**Forecasting Sales of stores Using LSTM**

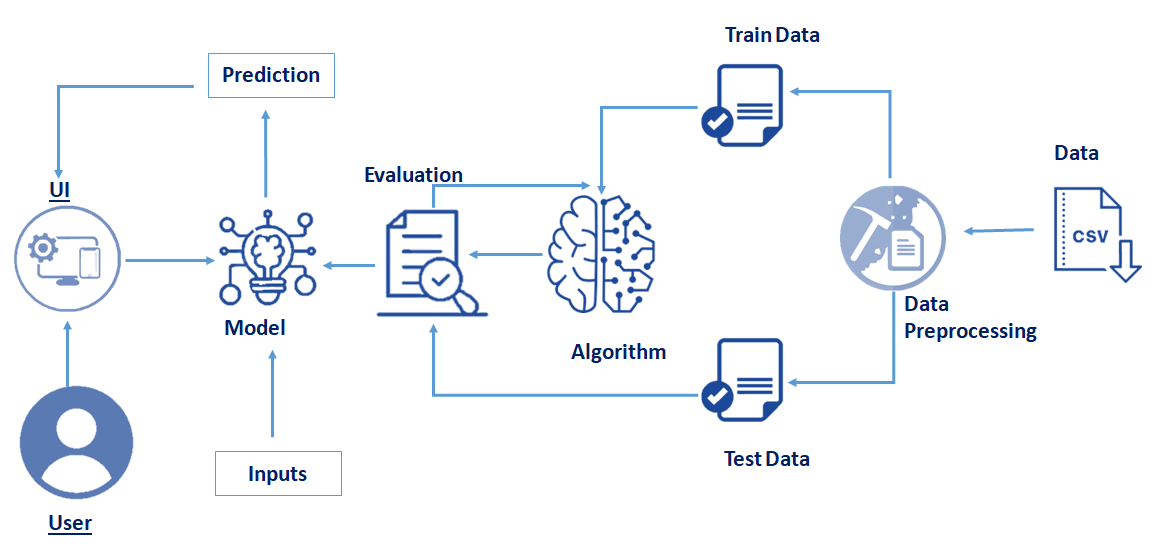
**Introduction to project:**

Sales forecasting is an essential task for the management of a store. Being able to estimate the quantity of products that a retail store is going to sell in the future will allow the owners of these shops to prepare the inventory that they will need.

In this project we are building a system which analyses the previous trends of sales which includes sales on various days and predict the future sales. The goal of this project is to forecast the sales of store by using time series analysis. Here time series analysis algorithms such as RNN (Recurrent Neural Network) & LSTM (Long Term Short Memory) are used to analyse the past trends of sales of store. Create and deploy flask based web Application and integrate AI model to it.

The objective of the project is to build a web application where the user gives the last ten days sales values and get the prediction for the 11 th day which is showcased on UI.

**Technical Architecture:**

****

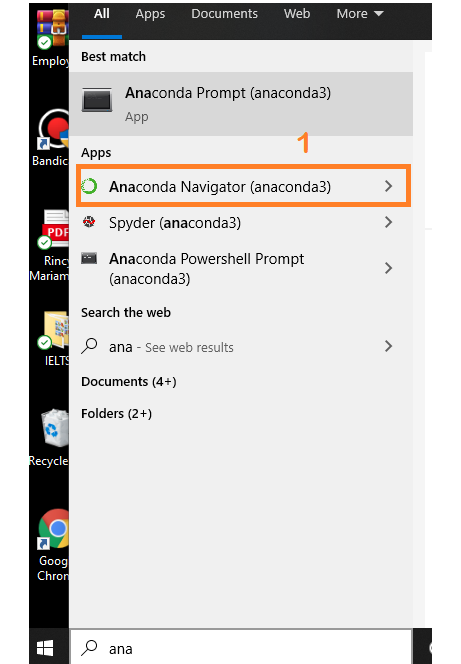
**Pre requisites:**

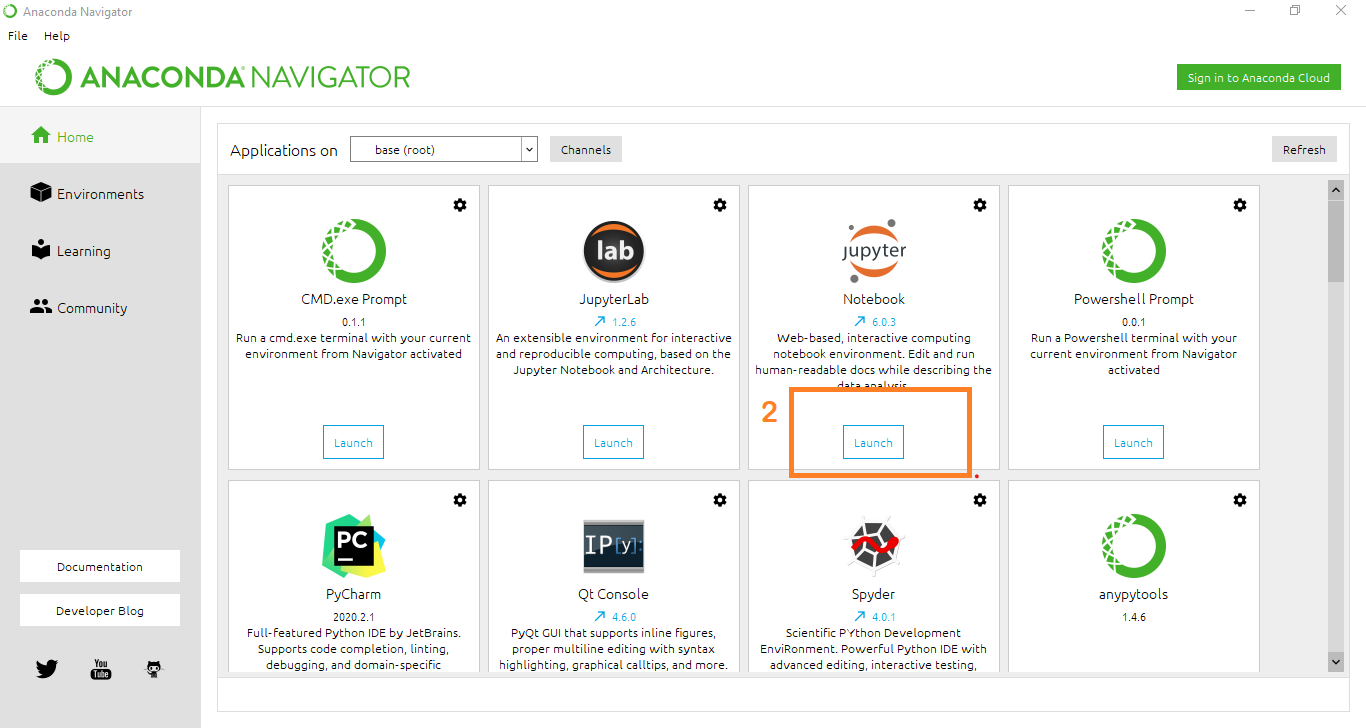
**To complete this project, you must require following software’s, concepts and packages**

* **Anaconda navigator:**
  + Refer to the link below to download anaconda navigator
  + **Link : https://www.youtube.com/watch?v=5mDYijMfSzs**
* **Python packages:**
  + Open anaconda prompt.
  + Type “pip install jupyter notebook” and click enter.
  + Type “pip install spyder” and click enter.
  + Type “pip install numpy” and click enter.
  + Type “pip install pandas” and click enter.
  + Type “pip install matplotlib” and click enter.
  + Type “pip install seaborn” and click enter.
  + Type “pip install sklearn” and click enter.
  + Type “pip install tensorflow==2.3.0” and click enter.
  + Type “pip install keras==2.4.3” and click enter.
  + Type “pip install Flask” and click enter.

