

# #100daysOfMLCode - Day1

24 August 2018 08:43

Day 1	Machine Learning	#100daysofml
-------	------------------	--------------

Humans have created huge amount of data in the complete human history but the pace of data creation now has increased enormously. After couple of years we will be sitting on humongous pile of data. Machine Learning is the potential to process that amount of data effectively.

## Enviroment Setup

Anaconda -- spyder IDE for Python  
RStudio for R

## Part 1 Data Pre-processing

Figuring out the difference between dependent variables and independent variables.  
In any ML model we use some independent variables to predict a dependent variable.

### Importing the Libraries

Python Libraries : numpy, matplotlib, pandas

R Libraries : no libraries for now, may need later

### Importing the Dataset

Python : `pd.read_csv()`

# #100daysOfMLCode - Day2

25 August 2018 15:44

Day 2	Machine Learning	#100daysOfMLCode
Part 1	Data Pre-processing	
Python		

## Importing the Dataset

```
< pd.read_csv() >
```

Checking data format. Use the preferred number formatting.

## Creating matrix of features/independent variables

```
< dataset.iloc[].values >
```

## Creating dependent variable vector

```
< dataset.iloc[].values >
```

**Missing Data** - filling the missing data with the mean along the axis

```
< Imputer() > from sklearn.preprocessing
```

R
---

## Importing the Dataset

```
< read.csv() >
```

! {indexes in R start from 1 unlike Python in which indexes start from 0}

**Missing Data** - filling the missing data with the mean along the axis

```
< ifelse(is.na()) >
```

Github -> <a href="https://github.com/amitforamit/sdsML">https://github.com/amitforamit/sdsML</a>
---

# #100daysOfMLCode - Day3

26 August 2018 21:05

Day 3	Machine Learning	#100daysOfMLCode
Part 1	Data Pre-processing	

Since Machine Learning models are based on mathematical equations so we need to encode Categorical variables as numbers.

Python
--------

## Encoding Categorical Variable

< [LabelEncoder\(\)](#) > from sklearn.preprocessing

## Dummy Encoding


To avoid any order attribute in case of same level categorical variables, need to create dummy columns < [OneHotEncoder\(\)](#) > from sklearn.preprocessing .It creates separate Columns for every categorical variables and assigning either 1 or 0 denoting the presence or absence of the variable.

R
---

## Encoding Categorical Variable

Use < [factor\(\)](#) > function without any order

Github -> <a href="https://github.com/amitforamit/sdsML">https://github.com/amitforamit/sdsML</a>
---

 curquest

# #100daysOfMLCode - Day4

27 August 2018 08:41

Day 4	Machine Learning	#100daysOfMLCode
-------	------------------	------------------

Part 1	Data Pre-processing
--------	---------------------

- Splitting the dataset into training set and test set. So, that we can test the performance of our Machine learning model which is done on test set.
- Feature Scaling - Bringing variables on the same scale, so that the model is not unfairly dominated by any single variable

Python
--------

Splitting dataset into train and test set

< train\_test\_split() > from sklearn.model\_selection

Feature Scaling

< StandardScaler() > from sklearn.preprocessing

R
---

Splitting dataset into train and test set

Using package < caTools >

< sample.split() > to split the dataset

Feature Scaling

< scale() >

Github -> <a href="https://github.com/amitforamit/sdsML">https://github.com/amitforamit/sdsML</a>
---



# #100daysOfMLCode - Day5

28 August 2018 06:37

Day 5	Machine Learning	#100daysOfMLCode
-------	------------------	------------------

Creating Generic Data Pre-processing Template for future uses

Part 2	Regression
--------	------------

Regression Models are used to predict real values based on independent variables. Various kind of Regression Models - >

Simple Linear, Multiple Linear, Decision Tree, Random Forest etc

Simple Linear Regression
--------------------------

**Linear Regression** - A trend line which best fits the data

**Simple Linear Regression** - It is a trend line which gives minimum value as sum of squares for data points with respect to the line. -- **Ordinary Least Squares**

$$y = b_0 + b_1 * X_1$$

y = Dependent Variable

X<sub>1</sub> = Independent Variable

b<sub>1</sub> = Coefficient (how the unit change in X<sub>1</sub> changes y)

b<sub>0</sub> = Constant



Github -> <a href="https://github.com/amitforamit/sdsML">https://github.com/amitforamit/sdsML</a>
---

