

LOAN APPROVAL STATUS

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

MAHENDRA DUTTA

BY

SUBHANKAR PRAMANIK



FUTURE INSTITUTE OF ENGINEERING

AND

MANAGEMENT

In association with



SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

1.	Title of the Project:	LOAN APPROVAL STATUS
2.	Project Members:	SUBHANKAR PRAMANIK
3.	Name of the guide:	Mr. MAHENDRA DUTTA
4.	Address:	Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified) SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal, 700091

Project Version Control History

Version	Primary Author	Description of Version	Date Completed
Final	SUBHANKAR PRAMANIK	Project Report	31 ST DECEMBER, 2021

Subhankar Pramanik.

Signature of Team Member

Date: 31/12/2021

For Office Use Only

Mahendra Dutta.

Signature of Approver

Date: 31/12/2021

**MR. MAHENDRA
DUTTA**

Project Proposal Evaluator

✓
Approved

Not Approved

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled “***LOAN APPROVAL STATUS***” in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT.**

LTD, SALT LAKE, KOLKATA, WEST BENGAL, is an authentic work carried out under the guidance of **MR. MAHENDRA DUTTA.** The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 31/12/2021

Name of the Students: SUBHANKAR PRAMANIK

Signature of the students:

Subhankar Pramanik.



Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled “**LOAN APPROVAL STATUS**” is a record of bonafide work, carried out by **SUBHANKAR PRAMANIK** under my guidance at **ARDENTCOMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®**. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor



MR. MAHENDRA DUTTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. MAHENDRA DUTTA***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

- Overview
- History of Python
- Environment Setup
- Basic Syntax
- Variable Types
- Functions
- Modules
- Packages
- Artificial Intelligence
 - Deep Learning
 - Neural Networks
 - Machine Learning
- Machine Learning
 - Supervised and Unsupervised Learning
 - NumPy
 - SciPy
 - Scikit-learn
 - Pandas
 - Regression Analysis
 - Classification Analysis
 - Matplotlib
 - Clustering
- **LOAN APPROVAL STATUS**

OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic datatypes and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

*If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ('Hello, Python!');***

However in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The keywords are **some predefined and reserved words in python that have special meanings**. Keywords are used to define the syntax of the coding. The keyword cannot be used as an identifier, function, and variable name. All the keywords in python are written in lower case except True and False. There are 33 keywords in python 3.7.

Keywords in Python				
All keywords has special meaning and purpose in python. Here is list of all keywords in python:				
False	None	True	and	as
assert	break	class	continue	def
del	elif	else	except	finally
for	from	global	if	import
in	is	lambda	nonlocal	not
or	pass	raise	return	try
while	with	yield		

tutorialstonight.com @tutorialstoni8

Lines & Indentation

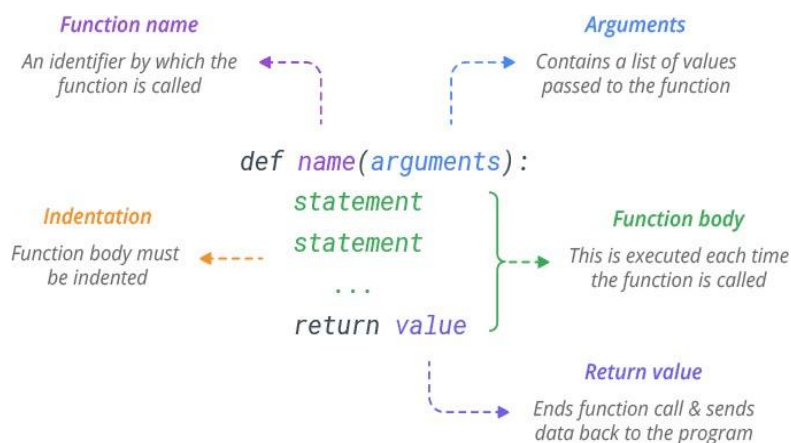
Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
print "True"
else:
print "False"
```

FUNCTIONS

Defining a Function



MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named

aname.py. Here's an example of a simple module, *support.py*

```
def print_func( par ):
    print "Hello : ", par
    return
```

The import Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
Import module1 [, module2 [... moduleN]
```

PACKAGES

A package is basically **a directory with Python files and a file with the name `__init__.py`**. This means that every directory inside of the Python path, which contains a file named `__init__.py`, will be treated as a package by Python. It's possible to put several modules into a Package.

Top Python Packages

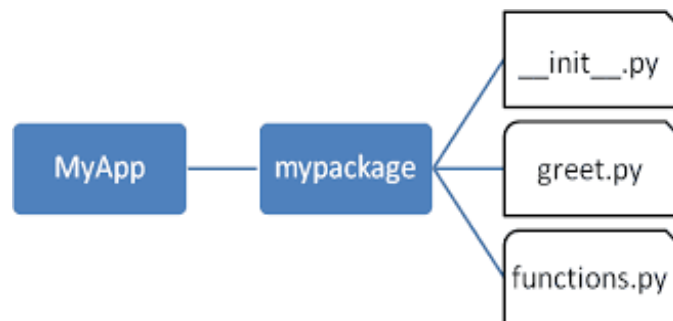
NumPy.

pandas.

Matplotlib.

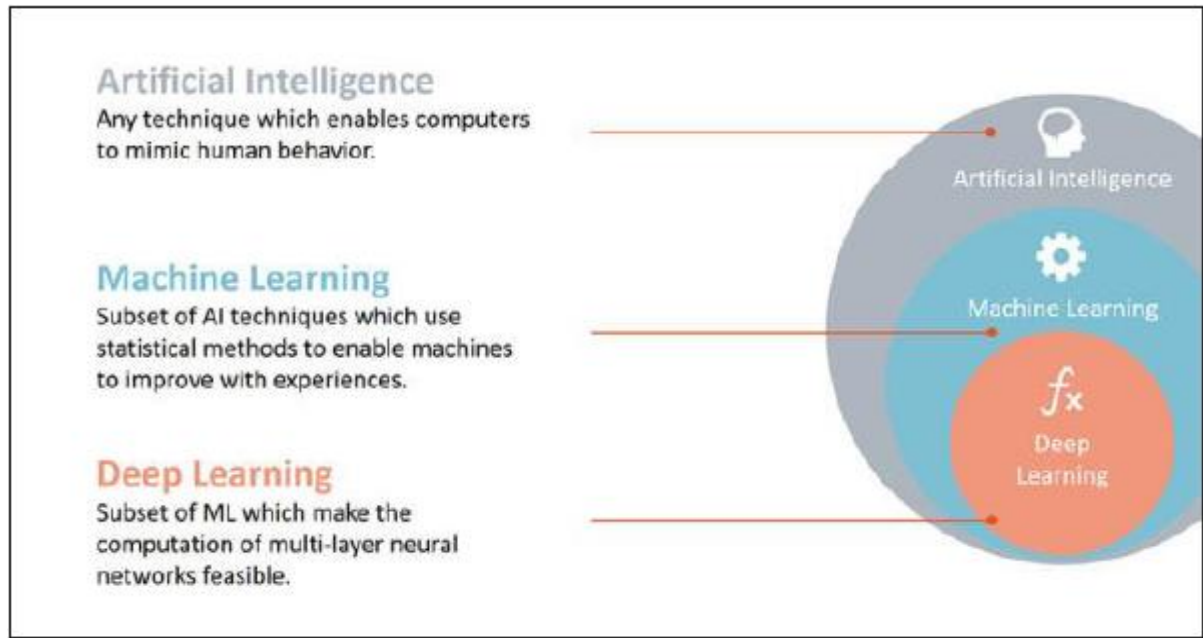
Seaborn.

scikit-learn.



