**MINOR PROJECT-I**

**15CS375L**

A presentation report on

***Predicting future sales for a company***

Submitted by:

RA1711003010327 - Neha Goswami

RA1711003010297 - Shashank Thakur

RA1711003010353 - Aditi A. Shukla

Under the supervision of :

Ms.Aswathy K Cherian (102217)

Assistant Professor (O.G)

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SRM Institute of Science and Technology, KATTANKULATHUR – 603203. JUNE,2019.**

**BONAFIDE CERTIFICATE**

Certified that the report on ....................................................................... , is a proof of successful completion of Minor Project–I programme undergone by……………………………………………………………… .............................................................................................................................................................................................. during the period .................................... to .................................... .

**Date Signature of the Teacher**

**In charge**

**ACKNOWLEDGEMENT**

I would like to express my special thanks of gratitude to our teacher **Mrs.Aswathy** ma'am who gave us the opportunity to make this project which helped us in doing a lot of research and we came to explore new things.

I would also like to express my sincere gratitude to **Mrs.Subalalitha** ma'am who encouraged us to research about this topic and provided us with required help throughout the making of the project.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Contents** | **Page** |
| 1. | Abstract | 5 |
| 2. | Introduction | 6 |
| 3. | Architectural diagrams | 7 |
| 4. | Implementation | 8 |
| 5. | Result | 14 |
| 6. | Analysis | 17 |
| 7. | Conclusion | 19 |
| 8. | Future Work | 20 |
| 9. | Reference | 21 |

**ABSTRACT**

Provided with daily historical sales data. The task is to forecast the future sales for a company. Creating a robust model that can handle such situations is the agenda of this project. It can be accomplished using data pipelining to load data, heal data and remove outliers.Time series forecasting is one of the major building blocks of Machine Learning. We will focus on Long Short-term Memory (LSTM) method. We will use [Keras](https://keras.io/" \t "https://towardsdatascience.com/_blank) in our project to implement LSTM.Keras is a high-level neural networks API, written in Python. It focuses on enabling fast experimentation. **Knowing the future sales helps the business as** it is a benchmark. It helps to calculate the incremental value of our new actions on top of the benchmark. It can be utilized for planning. According to this demands can be planned and supply actions by looking at the forecasts. It helps to see where to invest more. It acts as a guide for planning budgets and targets.

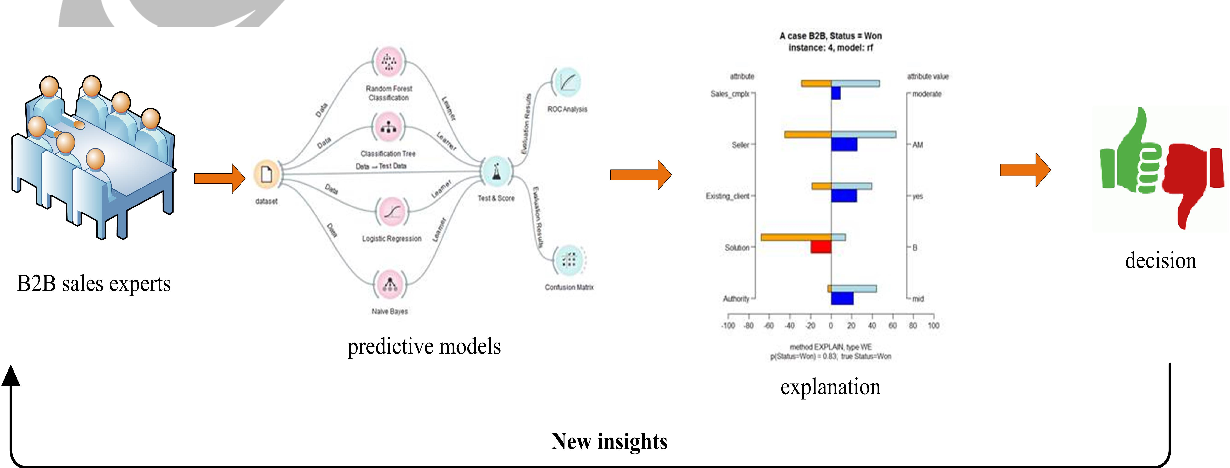
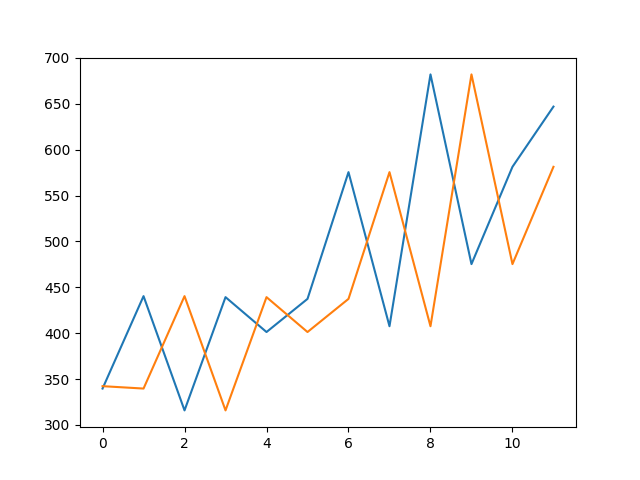
**KEY WORDS**

**Data Mining, Data science, Prediction, Sales**

**INTRODUCTION**

In this information age, the expertise of data analytics specialists or data scientists has become a critical success factor for organizations to understand and react to their environment. The shortage in skilled professionals and the resulting high cost causes a deficit of such experts in many domains . As a consequence, notably small and medium enterprises (SME) often miss the potential that lies in unexploited information. In the domain of e-commerce, sales forecasts provide an example of such critical information. Online retailers that are able to compute reliable forecasts, based on existing sales transactions can reduce losses caused by out of stock or non-selling items. Especially in short series product life cycle fields such as fashion, it is crucial to have accurate figures on upcoming sales even before production. Among SMEs cloud computing in general and software as a service (SaaS) in particular are popular solutions to share the costs for IT service development and operation. Therefore, the task of data analytics for product sales forecasting is a promising application for the new cloud service model. With the current system landscape of most online retailers, transactional data is scattered across various application system components and has to be preprocessed before it may be used. Data preprocessing consists of data cleaning, record selection, summarization, denormalization, variable creation and coding. It is considered as the most timeconsuming task in data analytics projects. However, collecting and cleansing data from various sources is a very customer specific task and therefore difficult to implement as a cloud service. The CATeLOG project which is financed by Dutch institute for advanced logistics (Dinalog) aims amongst others at the development of innovative, pluggable e-commerce services as well as a suitable platform architecture to facilitate the adoption of such services. For this work we have combined two areas of expertise within our research project to come up with a solution to develop state of the art sales forecasting logic and integrate it into a pluggable platform architecture. The research goal was to design and develop a cloud based sales forecasting service to allow small and medium enterprises to make use of advances in data analytics techniques. In section 2 we present the current research in sales forecasting and present a forecasting module which is the core component of the solution. In the third section we outline the concept of service pluggability and present the architecture for a pluggable service platform. In section 4 we present

**Architectural Diagrams**



**IMPLEMENTATION**

Time series forecasting is one of the major building blocks of Machine Learning

The implementation of our model will have 3 steps:

* Data Wrangling
* Data Transformation to make it stationary and supervised
* Building the model & evaluation

**Step 1: Define Network**: We will construct an Long short term neural network with a 1 input timestep and 1 input feature in the visible layer, 10 memory units in the Long Short Term hidden layer, and 1 neuron in the fully connected output layer with a linear (default) activation function.