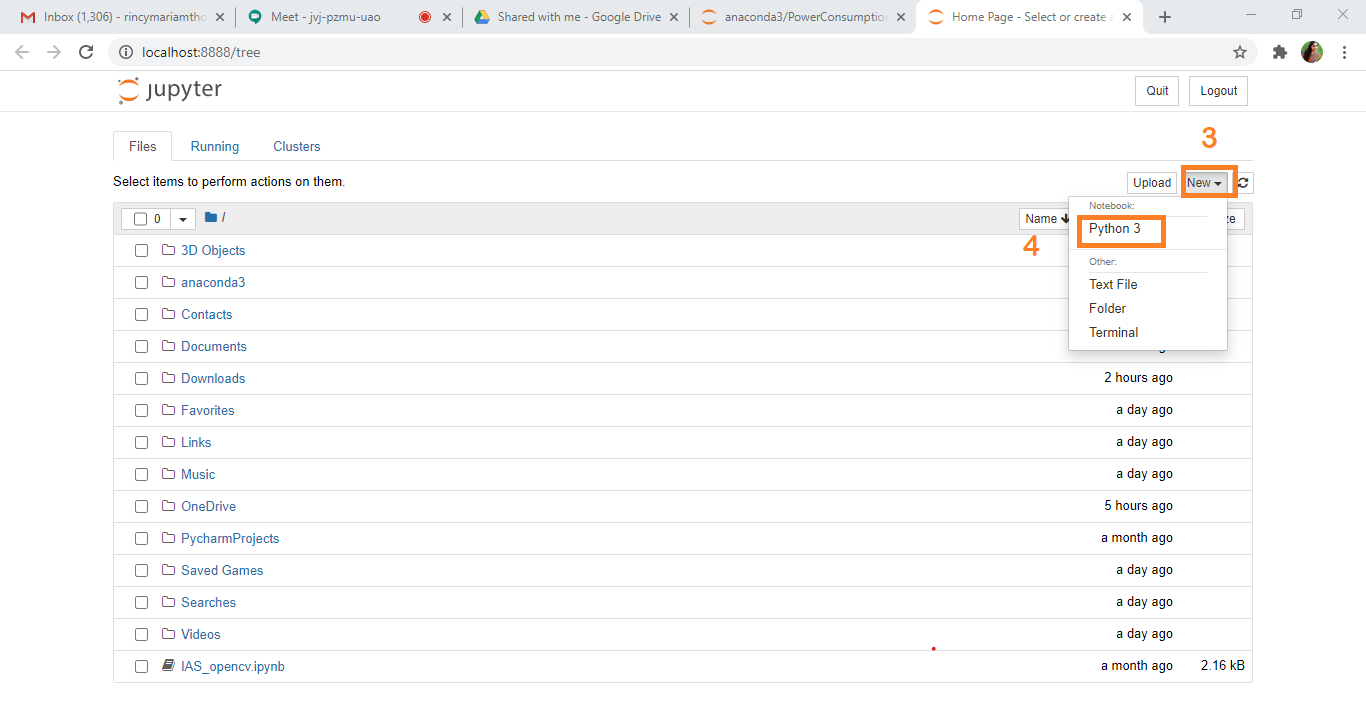
the above steps allow you to install keras and tensorflow in the anaconda environment

* **Deep Learning Concepts** 
  + **LSTM:** <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
  + **TimeSeriesForecasting:**https://analyticsindiamag.com/an-introductory-guide-to-time-series-forecasting/
  + **Flask Basics** : <https://www.youtube.com/watch?v=lj4I_CvBnt0>
* **Launch Jupyter**
  + Search for Anaconda Navigator and open Launch Jupyter notebook.





* Then you will be able to see that the jupyter notebook runs on local host:8888.
* To Create a new file Go to New 🡪Python3. The file in jupyter notebook is saved with .ipynb extension.



**Project Objectives:**

By the end of this project you will:

* know fundamental concepts and techniques of time series forecasting and LSTM
* gain a broad understanding of time series data.
* Knowhow to split the data for time series forecasting.
* know how to build a web application using Flask framework.

**Project Flow:**

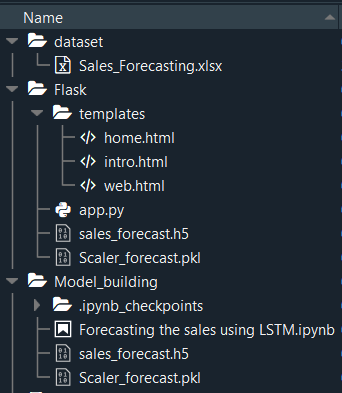
* User interacts with the UI (User Interface) to enter the data of the previous 10 days to get the future prediction
* Entered data is analyzed by the model which is integrated
* Once the model analyses the input the prediction of the next day is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

* Data Collection.
  + Collect the dataset or Create the dataset
* Data Preprocessing.
  + Import the Libraries.
  + Importing the dataset.
  + Analyse the data
  + Taking care of Missing Data
  + Feature Scaling
  + Data Visualization
  + Splitting Data into Train and Test.
  + Creating a datasets with a sliding window.
* Model Building
  + Import the model building Libraries
  + Initializing the model
  + Adding LSTM Layers
  + Adding Output Layer
  + Configure the Learning Process
  + Training the model
  + Model Evaluation
  + Save the Model
  + Test the Model
* Application Building
  + Create an HTML file
  + Build Python Code

**Project Structure:**

Create a Project folder which contains files as shown below



* We are building a Flask Application which needs HTML pages stored in the templates folder and a python script app.py for serverside scripting
* app.py - contains the actual python code that will import the app and start the development server.
* Forecasting the sales using LSTM.ipynb - This is where you define models for your application.
* sales\_forecast.h5 - This is our model weights file
* Scaler\_forecast.pkl-This is our scalar file
* templates - This is where you store your html templates i.e. home.html, web.html,intro.html
* requirements.txt - This is where you store your package dependancies.

**Milestone 1: Data Collection**

ML depends heavily on data, without data, a machine can't learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

**Activity 1: Download dataset /create dataset**:

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

The dataset used for this project was obtained from Kaggle. Please refer to the link given below to download the data set and to know about the dataset

* + **Link**: <https://github.com/Guided-Projects/Forecasting-Sales-of-a-Store-using-LSTM/tree/main/dataset>
  + Our Sales dataset contains

1. HQ - This attribute represents HeadQuarters of the company.

2. Country - This attribute represents the country of the outlet.

3. State\_of\_outlet - This attribute represents the state of the outlet.

4. City\_of\_outlet - This attribute represents the city of the outlet.

5. Month - This attribute represents month(ranging from 1 to 12 )

6. Day - This attribute represents day of the month(ranging from 1 to 31)

7. Year - This attribute represents year(ranging from 2005 to 2016)

8. Total\_Sales - Target variable (total sales on particular day).It contains total sales from 19886 to 2018.

**Milestone 2: Data Preprocessing**

Data Pre-processing includes the following main tasks

* + Import the Libraries.
  + Importing the dataset.
  + Analyse the data
  + Taking care of Missing Data
  + Data Visualization
  + Feature Scaling
  + Splitting Data into Train and Test.
  + Creating a datasets with sliding window.

**Activity 1: Import the Libraries**

The first step is usually importing the libraries that will be needed in the program.

The required libraries to be imported to Python script are:

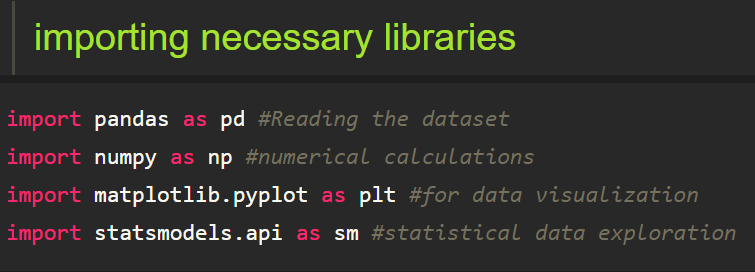
**Numpy:**

It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

**Pandas**- It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

**Matplotlib**- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python

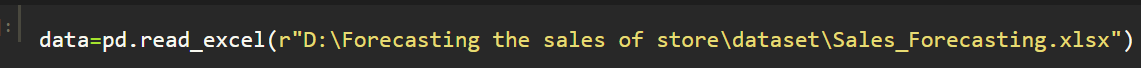
**statsmodels**- statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.



