**SCIKIT LEARN**

**How to make sure your data is in an acceptable format:**

**Import Libraries**

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.datasets import load\_iris

**Load the Dataset**

The Iris dataset is one of datasets scikit-learn comes with that do not require the downloading of any file from some external website. The code below loads the iris dataset.

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['species'] = data.target

df.head()

**Arrange Data into Features Matrix and Target Vector**

feature\_names = ['sepal length (cm)',

                 'sepal width (cm)',

                 'petal length (cm)',

                 'petal width (cm)']

**Multiple column features matrix to convert to NumPy Array**

df.loc[:, feature\_names]

**Convert to numpy array**

x = df.loc[:, feature\_names].values

**Make sure NumPy array is two dimensional**

x.shape

**Pandas series to convert to NumPy Array**

df.loc[:, 'species']

y = df.loc[:, 'species'].values

y.shape

scikit-learn expects data in a particular format.

**How to create a linear regression model using Scikit-Learn:**

**Import Libraries**

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

**Load the Dataset**

The dataset that is loaded below is a dataset which is designed to show that Scikit-Learn requires data to be free of missing values. If you don't remove or impute your missing values, you will get an error. The goal of this dataset is to use the feature column x to predict the target column y.

df = pd.read\_csv("data/linear.csv")

df.head()

|  | **x** | **y** |
| --- | --- | --- |
| 0 | 0.000000 | -51.000000 |
| 1 | 25.000000 | -12.000000 |
| 2 | 117.583220 | 134.907414 |
| 3 | 108.922466 | 134.085180 |
| 4 | 69.887445 | NaN |

**Remove Missing or Impute Values**

If you want to build models with your data, null values are (almost) never allowed. It is important to always see how many samples have missing values and for which columns.

df.shape

There are missing values in the y column which is what we will predict

df.isnull().sum()

x 0

y 8

dtype: int64

Remove entire rows from dataframe if they contain any nans in them or 'all'

# this may not be the best strategy for our dataset

df = df.dropna(how = 'any')

There are no more missing values

df.isnull().sum()

x 0

y 0

dtype: int64

**Arrange Data into Features Matrix and Target Vector**

 Convert x column to numpy array

X = df.loc[:, ['x']].values

Features Matrix needs to be at 2 dimensional

X.shape

(94, 1)

y = df.loc[:, 'y'].values

y.shape

(94,)

**What is `train\_test\_split`**

 Split the dataset into two pieces: a **training set** and a **testing set**. Typically, about 75% of the data goes to your training set and 25% goes to your test set.

2. Train the model on the **training set**.

3. Test the model on the **testing set** and evaluate the performance .

**Import Libraries**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_boston

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

**Load the Dataset**

housing = fetch\_california\_housing()

df = pd.DataFrame(housing.data, columns=housing.feature\_names)

df['target'] = housing.target

df.head()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=3)

**Linear Regression Model**

# Make a linear regression instance

reg = LinearRegression(fit\_intercept=True)

# Train the model on the training set.

reg.fit(X\_train, y\_train)

**Measuring Model Performance**

By measuring model performance on the test set, you can estimate how well your model is likely to perform on new data (out-of-sample data)

**Test the model on the testing set and evaluate the performance**

score = reg.score(X\_test, y\_test)

print(score)