**Step 2: Compile Network**: We will use the efficient Adaptive Moment Estimation optimization algorithm with default configuration and the mean squared error loss function because it is a regression problem.

**Step 3: Fit Network**: We will fit the network for 1,00 epochs and use a batch size equal to the number of patterns in the training set. We will also turn off all verbose output.

**Step 4: Evaluate Network**. We will evaluate the network on the training dataset. Typically we would evaluate the model on a test or validation set.

**Step 5: Make Predictions**. We will make predictions for the training input data. Again, typically we would make predictions on data where we do not know the right answer.

**PYTHON CODE :**

import matplotlib as mpl

import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))

plt.plot(df\_sales['date'],df\_sales['sales'])

# plt.plot(y,sales)

plt.show()

import warnings

warnings.filterwarnings("ignore")

! pip install plotly

!pip install cufflinks

import plotly.plotly as py

import plotly.offline as pyoff

import plotly.graph\_objs as go

import keras

from keras.layers import Dense

from keras.models import Sequential

from keras.optimizers import Adam

from keras.callbacks import EarlyStopping

from keras.utils import np\_utils

from keras.layers import LSTM

from sklearn.model\_selection import KFold, cross\_val\_score, train\_test\_split

pyoff.init\_notebook\_mode()

df\_sales = pd.read\_csv('/content/drive/My Drive/data/train.csv')

df\_sales.shape

df\_sales.head(10)

df\_sales['date'] = pd.to\_datetime(df\_sales['date'])

df\_sales['date'] = df\_sales['date'].dt.year.astype('str') + '-' + df\_sales['date'].dt.month.astype('str') + '-01'

df\_sales['date'] = pd.to\_datetime(df\_sales['date'])

df\_sales = df\_sales.groupby('date').sales.sum().reset\_index()

df\_sales.head()

import matplotlib as mpl

import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))

plt.plot(df\_sales['date'],df\_sales['sales'])

# plt.plot(y,sales)

​

plt.show()

df\_diff = df\_sales.copy()

df\_diff['prev\_sales'] = df\_diff['sales'].shift(1)

​

df\_diff.head()

#drop the null values and calculate the difference

df\_diff = df\_diff.dropna()

df\_diff['diff'] = (df\_diff['sales'] - df\_diff['prev\_sales'])

df\_diff.head(10)

import matplotlib as mpl

import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))

plt.plot( df\_diff['date'],df\_diff['diff'])

#plt.figure(figsize=(20000,10000))

​

df\_supervised = df\_diff.drop(['prev\_sales'],axis=1)

for inc in range(1,13):

field\_name = 'lag\_' + str(inc)

df\_supervised[field\_name] = df\_supervised['diff'].shift(inc)

df\_supervised.head(10)

df\_supervised.tail(6)

df\_supervised = df\_supervised.dropna().reset\_index(drop=True)

import statsmodels.formula.api as smf

​

model = smf.ols(formula='diff ~ lag\_1', data=df\_supervised)

​

model\_fit = model.fit()

​

regression\_adj\_rsq = model\_fit.rsquared\_adj

print(regression\_adj\_rsq)

import statsmodels.formula.api as smf

​

# Define the regression formula

model = smf.ols(formula='diff ~ lag\_1 + lag\_2 + lag\_3 + lag\_4 + lag\_5', data=df\_supervised)

​

# Fit the regression

model\_fit = model.fit()

​

# Extract the adjusted r-squared

regression\_adj\_rsq = model\_fit.rsquared\_adj

print(regression\_adj\_rsq)

import statsmodels.formula.api as smf

​

# Define the regression formula

model = smf.ols(formula='diff ~ lag\_1 + lag\_2 + lag\_3 + lag\_4 + lag\_5 + lag\_6 + lag\_7 + lag\_8 + lag\_9 + lag\_10 + lag\_11 + lag\_12', data=df\_supervised)

​

# Fit the regression

model\_fit = model.fit()

​

# Extract the adjusted r-squared

regression\_adj\_rsq = model\_fit.rsquared\_adj

print(regression\_adj\_rsq)

#import MinMaxScaler and create a new dataframe for LSTM model

from sklearn.preprocessing import MinMaxScaler

df\_model = df\_supervised.drop(['sales','date'],axis=1)

​

#split train and test set

train\_set, test\_set = df\_model[0:-6].values, df\_model[-6:].values

df\_model.info()

#apply Min Max Scaler

scaler = MinMaxScaler(feature\_range=(-1, 1))

scaler = scaler.fit(train\_set)

# reshape training set

train\_set = train\_set.reshape(train\_set.shape[0], train\_set.shape[1])

train\_set\_scaled = scaler.transform(train\_set)

​

# reshape test set

test\_set = test\_set.reshape(test\_set.shape[0], test\_set.shape[1])

test\_set\_scaled = scaler.transform(test\_set)

X\_train, y\_train = train\_set\_scaled[:, 1:], train\_set\_scaled[:, 0:1]

X\_train = X\_train.reshape(X\_train.shape[0], 1, X\_train.shape[1])

X\_test, y\_test = test\_set\_scaled[:, 1:], test\_set\_scaled[:, 0:1]

X\_test = X\_test.reshape(X\_test.shape[0], 1, X\_test.shape[1])

model = Sequential()

model.add(LSTM(4, batch\_input\_shape=(1, X\_train.shape[1], X\_train.shape[2]), stateful=True))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, y\_train, nb\_epoch=100, batch\_size=1, verbose=1, shuffle=False)

y\_pred = model.predict(X\_test,batch\_size=1)

y\_pred

y\_test

y\_pred = y\_pred.reshape(y\_pred.shape[0], 1, y\_pred.shape[1])

pred\_test\_set = []

for index in range(0,len(y\_pred)):

print (np.concatenate([y\_pred[index],X\_test[index]],axis=1))

pred\_test\_set.append(np.concatenate([y\_pred[index],X\_test[index]],axis=1))

pred\_test\_set = np.array(pred\_test\_set)

pred\_test\_set = pred\_test\_set.reshape(pred\_test\_set.shape[0], pred\_test\_set.shape[2])

#inverse transform

pred\_test\_set\_inverted = scaler.inverse\_transform(pred\_test\_set)

#create dataframe that shows the predicted sales

result\_list = []

sales\_dates = list(df\_sales[-7:].date)

act\_sales = list(df\_sales[-7:].sales)

for index in range(0,len(pred\_test\_set\_inverted)):

result\_dict = {}

result\_dict['pred\_value'] = int(pred\_test\_set\_inverted[index][0] + act\_sales[index])

result\_dict['date'] = sales\_dates[index+1]

result\_list.append(result\_dict)

df\_result = pd.DataFrame(result\_list)

df\_result

df\_sales.head()