Note: It’s conventional to refer to alias. When you add the alias name at the end of your import statement, your Jupyter Notebook understands that from this point on every time you type alias name, you are actually referring to the particular library.

**Activity 2**: **Importing the dataset**

* You might have your data in .csv files, .excel files
* Let’s load the excel data file into pandas using the **read\_excel() function.** We will need to locate the directory of the excel file at first (it’s more efficient to keep the dataset in the same directory as your program).



* If your dataset is in some other location, Then

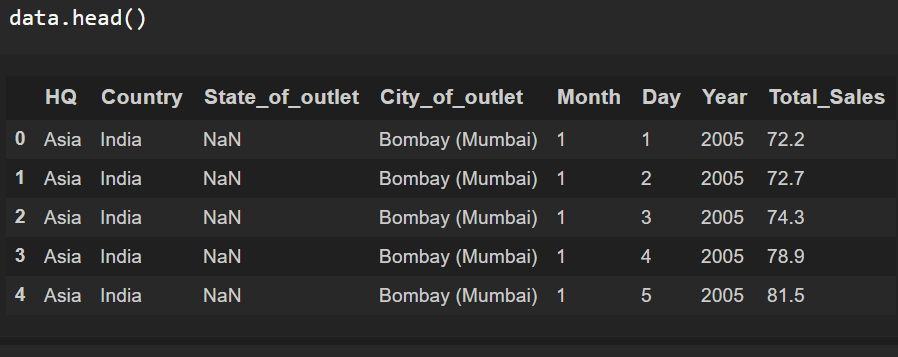
Data=pd.read\_excel(r”File\_location”)

**Note:**r stands for "raw" and will cause backslashes in the string to be interpreted as actual backslashes rather than special characters.

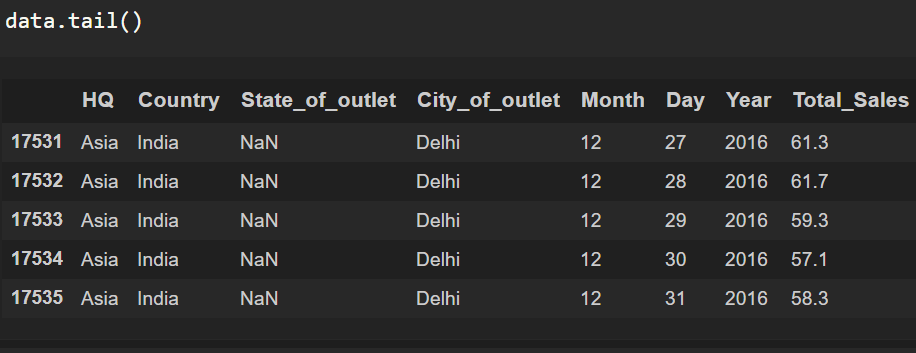
* If the dataset in same directory of your program, you can directly read it, without giving raw as r.

**Activity 3** : **Analyse the data**

* head() method is used to return top n (5 by default) rows of a DataFrame or series.



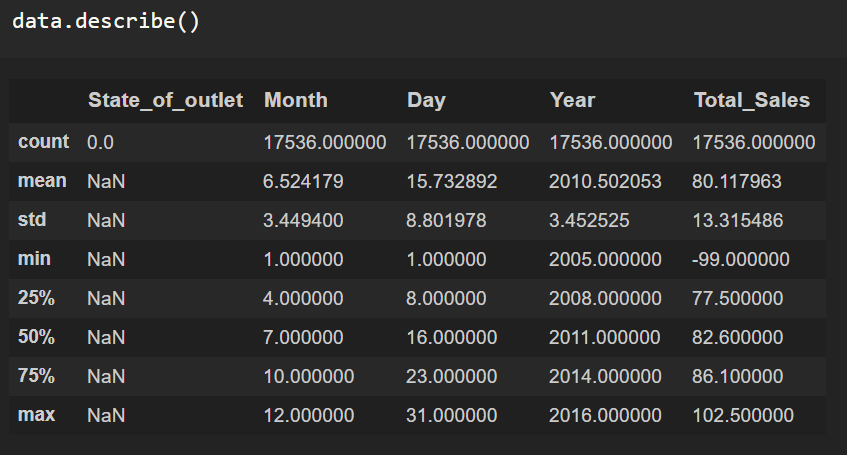
* tail() method is used to return bottom n (5 by default) rows of a DataFrame or series.



* describe() method computes a summary of statistics like count, mean, standard deviation, min, max, and quartile values.

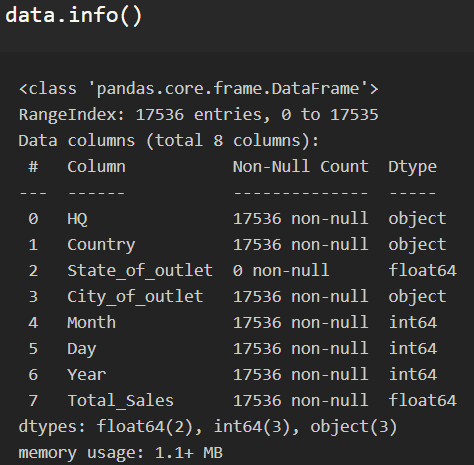


The output is as shown below



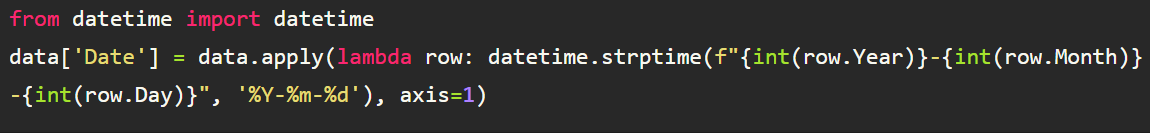
From the data we infer that there are 17536 records

* info() gives information about the data

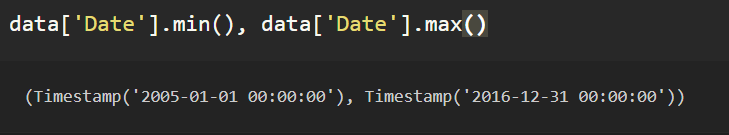


**Converting month, day, year into Date format**

As we have month, day ,year in separate columns,let us convert them into date format by using datetime module



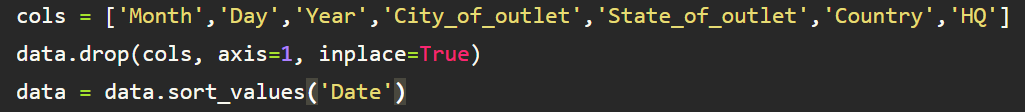
Let us take a look at how many years of data is present in our dataset.



We notice that we have Sales data from 2015 to 2016

**Dropping the columns:**

Let us drop the columns 'Month','Day','Year','City\_of\_outlet','State\_of\_outlet','Country','HQ'which are not required for the purpose for analysis and sorting the sales values using date column which we have created in the previous step.



Axis=1 indicates that drop the columns

The 'inplace=True' argument stands for the data frame has to make changes permanent

**Activity 4**: **Taking care of Missing Data**

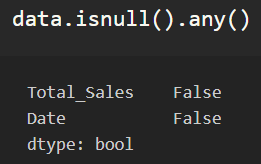
1. After loading the dataset, it is important to check the complete information of such as null values in a column or a row

2.Check whether any null values are there or not. if it is present then the following can be done,

a.Imputing data using Imputation method in sklearn

b.Filling NaN values with mean, median, and mode using fillna() method.

c.Delete the records



We can see that there are null values in the Total\_Sales Column.

Let us check how many numbers of null records present in the Total\_Sales column using sum() function.

Let us consider Total\_Sales column in the dataset and groupby the date value to get the sales sum of each order date.

