

**A Major Project Report on**  
**SOCIAL NETWORK ADS SALES PREDICTION SYSTEM**  
**USING DEEP LEARNING TECHNIQUES**

**Submitted In partial fulfillment of the Requirements**  
**for the award of the degree of**

**BACHELOR OF TECHNOLOGY**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**

**BY**

<b>BABLU</b>	<b>(17UJ1A0502)</b>
<b>CHAPPIDI RAJENDRA PRASAD</b>	<b>(17UJ1A0504)</b>
<b>BEMAPURI MANISH KUMAR</b>	<b>(17UJ1A0546)</b>
<b>AMGOTH ANIL</b>	<b>(17UJ1A0501)</b>
<b>R PARASHURAM</b>	<b>(17UJ1A0519)</b>

**Under the Esteemed Guidance of**  
**Mr. P. SUDHEER, M.Tech**  
**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND**  
**ENGINEERING MALLA REDDY COLLEGE OF ENGINEERING**  
**AND MANAGEMENT SCIENCE**

**(Approved by AICTE New Delhi & Affiliated to JNTU Hyderabad)**  
**Kistapur, Medchal, Medchal Dist- 501401.**

**2017-2021**



**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY**  
**HYDERABADKUKATPALLY, HYDERABAD-85.**



MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT  
SCIENCES (Approved by AICTE New Delhi & Affiliated to JNTU Hyderabad)  
Kistapur, Medchal, Medchal Dist- 501401.

### CERTIFICATE

This is to certify that the major project report entitled " Social  
Network Ads Sales Prediction System Using Deep Learning Techniques"  
being submitted by

<b>BABLU</b>	<b>(17UJ1A0502)</b>
<b>CHAPPIDI RAJENDRA PRASAD</b>	<b>(17UJ1A0504)</b>
<b>BEMAPURI MANISH KUMAR</b>	<b>(17UJ1A0546)</b>
<b>AMGOTH ANIL</b>	<b>(17UJ1A0501)</b>
<b>R PARASHURAM</b>	<b>(17UJ1A0519)</b>

in partial fulfillment for the award of the Degree of Bachelor of Technology  
in Computer Science and Engineering to the Jawaharlal Nehru Technological  
University , Hyderabad, during the academic year 2017-2021 is a record of  
bonafied work carried out under my guidance and supervision.

The results embodied in this project report have not been submitted to  
any other University or Institute for the award of any Degree or Diploma.

#### Internal Guide

**Mr. P. SUDHEER M.Tech.**

Assistant Professor  
Department.

#### Head of The Department

**Dr. D . MAHAMMAD RAFI, M.Tech, Ph.D**

Professor & Head of CSE  
Department of CSE

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

First and foremost, I express my deep debt of gratitude to the Chairperson and Managing Trustee **Mr.V. Malla Reddy** for their immense contribution in making this organization grow and providing me the state of the art facilities to do this project work.

My heartfelt gratefulness to Director of MREM **Mr.L. Venugopal Reddy** for their deep commitment and dedication, to bring this institution to the peak in terms of discipline and values.

I owe my full satisfaction in my project work to the extensive hands of cooperation of Principal Academics **Dr.CNV.SRIDHAR , M.Tech,Ph.D. .**

I would like to express my deep appreciation and sincere gratitude to my project internal guide , **Mr.P. Sudheer**, Assistant Professor , , Department of Computer Science & Engineering, Malla Reddy Engineering College and Management Sciences, for his guidance, support, encouragement, understanding and patience. I have been honored to work under his supervision and learn from her advice and useful insights throughout my project work.

I extend my gratitude to **Dr. D.Mahammad Rafi**, Professor and Head, Department of Computer Science and Engineering for his constant support to complete the project work.

I express my sincere thanks to all the teaching and non-teaching staffs of Malla Reddy Engineering College and Management Sciences who cooperated with me, which helped me to realize my dream.We would like to thank our internal project mates and department faculties for their full-fledged guidance and giving courage to carry out the project.I am very much thankful to one and all that helped me for the successful completion of my project.

<b>BABLU</b>	<b>(17UJ1A0502)</b>
<b>CHAPPIDI RAJENDRA PRASAD</b>	<b>(17UJ1A0504)</b>
<b>BEMAPURI MANISH KUMAR</b>	<b>(17UJ1A0546)</b>
<b>AMGOTH ANIL</b>	<b>(17UJ1A0501)</b>
<b>R PARASHURAM</b>	<b>(17UJ1A0519)</b>

## **ABSTRACT**

Social Media Advertising plays a vital role in today's business. It helps marketers to build relationships with their customers and increase sales. Marketers are using social media to advertise their products and generate sales. Social networking sites such as YouTube, Facebook, and Instagram are essential in today's competitive business for boosting the sales of the firm. This study aims to predict the impact of social media advertising on sales of a company. The purpose of this paper is to build a linear regression model that predicts sales based on money spent on YouTube advertisements. The data is analyzed using the python open-source software program.

The python is a powerful programming tool that can represent the dataset graphically concerning different parameters, and it also uses different packages available. The result of this research shows that YouTube advertising is a better predictor of company sales

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>iv</b>
	<b>LIST OF TABLES</b>	<b>xii</b>
	<b>LIST OF FIGURES</b>	<b>xiii</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>xvi</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Social Media Marketing	1
1.1.1	Social Media Monitoring	2
1.1.2	Sentiment Analysis For Social Media Marketing	2
1.1.3	Image Recognition For Social Media Marketing	3
1.1.4	Chatbots For Social Media Marketing	3
1.2	Objective	4
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>5</b>
2.1	Disadvantages Of Existing System	7
2.2	Summary of Existing System	7
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>8</b>
3.1	Problem Statement	8
3.2	Linear Regression Algorithm	8
3.2.1	Estimating The Coefficients	9
3.2.2	Estimate The Relevancy Of The Coefficients	10
3.2.3	Assess The Accuracy Of The Model	11
3.2.4	The Theory Of Multiple Linear Regression	12
3.2.5	Assess The Relevance Of A Predictor	12

3.2.6	Assess The Accuracy Of The Model	13
3.2.7	Adding Interaction	13
3.3	Software Modules/Libraries Used	14
3.3.1	Python Programming Language	14
3.3.2	Invoking The Interpreter	16
3.3.3	Argument Passing	17
3.3.4	Interactive Mode	18
3.4	Scikit Learn	25
3.5	Pandas	30
3.6	Matplotlib	61
3.6.1	Third Party Distributions Of Matplotlib	62
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>70</b>
4.1	System Design: Proposed System Flow Chart	70
4.2	Proposed System Data Flow Diagram	71
4.3	Proposed System Class Diagram	73
<b>5</b>	<b>SOFTWARE AND HARDWARE REQUIREMENTS</b>	<b>74</b>
5.1	Software Requirements	74
5.2	Hardware Requirements	74
<b>6</b>	<b>CODING</b>	<b>75</b>
6.1	Input Data Sets	75
6.2	Proposed System Source Code	76
<b>7</b>	<b>TESTING</b>	<b>81</b>
7.1	Proposed System Testing	81
7.1.1	Black Box Testing	81
7.1.2	Typing Of Black Box Testing	83

7.1.3 Black Box Testing Techniques	83
7.2 White Box Testing	84
<b>8 OUTPUT SCREENS</b>	<b>85</b>
8.1 Proposed System Output Screens	85
<b>9 CONCLUSION</b>	<b>89</b>
<b>10 FURTHER ENHANCEMENTS</b>	<b>90</b>
<b>11 REFERENCES</b>	<b>91</b>

**LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
2.1	Literature Survey	6
3.5	Data Structure	32
5.2	Hardware Requirement	74
6.2	Sales Data	78
7.1.1	Pros and Cons of Black Box Testing	83



## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Linear fit to a data set	10
3.5	Sepal Ratio	59
3.6.1	track of all the child Axes, a smattering of 'special' artists and the <b>canvas</b>	69
4.1	Proposed System Flow Chart	70
4.2	Proposed system data flow diagram	72
4.3	Proposed System class diagram	73
6.1	Input Data Sets for Sales Ads	75
6.2	Sales Data	76
8.1	Correlations of the Sales Data	86
8.2	Final Sales Prediction	87
8.3	YouTube Ads vs Sales Prediction	87
8.4	Google Ads vs Sales Prediction	88
8.5	Facebook Ads vs Sales Prediction	88
8.6	Best Ads vs Sales Prediction in YouTube	89

## LIST OF ABBREVIATIONS

ADS	Advertisement
ALGO	Algorithms
BD	Budget in Rupees
CLI	Command Line Interface
COEFF	Coefficient
CSV	Comma Separated Values
DS	Data Structures
FB	Facebook
GUI	Graphical User Interface
LANG	Language
LIB	Libraries
LR	Linear Regression
MAT	Matplotlib Library
TMOD	Model used For Training
OS	Operating System
PD	Pandas
PI	Constant Coefficient
RAM	Random Access Memory
REF	References
SK	Sklearn
SM	Social Media
SQL	Structured Query Language

## **CHAPTER - I**

### **INTRODUCTION**

#### **1. INTRODUCTION**

In today's competitive global market, social media have become a path where marketers can spread their marketing campaigns to a broader range of customers. The tools and methods for interacting with customers have transformed significantly with the occurrence of social media; so, businesses must understand utilizing social media in a manner that is constant with their business plan (Mangold and Faulds 2009). Social media marketing helps companies to gain a competitive advantage and increase their sale. Social media websites such as Face book, Twitter, and YouTube have nearby to five million visitors daily, hence an essential hub for marketing. Social media has transformed the face of advertising most obviously in that it has almost eliminated the necessity for print advertising, thus providing a greener environment. There are a massive number of social media sites for advertising, but YouTube advertising is selected for this research.

#### **1.1 SOCIAL MEDIA MARKETING**

##### **4 Ways Machine Learning Can Enhance Social Media Marketing:**

Instagram is a global platform where businesses can showcase their products to over 800 million total Instagrammers, of which over 500 million are active on the app at least once a day. Facebook and Twitter also allow businesses to provide customer support and spread the word about upcoming events and sales to a huge audience. 63% of customers prefer customer support on social media, compared to other avenues like phone or email. Major businesses such as GameStop, UNIQLO, and The Container Store are using social media to establish and maintain relationships with influencers and engage with customers in a casual setting. In recent years, social media marketing has become crucial for most businesses to remain competitive.

## **Social Network Ads Sales Prediction System Using Deep Learning Techniques**

At the same time, artificial intelligence and machine learning are becoming more integrated in many aspects of social media. AI is far from replacing human touch in the field of social media, but it is increasing both the quantity and quality of online interactions between businesses and their customers. Businesses can use machine learning in the following ways to create effective social media marketing strategies:

### **1.1.1 SOCIAL MEDIA MONITORING**

Social media monitoring is one of the more traditional tools for businesses looking to manage their social media accounts. Some platforms like Twitter and Instagram have built-in analytics tools that can measure the success of past posts, including number of likes, comments, clicks on a link, or views for a video. Third-party tools like Iconosquare (\*for Instagram and Facebook) can also provide similar social media insight and management services. These tools can also tell businesses a lot about their audiences, including demographic information and the peak times when their followers are most active on the platform. Social media algorithms generally prioritize more recent posts over older posts, so with this data, businesses can strategically schedule their posts at or a few minutes before the peak times.

In the future, businesses might be able to rely on AI for recommendations about which users to message directly, or which posts to comment on, that could likely lead to increased sales. These recommendations would partly be based on the information gathered through existing analytics tools for social media monitoring.

### **1.1.2 SENTIMENT ANALYSIS FOR SOCIAL MEDIA MARKETING**

Sentiment analysis, also called opinion mining or emotion AI, is judging the opinion of a text. The process uses both natural language processing (NLP) and machine learning to pair social media data with predefined labels such as positive, negative, or neutral. Then, the machine can develop agents that learn to understand the sentiments underlying new messages.

## **Social Network Ads Sales Prediction System Using Deep Learning Techniques**

Businesses can apply sentiment analysis in social media and customer support to collect feedback on a new product or design. Similarly, businesses can apply sentiment analysis to discover how people feel about their competitors or trending industry topics.

### **1.1.3 IMAGE RECOGNITION FOR SOCIAL MEDIA MARKETING**

Image recognition uses machine learning to train computers to recognize a brand logo or photos of certain products, without any accompanying text. This can be useful for businesses when their customers upload photos of a product without directly mentioning the brand or product name in a text. Potential customers might also upload a photo of your product with a caption saying “Where can I buy this?” If businesses can notice when that happens, they can use it as an opportunity to send targeted promotions to that person, or simply comment on the post to say thank you for their purchase, which could certainly lead to increased customer loyalty.

In addition, the customer might feel encouraged to post more photos of your products in the future, which leads to further brand promotion. Businesses may benefit from paying close attention when people post photos of their products, because social media posts with images generally receive higher user engagement compared to posts that are purely text. Facebook users are 2.3 times more likely to like or comment on posts with images, and Twitter users are 1.5 times more likely to retweet a tweet with images. This is important for marketing because social media algorithms are usually designed so that posts with high engagement, measured by how many users interacted with a post such as by liking, commenting or sharing that post with other users, show up at the top of user feeds.

### **1.1.4 CHATBOTS FOR SOCIAL MEDIA MARKETING**

Chatbots are an application of AI that mimic real conversations. They can be embedded in websites such as online stores, or through a third-party messaging platform like Face book messenger, and Twitter and Instagram’s direct messaging.

## **Social Network Ads Sales Prediction System Using Deep Learning Techniques**

Chatbots allow businesses to automate customer service without requiring human interaction, unless the customer specifically asks to speak or chat with a human representative. For businesses with a generally young customer base, chatbots are more likely to increase customer satisfaction. 60% of millennials have used chatbots, and 70% of them reported positive experiences.

The use of chatbots is not limited to situations when the customer has a specific question or complaint. Estee Lauder uses a chatbot embedded in Facebook messenger that uses facial recognition to pick out the right shade of foundation for its customers, and Airbnb has used Amazon Alexa to welcome guests and introduce them to local attractions and restaurants

Artificial intelligence can be a powerful tool for businesses looking to get ahead in social marketing. Receiving feedback on how customers feel about different products and learning how customers spend their time on social media platforms are valuable regardless of industry. Businesses can use the applications introduced in this article to better understand and meet customer needs, and ultimately build stronger relationships with their customers.

### **1.2 OBJECTIVE**

- To analyze the relationship between advertising budget and sales.
- To build a linear regression model using the training data set.
- To make predictions of the sales of test data set using a linear regression model.

### **1.3 SCOPE OF THE STUDY**

Artificial intelligence can be a powerful tool for businesses looking to get ahead in social marketing. Receiving feedback on how customers feel about different products and learning how customers spend their time on social media platforms are valuable regardless of industry. Businesses can use the applications introduced in this article to better understand and meet customer needs, and ultimately build stronger relationships with their customers.

## **CHAPTER - 2**

### **LITERATURE SURVEY**

#### **2. LITERATURE SURVEY**

Supriya Verma (2016). The study is about the effect and behavior of people toward YouTube advertisements. Advertisements play a significant role in creating awareness among people about the product or service. Earlier advertisements were shown and displayed on television, radio, and newspaper, but today's digital media have replaced and conquered the old form of advertisements. In digital media, YouTube has become a common platform for advertisements. While watching YouTube, we often come across pre-roll advertisements that sometimes have an option to skip and mid-roll advertisements, which are usually between the content we are viewing on YouTube. For this research sample size is 100.

The questionnaire was mailed to 100 respondents (18-35 years). 56% of respondents watch YouTube daily, and some of them use mobile to watch YouTube. 79% of respondents come across pre-roll advertisements. According to the research, 15% of respondents strongly agree that YouTube videos help in enhancing knowledge, and 22% agree that advertising is beneficial to consumers because it provides important information about goods and services. 64% of respondents are moderately influenced by their buying behavior. 44% said that YouTube advertisements are most influential in their buying behavior. It is recommended to advertise online as it has an impact on their buying behavior.

Yuksel (2016), who investigated the effects of user-generated content in YouTube videos on consumers' purchase intention, found that perceived credibility, perceived usefulness, and perceived video characteristics of information in the YouTube videos positively affect purchase intention.