#merge with actual sales dataframe

df\_sales\_pred = pd.merge(df\_sales,df\_result,on='date',how='left')

df\_sales\_pred

import matplotlib as mpl

import matplotlib.pyplot as plt

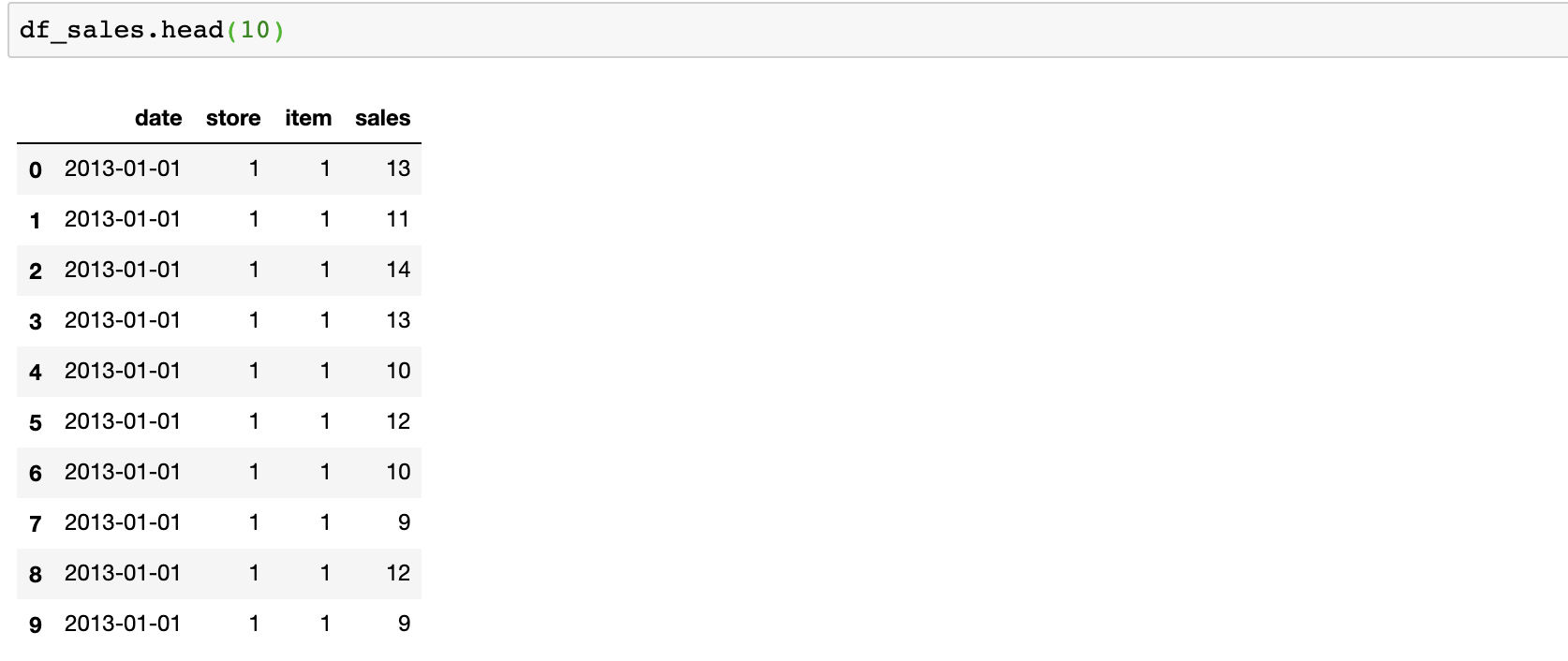
plt.figure(figsize=(15,10))

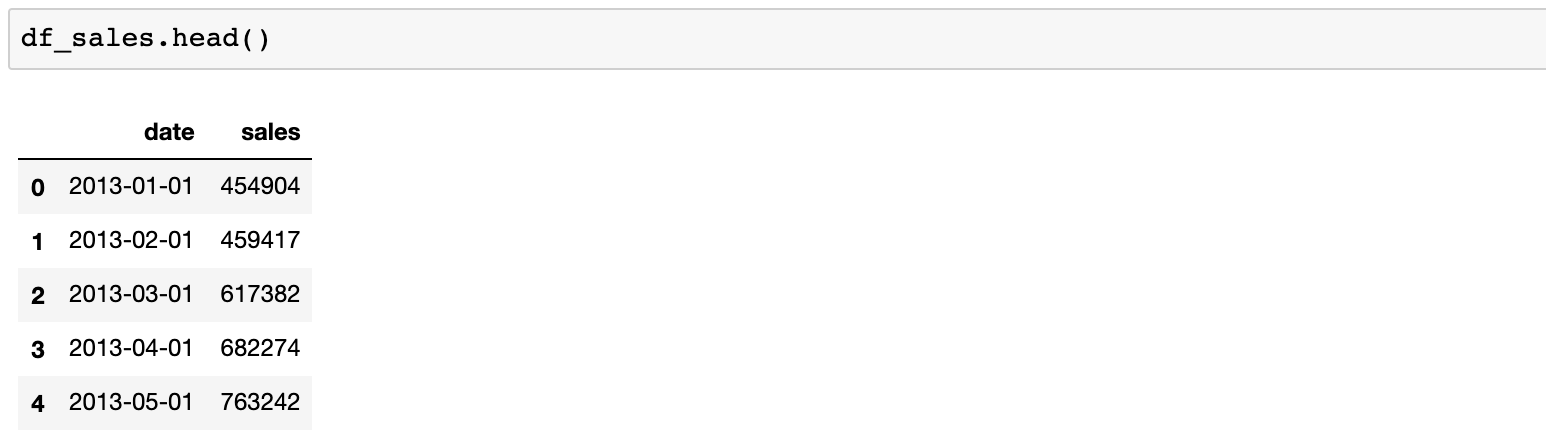
plt.plot(df\_sales\_pred['date'],df\_sales\_pred['sales'])