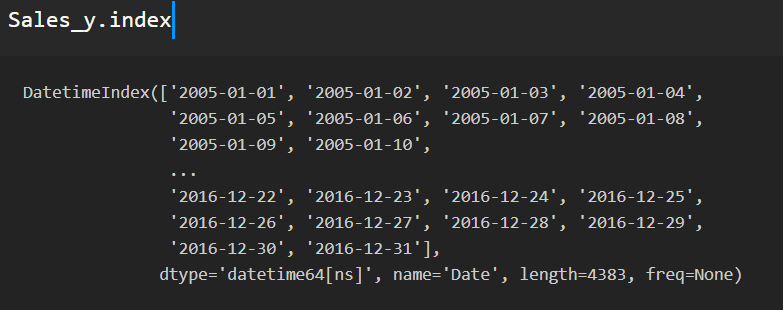


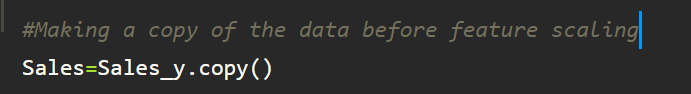
**reset\_index()** is a method to reset index of a Data Frame. reset\_index() method sets a list of integer ranging from 0 to length of data as index.

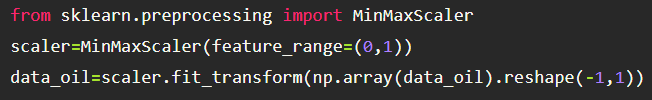
Let us set the index for Date



Let us check the index







**Activity 5** : **Data Visualization**

**Link**: <https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7>

* Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data.
* Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn’t visualized and understood properly.
* To visualize the dataset we need libraries called Matplotlib and Seaborn.
* The Matplotlib library is a Python 2D plotting library which allows you to generate plots, scatter plots, histograms, bar charts etc.

Let’s visualize our data using Matplotlib and searborn library.

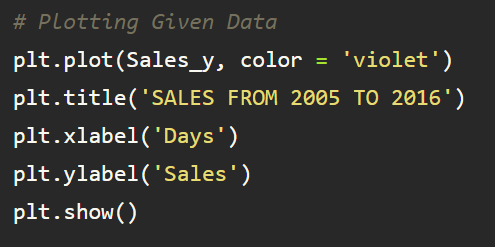
Before diving into the code, let's look at some of the basic properties we will be using when plotting.

xlabel: Set the label for the x-axis.

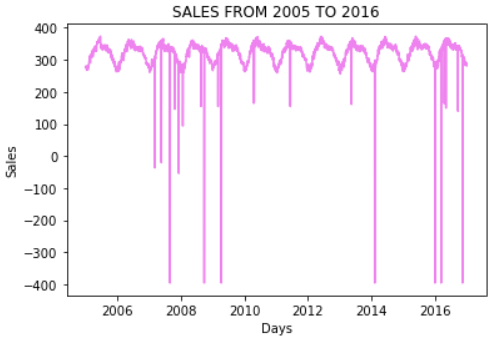
ylabel: Set the label for the y-axis.

title: Set a title for the axes.

Legend: Place a legend on the axes.

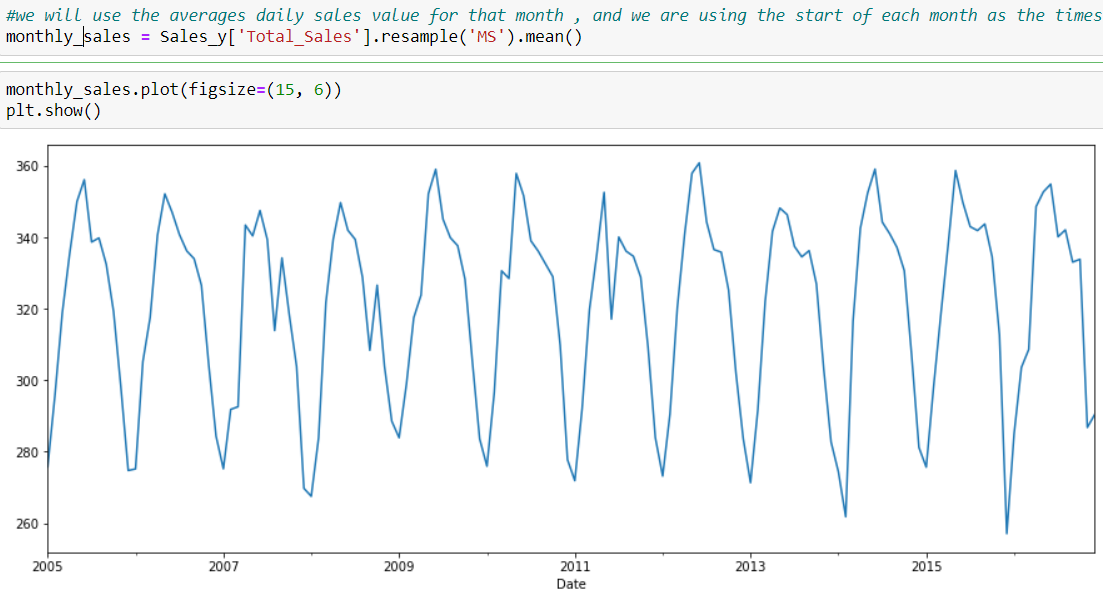


Let us check the sales from 2005 to 2016



From the graph we infer that, the Sales are increasing day by day but ,we cannot say that it is completely increasing as there is a significant drop in some years.

we will use the averages daily sales value for that month , and we are using the start of each month as the timestamp.



We notice that the time-series has seasonality pattern, such as sales are always low at the beginning of the year and high at the end of the year. There is always an upward trend within any single year with a couple of low months in the mid of the year.

Time series analysis provides a body of techniques to better understand a dataset.

Perhaps the most useful of these is the decomposition of a time series into 4 constituent parts:

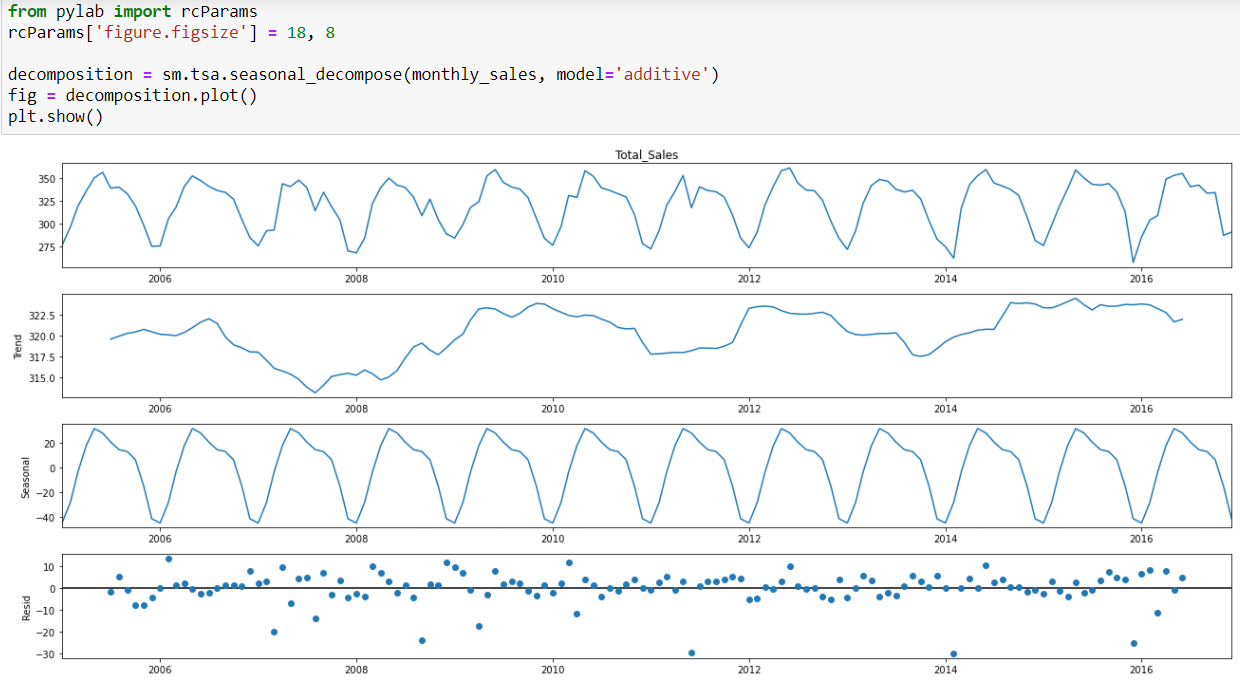
1.Level. The baseline value for the series if it were a straight line.