Lai and Chiang (2015) found that product placement and product involvement have a positive effect on purchase intention in the YouTube platform. In their study, Dehghani and Tumer (2015) determined that Facebook ads influenced the purchasing

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

intention by affecting brand value and brand image. Dehghani et al. (2015) found that the attitude toward YouTube ads influence positively purchasing intention.

S.NO	TITLE OF PAPER	AUTHOR	TECHNOLOGY	ADVANTAGE	DISADVANTAGE
1	Behavior of people toward YouTube advertisements	Supriya Verma	Deep Learning Technique	Promotion of Sales	Adds to Costs
2	Purchasing intention is a consumers	Hsu and Tsou	Machine Learning	Expansion of Production	Promotes Unnecessary Consumption
3	Generated content in YouTube videos	Yuksel	Deep Learning Technique	Large Turnover and Huge Profits	Leads to Price War
4	Facebook ads influenced the purchasing	Dehghani and Tumer	Deep Learning Technique	Information about Different Options and Comparative Prices	Unsafe Data
5	Attitude toward YouTube ads influence positively purchasing intention	Dehghani et al	Deep Learning Technique	Enhances Goodwill	Leads to Unequal Competition

Table 2.1 : Literature Survey

In the Comparison of various social media networks as discussed in the Literature Survey, the neural network model performance seems to be more effective and is of less disadvantages than other models. So, in this project work neural network is considered as a better existing system and due to this reason, it has been used as a reference during the comparison of performance to social media networks with high accuracy values.

the neural network model to initiate growth forecasting, which has the nature of "black box", increased calculations, prone to over fitting, and the empirical nature of model development. The amount of data required for normal operation of neural networks is very high, which may or may not be possible.



### **2.1 DISADVANTAGES OF EXISTING SYSTEM**

- ▶ **Hardware dependence:** Artificial neural networks require processors with parallel processing power, in accordance with their structure. For this reason, the realization of the equipment is dependent.
- ▶ **Unexplained behavior of the network:** This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network.
- ▶ **Determination of proper network structure:** There is no specific rule for determining the structure of artificial neural networks. Appropriate network structure is achieved through experience and trial and error.
- ▶ **Difficulty of showing the problem to the network:** ANNs can work with numerical information. Problems have to be translated into numerical values before being introduced to ANN. The display mechanism to be determined here will directly influence the performance of the network. This depends on the user's ability.
- ▶ **The duration of the network is unknown:** The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.

### **2.3 SUMMARY OF EXISTING SYSTEM**

This Literature Survey gives a brief study on evaluation of various social media networks and detailed study on various research issues on social media networks. The various media networks and different approaches on sales add detections have been analyzed. Social Network Ads Sales Prediction System is the most researched topic of neural networks. Although there are many other research topics that have been investigated in the literature, it is believed that this selected review has covered the most important aspects of neural networks in solving Social Network Ads Sales Prediction problems. Neural networks have been demonstrated to be a competitive alternative to traditional Social Network Ads Sales Prediction for many practical problems. However, while neural networks have shown much promise, many issues still remain unsolved or incompletely solved. Recently, several authors have pointed out the lack of the rigorous comparisons between neural network and other Social Network Ads Sales in the current literature . The Proposed linear regression algorithm Using Deep Learning Techniques is the efficient algorithm to address these issues in Social Network Ads Sales Prediction.

## **CHAPTER - 3**

### **SYSTEM ANALYSIS**

#### **3.1 PROBLEM STATEMENT**

Due to the short coming off neural network such as Hardware dependence, Unexplained behavior of the social media networks, Determination of proper network structure, Difficulty of showing the problem to the network, The duration of the network is unknown, it is quite difficult to produce accurate results from neural networks models.

Most of Social Network Ads Sales Prediction techniques have not Predict ads sales in social media with high accuracy. This raises a need to develop a linear regression algorithm Using Deep Learning Techniques which is able to improve the Social Network Ads Sales Prediction accuracy. Therefore, the problem statement of this project work is:

" To design two different non-neural network model and trying to find out the best one to overcome the short coming off neural network. We will be using linear regression algorithm Using Deep Learning Techniques which is very scalable and consumes lot less power."

#### **3.2 LINEAR REGRESSION ALGORITHM**

Linear regression is probably the simplest approach for statistical learning. It is a good starting point for more advanced approaches, and in fact, many fancy statistical learning techniques can be seen as an extension of linear regression. Therefore, understanding this simple model will build a good base before moving on to more complex approaches.

Linear regression is very good to answer the following questions:

- Is there a relationship between 2 variables?
- How strong is the relationship?

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

- Which variable contributes the most?
- How accurately can we estimate the effect of each variable?
- How accurately can we predict the target?
- Is the relationship linear? (duh)
- Is there an interaction effect?

### 3.2.1 ESTIMATING THE COEFFICIENTS

Let's assume we only have one variable and one target. Then, linear regression is expressed as:

$$Y = \beta_0 + \beta_1 X$$

Equation for a linear model with 1 variable and 1 target

In the equation above, the *betas* are the coefficients. These coefficients are what we need in order to make predictions with our model.

To find the parameters, we need to minimize the **least squares** or the **sum of squared errors**. Of course, the linear model is not perfect and it will not predict all the data accurately, meaning that there is a difference between the actual value and the prediction. The error is easily calculated with:

$$e_i = y_i - \hat{y}_i$$

Subtract the prediction from the true value

But why are the errors squared? We square the error, because the prediction can be either above or below the true value, resulting in a negative or positive difference respectively. If we did not square the errors, the sum of errors could decrease because of negative differences and not because the model is a good fit.

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

Also, squaring the errors penalizes large differences, and so the minimizing the squared errors “guarantees” a better model.

Let’s take a look at a graph to better understand.

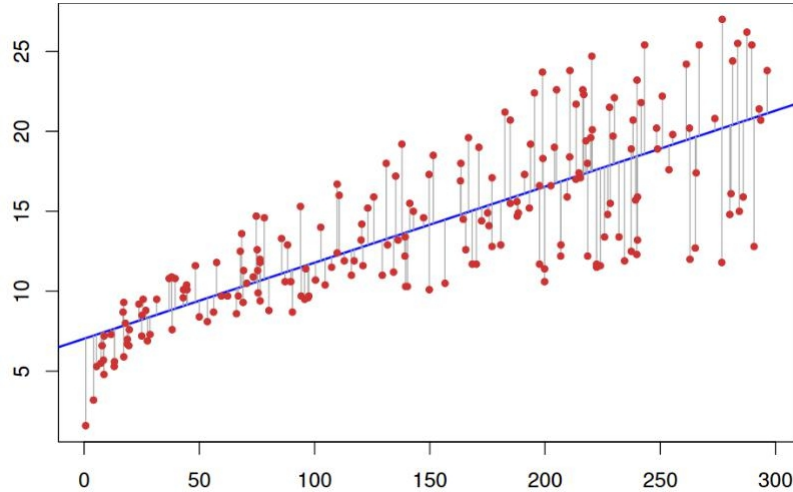


Figure 3.1 : Linear fit to a data set

In the graph above, the red dots are the true data and the blue line is linear model. The grey lines illustrate the errors between the predicted and the true values. The blue line is thus the one that minimizes the sum of the squared length of the grey lines.

After some math that is too heavy for this article, you can finally estimate the coefficients with the following equations:

$$\beta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

Where  $\bar{x}$  and  $\bar{y}$  represent the mean.

### 3.2.2 ESTIMATE THE RELEVANCY OF THE COEFFICIENTS

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

- Now that you have coefficients, how can you tell if they are relevant to predict your target?

The best way is to find the *p-value*. The *p-value* is used to quantify statistical significance; it allows to tell whether the null hypothesis is to be rejected or not.

- The null hypothesis?

For any modelling task, the hypothesis is that **there is some correlation** between the features and the target. The null hypothesis is therefore the opposite: **there is no correlation** between the features and the target.

So, finding the *p-value* for each coefficient will tell if the variable is statistically significant to predict the target. As a general rule of thumb, if the *p-value* is **less than 0.05**: there is a strong relationship between the variable and the target.

### 3.2.3 ASSESS THE ACCURACY OF THE MODEL

You found out that your variable was statistically significant by finding its *p-value*. Great!

- Now, how do you know if your linear model is any good?

To assess that, we usually use the RSE (residual standard error) and the  $R^2$  statistic.

$$RSE = \sqrt{\frac{1}{n-2}RSS} = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

RSE formula

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}, \quad TSS = \sum (y_i - \bar{y})^2$$

$R^2$  formula

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

The first error metric is simple to understand: the lower the residual errors, the better the model fits the data (in this case, the closer the data is to a linear relationship).

As for the  $R^2$  metric, it measures the **proportion of variability in the target that can be explained using a feature X**. Therefore, assuming a linear relationship, if feature X can explain (predict) the target, then the proportion is high and the  $R^2$  value will be close to 1. If the opposite is true, the  $R^2$  value is then closer to 0.

### 3.2.4 THE THEORY OF MULTIPLE LINEAR REGRESSION

In real life situations, there will never be a single feature to predict a target. So, do we perform linear regression on one feature at a time? Of course not. We simply perform multiple linear regression. The equation is very similar to simple linear regression; simply add the number of predictors and their corresponding coefficients:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

Multiple linear regression equation. **p** is the number of predictors

### 3.2.5 ASSESS THE RELEVANCY OF A PREDICTOR

Previously, in simple linear regression, we assess the relevancy of a feature by finding its *p-value*.

In the case of multiple linear regression, we use another metric: the F-statistic.

$$F = \frac{\frac{TSS - RSS}{p}}{\frac{RSS}{(n - p - 1)}}$$

F-statistic formula. **n** is the number of data points and **p** is the number of predictors

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

Here, the F-statistic is calculated for the overall model, whereas the *p-value* is specific to each predictor. If there is a strong relationship, then F will be much larger than 1. Otherwise, it will be approximately equal to 1.

➤ How *larger than 1* is large enough?

This is hard to answer. Usually, if there is a large number of data points, F could be slightly larger than 1 and suggest a strong relationship. For small data sets, then the F value must be way larger than 1 to suggest a strong relationship.

➤ Why can't we use the *p-value* in this case?

Since we are fitting many predictors, we need to consider a case where there are a lot of features (*p* is large). With a very large amount of predictors, there will always be about 5% of them that will have, by chance, a very small p-value **even though they are not statistically significant**. Therefore, we use the F-statistic to avoid considering unimportant predictors as significant predictors.

### 3.2.6 ASSESS THE ACCURACY OF THE MODEL

Just like in simple linear regression, the  $R^2$  can be used for multiple linear regression. However, know that adding more predictors will always increase the  $R^2$  value, because the model will necessarily better fit the training data.

Yet, this does not mean it will perform well on test data (making predictions for unknown data points).

### 3.2.7 ADDING INTERACTION

Having multiple predictors in a linear model means that some predictors may have an influence on other predictors.

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

For example, you want to predict the salary of a person, knowing her age and number of years spent in school. Of course, the older the person, the more time that person could have spent in school. So how do we model this interaction effect?

Consider this very simple example with 2 predictors:

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 \\ &= \beta_0 + \widetilde{\beta}_1 X_1 + \beta_2 X_2, \quad \widetilde{\beta}_1 = \beta_1 + \beta_3 X_2 \end{aligned}$$

Interaction effect in multiple linear regression

As you can see, we simply multiply both predictors together and associate a new coefficient. Simplifying the formula, we see now that the coefficient is influenced by the value of another feature.

As a general rule, if we include the interaction model, we should include the individual effect of a feature, even if its *p-value* is not significant. This is known as the **hierarchical principle**. The rationale behind this is that if two predictors are interacting, then including their individual contribution will have a small impact on the model.

### 3.3 SOFTWARE MODULES / LIBRARIES USED

#### 3.3.1 PYTHON PROGRAMMING LANGUAGE

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in July 2018. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural. It also has a comprehensive standard library. The Python interpreter and the



## **Social Network Ads Sales Prediction System Using Deep Learning Techniques**

extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.



Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a very-high-level language, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

- the high-level data types allow you to express complex operations in a single statement;
- statement grouping is done by indentation instead of beginning and ending brackets;
- no variable or argument declarations are necessary.

Python is extensible: if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

### 3.3.2 INVOKING THE INTERPRETER

The Python interpreter is usually installed as `/usr/local/bin/python3.7` on those machines where it is available; putting `/usr/local/bin` in your Unix shell's search path makes it possible to start it by typing the command:

```
python3.7
```

to the shell. [1] Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., `/usr/local/python` is a popular alternative location.)

On Windows machines, the Python installation is usually placed in `C:\Python37`, though you can change this when you're running the installer. To add this directory to your path, you can type the following command into the command prompt in a DOS box:

```
set path=%path%;C:\python37
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn't work, you can exit the interpreter by typing the following command: `quit()`.

The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support readline. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if `^P` is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is `python -c command [arg] ...`, which executes the statement(s) in command, analogous to the shell's `-c` option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using `python -m module [arg] ...`, which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing `-i` before the script.

### 3.3.3 ARGUMENT PASSING

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the `argv` variable in the `sys` module. You can access this list by executing `import sys`. The length of the list is at least one; when no script and no arguments are given, `sys.argv[0]` is an empty string. When the script name is given as `'-'` (meaning standard input), `sys.argv[0]` is set to `'-'`.

When `-c` command is used, `sys.argv[0]` is set to `'-c'`. When `-m` module is used, `sys.argv[0]` is set to the full name of the located module. Options found after `-c` command or `-m` module are not consumed by the Python interpreter's option processing but left in `sys.argv` for the command or module to handle.

### 3.3.4 INTERACTIVE MODE

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (`>>>`); for continuation lines it prompts with the secondary prompt, by default three dots (`...`). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
$ python3.7
Python 3.7 (default, Sep 16 2015, 09:25:04)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>>
>>>the_world_is_flat = True
>>> if the_world_is_flat:
...     print("Be careful not to fall off!")
... 
```

Be careful not to fall off!

In the following examples, input and output are distinguished by the presence or absence of prompts (`>>>` and `...`): to repeat the example, you must type everything after the prompt, when the prompt appears; lines that do not begin with a prompt are output from the interpreter. Note that a secondary prompt on a line by itself in an example means you must type a blank line; this is used to end a multi-line command.

Many of the examples in this manual, even those entered at the interactive prompt, include comments. Comments in Python start with the hash character, `#`, and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal. A hash character within a string literal is just a hash character. Since comments are to clarify code and are not interpreted by Python, they may be omitted when typing in examples.

### Some examples:

```
# this is the first comment
spam = 1 # and this is the second comment
# ... and now a third!
text = "# This is not a comment because it's inside quotes."
```

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or `EggsClass()`), and the

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

classes are instances of the metaclass type (itself an instance of itself), allowing metaprogramming and reflection.

Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long term plan is to support gradual typing<sup>[77]</sup> and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named mypy supports compile-time type checking.

Python has the usual C language arithmetic operators ( + , - , \* , / , % ). It also has \*\* for exponentiation, e.g.  $5**3 == 125$  and  $9**0.5 == 3.0$ , and a new matrix multiply @ operator is included in version 3.5.<sup>[80]</sup> Additionally, it has a unary operator (~), which essentially inverts all the bits of its one argument. For integers, this means  $\sim x = -x-1$ .<sup>[81]</sup> Other operators include bitwise shift operators  $x \ll y$ , which shifts x to the left y places, the same as  $x*(2**y)$ , and  $x \gg y$ , which shifts x to the right y places, the same as  $x/(2**y)$ .<sup>[82]</sup>

The behavior of division has changed significantly over time:<sup>[83][why?]</sup>

- Python 2.1 and earlier use the C division behavior. The / operator is integer division if both operands are integers, and floating-point division otherwise. Integer division rounds towards 0, e.g.  $7/3 == 2$  and  $-7/3 == -2$ .
- Python 2.2 changes integer division to round towards negative infinity, e.g.  $7/3 == 2$  and  $-7/3 == -3$ . The floor division // operator is introduced. So  $7//3$

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

$\text{== } 2$  ,  $-7//3 \text{ == } -3$  ,  $7.5//3 \text{ == } 2.0$  and  $-7.5//3 \text{ == } -3.0$  . Adding from `__future__`

`import division` causes a module to use Python 3.0 rules for division (see next).