ARTIFICIAL INTELLIGENCE

Introduction



According to the father of Artificial Intelligence, John McCarthy, it is “*The science and engineering of making intelligent machines, especially intelligent computer programs*”.

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems – The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.

To Implement Human Intelligence in Machines – Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as:-

Gaming – AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing – It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems – There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems – These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information

Or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

Speech Recognition – Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition – The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots – Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

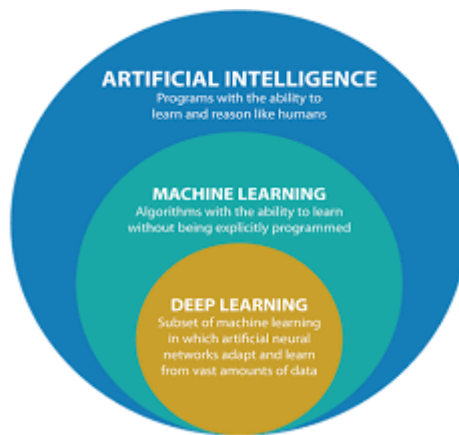
Application of AI

Deep Learning

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning.

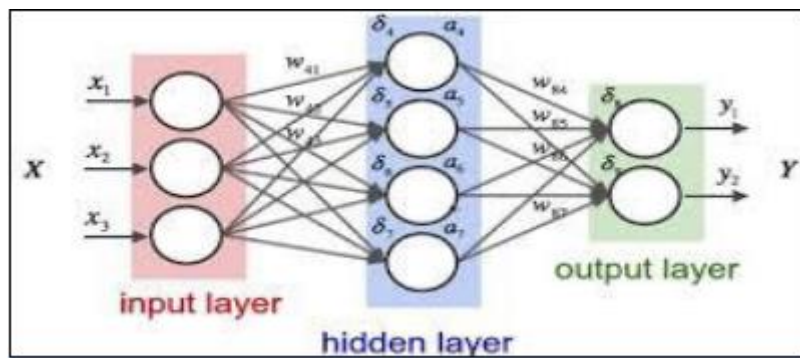
Deep learning is a class of machine learning algorithms that:

- Use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- Learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- Learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- Use some form of gradient descent for training via backpropagation.



NEURAL NETWORKING

Artificial neural networks (ANNs) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance on) tasks by considering examples, generally without task-specific programming

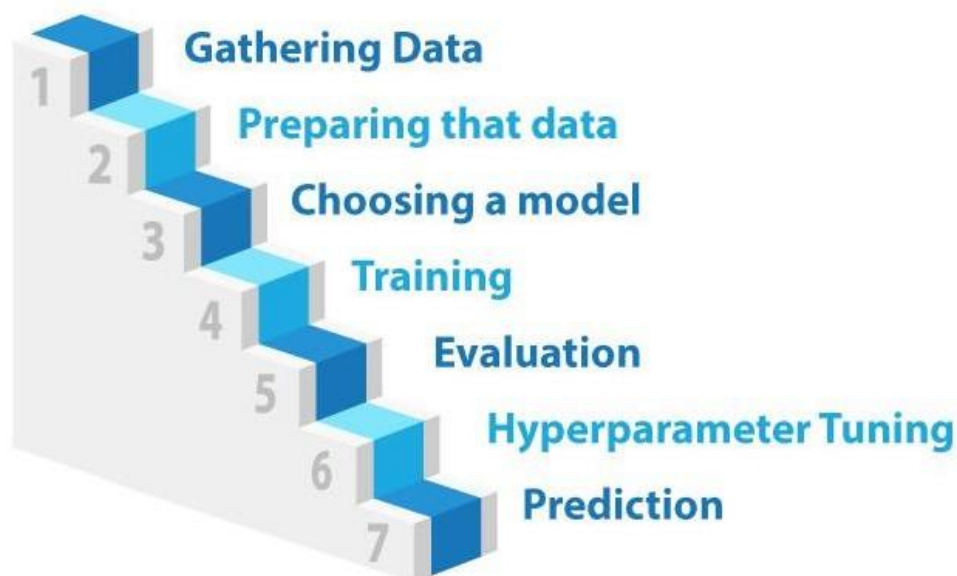


An ANN is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). Each connection between artificial neurons can transmit a signal from one to another.

MACHINE LEARNING

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: *The ability to learn*. Machine learning is actively being used today, perhaps in many more places than one would expect.

7 steps of Machine Learning



INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system: -

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labelled training data*.^[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[1., 0., 0.],  
 [ 0., 1., 2.]]
```

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

Import numpy as np

```
a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])
```

```
print 'Our array is:'
```

```
Print a
```

```
print '\n'
```

```
print "The items in the second column are:"
```

```
print a[:,1]
```

```
print '\n'
```

```
print "The items in the second row are:"
```

```
print a[1,:]
```

```
print '\n'
```

```
print "The items columns 1 onwards are:"
```

```
print a[:,1:]
```

OUTPUT

Our array is:

```
[[1 2 3]
 [3 4 5]
 [4 5 6]]
```

The items in the second column are:

```
[2 4 5]
```

The items in the second row are:

```
[3 4 5]
```

The items column 1 onwards are:

```
[[2 3]
 [4 5]
 [5 6]]
```

SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)

- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

Data Structures

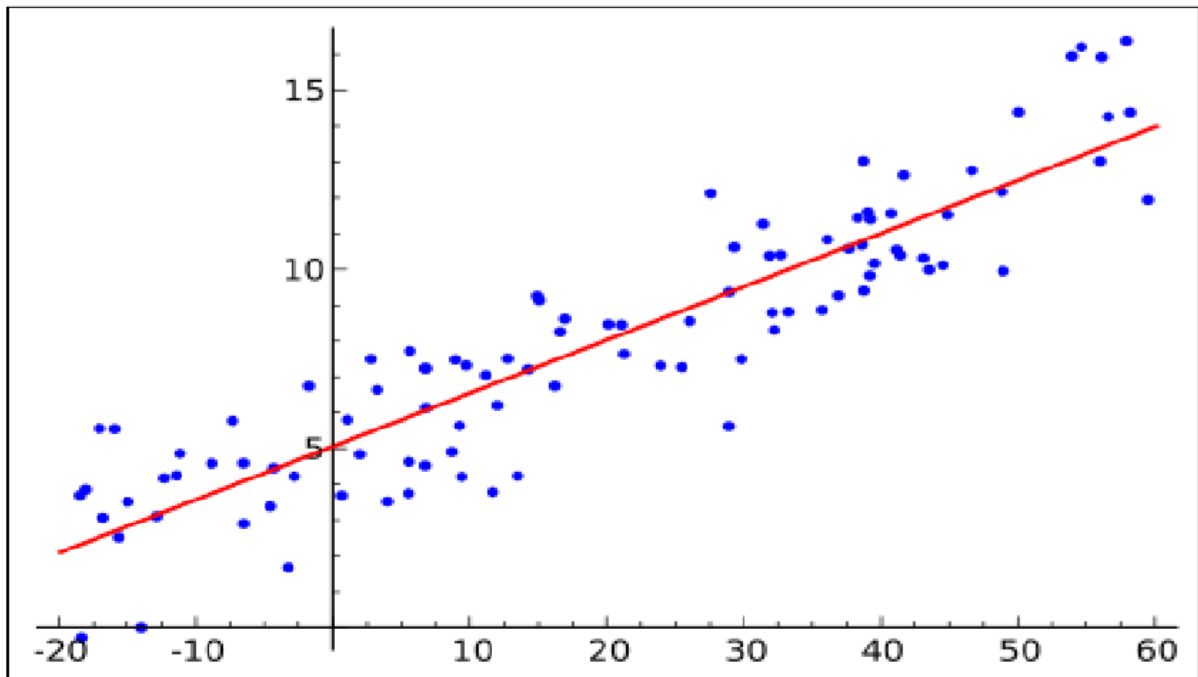
The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

Scikit-learn 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.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

- Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X . The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.
- In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

LOGISTIC REGRESSION

- Logistic regression, or logit regression, or logit model^[1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

- Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x .
- Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted $E(y | x)$, and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.
- Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function $E(y | x)$ is linear in the unknown parameters that are estimated from the data.

K-NEAREST-NEIGHBOUR ALGORITHM

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

NAÏVE BAYES CLASSIFIER ALGORITHM

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

SUPPORT VECTOR MACHINE ALGORITHM

- Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the kernel **trick**. Support Vector Machines are popular in machine learning. In this article, we will perform SVM classification with different types of kernels provided by sklearn.svm library’s SVC class. We will use the same problem statement used in logistic regression article.
- Four types of kernel tricks in SVM. They are-
 - i) Linear Kernel
 - ii) RBF Kernel
 - iii) Polynomial Kernel
 - iv) Sigmoid Kernel

RANDOM FOREST CLASSIFIER ALGORITHM

- Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.
- Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

DECISION TREE CLASSIFIER ALGORITHM

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.

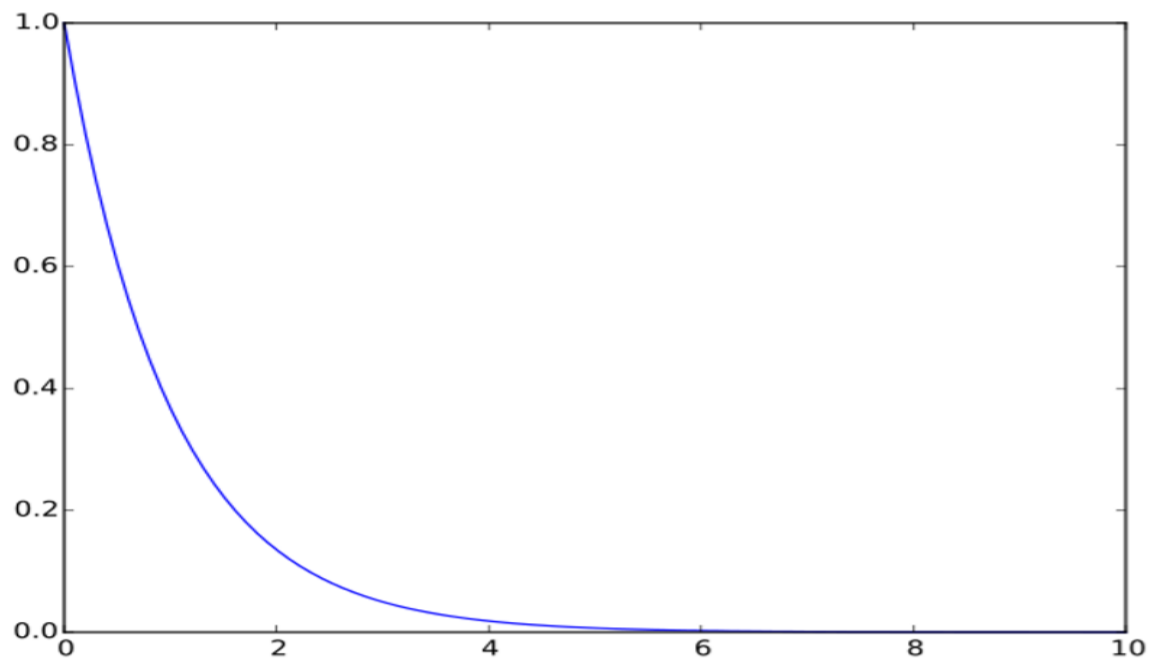
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

EXAMPLE

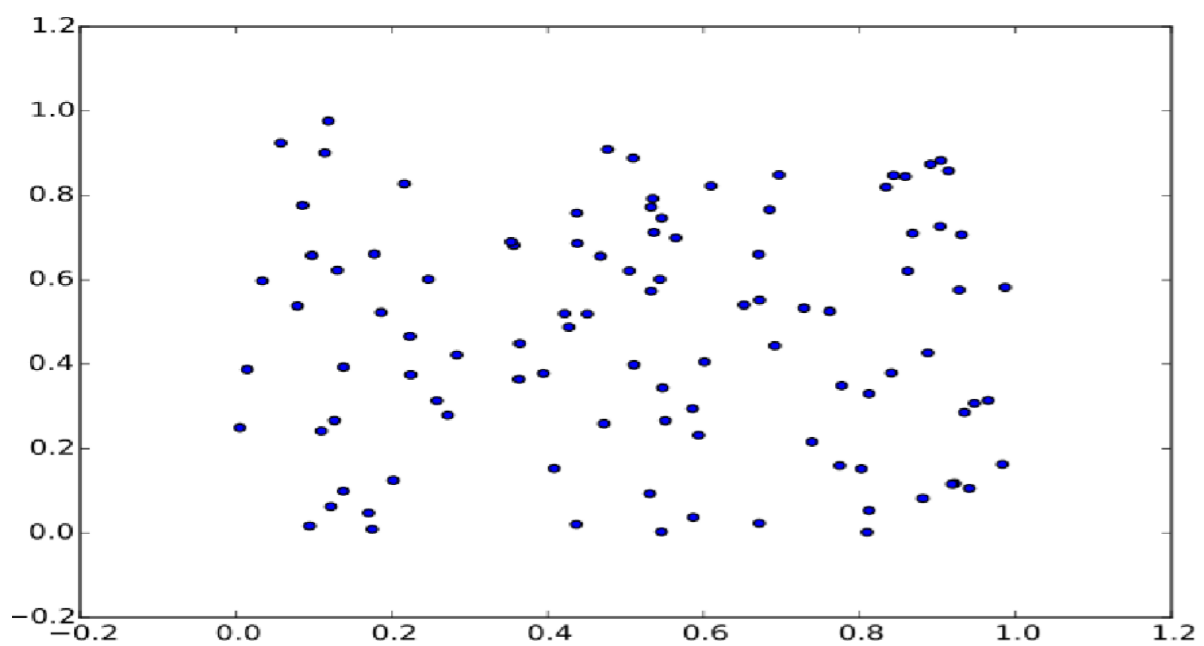
➤ LINE PLOT

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>>a=np.linspace(0,10,100)
>>>b=np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```

➤ SCATTER PLOT

```
>>>importmatplotlib.pyplotasplt  
>>>fromnumpy.randomimportrand  
>>>a=rand(100)  
>>>b=rand(100)  
>>>plt.scatter(a,b)  
>>>plt.show()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective

optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of weather from online website. We have downloaded the .csv files in which information was present.