2.Trend. The optional and often linear increasing or decreasing behavior of the series over time.

3.Seasonality. The optional repeating patterns or cycles of behavior over time.

4.Noise. The optional variability in the observations that cannot be explained by the model.

Let us visualize our data using the above method that allows us to decompose our time series data

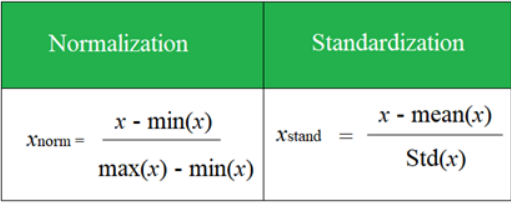


From the above plot, we notice that the sales is following a seasonality

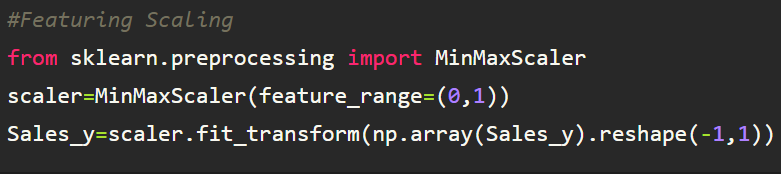
**Activity 6**: **Feature Scaling**

Feature scaling is a method used to normalize the range of independent variables or features of data.

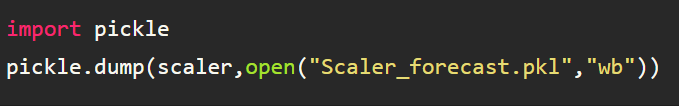
The next step is to scale the total sales prices between (0, 1) to avoid intensive computation. Common methods include **Standardization**and **Normalization .**



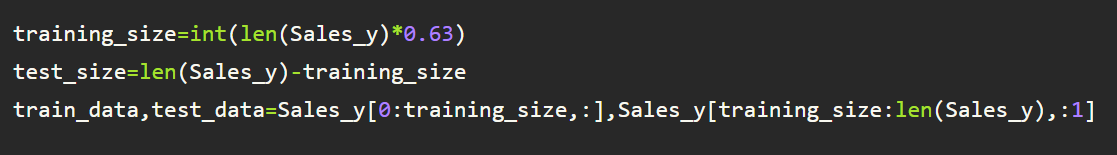
LSTM are sensitive to the scale of the data. so we apply MinMax scaler



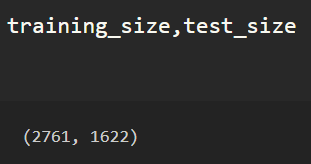
We have to save the scaler file to reuse it in building our application by using pickle module.



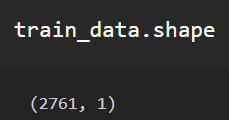
**Activity 7** : **Splitting Data into Train and Test**

* When you are working on a model and you want to train it, you have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.
* But the question is, how do you split the data?
* For time-series data, the sequence of values is important. A simple method that we can use is to split the ordered dataset into train and test datasets. The code below calculates the index of the split point and separates the data into the training datasets with 63% of the observations that we can use to train our model, leaving the remaining 27% for testing the model.x` 

The size of train and test data after splitting



Shape of the train data

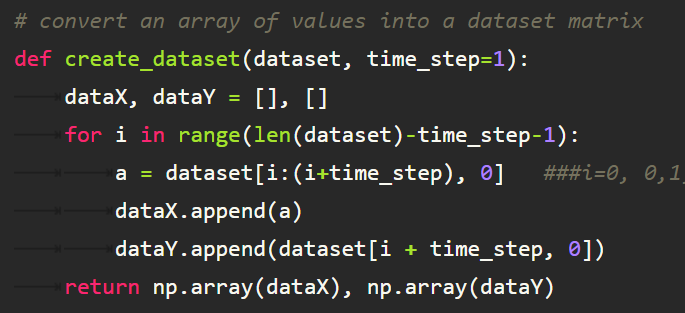


**Activity 8** : **Creating a datasets with sliding window.**

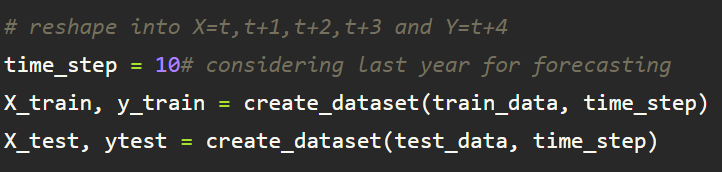
A special data structure is needed to cover n-time stamps, based on which LSTM will predict the n +1 tt price. Here the number of past timestamps is set to 10.

The function takes two arguments, the dataset which is a NumPy array that we want to convert into a dataset and the time\_step which is the number of previous time steps to use as input variables to predict the next time period, in this case, defaulted to 1.

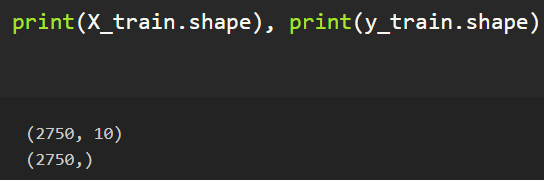
This default will create a dataset where X is the total\_sales at a given time (t) and Y is the sales at the next time (t + 1).



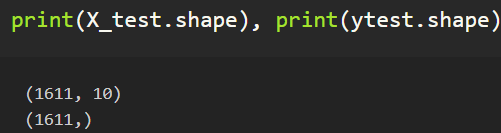
We are applying the function on training data and test data. Hence we get X\_train,y\_train and X\_test,ytest



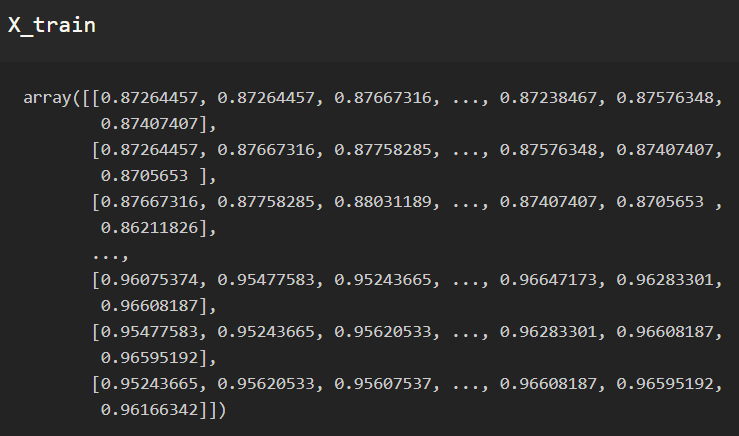
The shape of training data



The shape of test data

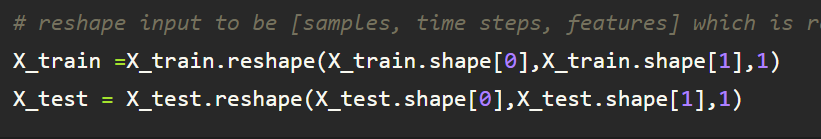


The data of X\_train is as follows



For LSTM model , it is necessary to reshape the X\_train and X\_test into 3 dimensional array before building the model.