- Python 3.0 changes `/` to be always floating-point division. In Python terms, the pre-3.0 `/` is classic division, the version-3.0 `/` is real division, and `//` is floor division.

Rounding towards negative infinity, though different from most languages, adds consistency. For instance, it means that the equation  $(a + b)//b \text{ == } a//b + 1$  is always true. It also means that the equation  $b*(a//b) + a\%b \text{ == } a$  is valid for both positive and negative values of  $a$  . However, maintaining the validity of this equation means that while the result of  $a\%b$  is, as expected, in the half-open interval  $[0, b)$ , where  $b$  is a positive integer, it has to lie in the interval  $(b, 0]$  when  $b$  is negative.

Python provides a `round` function for rounding a float to the nearest integer. For tie-breaking, versions before 3 use round-away-from-zero: `round(0.5)` is 1.0, `round(-0.5)` is -1.0. Python 3 uses round to even: `round(1.5)` is 2, `round(2.5)` is 2.

Python allows boolean expressions with multiple equality relations in a manner that is consistent with general use in mathematics. For example, the expression  $a < b < c$  tests whether  $a$  is less than  $b$  and  $b$  is less than  $c$  . C-derived languages interpret this expression differently: in C, the expression would first evaluate  $a < b$  , resulting in 0 or 1, and that result would then be compared with  $c$  .

Python has extensive built-in support for arbitrary precision arithmetic. Integers are transparently switched from the machine-supported maximum fixed-precision (usually 32 or 64 bits), belonging to the python type `int` , to arbitrary precision, belonging to the Python type `long` , where needed. The latter have an "L" suffix in their textual representation. (In Python 3, the distinction between the `int` and `long` types was

eliminated; this behavior is now entirely contained by the `int` class.) The `Decimal` type/class in module `decimal` (since version 2.4) provides decimal floating point numbers to arbitrary precision and several rounding modes. The `Fraction` type in module `fractions` (since version 2.6) provides arbitrary precision for rational numbers.

Due to Python's extensive mathematics library, and the third-party library NumPy that further extends the native capabilities, it is frequently used as a scientific scripting language to aid in problems such as numerical data processing and manipulation.

Rounding towards negative infinity, though different from most languages, adds consistency. For instance, it means that the equation  $(a + b) // b == a // b + 1$  is always true. It also means that the equation  $b * (a // b) + a \% b == a$  is valid for both positive and negative values of  $a$ . However, maintaining the validity of this equation means that while the result of  $a \% b$  is, as expected, in the half-open interval  $[0, b)$ , where  $b$  is a positive integer, it has to lie in the interval  $(b, 0]$  when  $b$  is negative.

The long term plan is to support gradual typing<sup>[77]</sup> and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named `mypy` supports compile-time type checking.

Python provides a `round` function for rounding a float to the nearest integer. For tie-breaking, versions before 3 use round-away-from-zero: `round(0.5)` is 1.0, `round(-0.5)` is -1.0. Python 3 uses round to even: `round(1.5)` is 2, `round(2.5)` is 2.

Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from `object` and are instances of type `object`). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.



### How to Install Python on Windows



Python doesn't come prepackaged with Windows, but that doesn't mean Windows users won't find the flexible programming language useful. It's not quite as simple as installing the newest version however, so let's make sure you get the right tools for the task at hand.

First released in 1991, Python is a popular high-level programming language used for general purpose programming. Thanks to a design philosophy that emphasizes readability it has long been a favorite of hobby coders and serious programmers alike. Not only is it an easy language (comparatively speaking, that is) to pick up but you'll find thousands of projects online that require you have Python installed to use the program.

#### ➤ Which Version Do You Need?

Unfortunately, there was a significant update to Python several years ago that created a big split between Python versions. This can make things a bit confusing to newcomers, but don't worry. We'll walk you through installing both major versions

When you visit the [Python for Windows download page](#), you'll immediately see the division. Right at the top, square and center, the repository asks if you want the latest release of Python 2 or Python 3 (2.7.13 and 3.6.1, respectively, as of this tutorial).



### ➤ **Newer is better, right?**

Maybe so, maybe not. The version you want depends on your end goal. That project is coded in Python and requires Python 2.7—you can't run the MCDungeon project with Python 3.6. In fact, if you're exploring hobby projects like MCDungeon, you'll find that nearly all of them use 2.7. If your goal is to get some project that ends in a ".py" extension up and running, then there's a very, *very* good chance you'll need 2.7 for it.

On the other hand, if you're looking to actually learn Python, we recommend installing both versions side by side (which you can do with zero risk and only a tiny bit of setup hassle). This lets you work with the newest version of the language, but also run older Python scripts (and test backwards compatibility for newer projects). Comparing the two versions is an article unto itself, though, so we'll defer to the Python project wiki where you can read their well written overview of the differences.

You can download just Python 2 or Python 3 if you're sure you only need a particular version. We're going the distance today and will be installing both of them, so we recommend you download both versions and do the same. Under the main entry for both versions you'll see an "x86-64" installer, as seen below.

- [Python 3.6.1 - 2017-03-21](#)
  - Download [Windows x86 web-based installer](#)
  - Download [Windows x86 executable installer](#)
  - Download [Windows x86 embeddable zip file](#)
  - Download [Windows x86-64 web-based installer](#)
  - Download [Windows x86-64 executable installer](#)
  - Download [Windows x86-64 embeddable zip file](#)
  - Download [Windows help file](#)

### **What's the Difference Between 32-bit and 64-bit Windows?**

This installer will install the appropriate 32-bit or 64-bit version on your computer automatically (here's [some further reading](#) if you want to know more about the differences between the two).

### 3.4 SCIKIT-LEARN

Defining scikit learn, it is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k - means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn was initially developed by David Cournapeau as a Google summer of code project in 2007. Later Matthieu Brucher joined the project and started to use it as a part of his thesis work. In 2010 INRIA got involved and the first public release (v0.1 beta) was published in late January 2010. The project now has more than 30 active contributors and has had paid sponsorship from INRIA, Google, Tinyclues and the Python Software Foundation.

In general, a learning problem considers a set of  $n$  samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and,

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or **features**.

Learning problems fall into a few categories:

supervised learning, in which the data comes with additional attributes that we want to predict (the scikit-learn supervised learning page). This problem can be either:

- classification: *samples* belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of a classification problem would be handwritten digit recognition, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the  $n$  samples provided, one is to try to label them with the correct category or class.
- regression: *if* the desired output consists of one or more continuous variables, then the task is called *regression*. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.

unsupervised learning, in which the training data consists of a set of input vectors  $x$  without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of *visualization* (Scikit-Learn unsupervised learning page).

### Training set and testing set

Machine learning is about learning some properties of a data set and then testing those properties against another data set. A common practice in machine learning is to evaluate an algorithm by splitting a data set into two. We call one of those sets the **training set**, on which we learn some properties; we call the other set the **testing set**, on which we test the learned properties.

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

### Loading an example dataset

scikit-learn comes with a few standard datasets, for instance the iris and digits datasets for classification and the boston house prices dataset for regression.

In the following, we start a Python interpreter from our shell and then load the iris and digits datasets. Our notational convention is that \$ denotes the shell prompt while >>> denotes the Python interpreter prompt:

```
$ python
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
```

A dataset is a dictionary-like object that holds all the data and some metadata about the data. This data is stored in the .data member, which is a n\_samples, n\_features array. In the case of supervised problem, one or more response variables are stored in the .target member. More details on the different datasets can be found in the dedicated section.

For instance, in the case of the digits dataset, digits.data gives access to the features that can be used to classify the digits samples:

```
>>> print(digits.data)
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]
```

and digits.target gives the ground truth for the digit dataset, that is the number corresponding to each digit image that we are trying to learn:

```
>>> digits.target
array([0, 1, 2, ..., 8, 9, 8])
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

### Shape of the data arrays

The data is always a 2D array, shape (n\_samples, n\_features), although the original data may have had a different shape. In the case of the digits, each original sample is an image of shape (8, 8) and can be accessed using:

```
>>>
```

```
>>>digits.images[0]
array([[ 0.,  0.,  5., 13.,  9.,  1.,   0.,   0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,   0.,   0.]])
```

The simple example on this dataset illustrates how starting from the original problem one can shape the data for consumption in scikit-learn.

### Loading from external datasets

To load from an external dataset, please refer to loading external datasets.

### Learning and predicting

In the case of the digits dataset, the task is to predict, given an image, which digit it represents. We are given samples of each of the 10 possible classes (the digits zero through nine) on which we *fit* an estimator to be able to *predict* the classes to which unseen samples belong.

In scikit-learn, an estimator for classification is a Python object that implements the methods `fit(X, y)` and `predict(T)`.

An example of an estimator is the class `sklearn.svm.SVC`, which implements support vector classification. The estimator's constructor takes as arguments the model's parameters.

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

For now, we will consider the estimator as a black box:

```
>>>
```

```
>>>fromsklearnimport svm  
>>>clf = svm.SVC(gamma=0.001, C=100.)
```

### Choosing the parameters of the model

In this example, we set the value of `gamma` manually. To find good values for these parameters, we can use tools such as grid search and cross validation.

The `clf` (for classifier) estimator instance is first fitted to the model; that is, it must *learn* from the model. This is done by passing our training set to the `fit` method. For the training set, we'll use all the images from our dataset, except for the last image, which we'll reserve for our predicting. We select the training set with the `[:-1]` Python syntax, which produces a new array that contains all but the last item from `digits.data`:

```
>>>
```

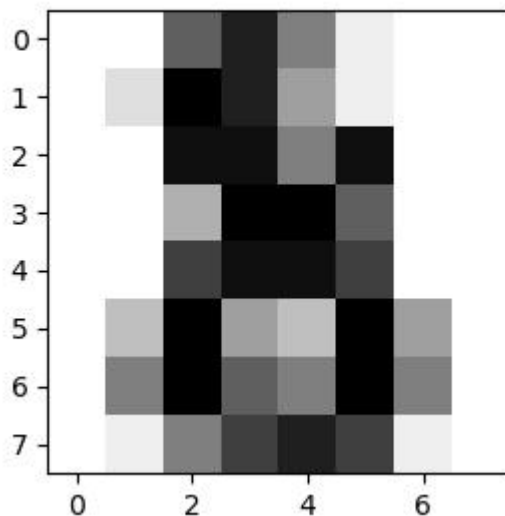
```
>>>clf.fit(digits.data[:-1], digits.target[:-1])  
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

Now you can *predict* new values. In this case, you'll predict using the last image from `digits.data`. By predicting, you'll determine the image from the training set that best matches the last image.

```
>>>
```

```
>>>clf.predict(digits.data[-1:])  
array([8])
```

The corresponding image is:



### 3.5 PANDAS

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

**pandas** is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure



## Social Network Ads Sales Prediction System Using Deep Learning Techniques

The two primary data structures of pandas, **Series** (1-dimensional) and **DataFrame** (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, **DataFrame** provides everything that R's `data.frame` provides and much more.

pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

- Easy handling of **missing data** (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be **inserted and deleted** from DataFrame and higher dimensional objects
- Automatic and explicit **data alignment**: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let *Series*, *DataFrame*, etc. automatically align the data for you in computations
- Powerful, flexible **group by** functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it **easy to convert** ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets
- Intuitive **merging** and **joining** data sets
- Flexible **reshaping** and pivoting of data sets
- **Hierarchical** labeling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from **flat files** (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast **HDF5 format**
- **Time series**-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. pandas is the ideal tool for all of these tasks.

### Some other notes

pandas is **fast**. Many of the low-level algorithmic bits have been extensively tweaked in Cythoncode. However, as with anything else generalization usually sacrifices performance. So if you focus on one feature for your application you may be able to create a faster specialized tool.

pandas is a dependency of statsmodels, making it an important part of the statistical computing ecosystem in Python.

- pandas has been used extensively in production in financial applications.

## 3.5 DATA STRUCTURES

Dimensions	Name	Description
1	Series	1D labeled homogeneously-typed array
2	DataFrame	General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column



### Why more than one data structure?

The best way to think about the pandas data structures is as flexible containers for lower dimensional data. For example, DataFrame is a container for Series, and Series is a container for scalars. We would like to be able to insert and remove objects from these containers in a dictionary-like fashion.

Also, we would like sensible default behaviors for the common API functions which take into account the typical orientation of time series and cross-sectional data sets. When using ndarrays to store 2- and 3-dimensional data, a burden is placed on the user to

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

consider the orientation of the data set when writing functions; axes are considered more or less equivalent (except when C- or Fortran-contiguousness matters for performance). In pandas, the axes are intended to lend more semantic meaning to the data; i.e., for a particular data set there is likely to be a “right” way to orient the data. The goal, then, is to reduce the amount of mental effort required to code up data transformations in downstream functions.

For example, with tabular data (DataFrame) it is more semantically helpful to think of the **index** (the rows) and the **columns** rather than axis 0 and axis 1. Iterating through the columns of the DataFrame thus results in more readable code:

```
for col in df.columns:

    series = df[col]

# do something with series
```

### Mutability and copying of data

All pandas data structures are value-mutable (the values they contain can be altered) but not always size-mutable. The length of a Series cannot be changed, but, for example, columns can be inserted into a DataFrame. However, the vast majority of methods produce new objects and leave the input data untouched. In general we like to **favor immutability** where sensible.



### INTRO TO DATA STRUCTURES

We'll start with a quick, non-comprehensive overview of the fundamental data structures in pandas to get you started. The fundamental behavior about data types, indexing, and axis labeling / alignment apply across all of the objects. To get started, import NumPy and load pandas into your namespace:

```
In [1]: import numpy as np
```

```
In [2]: import pandas as pd
```

Here is a basic tenet to keep in mind: **data alignment is intrinsic**. The link between labels and data will not be broken unless done so explicitly by you.

We'll give a brief intro to the data structures, then consider all of the broad categories of functionality and methods in separate sections.



### Series

**Series** is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the **index**. The basic method to create a Series is to call:

```
>>> s = pd.Series(data, index=index)
```

Here, data can be many different things:

- a Python dict
- an ndarray
- a scalar value (like 5)

The passed **index** is a list of axis labels. Thus, this separates into a few cases depending on what **data** is:



### From ndarray

If data is an ndarray, **index** must be the same length as **data**. If no index is passed, one will be created having values `[0, ..., len(data) - 1]`.

```
In [3]: s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])
```

```
In [4]: s
```

```
Out[4]:
```

```
a    0.469112
```

```
b -0.282863
c -1.509059
d -1.135632
e 1.212112
dtype: float64

In [5]: s.index

Out[5]: Index(['a', 'b', 'c', 'd', 'e'], dtype='object')

In [6]: pd.Series(np.random.randn(5))

Out[6]:

0 -0.173215
1 0.119209
2 -1.044236
3 -0.861849
4 -2.104569
dtype: float64
```

### Note

pandas supports non-unique index values. If an operation that does not support duplicate index values is attempted, an exception will be raised at that time. The reason for being lazy is nearly all performance-based (there are many instances in computations, like parts of GroupBy, where the index is not used).



### From dict

Series can be instantiated from dicts:

**Department of CSE, MREM**

```
In [7]: d = {'b': 1, 'a': 0, 'c': 2}
```

```
In [8]: pd.Series(d)
```

```
Out[8]: b    1
```

```
      a    c    2
```

```
dtype:
```

```
int64
```

### Note

When the data is a dict, and an index is not passed, the Series index will be ordered by the dict's insertion order, if you're using Python version  $\geq 3.6$  and Pandas version  $\geq 0.23$ .

If you're using Python  $< 3.6$  or Pandas  $< 0.23$ , and an index is not passed, the Series index will be the lexically ordered list of dict keys.

In the example above, if you were on a Python version lower than 3.6 or a Pandas version lower than 0.23, the Series would be ordered by the lexical order of the dict keys (i.e. ['a', 'b', 'c'] rather than ['b', 'a', 'c']).