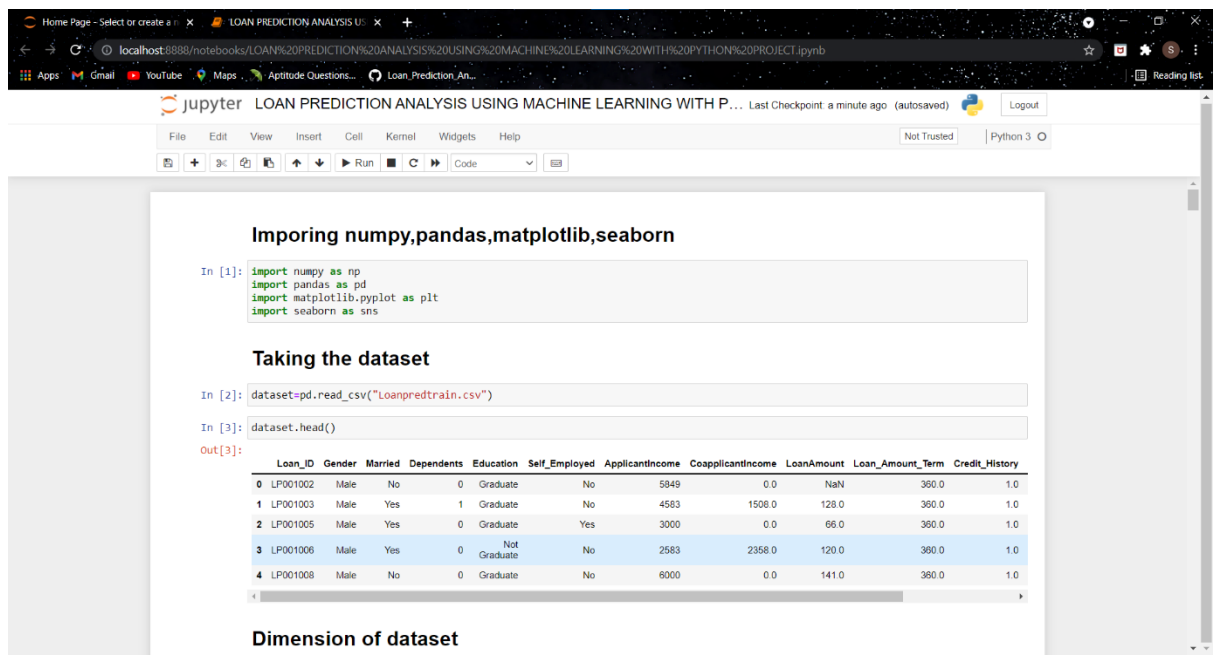
Data Formatting: The collected data is formatted into suitable data sets. We check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model, Ridge Linear model, Lasso Linear Model and Bayesian Ridge Linear Model.

Training: The data set was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.

Actual Codes For Loan Approval Status



The screenshot shows a Jupyter Notebook titled "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The code in the first cell imports numpy, pandas, matplotlib, and seaborn. The second cell loads a CSV file named "Loanpredtrain.csv" into a variable named 'dataset'. The third cell displays the first five rows of the dataset using 'dataset.head()'. The output shows a table with 13 columns: Loan_ID, Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, and Credit_History.

```
Importing numpy,pandas,matplotlib,seaborn

In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

Taking the dataset

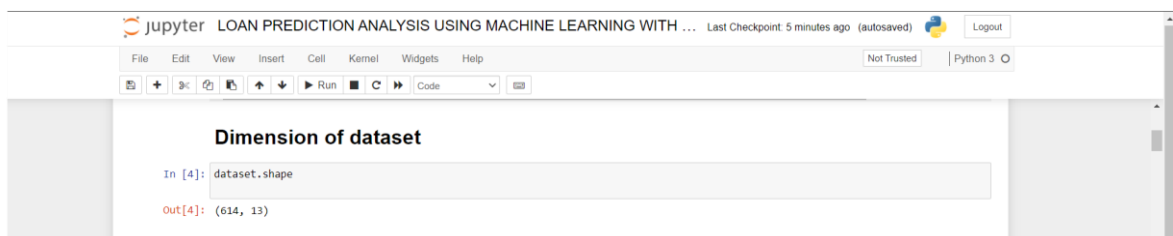
In [2]: dataset=pd.read_csv("Loanpredtrain.csv")

In [3]: dataset.head()
Out[3]:
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5949	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0

Dimension of dataset

Explanation: Loading the csv file into the dataframe 'dataset'.

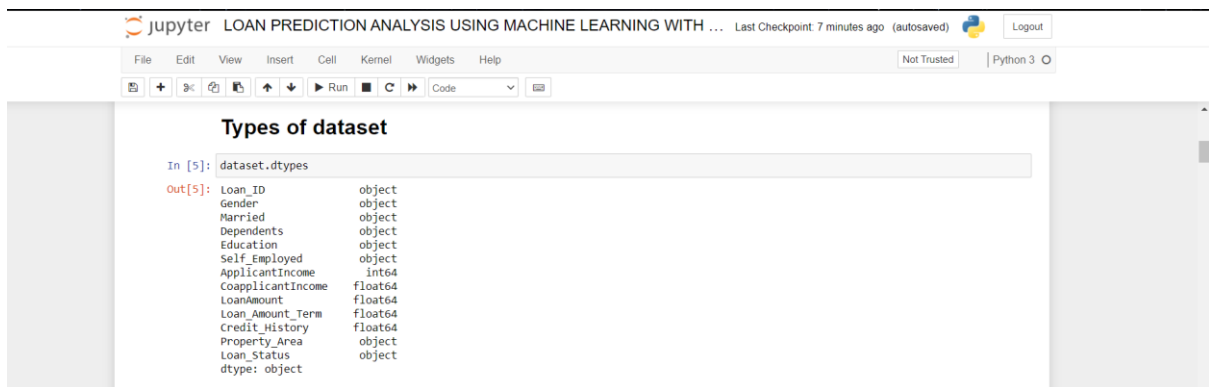


The screenshot shows a Jupyter Notebook titled "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ...". The code in the fourth cell checks the shape of the dataset using 'dataset.shape'. The output shows the dimensions as (614, 13).

```
Dimension of dataset

In [4]: dataset.shape
Out[4]: (614, 13)
```

Explanation: From this we can see that the dataset has 614 rows and 13 columns.



```
Types of dataset
In [5]: dataset.dtypes
Out[5]:
```

Loan_ID	object
Gender	object
Married	object
Dependents	object
Education	object
Self_Employed	object
ApplicantIncome	int64
CoapplicantIncome	float64
LoanAmount	float64
Loan_Amount_Term	float64
Credit_History	float64
Property_Area	object
Loan_Status	object
dtype:	object

Explanation: There are all together 13 columns in the dataframe. The dataframe contains 7 columns which consists of categorical data

- 1) Loan_ID
- 2) Gender
- 3) Married
- 4) Education
- 5) Self_Employed
- 6) Property_Area
- 7) Loan_Status

These 7 columns of categorical data are all nominal. The rest of the 6 columns out of 13 are quantitative and are as follow:

- 1) Dependents
- 2) ApplicantIncome
- 3) CoapplicantIncome

4)LoanAmount

5)Loan_Amount_Term

6)Credit_History

```
In [6]: dataset.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Out[6]: Loan_ID      0
Gender      13
Married      3
Dependents   15
Education    0
Self_Employed 32
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount   22
Loan_Amount_Term 14
Credit_History 50
Property_Area 0
Loan_Status  0
dtype: int64
```

Explanation:Number of null and NaNvalues present in each column.

jupyter LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ... Last Checkpoint: 8 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Filling the NULL values

```
In [7]: # Filling the NULL values
dataset.Gender=dataset.Gender.fillna("Male")
dataset.Married=dataset.Married.fillna("Yes")
dataset.Dependents=dataset.Dependents.fillna('0')
dataset.Self_Employed = dataset.Self_Employed.fillna('No')
dataset.LoanAmount=dataset.LoanAmount.fillna(dataset.LoanAmount.mean())
dataset.Loan_Amount_Term=dataset.Loan_Amount_Term.fillna(360.0)
dataset.Credit_History = dataset.Credit_History.fillna(1.0)
dataset.apply(lambda x: sum(x.isnull()),axis=0)
print(dataset)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed
0	LP001002	Male	No	0	Graduate	No
1	LP001003	Male	Yes	1	Graduate	No
2	LP001005	Male	Yes	0	Graduate	Yes
3	LP001006	Male	Yes	0	Not Graduate	No
4	LP001008	Male	No	0	Graduate	No
...
609	LP002978	Female	No	0	Graduate	No
610	LP002979	Male	Yes	3+	Graduate	No
611	LP002983	Male	Yes	1	Graduate	No
612	LP002984	Male	Yes	2	Graduate	No
613	LP002990	Female	No	0	Graduate	Yes

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	5849	0.0	146.412162	360.0
1	4583	1508.0	128.000000	360.0
2	3000	0.0	66.000000	360.0
3	2583	2358.0	120.000000	360.0
4	6000	0.0	141.000000	360.0
...
609	2900	0.0	71.000000	360.0
610	4106	0.0	40.000000	180.0

LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ... Last Checkpoint: 9 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	0.0	66.000000	360.0									
3	2583	2358.0	120.000000	360.0								
4	6000	0.0	141.000000	360.0								
...
609	2900	0.0	71.000000	360.0								
610	4106	0.0	40.000000	180.0								
611	8072	240.0	253.000000	360.0								
612	7583	0.0	187.000000	360.0								
613	4583	0.0	133.000000	360.0								

[614 rows x 13 columns]

Explanation:Filling the null or NaN values with 0 or mean.

LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ... Last Checkpoint: 10 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

After filling the null values the dataset

In [8]: dataset.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Loan_ID               614 non-null   object
1   Gender                614 non-null   object
2   Married               614 non-null   object
3   Dependents            614 non-null   object
4   Education             614 non-null   object
5   Self_Employed         614 non-null   object
6   ApplicantIncome       614 non-null   int64
7   CoapplicantIncome     614 non-null   float64
8   LoanAmount            614 non-null   float64
9   Loan_Amount_Term      614 non-null   float64
10  Credit_History         614 non-null   object
11  Property_Area         614 non-null   object
12  Loan_Status           614 non-null   object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Explanation:All the NaN values are replaced with suitable value as we observe the dataset to see that all the columns are non-null object.

The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ...". The notebook contains two code cells. The first cell, labeled "In [9]:", executes the command `dataset.groupby(['Gender', 'Loan_Status'])['Education'].count()`. The output, labeled "Out[9]:", is a table showing the count of education levels for each gender and loan status. The second cell, labeled "In [10]:", executes the command `dataset.groupby(['Education', 'Loan_Status'])['Loan_Status'].count()`. The output, labeled "Out[10]:", is a table showing the count of loan statuses for each education level.

```
In [9]: dataset.groupby(['Gender', 'Loan_Status'])['Education'].count()
Out[9]: Gender  Loan_Status
Female      N           37
           Y           75
Male        N          155
           Y          347
Name: Education, dtype: int64
```

Further analysis gave us following points:

More number of male applicants' loans got approved as compared to female applicants.

```
In [10]: #More number of male applicants' Loans got approved as compared to female applicants.
dataset.groupby(['Education', 'Loan_Status'])['Loan_Status'].count()
Out[10]: Education  Loan_Status
Graduate          N          140
                Y          340
Not Graduate      N           52
                Y           82
Name: Loan_Status, dtype: int64
```

Explanation: As we see the male applicants has the higher loan approval than the female applicants.

The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ...". The notebook contains two code cells. The first cell, labeled "In [11]:", executes the command `dataset.groupby(['Property_Area', 'Loan_Status'])['Loan_Status'].count()`. The output, labeled "Out[11]:", is a table showing the count of loan statuses for each property area. The second cell, labeled "In [12]:", executes the command `dataset.groupby(['Self_Employed', 'Loan_Status'])['Loan_Status'].count()`. The output, labeled "Out[12]:", is a table showing the count of loan statuses for each self-employment status.

```
In [11]: #More Loans of "Graduates" got approved as compared to "Not Graduates"
dataset.groupby(['Property_Area', 'Loan_Status'])['Loan_Status'].count()
Out[11]: Property_Area  Loan_Status
Rural              N           69
                Y          110
Semiurban          N           54
                Y          179
Urban              N           69
                Y          133
Name: Loan_Status, dtype: int64
```

More loans of "Graduates" got approved as compared to "Not Graduates"

If the property is situated in semiurban area, the chances of getting loan approved is high as compared to Rural or Urban.

```
In [12]: #If the property is situated in semiurban area, the chances of getting Loan approved is high as compared to Rural or Urban.
dataset.groupby(['Self_Employed', 'Loan_Status'])['Loan_Status'].count()
Out[12]: Self_Employed  Loan_Status
No                N          166
                Y          366
Yes               N           26
                Y           56
Name: Loan_Status, dtype: int64
```

Explanation: As we see here more graduates got more loan approval than the non-graduates and if the property is in semi urban area there is a higher probability of approval of loan than rural or urban.

jupyter LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ... Last Checkpoint: 13 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run

Loan term of 360 got maximum approval.

```
In [13]: #Loan term of 360 got maximum approval.
dataset.groupby(['Loan_Amount_Term','Loan_Status'])['Loan_Status'].count()
```

```
Out[13]: Loan_Amount_Term  Loan_Status
12.0      Y              1
36.0      N              2
60.0      Y              2
84.0      N              1
120.0     Y              3
180.0     N             15
240.0     Y             29
300.0     N              3
360.0     Y             367
480.0     N              9
         Y              6
Name: Loan_Status, dtype: int64
```

Explanation: The loan term which is 360 has got the most approval.

jupyter LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ... Last Checkpoint: 14 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run

Most of the loan applications of self employed individuals got rejected.

```
In [14]: #Most of the loan applications of self employed individuals got rejected.
dataset.groupby(['Self_Employed','Loan_Status'])['Loan_Status'].count()
```

```
Out[14]: Self_Employed  Loan_Status
No      N             166
        Y             366
Yes     N              26
        Y              56
Name: Loan_Status, dtype: int64
```

Loans of applicants with 0 dependents got more approved as compared to other applicants. Mainly, because it has been perceived that applicants with 0 dependents have lesser liabilities.

```
In [15]: #Loans of applicants with 0 dependents got more approved as compared to other applicants. Mainly, because it has been perceived
dataset.groupby(['Dependents','Loan_Status'])['Loan_Status'].count()
```

```
Out[15]: Dependents  Loan_Status
0      N             113
        Y             247
1      N              36
        Y              66
2      N              25
        Y              76
3+     N              19
        Y              33
Name: Loan_Status, dtype: int64
```

Explanation: As we see here who are self-employed individuals their application got rejected.

The dependents of value 0 got more approved as compared to the other applicants because it is seen that applicants have dependency 0 have lesser liabilities.