# reshape input to be [samples, time steps, features] which is required for LSTM



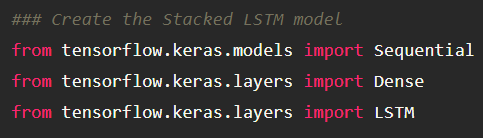
**Milestone 3: Model Building**

Model Building Includes:

* + Import the model building Libraries
  + Initializing the model
  + Adding LSTM Layers
  + Adding Output Layer
  + Configure the Learning Process
  + Training the model
  + Model Evaluation
  + Save the Model
  + Test the Model

**Activity 1 : Importing the Model Building Libraries**

Importing the necessary libraries



**Activity 2 : Initializing the model**

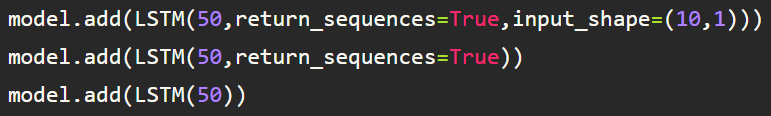
Sequential model is a linear stack of layers.You can create a Sequential model by passing a list of layer instances to the constructor: from keras. models import Sequential from keras as follows.



**Activity 3 : Adding LSTM Layers**

* Note for the LSTM layer, units is the number of LSTM neurons in the layer. 50 neurons will give the model high dimensionality, enough to capture the upwards and downward trends.
* return\_sequences is True as we need to add another LSTM layer after the current one. input\_shape corresponds to the number of time stamps and the number of indicators.

Following the above same method, add 2nd, 3rd LSTM layer



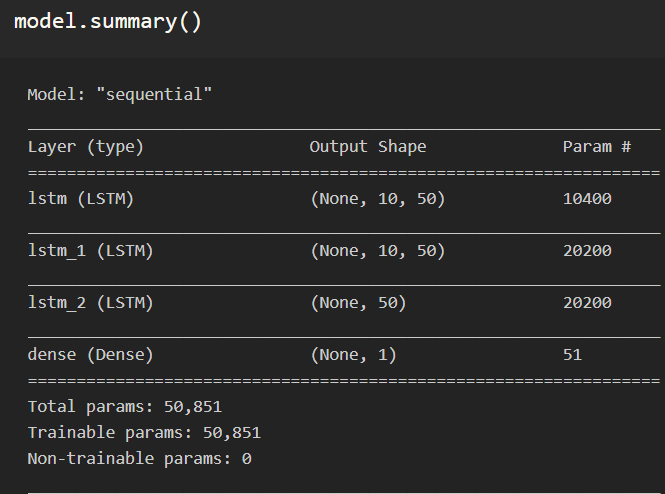
**Activity 4: Adding output Layers**

Dense layer is deeply connected neural network layer. It is most common and frequently used layer.

Finally, add the output layer. The output dimension is 1 since we are predicting 1 price each time.



Understanding the model is very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.



**Activity 5 : Configure The Learning Process**

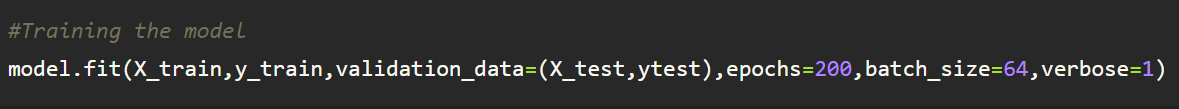
* The compilation is the final step in creating a model. Once the compilation is done, we can move on to training phase.Loss function is used to find error or deviation in the learning process. Keras requires loss function during model compilation process.
* Optimization is an important process which optimize the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
* Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in training process



**Activity 6: Train The model**

Now ,let us train our model

RNN weights are updated every 64 stock prices with a batch size of 64. Try more batches and epochs if the loss of the model is not converging.



**Arguments:**

* Epochs : an integer and number of epochs we want to train our model for.
* validation\_data can be either:

- an inputs and targets list

- a generator

- an inputs, targets, and sample\_weights list which can be used to evaluate

the loss and metrics for any model after any epoch has ended.

**Activity 7**: **Model Evaluation**

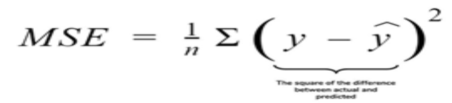
Finally, we need to check to see how well our model is performing on the test data.

**Regression Evaluation Metrics:**

### Mean Squared Error (MSE):

MSE or Mean Squared Error is one of the most preferred metrics for regression problems. It is simply the average of the squared difference between the target value and the value predicted by the regression model.

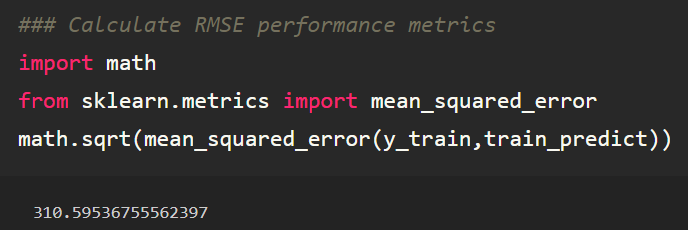
As it squares the differences, it penalizes even a small error which leads to over-estimation of how bad the model is. It is preferred more than other metrics because it is differentiable and hence can be optimized better.