If an index is passed, the values in data corresponding to the labels in the index will be pulled out.

```
In [9]: d = {'a': 0., 'b': 1., 'c': 2.}
```

```
In [10]: pd.Series(d)
```

```
Out[10]:
```

```
a    0.0
```

```
b    1.0
```

```
c 2.0
```

```
dtype: float64
```

```
In [11]: pd.Series(d, index=['b', 'c', 'd', 'a'])
```

```
Out[11]:
```

```
b 1.0
```

```
c 2.0
```

```
d NaN
```

```
a 0.0 dtype:
```

```
float64
```

### Note

NaN (not a number) is the standard missing data marker used in pandas.



#### From scalar value

If data is a scalar value, an index must be provided. The value will be repeated to match the length of **index**.

```
In [12]: pd.Series(5., index=['a', 'b', 'c', 'd', 'e'])
```

```
Out[12]:
```

```
a 5.0
```

```
b 5.0
```

```
c 5.0
```

```
d 5.0
```

```
e 5.0
```

```
dtype: float64
```

Series is ndarray-like

Series acts very similarly to a ndarray, and is a valid argument to most NumPy functions. However, operations such as slicing will also slice the index.

```
In [13]: s[0]
```

```
Out[13]: 0.46911229990718628
```

```
In [14]: s[:3]
```

```
Out[14]:
```

```
a  0.469112
```

```
b -0.282863
```

```
c -1.509059
```

```
dtype: float64
```

```
In [15]: s[s > s.median()]
```

```
Out[15]:
```

```
a  0.469112
```

```
e  1.212112
```

```
dtype: float64
```

```
In [16]: s[[4, 3, 1]]
```

```
Out[16]:
```

```
e  1.212112
```



```
d -1.135632
```

```
b -0.282863
```

```
dtype: float64
```

```
In [17]: np.exp(s)
```

```
Out[17]:
```

```
a 1.598575
```

```
b 0.753623
```

```
c 0.221118
```

```
d 0.321219
```

```
e 3.360575
```

```
dtype: float64
```

### Note

We will address array-based indexing like `s[[4, 3, 1]]` in section.

Like a NumPy array, a pandas Series has a **dtype**.

```
In [18]: s.dtype
```

```
Out[18]: dtype('float64')
```

This is often a NumPy dtype. However, pandas and 3rd-party libraries extend NumPy's type system in a few places, in which case the dtype would be a **ExtensionDtype**. Some examples within pandas are Categorical Data and Nullable Integer Data Type. See dtypes for more.

If you need the actual array backing a Series, use **Series.array**.

```
In [19]: s.array
```

```
Out[19]:
```

```
<PandasArray>
```

```
[ 0.46911229990718628, -0.28286334432866328, -1.5090585031735124,  
 -1.1356323710171934,  1.2121120250208506]
```

```
Length: 5, dtype: float64
```

Accessing the array can be useful when you need to do some operation without the index (to disable automatic alignment, for example).

**Series.array** will always be an **ExtensionArray**. Briefly, an **ExtensionArray** is a thin wrapper around one or more *concrete* arrays like a **numpy.ndarray**. Pandas knows how to take an **ExtensionArray** and store it in a **Series** or a column of a **DataFrame**. See **dtypes** for more.

While **Series** is **ndarray**-like, if you need an *actual* **ndarray**, then use **Series.to\_numpy()**.

```
In [20]: s.to_numpy()
```

```
Out[20]: array([ 0.4691, -0.2829, -1.5091, -1.1356,  1.2121])
```

Even if the **Series** is backed by a **ExtensionArray**, **Series.to\_numpy()** will return a **NumPy ndarray**.

Series is dict-like

A **Series** is like a fixed-size dict in that you can get and set values by index label:

```
In [21]: s['a']
```

```
Out[21]: 0.46911229990718628
```

```
In [22]: s['e'] =12.
```

```
In [23]: s
```

```
Out[23]:
```

```
a    0.469112
```

```
b   -0.282863
```

```
c   -1.509059
```

```
d   -1.135632
```

```
e   12.000000
```

```
dtype: float64
```

```
In [24]: 'e' in s
```

```
Out[24]: True
```

```
In [25]: 'f' in s
```

```
Out[25]: False
```

If a label is not contained, an exception is raised:

```
>>>s['f']
```

```
KeyError: 'f'
```

Using the get method, a missing label will return None or specified default:

```
In [26]: s.get('f')
```

```
In [27]: s.get('f', np.nan)
```

```
Out[27]: nan
```

See also the section on attribute access.

### Vectorized operations and label alignment with Series

When working with raw NumPy arrays, looping through value-by-value is usually not necessary. The same is true when working with Series in pandas. Series can also be passed into most NumPy methods expecting an ndarray.

```
In [28]: s + s
```

```
Out[28]:
```

```
a    0.938225
```

```
b   -0.565727
```

```
c   -3.018117
```

```
d   -2.271265
```

```
e   24.000000
```

```
dtype: float64
```

```
In [29]: s * 2
```

```
Out[29]:
```

```
a    0.938225
```

```
b   -0.565727
```

```
c   -3.018117
```

```
d   -2.271265
```

```
e 24.000000  
  
dtype: float64  
  
In [30]: np.exp(s)  
  
Out[30]:  
  
a 1.598575  
b 0.753623  
c 0.221118  
d 0.321219  
e 162754.79141  
  
9 dtype: float64
```

A key difference between Series and ndarray is that operations between Series automatically align the data based on label. Thus, you can write computations without giving consideration to whether the Series involved have the same labels.

```
In [31]: s[1:] + s[:-1]  
  
Out[31]:  
  
a NaN  
b -0.565727  
c -3.018117  
d -2.271265  
e NaN
```

```
dtype: float64
```

The result of an operation between unaligned Series will have the **union** of the indexes involved. If a label is not found in one Series or the other, the result will be marked as missing NaN. Being able to write code without doing any explicit data alignment grants immense freedom and flexibility in interactive data analysis and research. The integrated data alignment features of the pandas data structures set pandas apart from the majority of related tools for working with labeled data.

### Note

In general, we chose to make the default result of operations between differently indexed objects yield the **union** of the indexes in order to avoid loss of information. Having an index label, though the data is missing, is typically important information as part of a computation. You of course have the option of dropping labels with missing data via the **dropna** function.

Name attribute

Series can also have a name attribute:

```
In [32]: s = pd.Series(np.random.randn(5), name='something')
```

```
In [33]: s
```

```
Out[33]:
```

```
0   -0.494929
1    1.071804
2    0.721555
3   -0.706771
4   -1.039575
```

```
Name: something, dtype: float64
```

```
In [34]: s.name
```

```
Out[34]: 'something'
```

The Series name will be assigned automatically in many cases, in particular when taking 1D slices of DataFrame as you will see below.

*New in version 0.18.0.*

You can rename a Series with the **pandas.Series.rename()** method.

```
In [35]: s2 = s.rename("different")
```

```
In [36]: s2.name
```

```
Out[36]: 'different'
```

Note that `s` and `s2` refer to different objects.

### DataFrame

**DataFrame** is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object. Like Series, DataFrame accepts many different kinds of input:

- Dict of 1D ndarrays, lists, dicts, or Series
- 2-D numpy.ndarray
- Structured or record ndarray
- A Series
- Another DataFrame

Along with the data, you can optionally pass **index** (row labels) and **columns** (column labels) arguments. If you pass an index and / or columns, you are guaranteeing the index

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

and / or columns of the resulting DataFrame. Thus, a dict of Series plus a specific index will discard all data not matching up to the passed index.

If axis labels are not passed, they will be constructed from the input data based on common sense rules.

### Note

When the data is a dict, and columns is not specified, the DataFrame columns will be ordered by the dict's insertion order, if you are using Python version  $\geq 3.6$  and Pandas  $\geq 0.23$ .

If you are using Python  $< 3.6$  or Pandas  $< 0.23$ , and columns is not specified, the DataFrame columns will be the lexically ordered list of dict keys.

From dict of Series or dicts

The resulting **index** will be the **union** of the indexes of the various Series. If there are any nested dicts, these will first be converted to Series. If no columns are passed, the columns will be the ordered list of dict keys.

```
In [37]: d = {'one': pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
.....: 'two': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])} .....
```

```
In [38]: df = pd.DataFrame(d)
```

```
In [39]: df
```

```
Out[39]:
```

```
   one two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d NaN  4.0
```



```
In [40]: pd.DataFrame(d, index=['d', 'b', 'a'])
```

```
Out[40]:
```

```
one two
```

```
d NaN 4.0
```

```
b 2.0 2.0
```

```
a 1.0 1.0
```

```
In [41]: pd.DataFrame(d, index=['d', 'b', 'a'], columns=['two', 'three'])
```

```
Out[41]:
```

```
two three
```

```
d 4.0 NaN
```

```
b 2.0 NaN
```

```
a 1.0 NaN
```

The row and column labels can be accessed respectively by accessing the **index** and **columns** attributes:

### Note

When a particular set of columns is passed along with a dict of data, the passed columns override the keys in the dict.

```
In [42]: df.index
```

```
Out[42]: Index(['a', 'b', 'c', 'd'], dtype='object')
```

```
In [43]: df.columns
```

```
Out[43]: Index(['one', 'two'], dtype='object')
```

From dict of ndarrays / lists

The ndarrays must all be the same length. If an index is passed, it must clearly also be the same length as the arrays. If no index is passed, the result will be `range(n)`, where `n` is the array length.

```
In [44]: d = {'one': [1., 2., 3., 4.],
```

```
.....: 'two': [4., 3., 2., 1.]}
```

```
.....:
```

```
In [45]: pd.DataFrame(d)
```

```
Out[45]:
```

```
   one two
```

```
0  1.0  4.0
```

```
1  2.0  3.0
```

```
2  3.0  2.0
```

```
3  4.0  1.0
```

```
In [46]: pd.DataFrame(d, index=['a', 'b', 'c', 'd'])
```

```
Out[46]:
```

```
   one two
```

```
a  1.0  4.0
```

```
b  2.0  3.0
```

```
c  3.0  2.0
```

```
d  4.0  1.0
```

From structured or record array

This case is handled identically to a dict of arrays.

```
In [47]: data = np.zeros((2, ), dtype=[('A', 'i4'), ('B', 'f4'), ('C', 'a10')])
```

```
In [48]: data[:] = [(1, 2., 'Hello'), (2, 3., "World")]
```

```
In [49]: pd.DataFrame(data)
```

**Out[49]:**

```
   A   B   C
0 1 2.0 b'Hello'
1 2 3.0 b'World'
```

```
In [50]: pd.DataFrame(data, index=['first', 'second'])
```

**Out[50]:**

```
   A   B   C
first 1 2.0 b'Hello'
second 2 3.0 b'World'
```

```
In [51]: pd.DataFrame(data, columns=['C', 'A', 'B'])
```

**Out[51]:**

```
   C   A   B
0 b'Hello' 1 2.0
1 b'World' 2 3.0
```

**Note**

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

DataFrame is not intended to work exactly like a 2 -dimensional NumPy ndarray.

From a list of dicts

```
In [52]: data2 = [{ 'a': 1, 'b': 2}, { 'a': 5, 'b': 10, 'c': 20}]
```

```
In [53]: pd.DataFrame(data2)
```

Out[53]:

```
   a  b    c
0  1  2 NaN
1  5 10 20.0
```

```
In [54]: pd.DataFrame(data2, index=['first', 'second'])
```

Out[54]:

```
   a  b    c
first 1  2 NaN
second 5 10 20.0
```

```
In [55]: pd.DataFrame(data2, columns=['a', 'b'])
```

Out[55]:

```
   a  b
0  1  2
1  5 10
```

From a dict of tuples

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

You can automatically create a MultiIndexed frame by passing a tuples dictionary.

```
In [56]: pd.DataFrame({'a', 'b'): {'(A', 'B)': 1, ('A', 'C)': 2},
```

```
.....:      ('a', 'a'): {'(A', 'C)': 3, ('A', 'B)': 4},
```

```
.....:      ('a', 'c'): {'(A', 'B)': 5, ('A', 'C)': 6},
```

```
.....:      ('b', 'a'): {'(A', 'C)': 7, ('A', 'B)': 8},
```

```
.....:      ('b', 'b'): {'(A', 'D)': 9, ('A', 'B)': 10}})
```

```
.....:
```

```
Out[56]:
```

```
      a      b
```

```
      b  a  c  a  b
```

```
A B 1.0  4.0 5.0 8.0 10.0
```

```
C  2.0 3.0 6.0  7.0   NaN
```

```
D NaN  NaN  NaN NaN   9.0
```

### From a Series

The result will be a DataFrame with the same index as the input Series, and with one column whose name is the original name of the Series (only if no other column name provided).



### Missing Data

Much more will be said on this topic in the Missing data section. To construct a DataFrame with missing data, we use `np.nan` to represent missing values. Alternatively, you may pass a `numpy.MaskedArray` as the data argument to the DataFrame constructor, and its masked entries will be considered missing.

### Alternate Constructors



#### **DataFrame.from\_dict**

`DataFrame.from_dict` takes a dict of dicts or a dict of array-like sequences and returns a `DataFrame`. It operates like the `DataFrame` constructor except for the `orient` parameter which is 'columns' by default, but which can be set to 'index' in order to use the dict keys as row labels.

```
In [57]: pd.DataFrame.from_dict(dict([('A', [1, 2, 3]), ('B', [4, 5, 6]))])
```

**Out[57]:**

```
   A  B
0  1  4
1  2  5
2  3  6
```

If you pass `orient='index'`, the keys will be the row labels. In this case, you can also pass the desired column names:

```
In [58]: pd.DataFrame.from_dict(dict([('A', [1, 2, 3]), ('B', [4, 5, 6]))),
```

```
....:                                orient='index', columns=['one', 'two', 'three'])
```

```
....:
```

**Out[58]:**

```
   one two three
A    1   2     3
B    4   5     6
```



### **DataFrame.from\_records**

`DataFrame.from_records` takes a list of tuples or an ndarray with structured dtype. It works analogously to the normal `DataFrame` constructor, except that the resulting `DataFrame` index may be a specific field of the structured dtype. For example:

```
In [59]: data
```

```
Out[59]:
```

```
array([(1, 2., b'Hello'), (2, 3., b'World')],
```

```
      dtype=[('A', '<i4'), ('B', '<f4'), ('C', 'S10')])
```

```
In [60]: pd.DataFrame.from_records(data, index='C')
```

```
Out[60]:
```

```
      A  B  C
```

```
b'Hello' 1  2.0
```

```
b'World' 2  3.0
```

### Column selection, addition, deletion

You can treat a `DataFrame` semantically like a dict of like-indexed `Series` objects. Getting, setting, and deleting columns works with the same syntax as the analogous dict operations:

```
In [61]: df['one']
```

```
Out[61]:
```

```
a    1.0
```

```
b    2.0
```

```
c 3.0
```

```
d NaN
```

```
Name: one, dtype: float64
```

```
In [62]: df['three'] = df['one'] * df['two']
```

```
In [63]: df['flag'] = df['one'] > 2
```

```
In [64]: df
```

```
Out[64]:
```

```
   one two three  flag
a  1.0  1.0   1.0 False
b  2.0  2.0   4.0 False
c  3.0  3.0   9.0  True
d  NaN  4.0   NaN False
```

Columns can be deleted or popped like with a dict:

```
In [65]: del df['two']
```

```
In [66]: three = df.pop('three')
```

```
In [67]: df
```

```
Out[67]:
```

```
   one  flag
a  1.0 False
b  2.0 False
```



## Social Network Ads Sales Prediction System Using Deep Learning Techniques

```
c 3.0  True
d NaN  False
```

When inserting a scalar value, it will naturally be propagated to fill the column:

```
In [68]: df['foo'] = 'bar'
```

```
In [69]: df
```

```
Out[69]:
```

```
   one  flag  foo
a  1.0  False  bar
b  2.0  False  bar
c  3.0   True  bar
d  NaN  False  bar
```

When inserting a Series that does not have the same index as the DataFrame, it will be conformed to the DataFrame's index:

```
In [70]: df['one_trunc'] = df['one'][:2]
```

```
In [71]: df
```

```
Out[71]:
```

```
   one  flag  foo  one_trunc
a  1.0  False  bar         1.0
b  2.0  False  bar         2.0
c  3.0   True  bar         NaN
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

```
d NaN False bar NaN
```

You can insert raw ndarrays but their length must match the length of the DataFrame's index. By default, columns get inserted at the end. The insert function is available to insert at a particular location in the columns:

```
In [72]: df.insert(1, 'bar', df['one'])
```

```
In [73]: df
```

```
Out[73]:
```

```
   one bar  flag foo one_trunc
a 1.0 1.0 False bar      1.0
b 2.0 2.0 False bar      2.0
c 3.0 3.0  True bar      NaN
d NaN NaN False bar      NaN
```

### Assigning New Columns in Method Chains

Inspired by `dplyr`'s mutate verb, DataFrame has an **assign()** method that allows you to easily create new columns that are potentially derived from existing columns.