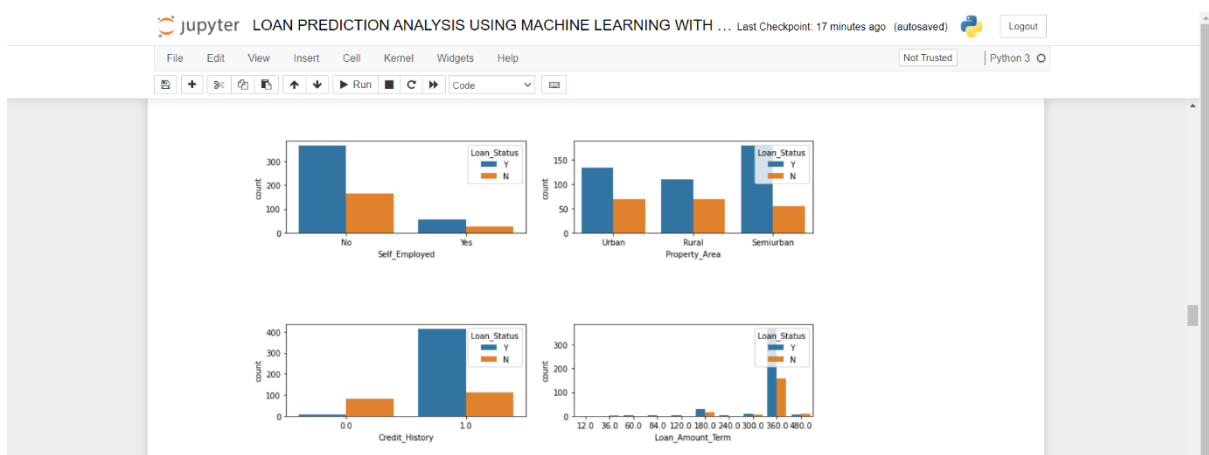
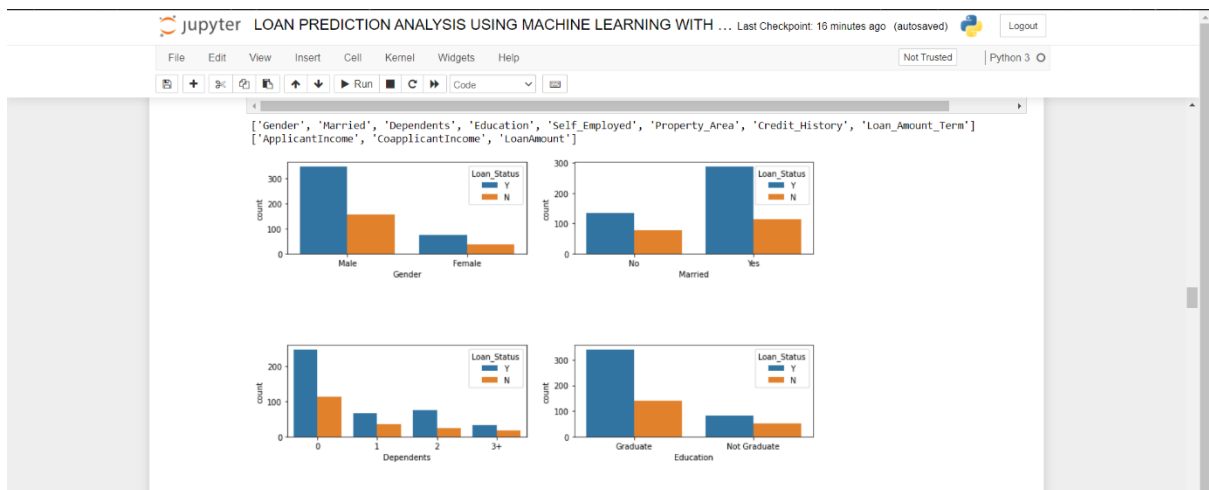
```

jupyter LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ... Last Checkpoint: 15 minutes ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
name: Loan_Status, dtype: int0a

In [16]: #categorical columns
categorical_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Property_Area', 'Credit_History', 'Loan_Amount_Term']
print(categorical_columns)
numerical_columns = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount']
print(numerical_columns)
X = dataset.iloc[:, 1:12].values
y = dataset.iloc[:, 12].values
fig, axes = plt.subplots(4, 2, figsize=(12, 15))
for idx, cat_col in enumerate(categorical_columns):
    row, col = idx//2, idx%2
    sns.countplot(x=cat_col, data=dataset, hue='Loan_Status', ax=axes[row, col])
plt.subplots_adjust(hspace=1)

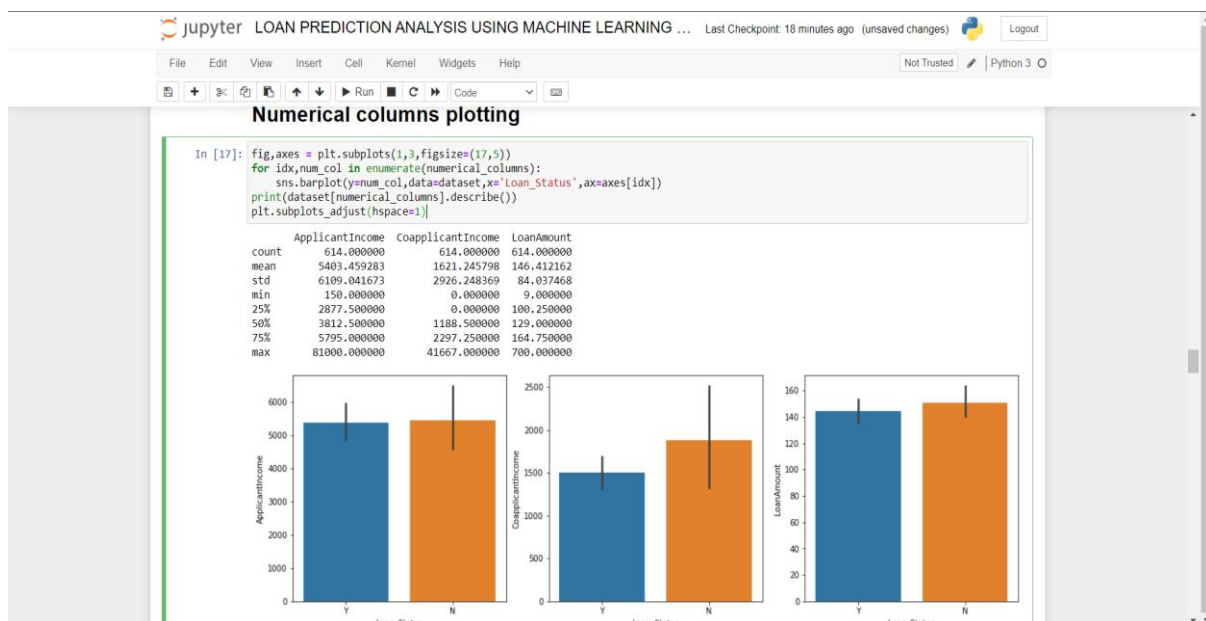
```

Explanation: now we have extracted the columns from the dataset by iloc and for plotting we divided into categorical and numerical columns.

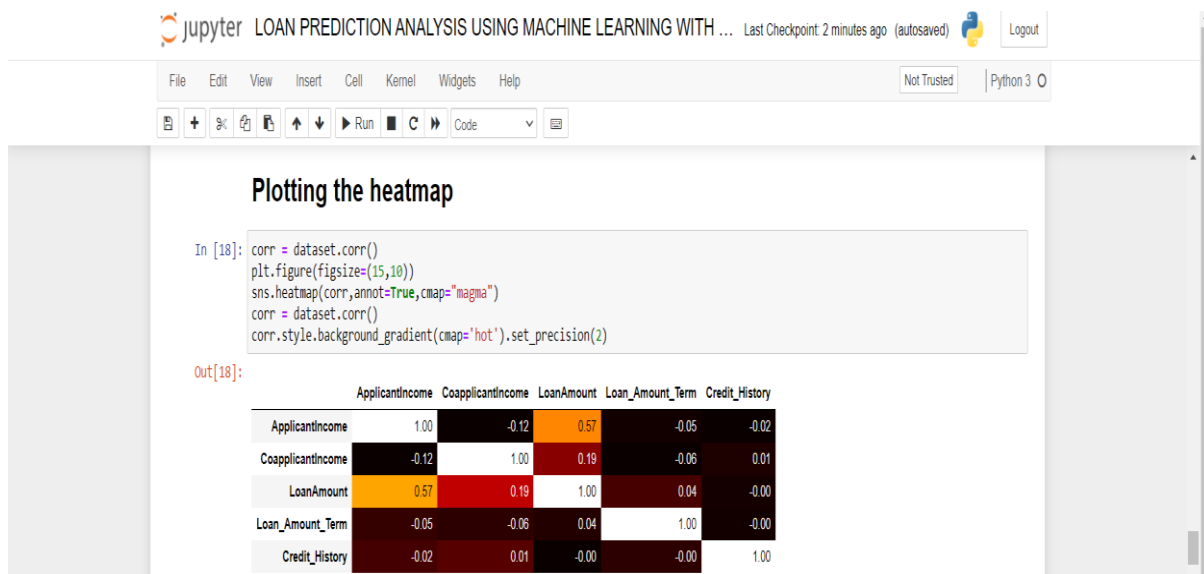


Explanation: Now we have plotted the categorical columns by the seaborn and matplotlib and comparing