### 2.RMSE:Root Mean Square Error:

### RMSE is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

### 

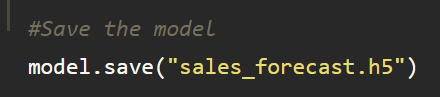


**Activity 8: Save the Model**

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

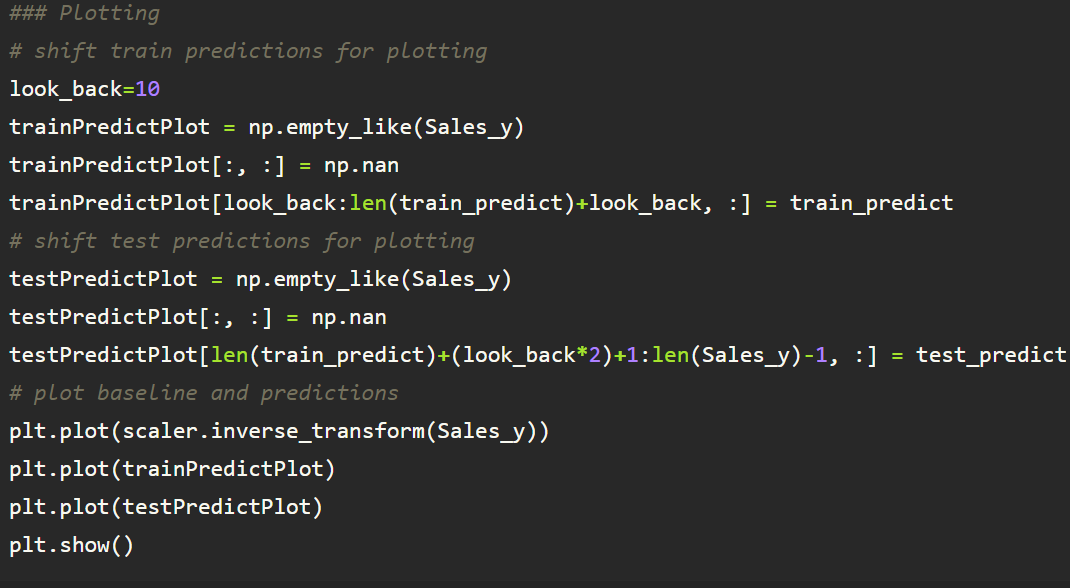


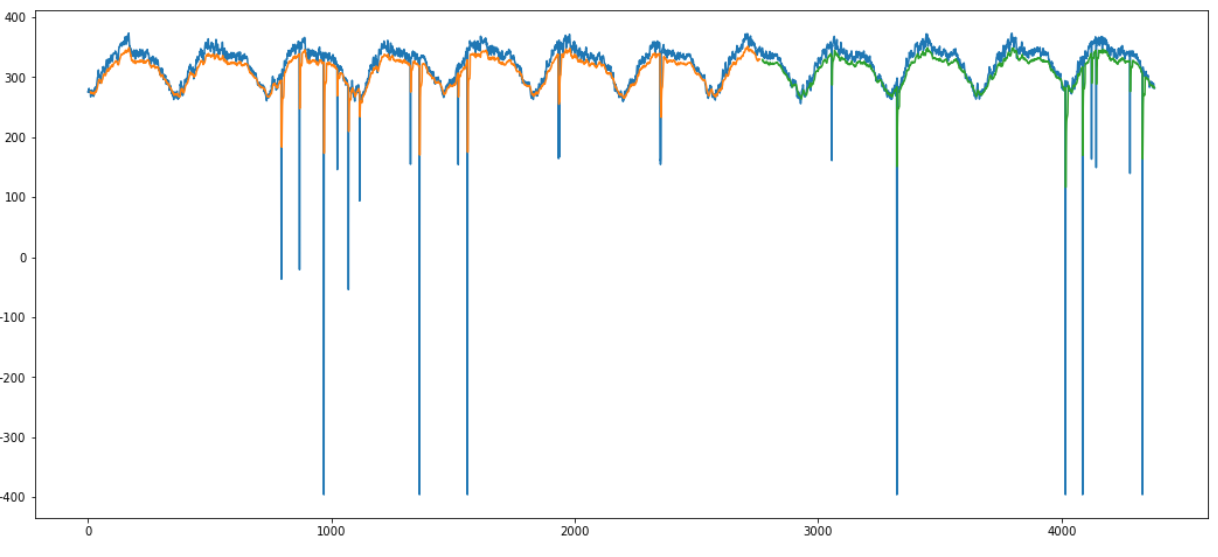


**Activity 9: Test the Model**

Finally, we can generate predictions using the model for both the train and test to visualize the model.

We must shift the predictions so that they aline on the x-axis with the original dataset. Once prepared, the data is plotted, showing the original dataset in blue, the predictions for the train dataset in green the predictions on the unseen test dataset in orange.

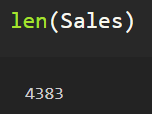




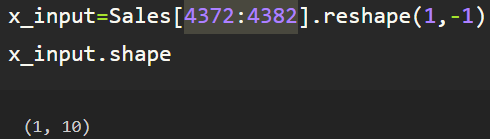
**Prediction for next 10 days:**

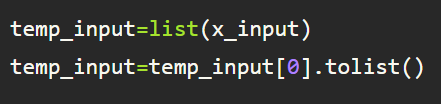
Now let us predict the sales for next 10 days.

As the length of the test data is 4383, We are taking previous 10 days input i.e., from index 4372 to 4382 predict 4383 th day output

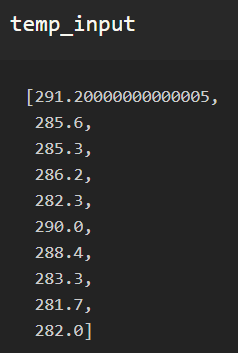


Create the input and reshape it and convert it into list





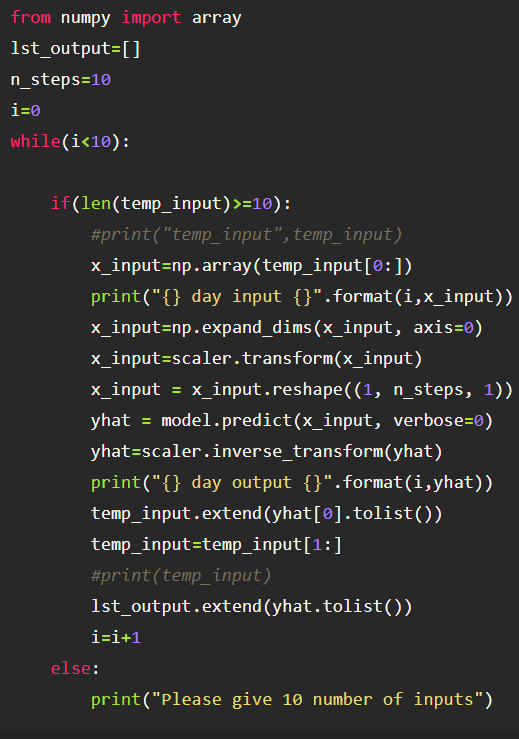
We can see temp\_input contains last 10 days price list



For predicting next 10 days sales we consider n\_steps=10

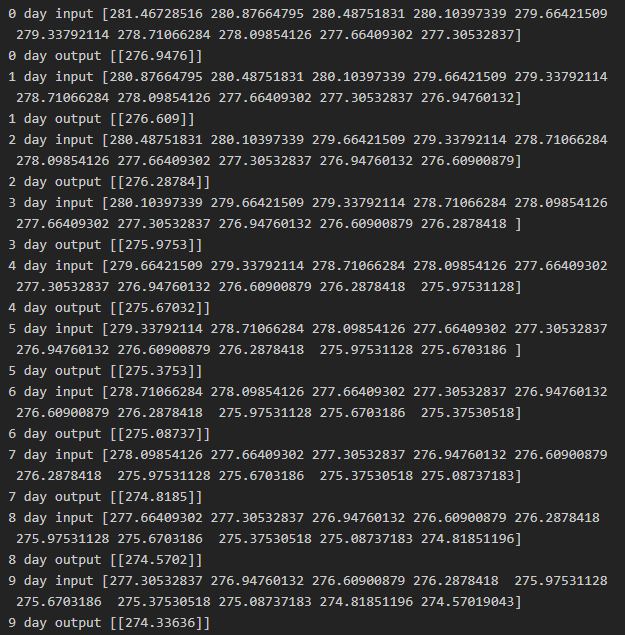
We create the input for prediction, index starting from the date 10 days before the first date in the test dataset. Then, reshape the inputs to have only 1 column and predict using model\_predict predefined function.

This can be done using the below code



The output is as shown follows

We can infer that it is taking 10 inputs and predicting the day 11 th output.



**Milestone 4: Application Building**

**Application building Involves following steps**

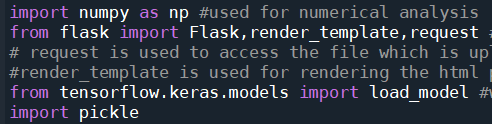
* + Create an HTML file
  + Build Python Code
  + **Link**: <https://thesmartbridge.com/documents/spsaimldocs/FlaskML.pdf>

**Activity 1 : Create an HTML File**

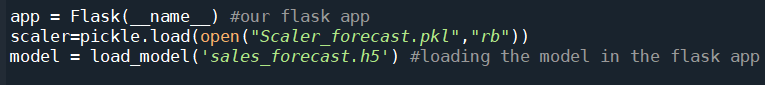
* We use HTML to create the front end part of the web page.
* Here, we created 2 html pages- index.html, web.html.
* index.html displays the home page.
* web.html accepts the values from the input and displays the prediction.
* For more information regarding HTML refer the link below.
  + **Link:** [**https://www.w3schools.com/bootstrap/bootstrap\_forms\_inputs.asp**](https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp)
  + **Link:**<https://www.w3schools.com/css/>

**Activity 2 : Build python code**

* Let us build flask file ‘app.py’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.
* App starts running when “\_\_name\_\_” constructor is called in main.
* render\_template is used to return html file.
* “GET” method is used to take input from the user.
* “POST” method is used to display the output to the user.
* Importing Libraries

****

* + Initialising the flask application and loading the scaler and model file created when building our model

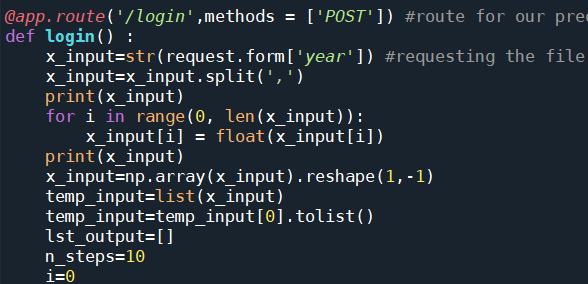


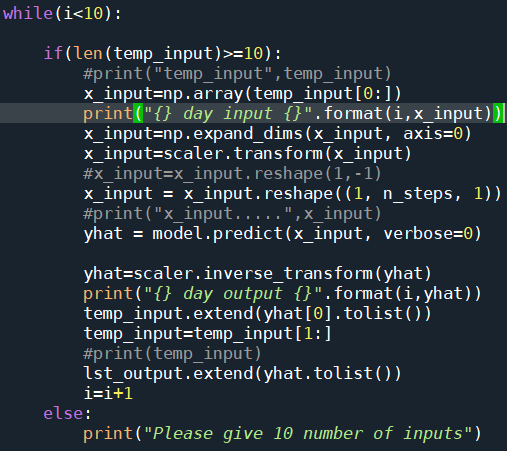
* + Routing to the html Page



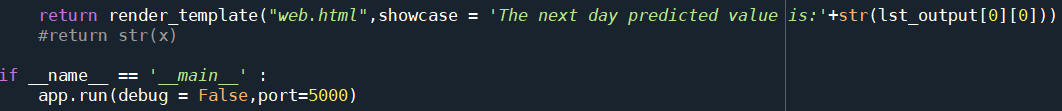
* For predicting next day’s sales of a store we consider n\_steps=10.