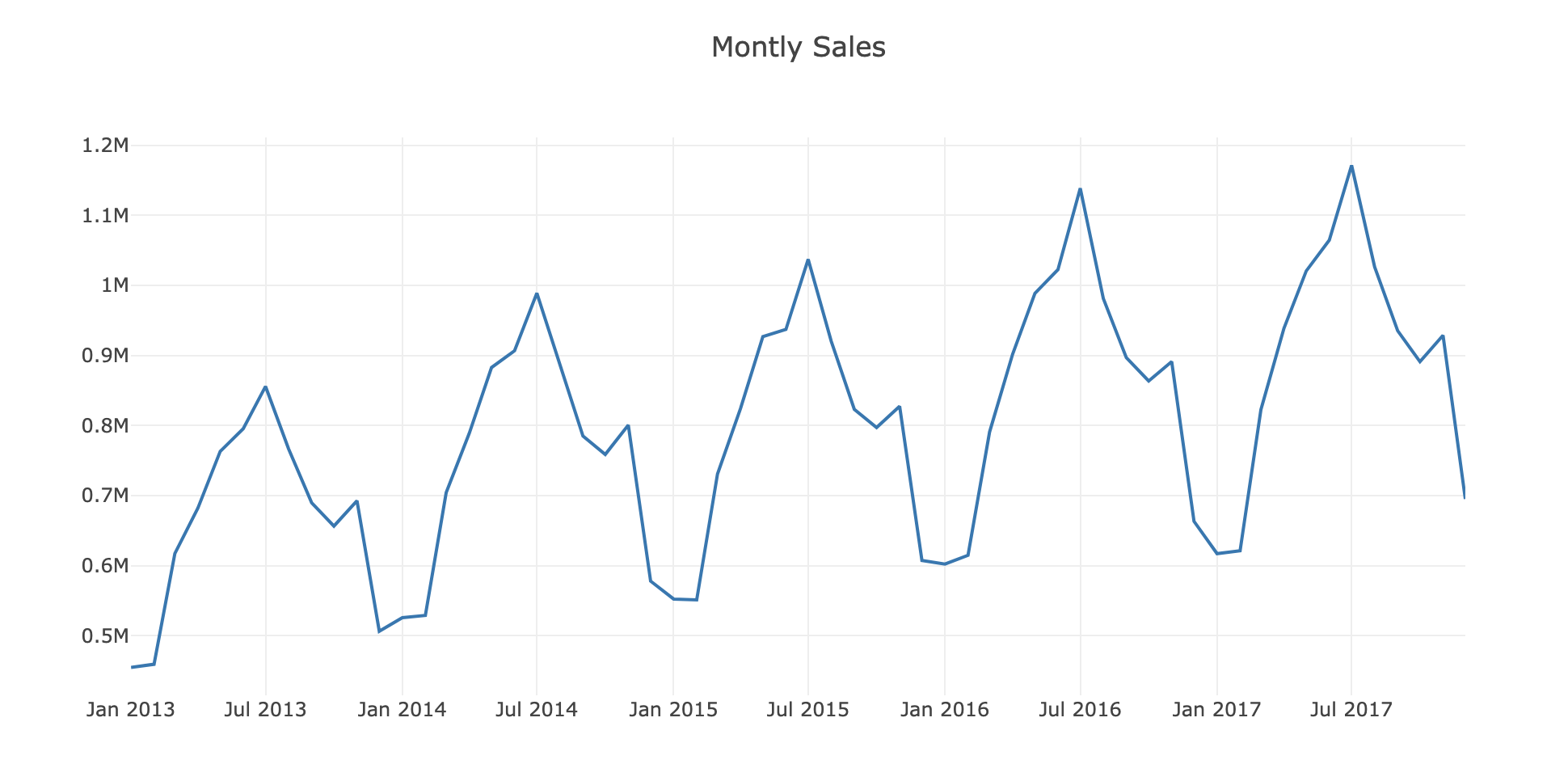
plt.plot(df\_sales\_pred['date'],df\_sales\_pred['pred\_value'])

#plt.figure(figsize=(20000,10000))

