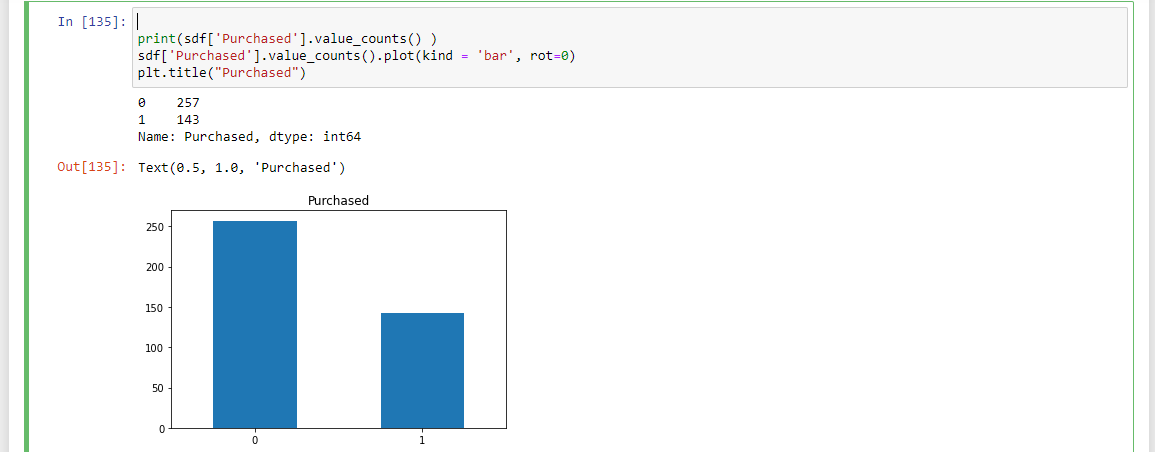
* Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.
* Formula for standardization:
* x`= (x-u)/sigma
* u is the mean of the feature values and sigma is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range**.**



### Handling Imbalance Data:

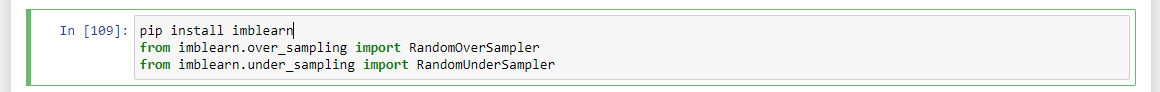
The number of observations belonging to one class is significantly lower than those belonging to other classes.

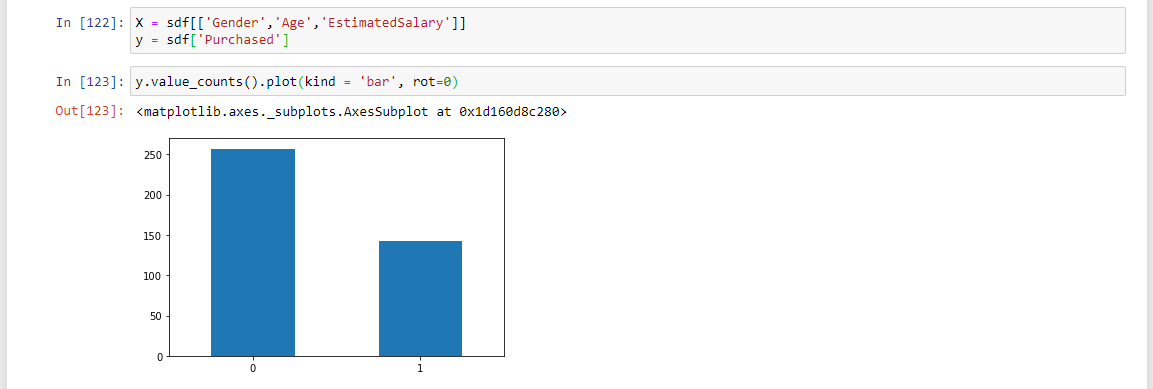
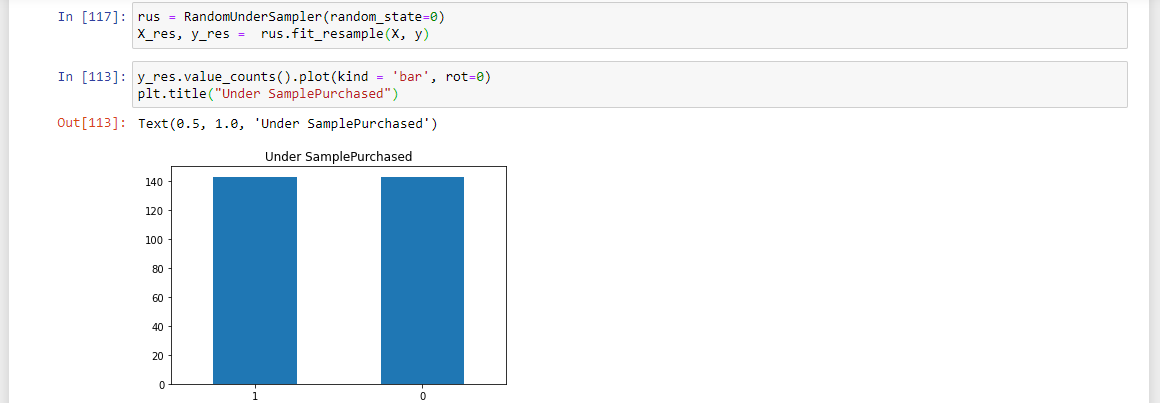




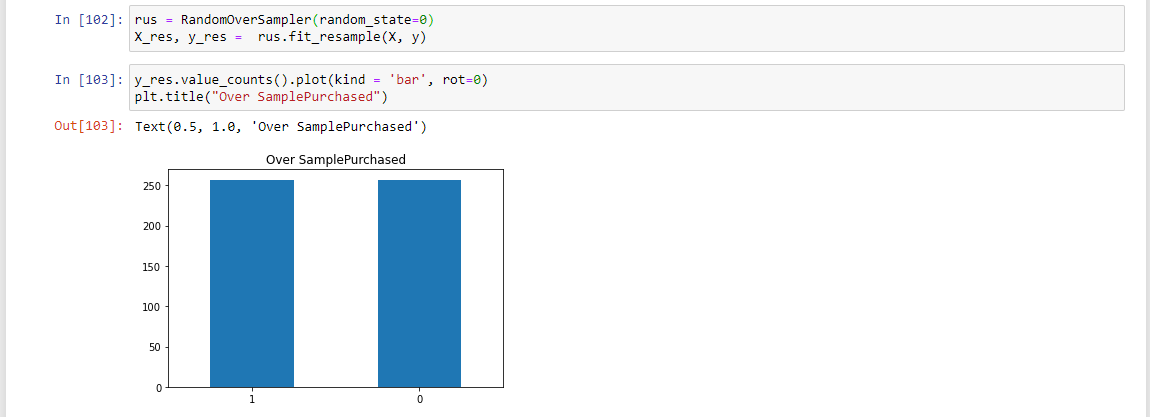
Two Techniques to handle

* **Under Sampling:** This aims to balance class distribution by randomly eliminating majority class examples



* **Over Sampling:** Increase the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.



### 