We take the input for prediction, index starting from the date 10 days before the first date in the test dataset. Then, reshape the inputs to have only 1 column and predict using model\_predict predefined function.





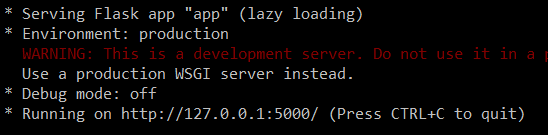
Rendering the template to showcase on UI



**Activity 3: Run The app in local browser**

* + Open anaconda prompt from the start menu
  + Navigate to the folder where your python script is.
  + Now type “python app.py” command
  + Navigate to the localhost where you can view your web page

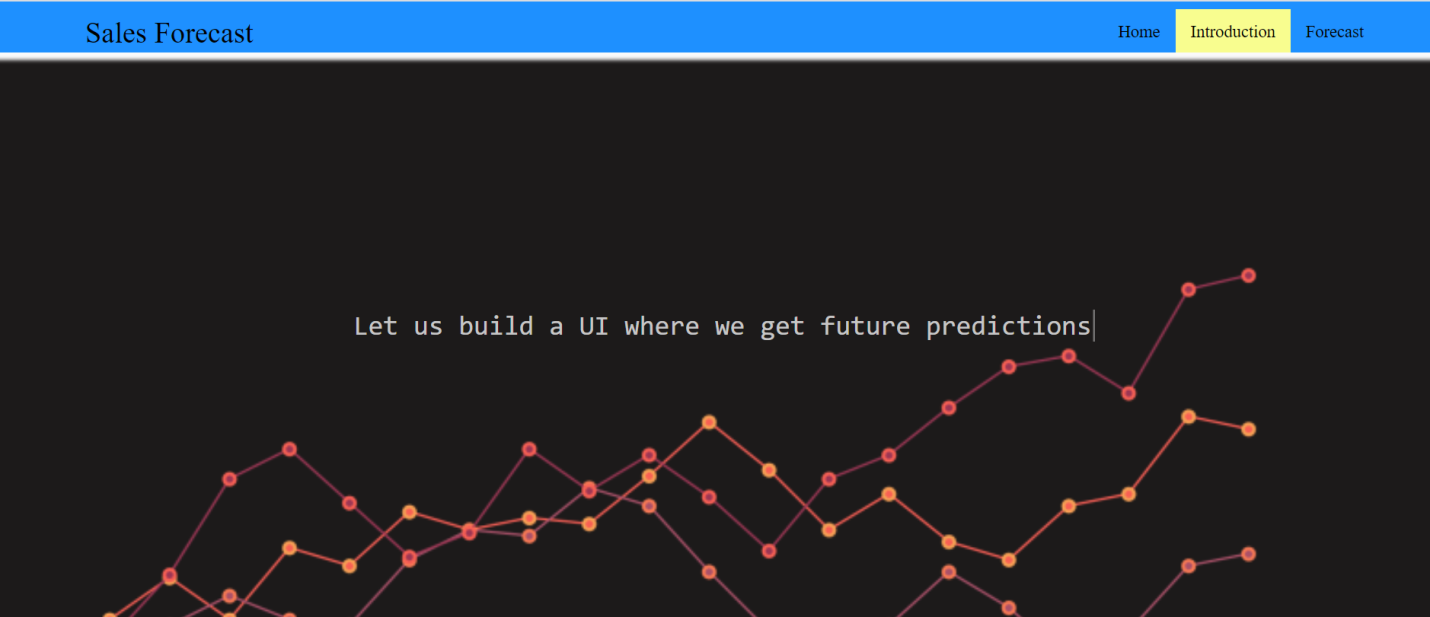




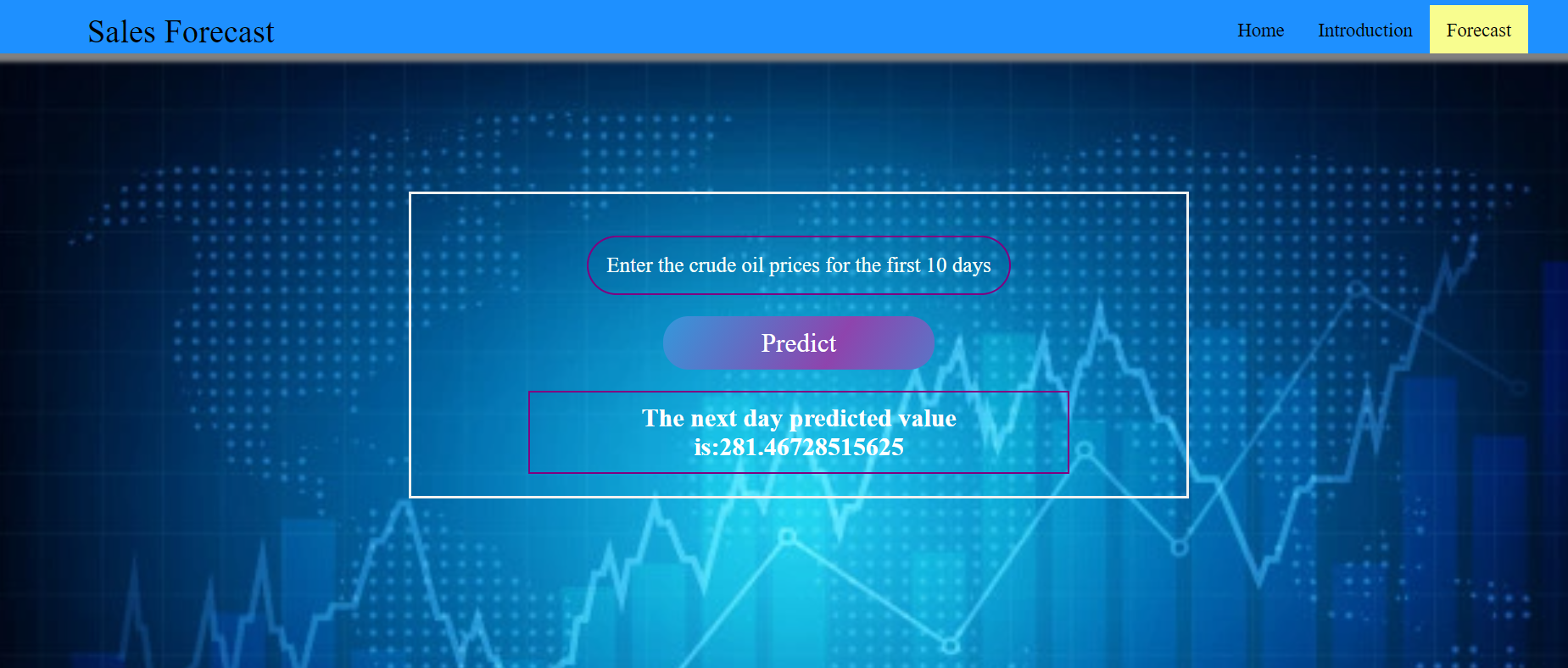
**Activity 4: Showcasing prediction on UI**

****

This is our home page where of the project.

****

As this is is a time series approach, the user has to give past 10 days sales as input to predict the future sales of store.

****

Here we give the input- last ten days and then we click on the predict button to predict the next day’s price.

As we see the predicted output is displayed on the User Interface