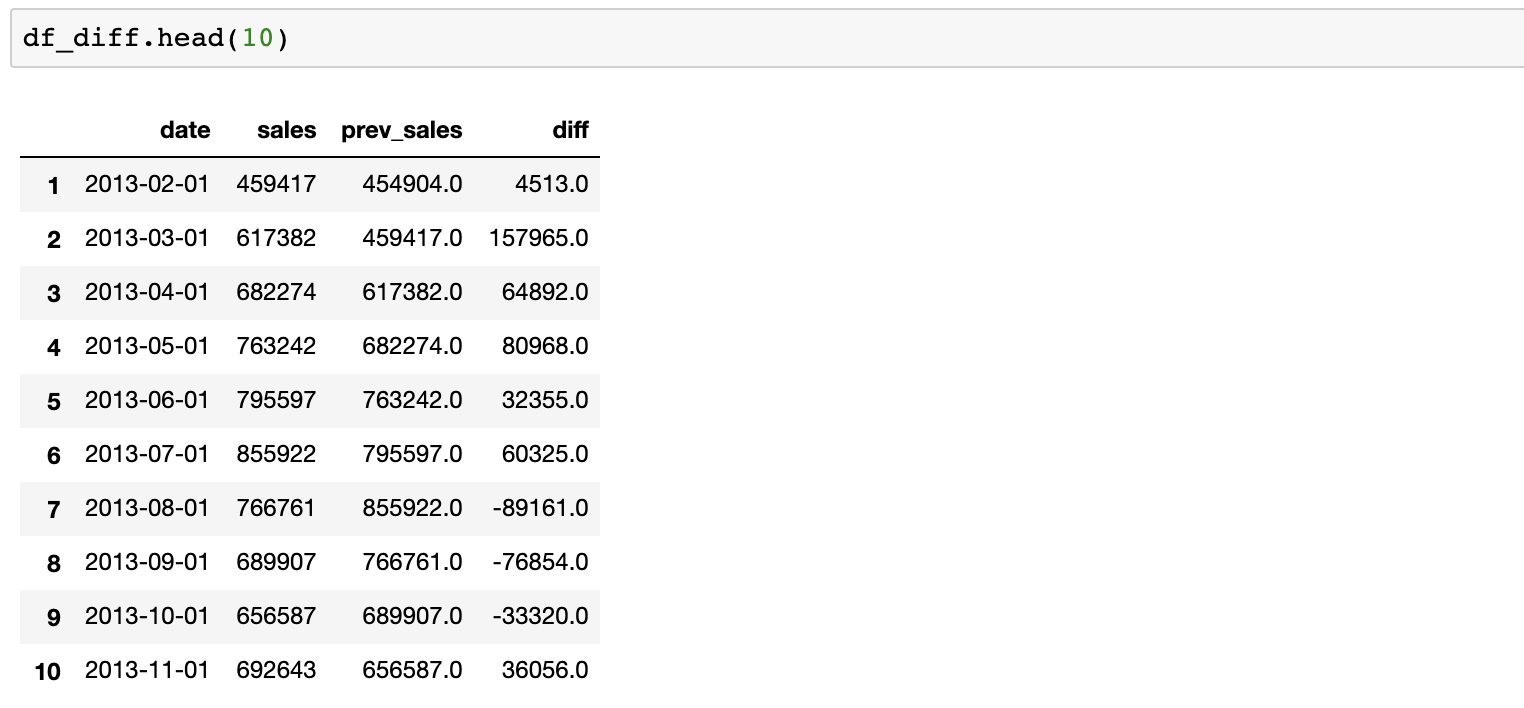
**RESULT**

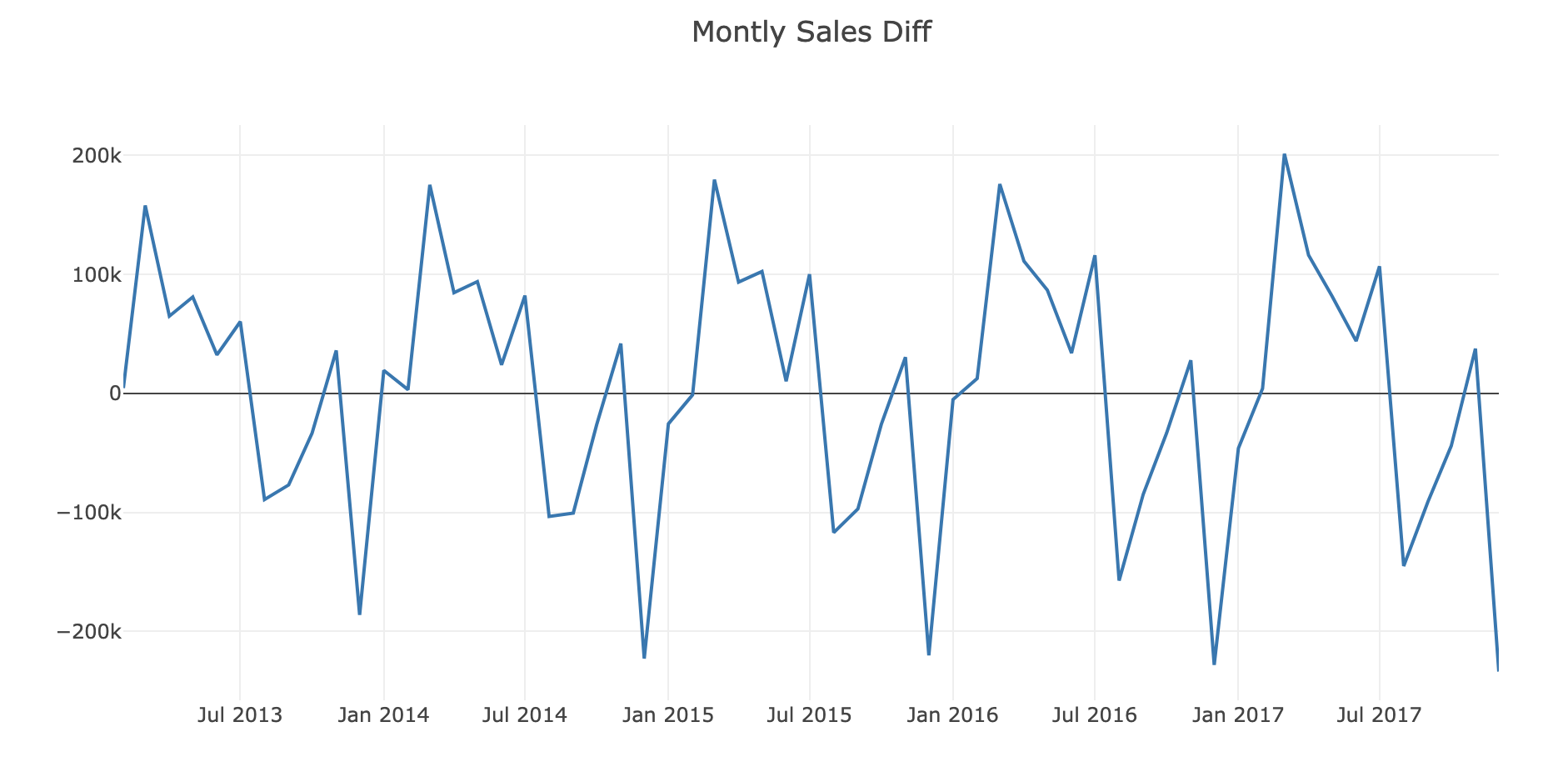




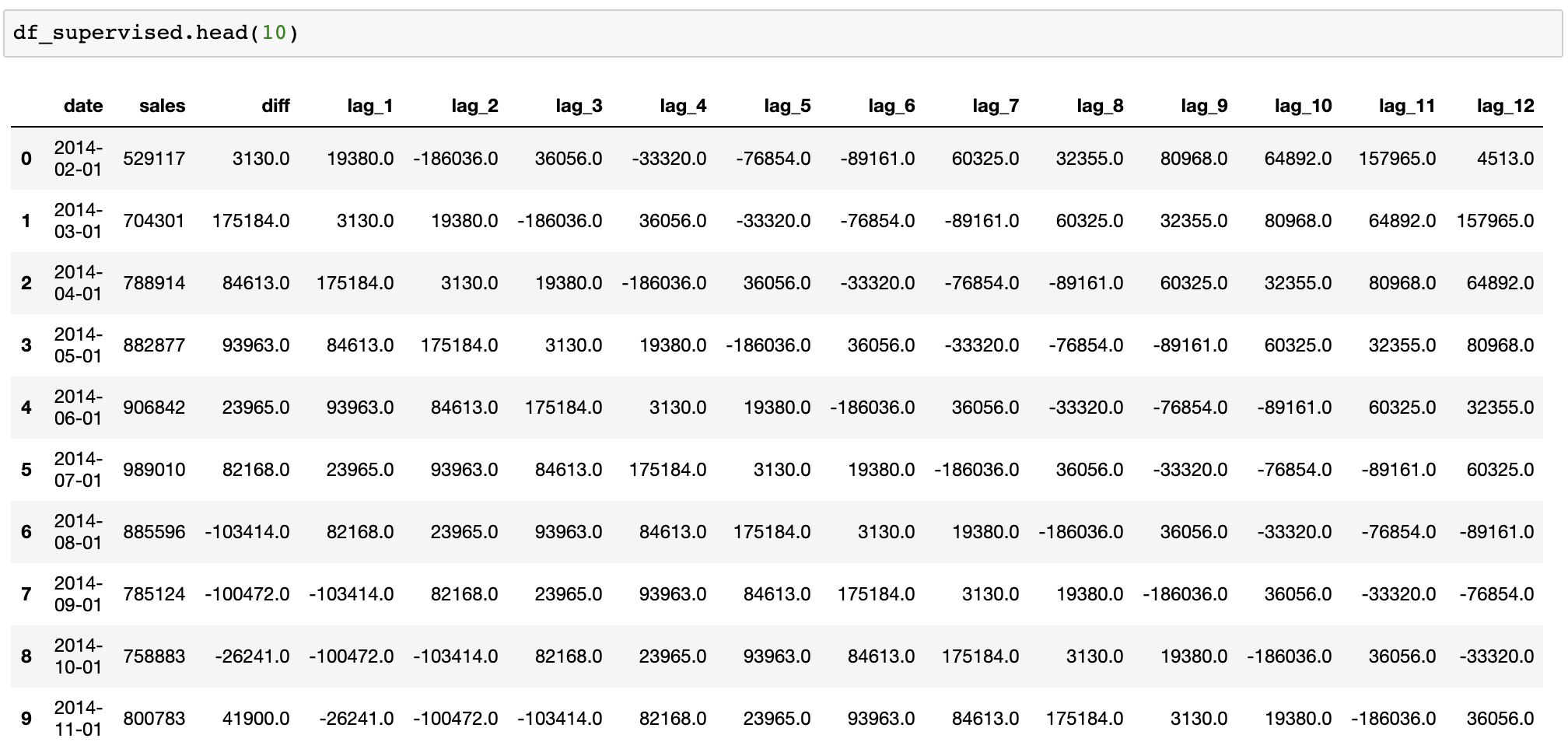


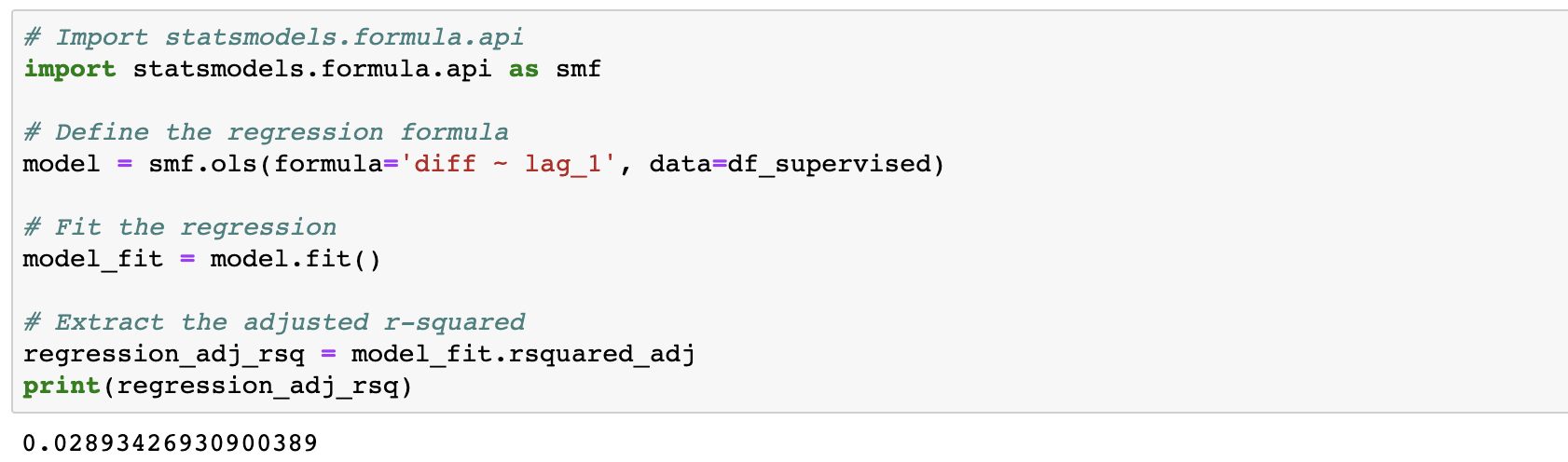
**NOT STATIONARY**

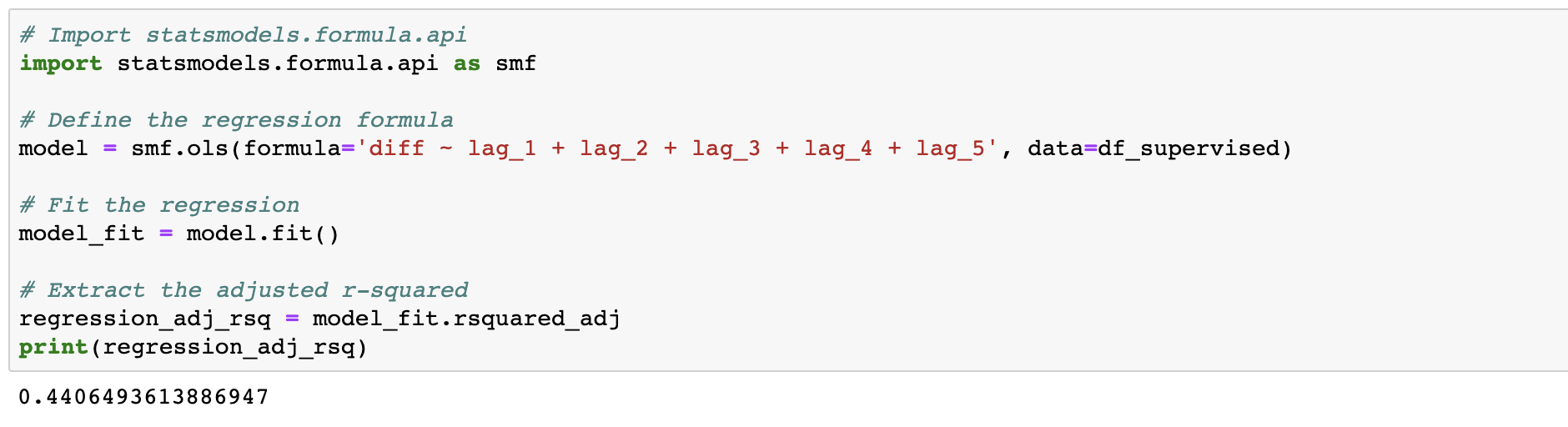


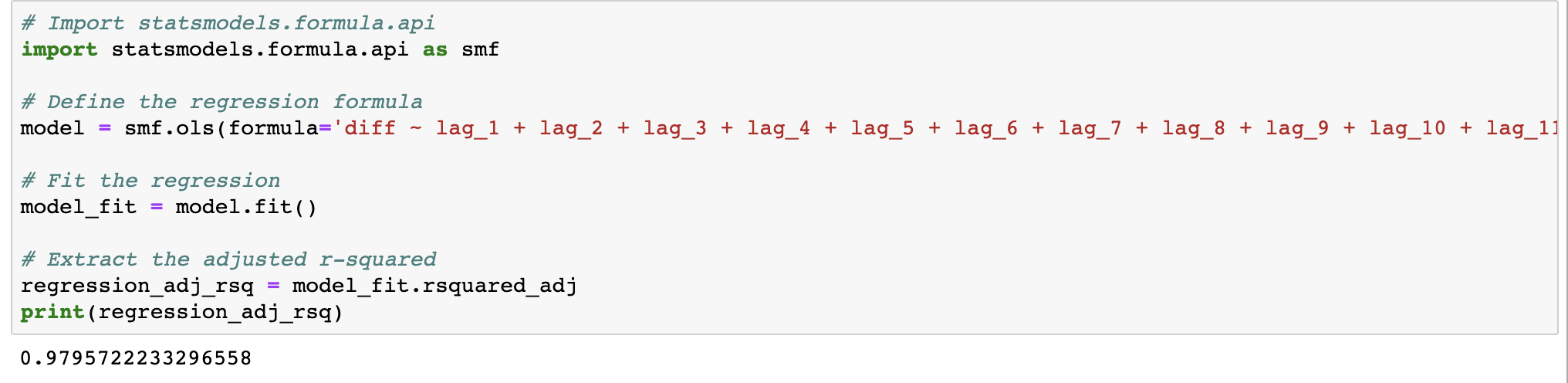


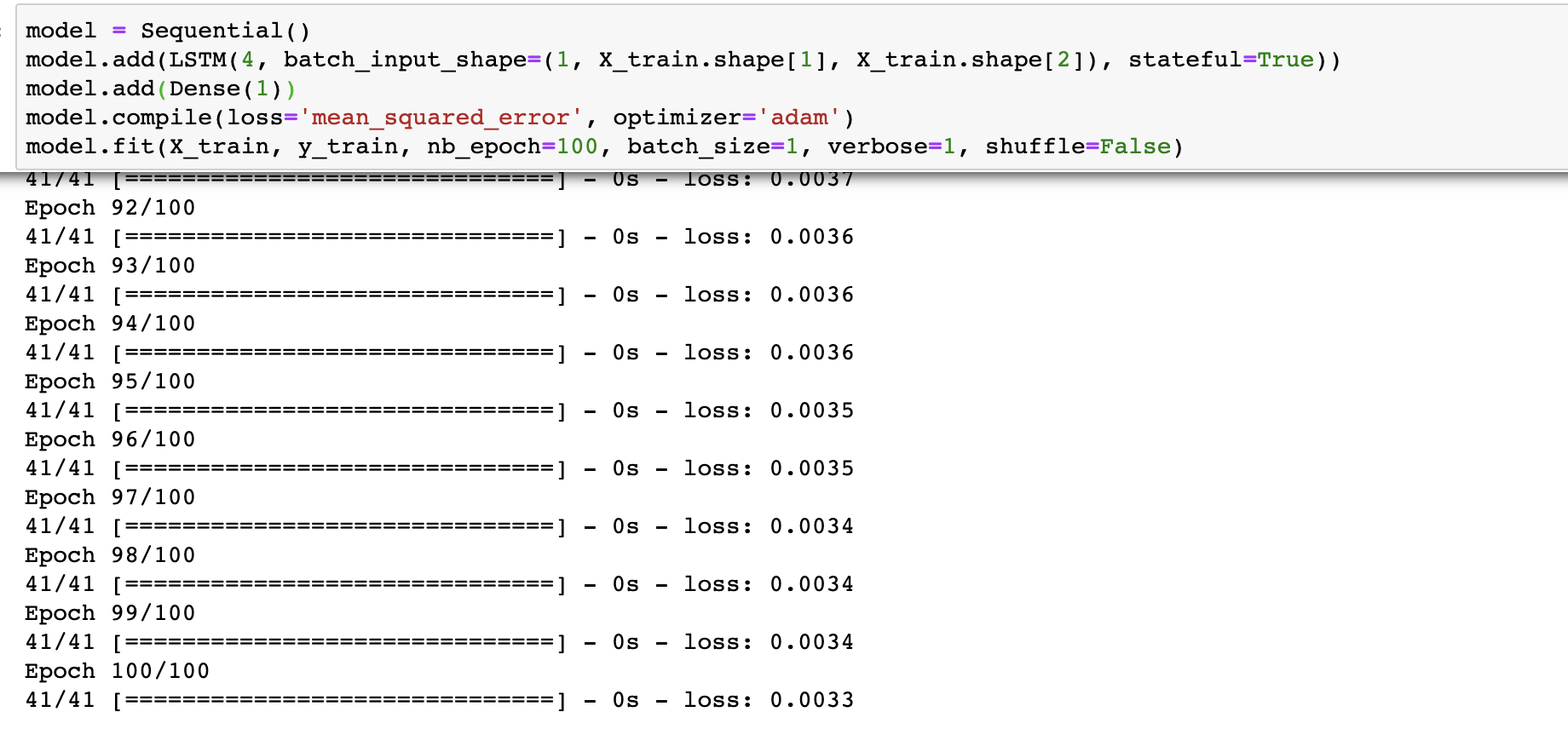
**STATIONARY**

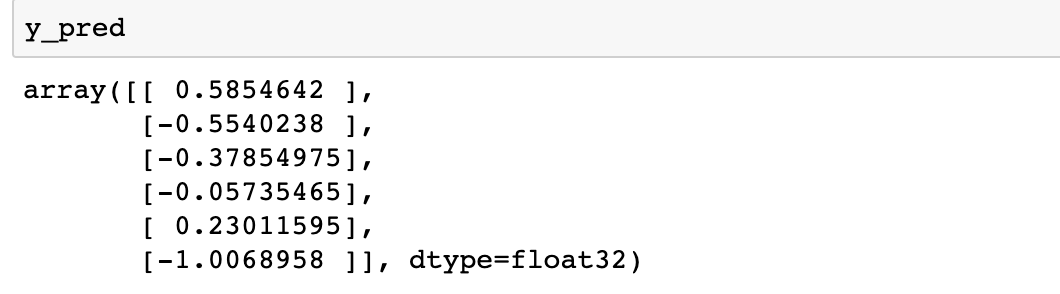


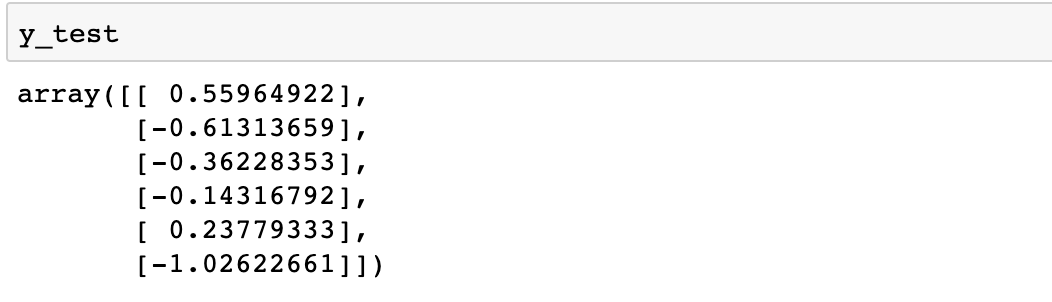


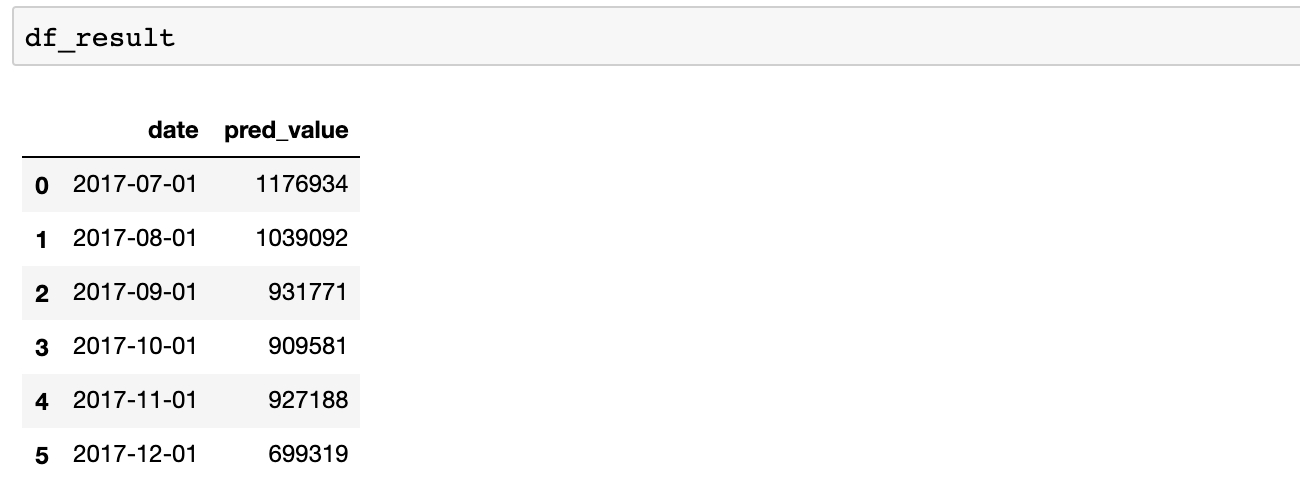


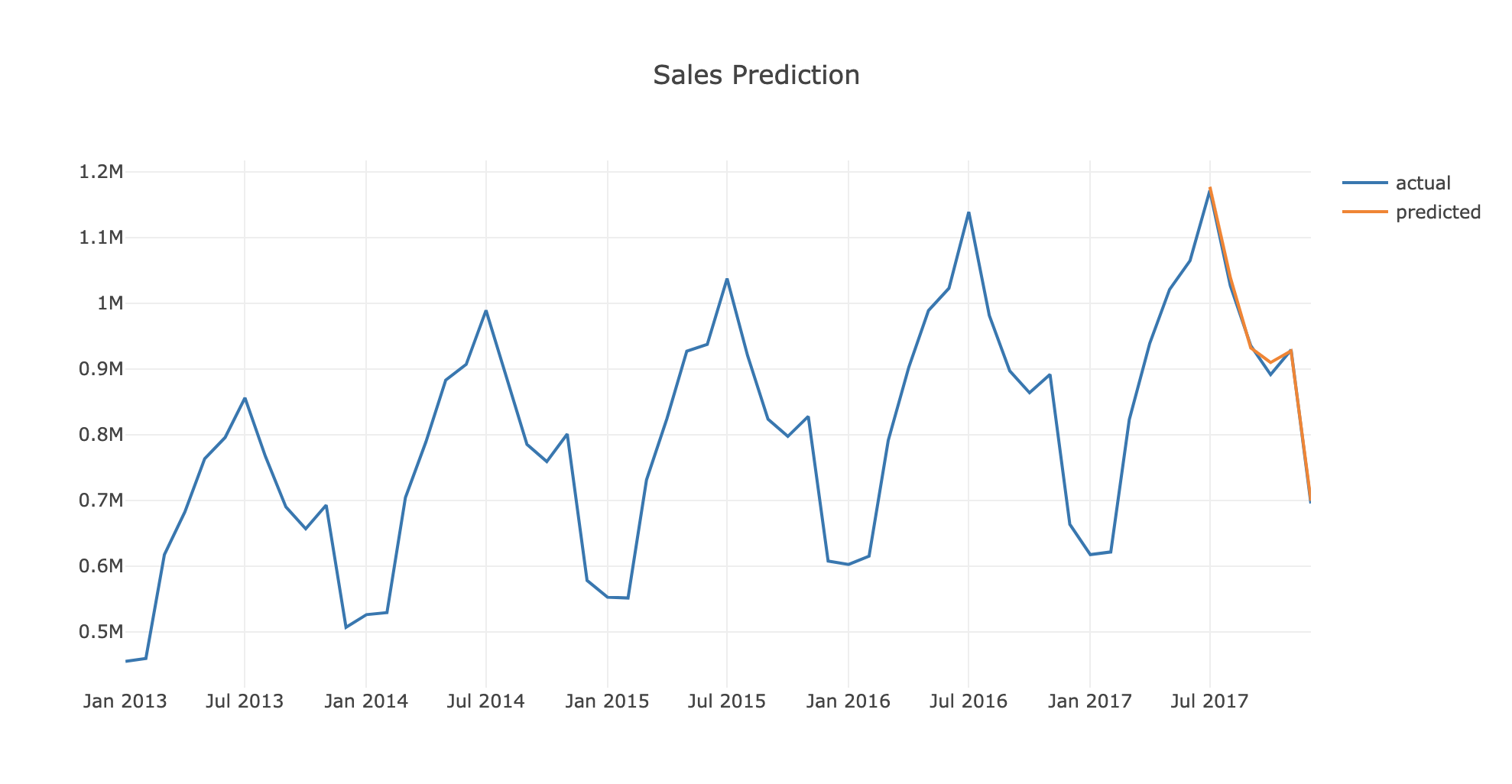












**Actual Vs Predicted**

In this tutorial, you discovered the steps and the tools for a time series forecasting project with Python.

We have covered a lot of ground in this tutorial; specifically:

* How to develop a test harness with a performance measure and evaluation method and how to quickly develop a baseline forecast and skill.
* How to use time series analysis to raise ideas for how to best model the forecast problem.
* How to develop an LSTM model, save it, and later load it to make predictions on new data.

**ANALYSIS**

We use the [dataset](https://www.kaggle.com/c/demand-forecasting-kernels-only" \t "_blank) from a Kaggle competition. It represents the daily sales for each store and item.

Like always we start with importing the required libraries and importing our data from CSV.

Our task is to forecast monthly total sales. We need to aggregate our data at the monthly level and sum up the sales column.

After applying the code above, **df\_sales**is now showing the aggregated sales we need:

**Data Transformation**

To model our forecast easier and more accurate, we will do the transformations below:

* We will convert the data to stationary if it is not