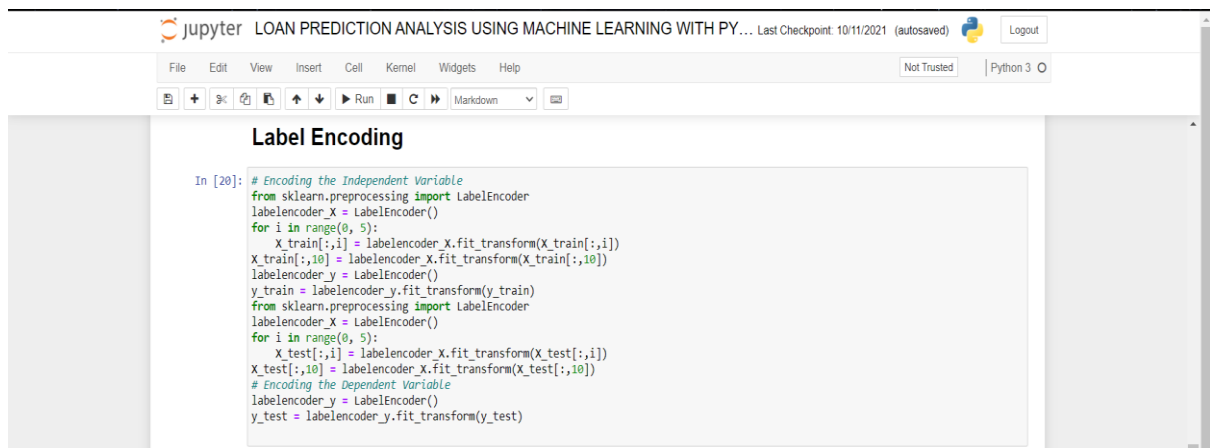
with the Loan_Status to see how the categorical columns are affecting the loan approval as we did previously on groupby function by values and here by seeing the bar plotting. Those who have credit history 1.0 has the higher approval.



Explanation: As we plotted categorical columns before now we are plotting the numerical columns to see how the applicants income and loan amount related. We see that applicant income and coapplicant income and loan amount in these categories probability of getting loan will be high when income will be greater and loan amount must be smaller.



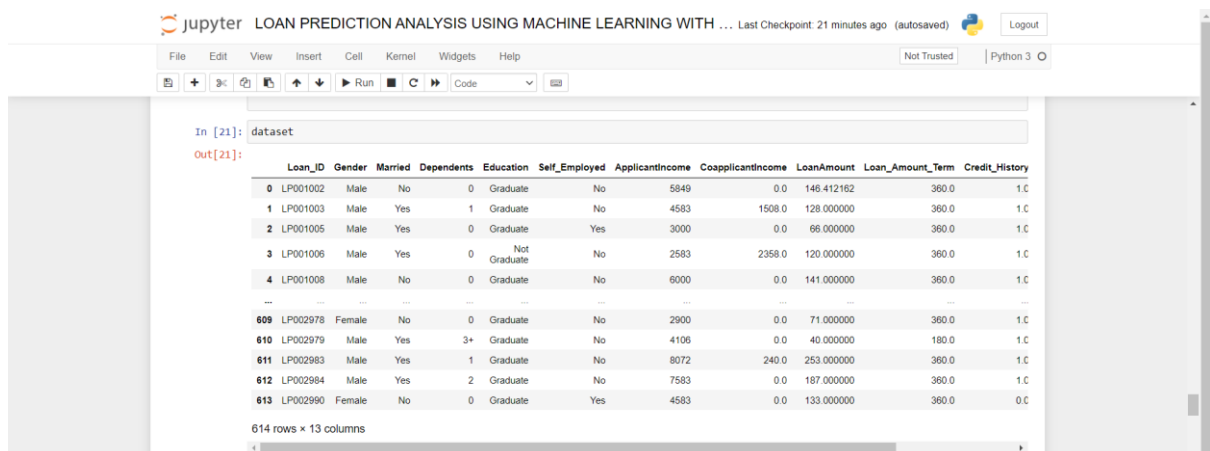
Explanation:Correlation ranges from -1 to +1. Values closer to zero means there is no linear trend between the two variables. The closer to 1 the correlation is the more positively correlated,that is as one increases so does the other and the closer to 1 the stronger this relationship is. 0.5 is the threshold point.



The screenshot shows a Jupyter Notebook interface with a title bar that reads "jupyter LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH PY...". The notebook contains a code cell with the following Python code for label encoding:

```
In [20]: # Encoding the Independent Variable
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
for i in range(0, 5):
    X_train[:,i] = labelencoder_X.fit_transform(X_train[:,i])
X_train[:,10] = labelencoder_X.fit_transform(X_train[:,10])
labelencoder_y = LabelEncoder()
y_train = labelencoder_y.fit_transform(y_train)
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
for i in range(0, 5):
    X_test[:,i] = labelencoder_X.fit_transform(X_test[:,i])
X_test[:,10] = labelencoder_X.fit_transform(X_test[:,10])
# Encoding the Dependent Variable
labelencoder_y = LabelEncoder()
y_test = labelencoder_y.fit_transform(y_test)
```

Explanation: To convert this kind of categorical text data into model-understandable numerical data, we use the Label Encoder class. By the label encoding the data is ready for model.



The screenshot shows a Jupyter Notebook interface with a title bar that reads "jupyter LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ...". The notebook contains a code cell with the following Python code:

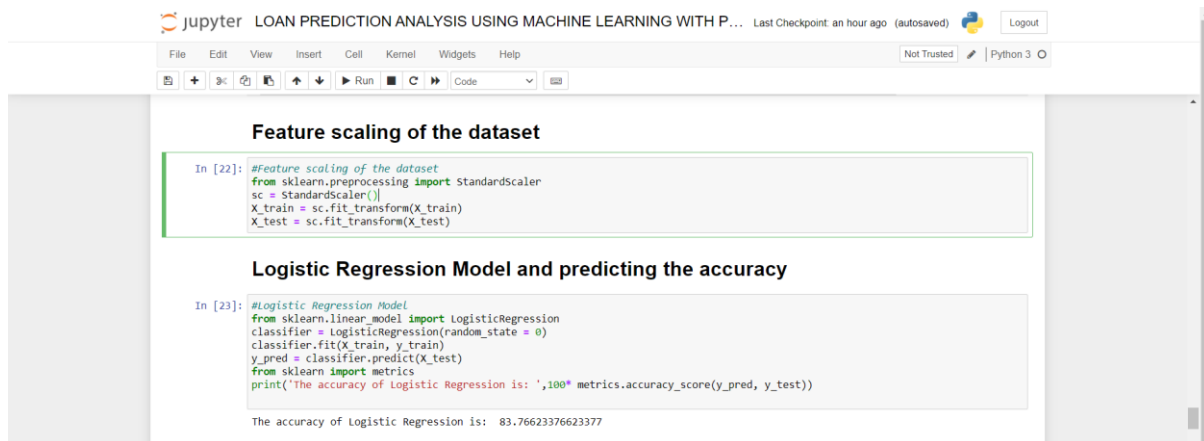
```
In [21]: dataset
```

The output of the code cell is a table with 13 columns and 614 rows. The columns are: Loan_ID, Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, and Credit_History. The table shows the first 13 rows of the dataset, with the last row being the 614th row.