```
In [74]: iris = pd.read_csv('data/iris.data')
```

```
In [75]: iris.head()
```

```
Out[75]:
```

```
   SepalLength SepalWidth PetalLength PetalWidth      Name
0         5.1         3.5         1.4         0.2 Iris-setosa
1         4.9         3.0         1.4         0.2 Iris-setosa
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

```
2      4.7      3.2      1.3      0.2      Iris-setosa
3      4.6      3.1      1.5      0.2      Iris-setosa
4      5.0      3.6      1.4      0.2      Iris-setosa
```

```
In [76]: (iris.assign(sepal_ratio=iris['SepalWidth'] / iris['SepalLength'])
```

```
.....: .head())
```

```
.....:
```

```
Out[76]:
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Name	sepal_ratio
0	5.1	3.5	1.4	0.2	Iris-setosa	0.686275
1	4.9	3.0	1.4	0.2	Iris-setosa	0.612245
2	4.7	3.2	1.3	0.2	Iris-setosa	0.680851
3	4.6	3.1	1.5	0.2	Iris-setosa	0.673913
4	5.0	3.6	1.4	0.2	Iris-setosa	0.720000

In the example above, we inserted a precomputed value. We can also pass in a function of one argument to be evaluated on the DataFrame being assigned to.

```
In [77]: iris.assign(sepal_ratio=lambda x: (x['SepalWidth'] / x['SepalLength'])).head()
```

```
Out[77]:
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Name	sepal_ratio
0	5.1	3.5	1.4	0.2	Iris-setosa	0.686275
1	4.9	3.0	1.4	0.2	Iris-setosa	0.612245

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

2	4.7	3.2	1.3	0.2 Iris-setosa	0.680851
3	4.6	3.1	1.5	0.2 Iris-setosa	0.673913
4	5.0	3.6	1.4	0.2 Iris-setosa	0.720000

`assign` **always** returns a copy of the data, leaving the original DataFrame untouched. Passing a callable, as opposed to an actual value to be inserted, is useful when you don't have a reference to the DataFrame at hand. This is common when using `assign` in a chain of operations. For example, we can limit the DataFrame to just those observations with a Sepal Length greater than 5, calculate the ratio, and plot:

```
In [78]: (iris.query('SepalLength > 5'))
```

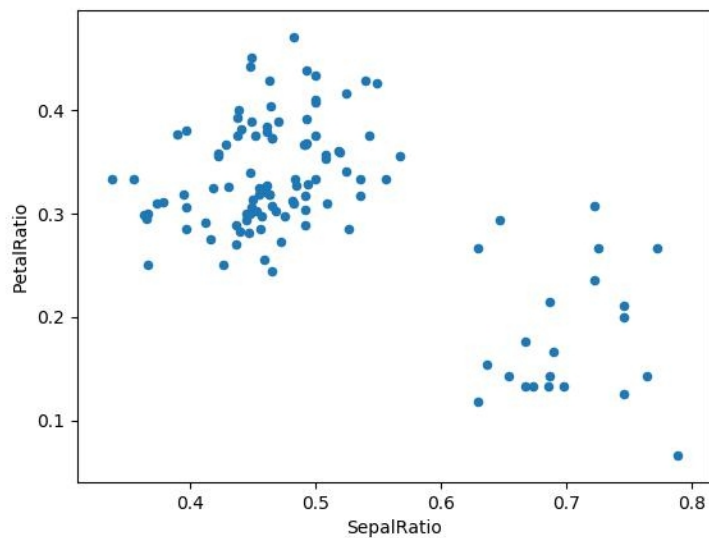
```
....: .assign(SepalRatio=lambda x: x.SepalWidth / x.SepalLength,
```

```
....:         PetalRatio=lambda x: x.PetalWidth / x.PetalLength)
```

```
....: .plot(kind='scatter', x='SepalRatio', y='PetalRatio'))
```

```
....:
```

```
Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2b527b1a58>
```



Since a function is passed in, the function is computed on the DataFrame being assigned to. Importantly, this is the DataFrame that's been filtered to those rows with sepal length greater than 5. The filtering happens first, and then the ratio calculations. This is an example where we didn't have a reference to the *filtered* DataFrame available.

The function signature for `assign` is simply `**kwargs`. The keys are the column names for the new fields, and the values are either a value to be inserted (for example, a Series or NumPy array), or a function of one argument to be called on the DataFrame. A *copy* of the original DataFrame is returned, with the new values inserted.

- **Changed in version 0.23.0.:** Starting with Python 3.6 the order of `**kwargs` is preserved. This allows for *dependent* assignment, where an expression later in `**kwargs` can refer to a column created earlier in the same `assign()`.

```
In [79]: dfa = pd.DataFrame({"A": [1, 2, 3],  
.....: "B": [4, 5, 6]})  
  
.....:  
  
In [80]: dfa.assign(C=lambda x: x['A'] + x['B'],  
.....:             D=lambda x: x['A'] + x['C'])
```

```
....:
```

```
Out[80]:
```

```
   ABC    D
0 145    6
1 257    9
2 369   12
```

In the second expression, `x['C']` will refer to the newly created column, that's equal to `dfa['A'] + dfa['B']`.

To write code compatible with all versions of Python, split the assignment in two.

```
In [81]: dependent = pd.DataFrame({"A": [1, 1, 1]})
```

```
In [82]: (dependent.assign(A=lambda x: x['A'] +1)
```

```
....: .assign(B=lambda x: x['A'] +2))
```

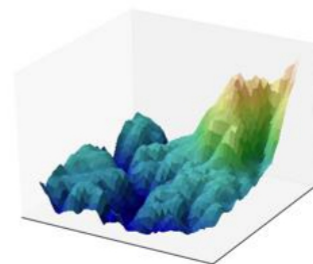
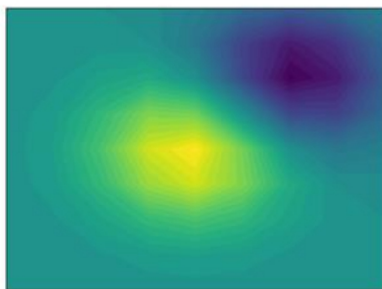
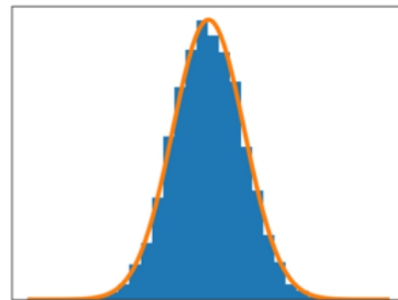
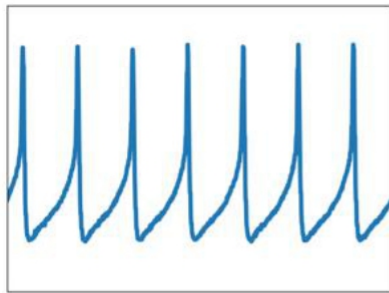
```
....:
```

```
Out[82]:
```

```
   A  B
0  2  4
1  2  4
2  4  4
```

### 3.6 MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.



Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

### ➤ **Installing an official release**

Matplotlib and its dependencies are available as wheel packages for macOS, Windows and Linux distributions:

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

Although not required, we suggest also installing IPython for interactive use. To easily install a complete Scientific Python stack, see Scientific Python Distributions below.

### ➤ **macOS**

To use the native OSX backend you will need a framework build of Python.

### ➤ **Test data**

The wheels (\*.whl) on the PyPI download page do not contain test data or example code.

If you want to try the many demos that come in the Matplotlib source distribution, download the \*.tar.gz file and look in the examples subdirectory.

To run the test suite:

- extract the lib/matplotlib/tests or lib/mpl\_toolkits/tests directories from the source distribution;
- install test dependencies: pytest, Pillow, MiKTeX, GhostScript, ffmpeg, avconv, ImageMagick, and Inkscape;
- run `python -m pytest`.

### 3.6.1 THIRD-PARTY DISTRIBUTIONS OF MATPLOTLIB

#### ➤ **Scientific Python Distributions**

Anaconda and Canopy and ActiveState are excellent choices that "just work" out of the box for Windows, macOS and common Linux platforms. WinPython is an option



## Social Network Ads Sales Prediction System Using Deep Learning Techniques

for Windows users. All of these distributions include Matplotlib and *lots* of other useful (data) science tools.

### ➤ **Linux: using your package manager**

If you are on Linux, you might prefer to use your package manager. Matplotlib is packaged for almost every major Linux distribution.

- Debian / Ubuntu: `sudo apt-get install python3-matplotlib`
- Fedora: `sudo dnf install python3-matplotlib`
- Red Hat: `sudo yum install python3-matplotlib`
- Arch: `sudo pacman -S python-matplotlib`

### ➤ **Installing from source**

If you are interested in contributing to Matplotlib development, running the latest source code, or just like to build everything yourself, it is not difficult to build Matplotlib from source. Grab the latest *tar.gz* release file from the PyPI files page, or if you want to develop Matplotlib or just need the latest bugfixed version, grab the latest git version. Install from source.

The standard environment variables `CC`, `CXX`, `PKG_CONFIG` are respected. This means you can set them if your toolchain is prefixed. This may be used for cross compiling.

```
export CC=x86_64-pc-linux-gnu-gcc
export CXX=x86_64-pc-linux-gnu-g++
export PKG_CONFIG=x86_64-pc-linux-gnu-pkg-config
```

Once you have satisfied the requirements detailed below (mainly Python, NumPy, libpng and FreeType), you can build Matplotlib.

```
cd matplotlib
python -mpip install .
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

We provide a `setup.cfg` file which you can use to customize the build process. For example, which default backend to use, whether some of the optional libraries that Matplotlib ships with are installed, and so on. This file will be particularly useful to those packaging Matplotlib.

If you have installed prerequisites to nonstandard places and need to inform Matplotlib where they are, edit `setuptools.py` and add the base dirs to the `basedir` dictionary entry for your `sys.platform`; e.g., if the header of some required library is in `/some/path/include/someheader.h`, put `/some/path` in the `basedir` list for your platform.



### Dependencies

Matplotlib requires the following dependencies:

- Python ( $\geq 3.5$ )
- FreeType ( $\geq 2.3$ )
- libpng ( $\geq 1.2$ )
- NumPy ( $\geq 1.10.0$ )
- setuptools
- cycler ( $\geq 0.10.0$ )
- dateutil ( $\geq 2.1$ )
- kiwisolver ( $\geq 1.0.0$ )
- pyparsing

Optionally, you can also install a number of packages to enable better user interface toolkits. See [What is a backend?](#) for more details on the optional Matplotlib backends and the capabilities they provide.

- tk ( $\geq 8.3$ ,  $\neq 8.6.0$  or  $8.6.1$ ): for the Tk-based backends;
- PyQt4 ( $\geq 4.6$ ) or PySide ( $\geq 1.0.3$ ): for the Qt4-based backends;
- PyQt5: for the Qt5-based backends;
- PyGObject or pgi ( $\geq 0.0.11.2$ ): for the GTK3-based backends;
- wxpython ( $\geq 4$ ): for the WX-based backends;
- cairocffi ( $\geq 0.8$ ) or pycairo: for the cairo-based backends;
- Tornado: for the WebAgg backend;

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

For better support of animation output format and image file formats, LaTeX, etc., you can install the following:

- ffmpeg/avconv: for saving movies;
- ImageMagick: for saving animated gifs;
- Pillow ( $\geq 3.4$ ): for a larger selection of image file formats: JPEG, BMP, and TIFF image files;
- LaTeX and GhostScript ( $\geq 9.0$ ) : for rendering text with LaTeX.



### Building on Linux

It is easiest to use your system package manager to install the dependencies.

If you are on Debian/Ubuntu, you can get all the dependencies required to build Matplotlib with:

```
sudo apt-get build-dep python-matplotlib
```

If you are on Fedora, you can get all the dependencies required to build Matplotlib with:

```
sudo dnf builddep python-matplotlib
```

If you are on RedHat, you can get all the dependencies required to build Matplotlib by first installing yum-builddep and then running:

```
su -c "yum-builddep python-matplotlib"
```

These commands do not build Matplotlib, but instead get and install the build dependencies, which will make building from source easier.



### Building on macOS

The build situation on macOS is complicated by the various places one can get the libpng and FreeType requirements (MacPorts, Fink, /usr/X11R6), the different architectures (e.g., x86, ppc, universal), and the different macOS versions (e.g., 10.4 and 10.5). We recommend that you build the way we do for the macOS release: get the source from the tarball or the git repository and install the required dependencies through a third -

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

party package manager. Two widely used package managers are Homebrew, and MacPorts. The following example illustrates how to install libpng and FreeType using brew:

```
brew install libpng freetype pkg-config
```

If you are using MacPorts, execute the following instead:

```
port install libpng freetype pkgconfig
```

After installing the above requirements, install Matplotlib from source by executing:

```
python -mpip install .
```

Note that your environment is somewhat important. Some conda users have found that, to run the tests, their PYTHONPATH must include `/path/to/anaconda/.../site-packages` and their DYLD\_FALLBACK\_LIBRARY\_PATH must include `/path/to/anaconda/lib`.



### Building on Windows

The Python shipped from <https://www.python.org> is compiled with Visual Studio 2015 for 3.5+. Python extensions should be compiled with the same compiler, see e.g. <https://packaging.python.org/guides/packaging-binary-extensions/#setting-up-a-build-environment-on-windows> for how to set up a build environment.

Since there is no canonical Windows package manager, the methods for building FreeType, zlib, and libpng from source code are documented as a build script at `matplotlib-winbuild`.

There are a few possibilities to build Matplotlib on Windows:

- Wheels via `matplotlib-winbuild`
- Wheels by using conda packages (see below)

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

- Conda packages (see below)

### Wheel builds using conda packages

This is a wheel build, but we use conda packages to get all the requirements. The binary requirements (png, FreeType,...) are statically linked and therefore not needed during the wheel install.

Set up the conda environment. Note, if you want a qt backend, add pyqt to the list of conda packages.

```
conda create -n "matplotlib_build" python=3.7 numpy python-dateutil pyparsing tornado  
cyclcr tk libpng zlib freetype msinttypes  
  
conda activate matplotlib_build
```

For building, call the script `build_alllocal.cmd` in the root folder of the repository:

```
build_alllocal.cmd
```



### General Concepts

matplotlib has an extensive codebase that can be daunting to many new users. However, most of matplotlib can be understood with a fairly simple conceptual framework and knowledge of a few important points.

Plotting requires action on a range of levels, from the most general (e.g., 'contour this 2-D array') to the most specific (e.g., 'color this screen pixel red'). The purpose of a plotting package is to assist you in visualizing your data as easily as possible, with all the necessary control -- that is, by using relatively high-level commands most of the time, and still have the ability to use the low-level commands when needed.

Therefore, everything in matplotlib is organized in a hierarchy. At the top of the hierarchy is the matplotlib "state-machine environment" which is provided by the matplotlib.pyplot module. At this level, simple functions are used to add plot elements (lines, images, text, etc.) to the current axes in the current figure.

### Note

Pyplot's state-machine environment behaves similarly to MATLAB and should be most familiar to users with MATLAB experience.

The next level down in the hierarchy is the first level of the object-oriented interface, in which pyplot is used only for a few functions such as figure creation, and the user explicitly creates and keeps track of the figure and axes objects. At this level, the user uses pyplot to create figures, and through those figures, one or more axes objects can be created. These axes objects are then used for most plotting actions.

For even more control -- which is essential for things like embedding matplotlib plots in GUI applications -- the pyplot level may be dropped completely, leaving a purely object-oriented approach.