* Converting from time series to supervised for having the feature set of our model
* Scale the data

First off, how do we check if the data is not stationary

Obviously, it is not stationary and has an increasing trend over the months. One method is to get the difference in sales compared to the previous month and build the model on it

Perfect! Now we can start building our feature set. We need to use previous monthly sales data to forecast the next ones. The look-back period may vary for every model. Ours will be 12 for this example.

So what we need to do is to create columns from lag\_1 to lag\_12 and assign values by using **shift()**method

**How useful are our features for prediction?**

Adjusted R-squared is the answer. It tells us how good our features explain the variation in our label (lag\_1 to lag\_12 for diff)

Basically, we fit a linear regression model (OLS — Ordinary Least Squares) and calculate the Adjusted R-squared. For the example above, we just used **lag\_1** to see how much it explains the variation in column **diff**. The output of this code block is:

lag\_1 explains 3% of the variation. Let’s check out others:

Adding four more features increased the score from 3% to 44%.

The result is impressive as the score is 98%. Now we can confidently build our model after scaling our data. But there is one more step before scaling. We should split our data into train and test sets. As the test set, we have selected the last 6 months’ sales.

## Building the model

Everything is ready to build our first deep learning model. We create feature and label sets from scaled datasets.

The code block above prints how the model improves itself and reduce the error in each epoch.

We do the prediction and see how the results look like: Results look similar but it doesn’t tell us much because these are scaled data that shows the difference. How we can see the actual sales prediction?

First, we need to do the inverse transformation for scaling.

Second, we need to build the dataframe has the dates and the predictions. Transformed predictions are showing the difference. We should calculate the predicted sales numbers.

**CONCLUSION**

* LSTM is great tool for anything that has a sequence. Since the meaning of a word depends on the ones that preceded it. This paved the way for NLP and narrative analysis to leverage Neural Networks.
* LSTM can be used for text generation. You can train the model on the text of a writer, say, and the model will be able to generate new sentences that mimics the style and interests of the writer.
* Sequence-to-Sequence LSTM models are the state of the technique for translations. They also have a wide array of applications like time series forecasting.
* Sales forecasting is a crucial part of the financial planning of a business. It's a self-assessment tool that uses past and current sales statistics to intelligently predict future performance.
* With an accurate sales forecast in hand, you can plan for the future. If your sales forecast says that during December you make 30 percent of your yearly sales, then you need to ramp up manufacturing in September to prepare for the rush. It might also be smart to invest in more seasonal salespeople and start a targeted marketing campaign right after Thanksgiving. One simple sales forecast can inform every other aspect of your business.
* Time series forecasting is one of the major building blocks of Machine Learning