```
# sphinx_gallery_thumbnail_number = 3
import matplotlib.pyplot as plt
import numpy as np
```



### Parts of a Figure

The **whole** figure. The figure keeps track of all the child Axes, a smattering of 'special' artists (titles, figure legends, etc), and the **canvas**. (Don't worry too much about the canvas, it is crucial as it is the object that actually does the drawing to get you your plot, but as the user it is more-or-less invisible to you). A figure can have any number of Axes, but to be useful should have at least one.

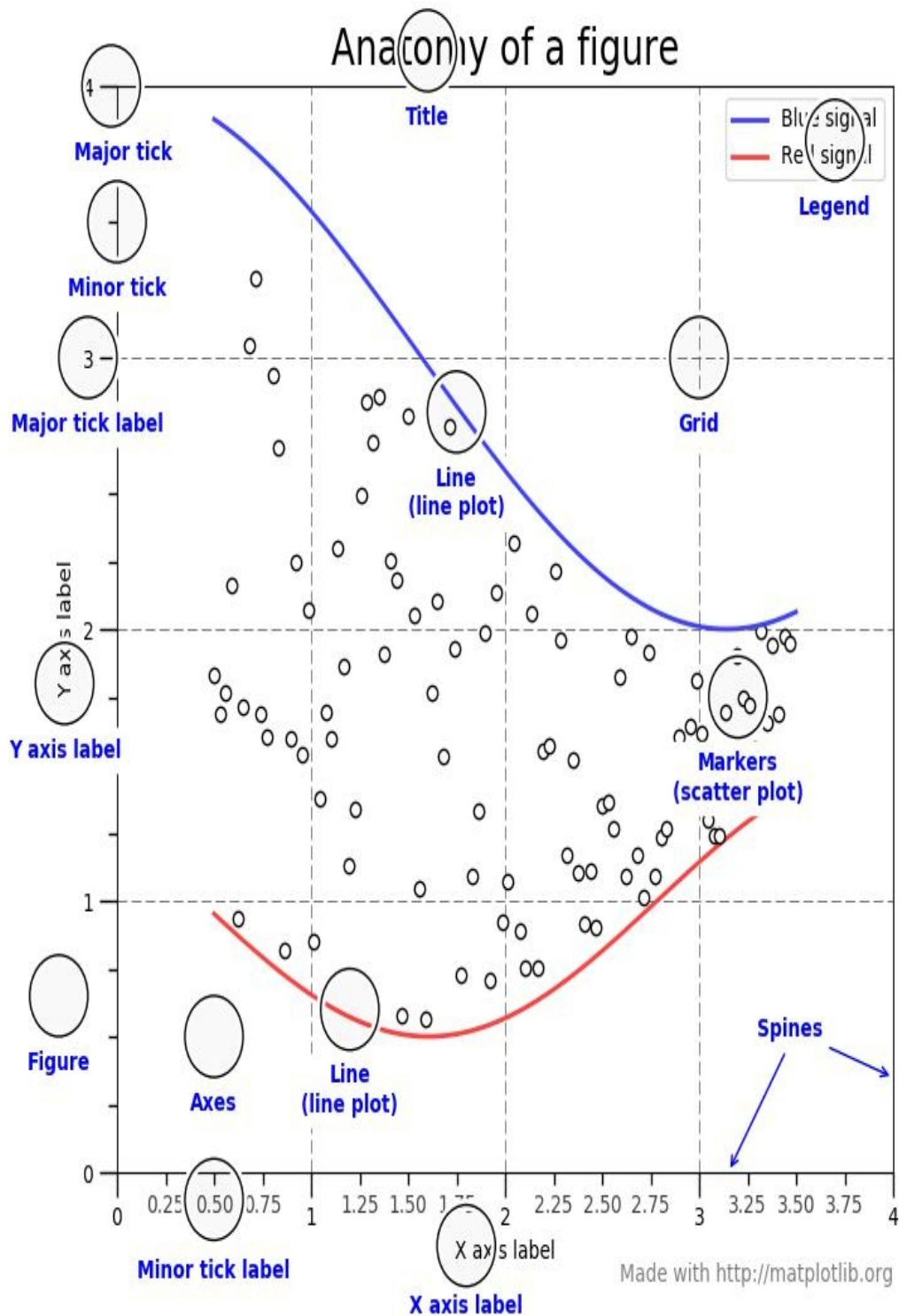


Figure : track of all the child Axes, a smattering of 'special' artists and the **canvas**

## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 SYSTEM DESIGN : PROPOSED SYSTEM FLOW CHART

A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields

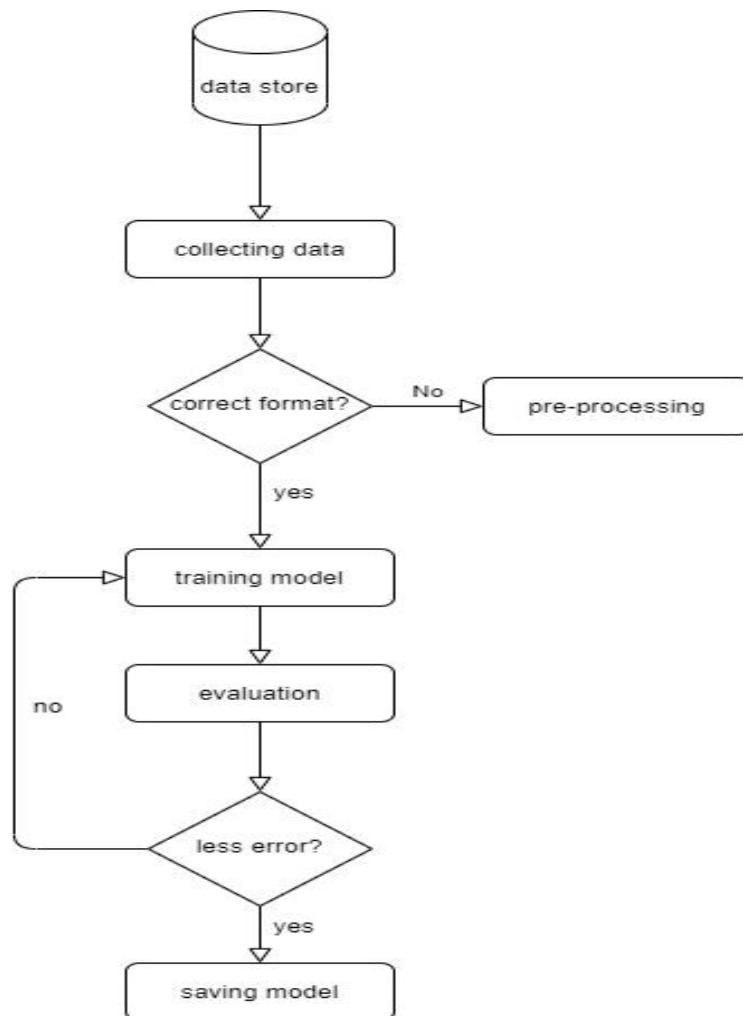


Figure 4.1 : Proposed System Flow Chart



### 4.2 PROPOSED SYSTEM DATA FLOW DIAGRAM

#### ➤ DFD Symbols

There are **four basic symbols** that are used to represent a data-flow diagram.

#### ➤ Process

A process receives input data and produces output with a different content or form.

Processes can be as simple as collecting input data and saving in the database, or it can be complex as producing a report containing monthly sales of all retail stores in the northwest region.

Every process has a name that identifies the function it performs.

The name consists of a verb, followed by a singular noun.

Example:

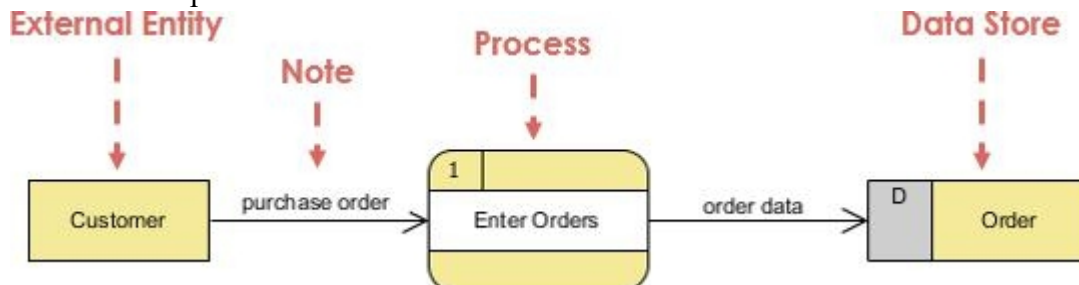
- Apply Payment
- Calculate Commission
- Verify Order

#### ➤ Notation

- A rounded rectangle represents a process
- Processes are given IDs for easy referencing



Process Example



## Social Network Ads Sales Prediction System Using Deep Learning Techniques

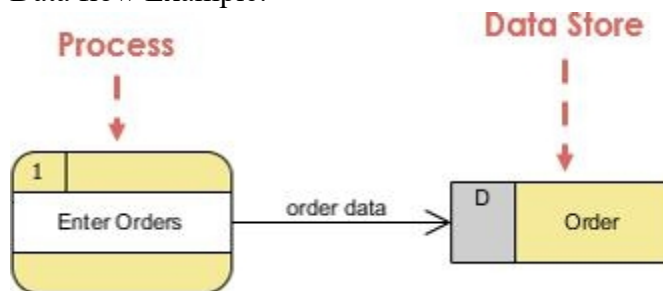


### Data Flow

A data-flow is a path for data to move from one part of the information system to another. A data-flow may represent a single data element such the Customer ID or it can represent a set of data element (or a data structure). Example:

- Customer\_info (LastName, FirstName, SS#, Tel #, etc.)
- Order\_info (OrderId, Item#, OrderDate, CustomerID, etc.).

Data flow Example:



Notation

- Straight lines with incoming arrows are input data flow
- Straight lines with outgoing arrows are output data flows

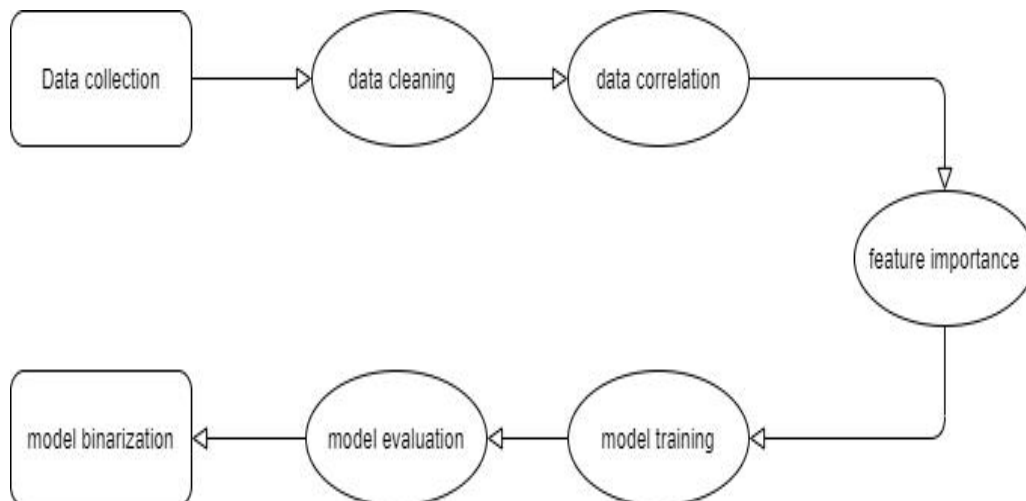


Figure 4.2 : Proposed System Data Flow Diagram

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

### 4.1.1 Proposed System Class Diagram

Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.

Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.

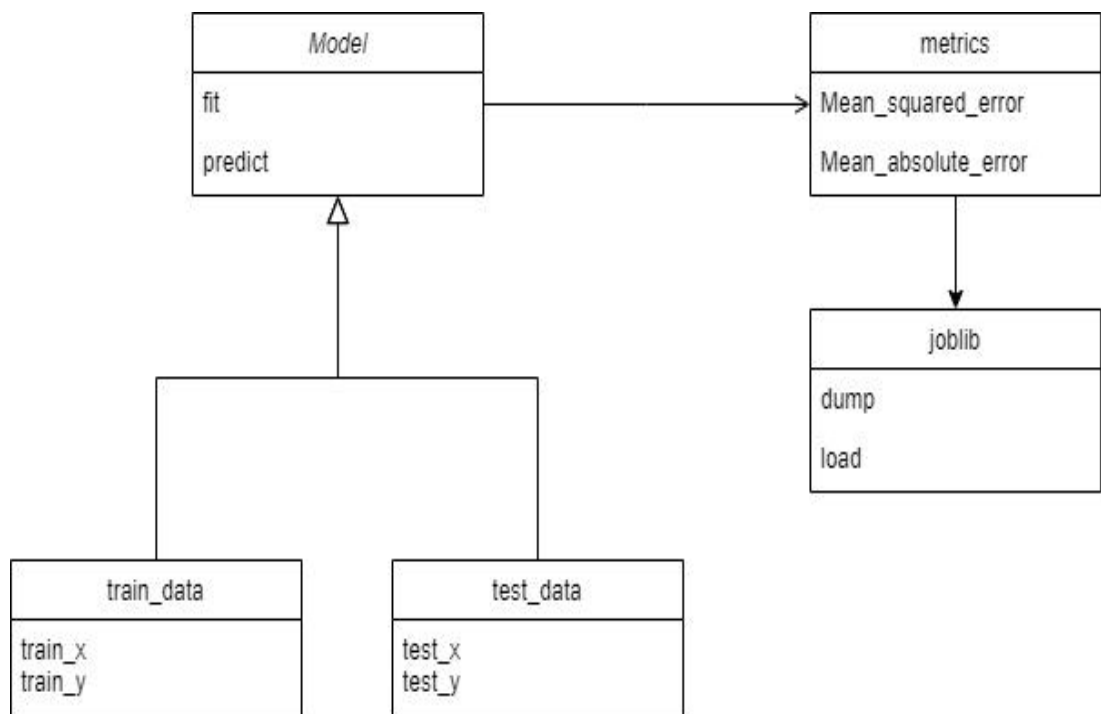


Figure 4.3 : Proposed System Class Diagram

## CHAPTER 5

### SOFTWARE AND HARDWARE REQUIREMENTS

#### 5.1 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. In this system study the following software requirements used.

- OS: Windows XP + / Linux/ MacOS
- Python 3
- Scikit-learn
- Pandas

#### 5.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. In this system study the following hardware requirements used.

Component	Minimum requirement
Processor	64-bit, i3 processor, 2.5 GHz minimum per core
RAM	8 GB for developer or evaluation use 16 GB for single server and multiple server farm installation for production use
Hard disk	120 GB

## CHAPTER 6

### CODING

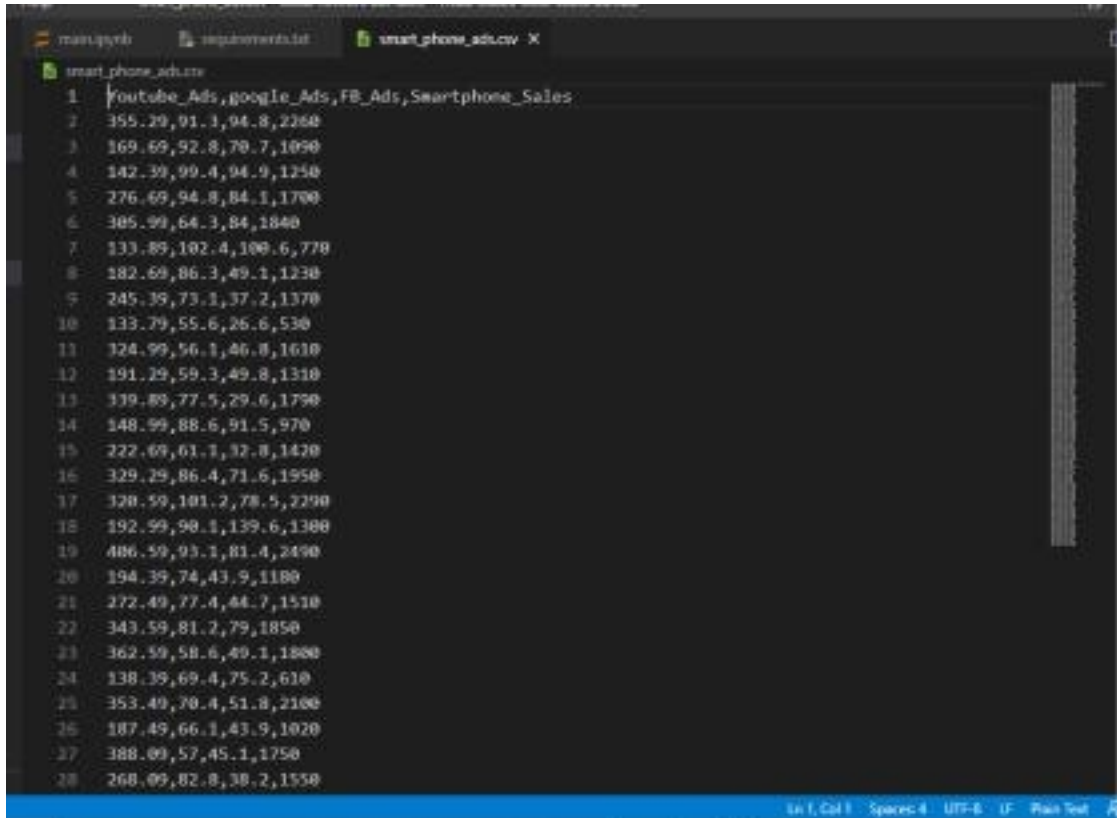
#### 6.1 INPUT DATA SETS

Youtube_Ads																			
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Youtube	google	FB_Ads	Smartphone_Sales																
355.29	91.3	94.8	2260																
169.69	92.8	70.7	1090																
142.39	99.4	94.9	1250																
276.69	94.8	84.1	1700																
305.99	64.3	84	1840																
133.89	102.4	100.6	770																
182.69	86.3	49.1	1230																
245.39	73.1	37.2	1370																
133.79	55.6	26.6	530																
324.99	56.1	46.8	1610																
191.29	59.3	49.8	1310																
339.89	77.5	29.6	1790																
148.99	88.6	91.5	970																
222.69	61.1	32.8	1420																
329.29	86.4	71.6	1950																
320.59	101.2	78.5	2290																
192.99	90.1	139.6	1300																
406.59	93.1	81.4	2490																
194.39	74	43.9	1180																
272.49	77.4	44.7	1510																
343.59	81.2	79	1850																
362.59	58.6	49.1	1800																

Figure 6.1 : Input Data Sets for Sales Ads

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

➤ Sales Data :  
Smartphoneads.csv



	Youtube_Ads	google_Ads	FB_Ads	Smartphone_Sales
1	355.29	91.1	94.8	2260
2	169.69	92.8	70.7	1090
3	142.39	99.4	94.9	1250
4	276.69	94.8	84.1	1700
5	305.99	64.3	84	1840
6	133.89	102.4	100.6	770
7	182.69	86.3	49.1	1230
8	245.39	73.1	37.2	1370
9	133.79	55.6	26.6	530
10	124.99	56.1	46.8	1610
11	191.29	59.3	49.8	1310
12	139.89	77.5	29.6	1790
13	148.99	88.6	91.5	970
14	227.69	61.7	37.8	1470
15	329.29	86.4	71.6	1950
16	120.59	101.2	78.5	2290
17	192.99	90.1	139.6	1300
18	406.59	93.1	81.4	2490
19	194.39	74.43	9	1180
20	272.49	77.4	44.7	1510
21	343.59	81.2	79	1850
22	362.59	58.6	40.1	1800
23	138.39	69.4	75.2	610
24	353.49	70.4	51.8	2100
25	187.49	66.1	43.9	1020
26	388.09	57	45.1	1750
27	268.09	82.8	30.2	1550

Figure 6.2 : Sales Data

### 6.2 PROPOSED SYSTEM SOURCE CODE:

```
#!/usr/bin/env python
# coding: utf-8
# # Loading Libraries
# In[1]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
# # Loading data and analyzing
# In[2]:
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

```
data = pd.read_csv('smart_phone_ads.csv')
data.head()
# In[3]:
data.info()
# ## Looking at the data correlation
# In[4]:
plt.figure(figsize = (10,8))
sns.heatmap(data.corr(), annot = True, cmap = "coolwarm")
plt.title("----Correlation----", size = 23, color = "r")
plt.show()
# ## Frequecydistribution(Histogram)
# In[5]:
#plotting All columns by using Histogram
data.hist(figsize = (15,8), color = "g")
plt.tight_layout()
plt.show()
# ## scatter plot
# In[6]:
sns.pairplot(data )
plt.show()
# # spliting data in training and testing set
# In[7]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data["Youtube_Ads"],
data["Smartphone_Sales"])
# # Using linear Regression to predict the sales
# In[8]:
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train.values.reshape(-1, 1), y_train)
# In[9]:
predict = reg.predict(x_test.values.reshape(-1,1))
# # visualizing the predicted results
# In[10]:
sns.distplot(predict,bins=5)
plt.title("sales histogram", size=20)
plt.show()
# # comparing with the actual values
# In[12]:
plt.scatter(x_test,y_test)
plt.plot(x_test,predict,'r')
plt.xlabel("youtube_ads")
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

```
plt.ylabel("Sales")
plt.title("Predicted vs actual values")
plt.show()
# # calculating the error
# In[12]:
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, predict))
```



### Loading

**Libraries** In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```



### Loading data and

**analyzing** In [2]:

```
data = pd.read_csv('smart_phone_ads.csv')
data.head()
Out[2]:
```

	Youtube_Ads	google_Ads	FB_Ads	Smartphone_Sales
0	355.29	91.3	94.8	2260
1	169.69	92.8	70.7	1090
2	142.39	99.4	94.9	1250
3	276.69	94.8	84.1	1700
4	305.99	64.3	84.0	1840

In [3]:

```
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
```



## Social Network Ads Sales Prediction System Using Deep Learning Techniques

```
Data columns (total 4 columns):
# Column      Non-Null Count  Dtype
---  -
0 Youtube_Ads    200 non-null   float64
1 google_Ads     200 non-null   float64
2 FB_Ads         200 non-null   float64
3 Smartphone_Sales 200 non-null   int64
dtypes: float64(3), int64(1)
memory usage: 6.4 KB
```



### Looking at the data

**correlation** In [4]:

```
plt.figure(figsize = (10,8))
sns.heatmap(data.corr(), annot = True, cmap = "coolwarm")
plt.title("----Correlation----", size = 23, color = "r")
plt.show()
```



### Frequency distribution(Histogram)

In [5]:

```
# plotting All columns by using
Histogram data.hist(figsize = (15,8), color
= "g") plt.tight_layout()
plt.show() C:\Users\Bablu\anaconda3\lib\site-
packages\pandas\plotting\_matplotlib\tools.py:331:
```



### MatplotlibDeprecationWarning:

The `is_first_col` function was deprecated in Matplotlib 3.4 and will be removed two minor releases later. Use `ax.get_subplotspec().is_first_col()` instead.

if `ax.is_first_col()`:



```
scatter plot
In [6]:

sns.pairplot(data )

plt.show()
```

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

splitting data in training and testing set

In [7]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data["Youtube_Ads"],
data["Smartphone_Sales"])
```



**Using linear Regression to predict the sales**

In [8]:**from** sklearn.linear\_model **import** LinearRegression

```
reg = LinearRegression()
reg.fit(x_train.values.reshape(-1, 1), y_train)
Out[8]:LinearRegression()In [9]:

predict = reg.predict(x_test.values.reshape(-1,1))
```



**visualizing the predicted results**

In [10]:

```
sns.distplot(predict,bins=5)
plt.title("sales histogram", size=20) plt.show() C:\Users\Bablu\anaconda3\lib\site-
packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
```

```
warnings.warn(msg, FutureWarning)
```



**comparing with the actual values** In [12]:

```
plt.scatter(x_test,y_test)
plt.plot(x_test,predict,'r')
plt.xlabel("youtube_ads")
plt.ylabel("Sales") plt.title("Predicted
vs actual values")
plt.show()
```

**calculating the error**

In [12]:

```
from sklearn.metrics import mean_squared_error

print(mean_squared_error(y_test, predict))
45165.10961129835

In [ ]:
```

## **CHAPTER 7**

### **TESTING**

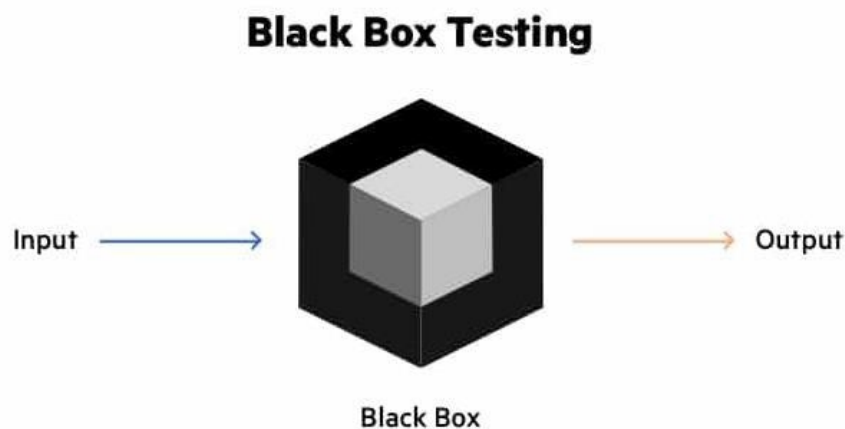
#### **7.1 PROPOSED SYSTEM TESTING**

##### **7.1.1 BLACK BOX TESTING**

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

An example of a security technology that performs black box testing is Dynamic Application Security Testing (DAST), which tests products in staging or production and provides feedback on compliance and security issues.





### Black Box Testing Pros and Cons

	Pros	Cons
1.	Testers do not require technical knowledge, programming or IT skills	Difficult to automate
2.	Testers do not need to learn implementation details of the system	Requires prioritization, typically infeasible to test all user paths
3.	Tests can be executed by crowdsourced or outsourced testers	Difficult to calculate test coverage
4.	Low chance of false positives	If a test fails, it can be difficult to understand the root cause of the issue
5.	Tests have lower complexity, since they simply model common user behavior	Tests may be conducted at low scale or on a non-production-like environment

### 7.1.2 Types Of Black Box Testing

Black box testing can be applied to three main types of tests: functional, non-functional, and regression testing.

### i) Functional Testing

Black box testing can test specific functions or features of the software under test. For example, checking that it is possible to log in using correct user credentials, and not possible to log in using wrong credentials.

Functional testing can focus on the most critical aspects of the software (smoke testing/sanity testing), on integration between key components (integration testing), or on the system as a whole (system testing).

### ii) Non-Functional Testing

Black box testing can check additional aspects of the software, beyond features and functionality. A non-functional test does not check “if” the software can perform a specific action but “how” it performs that action.

Black box tests can uncover if software is:

Usable and easy to understand for its users

- Performant under expected or peak loads
- Compatible with relevant devices, screen sizes, browsers or operating systems
- Exposed to security vulnerabilities or common security threats

### 7.1.3 Black Box Testing Techniques

**i) Equivalence Partitioning:** Testers can divide possible inputs into groups or “partitions”, and test only one example input from each group. For example, if a system requires a user’s birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the “under 18” group and one date in the “over 18” group.

**ii) Boundary Value Analysis :** Testers can identify that a system has a special response around a specific boundary value. For example, a specific field may accept only values between 0 and 99. Testers can focus on the boundary values (-1, 0, 99 and 100), to see if the system is accepting and rejecting inputs correctly.

### **iii) Decision Table Testing**

Many systems provide outputs based on a set of conditions. Testers can then identify “rules” which are a combination of conditions, identify the outcome of each rule, and design a test case for each rule.

For example, a health insurance company may provide different premium based on the age of the insured person (under 40 or over 40) and whether they are a smoker or not. This generates a decision table with four rules and up to four outcomes—below is an example with three possible outcomes.

## **7.2 WHITE BOX TESTING**

White-box testing's basic procedures require the tester to have an in-depth knowledge of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created. The following are the three basic steps that white-box testing takes in order to create test cases:

1. Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code and security specifications This is the preparation stage of white-box testing to lay out all of the basic information.
2. Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
3. Output involves preparing final report that encompasses all of the above preparations and result

## CHAPTER 8

### OUTPUT SCREENS

#### 8.1 PROPOSED SYSTEM OUTPUT SCREENS

##### ➤ INTERMEDIATE RESULTS:

Finding the Correlation of the Data:

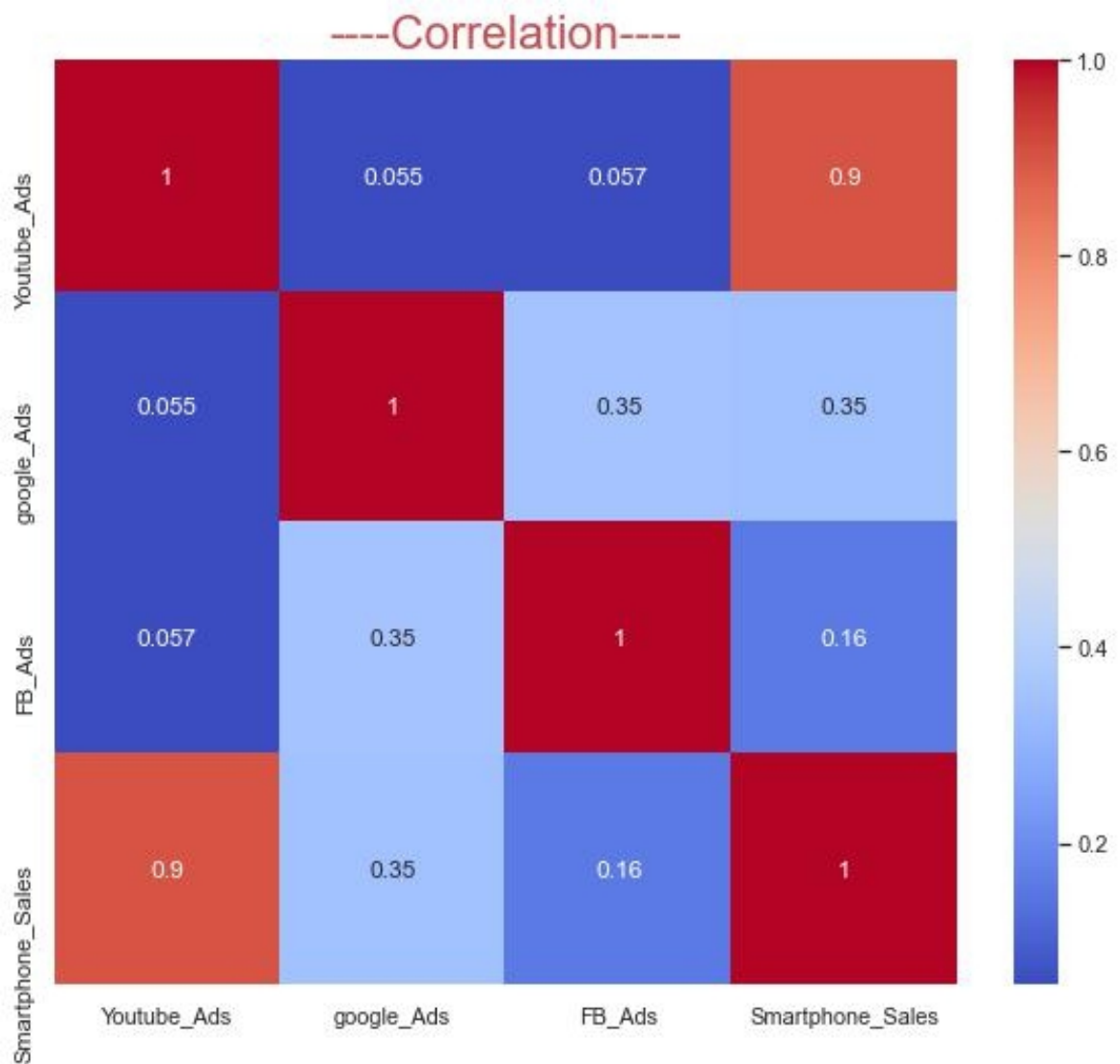


Figure 8.1 : Correlation of the Sales Data

## Social Network Ads Sales Prediction System Using Deep Learning Techniques

### ➤ HISTOGRAM:

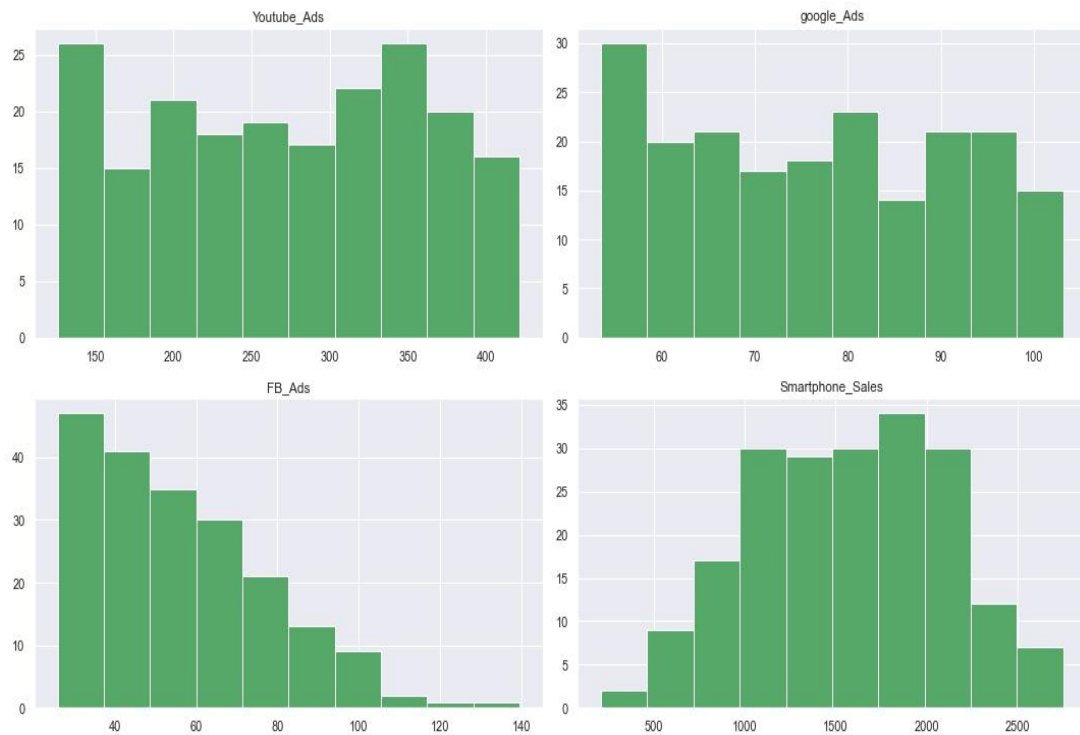
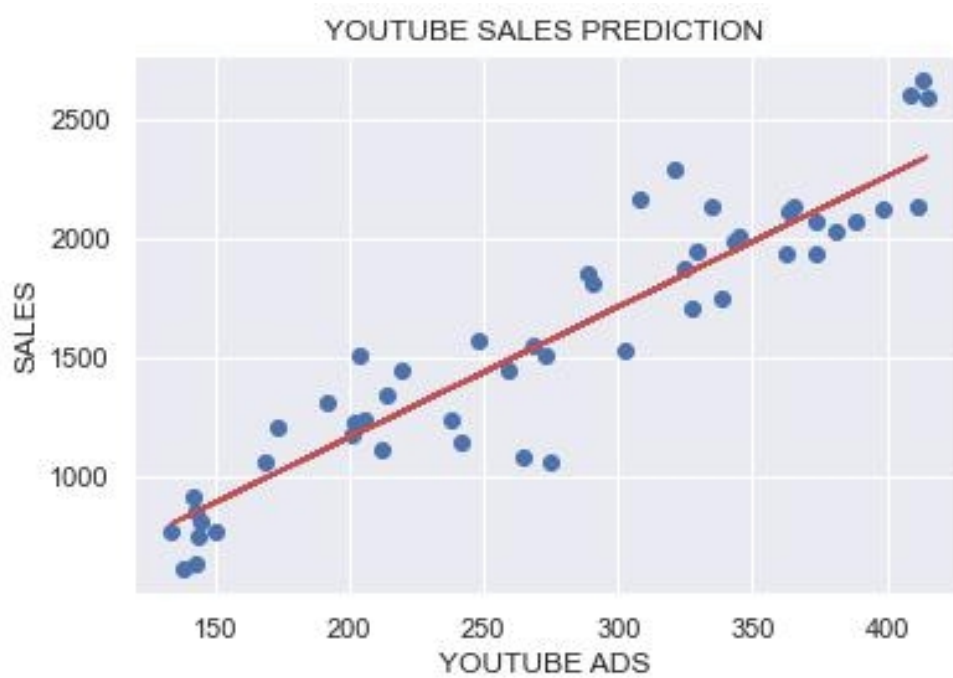


Figure 8.2: Final Sales Prediction

### ➤ YOUTUBE PREDICTION:





## Social Network Ads Sales Prediction System Using Deep Learning Techniques

Figure 8.3: Youtube Ads Vs. Sales

Predictions GOOGLE ADS PREDICTION:

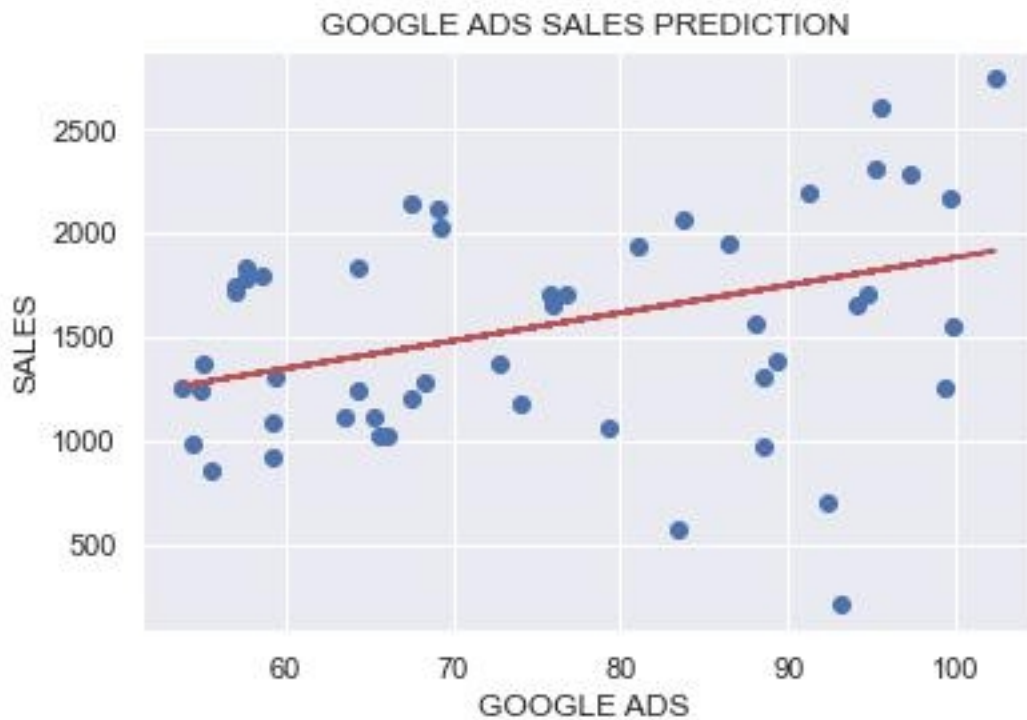


Figure 8.4 : Google Ads Vs. Sales Predictions

FACEBOOKS ADS PREDICTION:

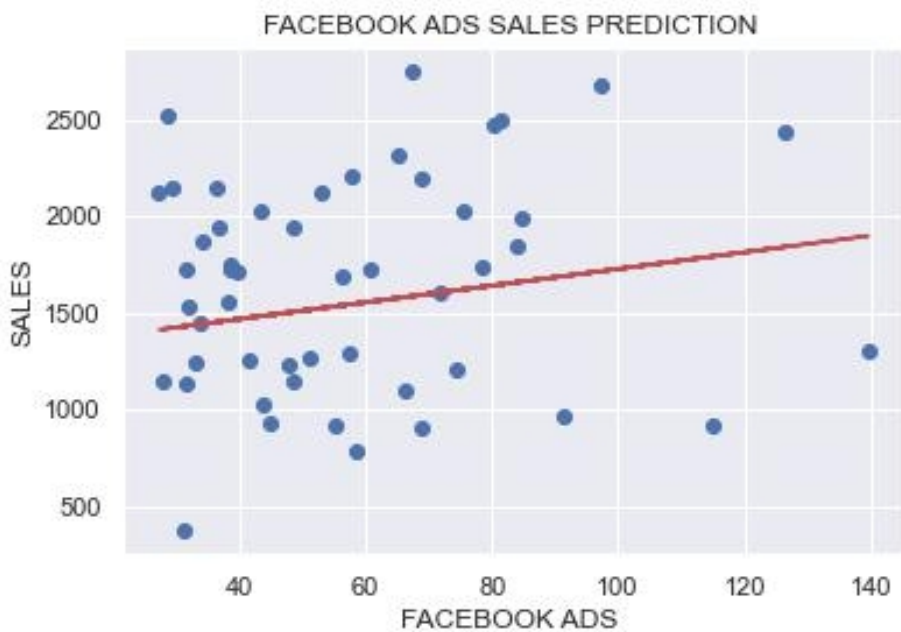


Figure 8.5 : Facebook Ads Vs. Sales Predictions

➤ **best prediction is youtube prediction**

From the above prediction the best prediction is occurred when we use the youtube prediction So youtube prediction is the final prediction of the model.

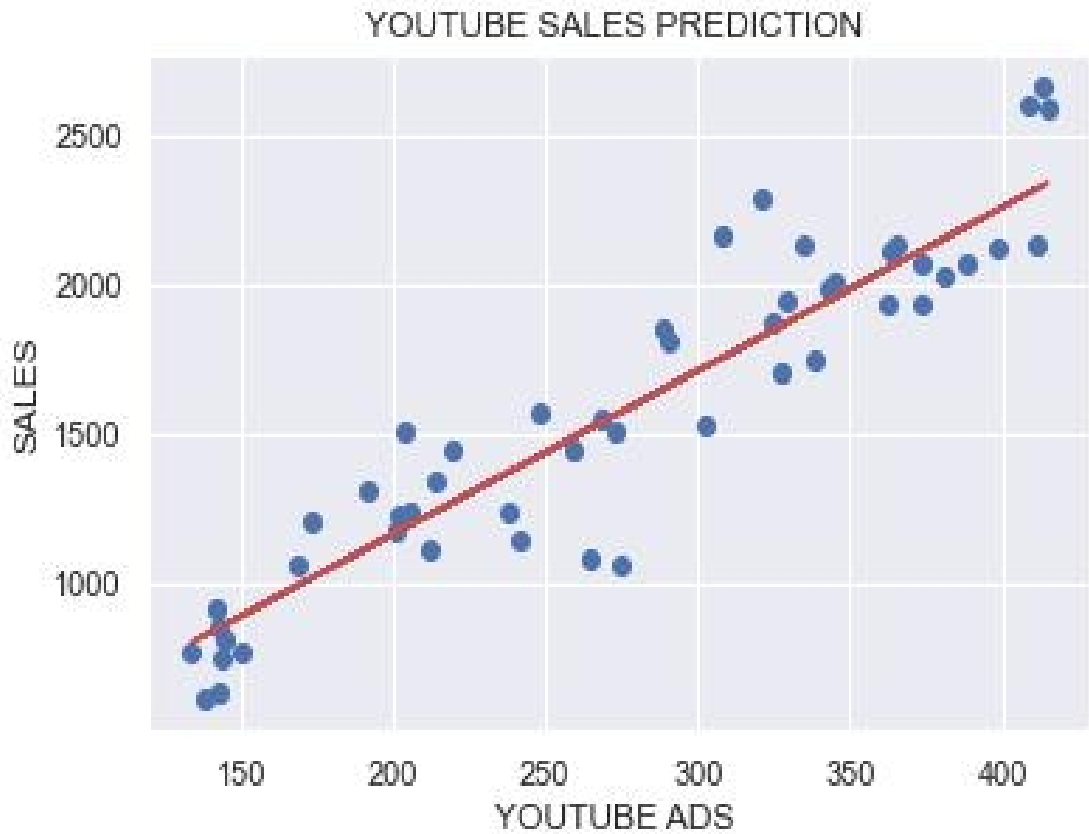


Figure 8.6 : Best Ads Vs. Sales Predictions in Youtube

## **CHAPTER 9**

### **CONCLUSIONS**

The linear regression algorithm Using Deep Learning Techniques were proposed, The implementation of the proposed methods was done by using python open-source software program

For experimentation, the **Input Data Sets for Sales Ads** given in the UCI machine learning repository such as, Facebook Ads and Sales data sets, youtube Ads and Sales data sets , google Ads and Sales data sets etc. was utilized to social network ads and sales prediction, From the above predictions by using different data sets youtube prediction is to be considered as the best prediction from result analysis by using proposed method i.e. linear regression algorithm Using Deep Learning Techniques.

This result impacts or improves the sales factor, profitability and consumer's factors. It is fact that the brands of company are viewed worldwide and also match with other similar brands of various companies

## **CHAPTER 10**

### **FURTHER ENHANCEMENTS**

➤ The proposed work can be extended in future by improving social network ads and sales prediction based on multiple data sets at a time, so that users can monitor their ads and sales prediction more accurately.

➤ Linear regression algorithm Using Deep Learning Techniques may be guaranteed as to provide high accuracy and not to allow the social media dataset loss. proposed technique can be enhanced by considering more number of multiple data sets at a time for their ads and sales prediction.

➤ Linear regression techniques can be implemented in future in the various fields like the stock marketing, bitcoin value prediction and the many other varies business, this technique may be applied to multiple regression and logistic regression. the future scope can be very much accurate and error can be reduced and model can be made more robust.

**CHAPTER 11**

**REFERENCES**



**REFERENCES**

1. Afuah, A., and Tucci, C. L. 2003. “A model of the Internet as creative destroyer,” *IEEE Transactions on Engineering Management* (50:4), pp. 395–402.
2. Agarwal, R., and Dhar, V. 2014. “Editorial—Big data, data science, and analytics: The opportunity and challenge for IS research” *Information Systems Research* (25:3), pp. 443-448.
3. Aldrich, H. 1999. *Organizations evolving*: Sage.
4. Alvarez, S. A., and Barney, J. B. 2007. “Discovery and creation: Alternative theories of entrepreneurial action,” *Strategic Entrepreneurship Journal* (1:1-2), pp. 11–26.
5. Archak, N., Ghose, A., and Ipeirotis, P. G. 2011. “Deriving the pricing power of product features by mining consumer reviews,” *Management Science* (57:8), pp. 1485–1509.
6. Armstrong, J. S. 2001. *Principles of forecasting: a handbook for researchers and practitioners*: Springer Science & Business Media.
7. Atanasov, P. D., Rescober, P., Stone, E., Swift, S. A., Servan-Schreiber, E., Tetlock, P. E., Ungar, L., and Mellers, B. 2017. “Distilling the wisdom of crowds: Prediction markets versus prediction polls,” *Management science* (63:3), pp. 691–706.
8. Attenberg, J., Ipeirotis, P., and Provost, F. 2015. “Beat the Machine: Challenging Humans to Find a Predictive Model's “Unknown Unknowns”,” *Journal of Data and Information Quality (JDIQ)* (6:1), pp. 1-15.