* There are many methods in the literature to achieve this like Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving-Average (SARIMA), Vector Autoregression (VAR), and so on..

**FUTURE WORK**

Forecasts in future can be helpful in suitable capital planning. It can be used to provide information for major strategic decisions. It can help in saving the wastages in material, man hours, machine time and capacity. Planning of a new unit must start with an analysis of the long term demand potential of the products of the firm.It can also be used to provide information for tactical decisions.

Forecasting reduces the risk associated with business fluctuations which generally produce harm­ful effects in business, create unemployment, induce speculation, discourage capital formation and reduce the profit margin.

Forecasting is indispensable and it plays a very important part in the determi­nation of various policies. In modern times forecasting has been put on scientific footing so that the risks associated with it have been considerably minimised and the chances of precision increased.

**REFERENCE**

<https://www.kaggle.com/c/competitive-data-science-predict-future-sales>

<https://datascience.stackexchange.com/questions/46319/how-to-predict-future-sales-based-upon-last-year-sales-data>

<https://machinelearningmastery.com/time-series-forecast-study-python-monthly-sales-french-champagne/>

<https://en.wikipedia.org/wiki/Forecasting>

<https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/>

<https://www.bistasolutions.com/resources/blogs/5-statistical-methods-for-forecasting-quantitative-time-series/>

<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>