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.C
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.C
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.C
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.C
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.C
...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.000000	360.0	1.C
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.000000	180.0	1.C
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.000000	360.0	1.C
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.000000	360.0	1.C
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.000000	360.0	0.C

614 rows x 13 columns

Explanation: after label encoding the dataset is this.



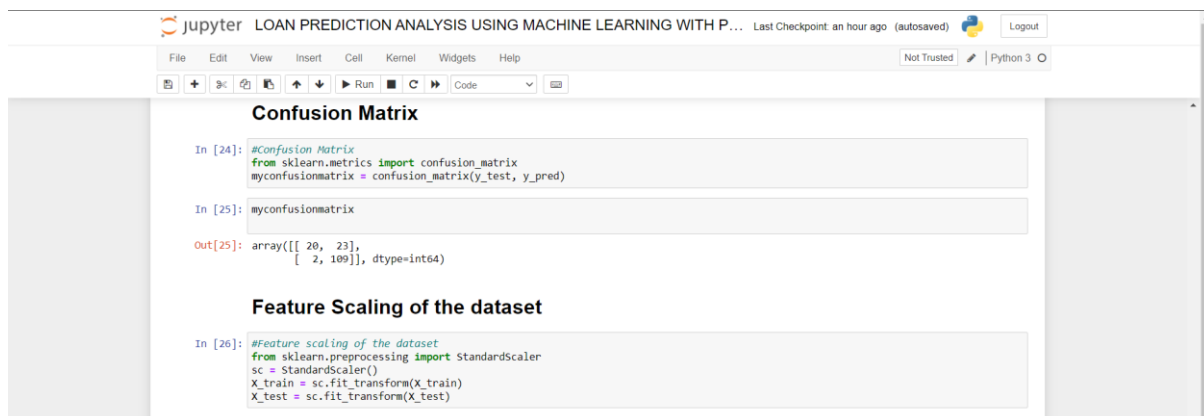
The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook contains two code cells. The first cell, titled "Feature scaling of the dataset", imports StandardScaler from sklearn.preprocessing and applies it to X_train and X_test. The second cell, titled "Logistic Regression Model and predicting the accuracy", imports LogisticRegression from sklearn.linear_model, fits the model on X_train and y_train, predicts on X_test, and prints the accuracy score. The output of the second cell shows an accuracy of 83.76623376623377.

```
In [22]: #Feature scaling of the dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)

In [23]: #Logistic Regression Model
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn import metrics
print('The accuracy of Logistic Regression is: ',100* metrics.accuracy_score(y_pred, y_test))

The accuracy of Logistic Regression is: 83.76623376623377
```

Explanation: We used StandardScaler feature scaling to normalize the range of independent variables or features of data. After the feature scaling of the dataset we used logistic regression model to predict the accuracy of the model and got an accuracy of 83.766.



The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook contains three code cells. The first cell, titled "Confusion Matrix", imports confusion_matrix from sklearn.metrics and prints the result. The second cell, titled "Feature Scaling of the dataset", imports StandardScaler from sklearn.preprocessing and applies it to X_train and X_test. The output of the first cell shows a confusion matrix array.

```
In [24]: #Confusion Matrix
from sklearn.metrics import confusion_matrix
myconfusionmatrix = confusion_matrix(y_test, y_pred)

In [25]: myconfusionmatrix

Out[25]: array([[ 20,  23],
               [  2, 109]], dtype=int64)

In [26]: #Feature scaling of the dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Explanation: Here we use the confusion matrix to see the performance of a classification model. The matrix compares the target values with the predicted value by the Machine Learning Model.

Again for the next model we have done the feature scaling of the dataset.

A Jupyter Notebook interface titled "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook shows a code cell where the training set is assigned to `X_train`. The output displays a NumPy array of 10 rows and 5 columns of numerical data.

```
In [27]: X_train
```

```
Out[27]: array([[ 0.46575875,  0.69217027,  0.20631248, ...,  0.27778225,
                  0.41649656,  1.20186498],
                [ 0.46575875, -1.44473123, -0.77207659, ...,  0.27778225,
                  0.41649656, -1.31684978],
                [ 0.46575875, -1.44473123,  1.18470154, ...,  0.27778225,
                  0.41649656, -1.31684978],
                ...,
                [ 0.46575875,  0.69217027,  2.1630906 , ...,  0.27778225,
                  0.41649656, -0.0574924 ],
                [ 0.46575875,  0.69217027, -0.77207659, ...,  0.27778225,
                  0.41649656,  1.20186498],
                [-2.14703426,  0.69217027, -0.77207659, ...,  0.27778225,
                  0.41649656, -0.0574924 ]])
```

A Jupyter Notebook interface titled "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook shows a code cell where the testing set is assigned to `X_test`. The output displays a NumPy array of 10 rows and 5 columns of numerical data.

```
In [28]: X_test
```

```
Out[28]: array([[ 0.49186938, -1.18585412, -0.6387615 , ...,  0.2610401 ,
                  0.39735971, -0.01699069],
                [-2.03306009, -1.18585412, -0.6387615 , ...,  0.2610401 ,
                  0.39735971, -0.01699069],
                [ 0.49186938,  0.84327404, -0.6387615 , ...,  0.2610401 ,
                  0.39735971,  1.29129257],
                ...,
                [ 0.49186938,  0.84327404, -0.6387615 , ...,  0.2610401 ,
                  0.39735971,  1.29129257],
                [ 0.49186938, -1.18585412, -0.6387615 , ...,  0.2610401 ,
                  0.39735971, -0.01699069],
                [ 0.49186938,  0.84327404,  0.40772011, ...,  0.2610401 ,
                  -2.51661148, -0.01699069]])
```

Explanation: Training and testing set to ready for prediction.

A Jupyter Notebook interface titled "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook shows two code cells. The first cell fits a K-Nearest-Neighbour model and prints the accuracy. The second cell imports and uses `StandardScaler` for feature scaling.

```
In [29]: #K-NEAREST-NEIGHBOUR CLASSIFICATION MODEL
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(weights='distance')
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Measuring Accuracy
from sklearn.metrics import accuracy_score
print('The accuracy of KNN is: ',100* accuracy_score(y_pred, y_test))

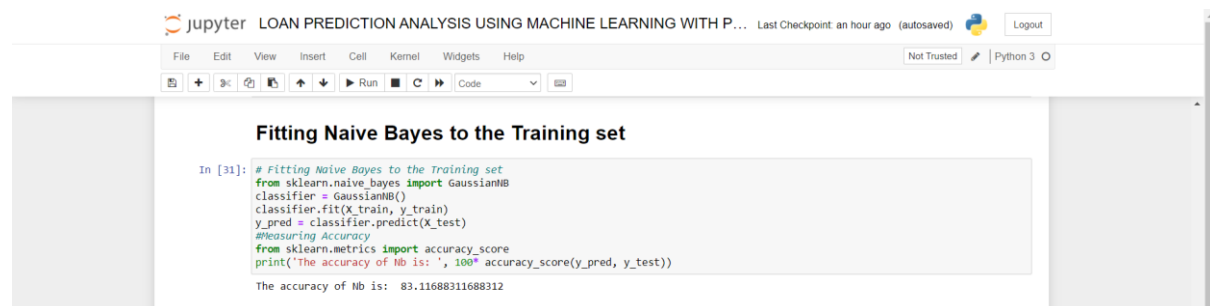
The accuracy of KNN is: 79.22077922077922
```

Feature Scaling of the dataset

```
In [30]: #Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Explanation: We used `StandardScaler` feature scaling to normalize the range of independent variables or

features of data. After the feature scaling of the dataset we used K-Nearest-Neighbour model to predict the accuracy of the model and got an accuracy of 79.22.

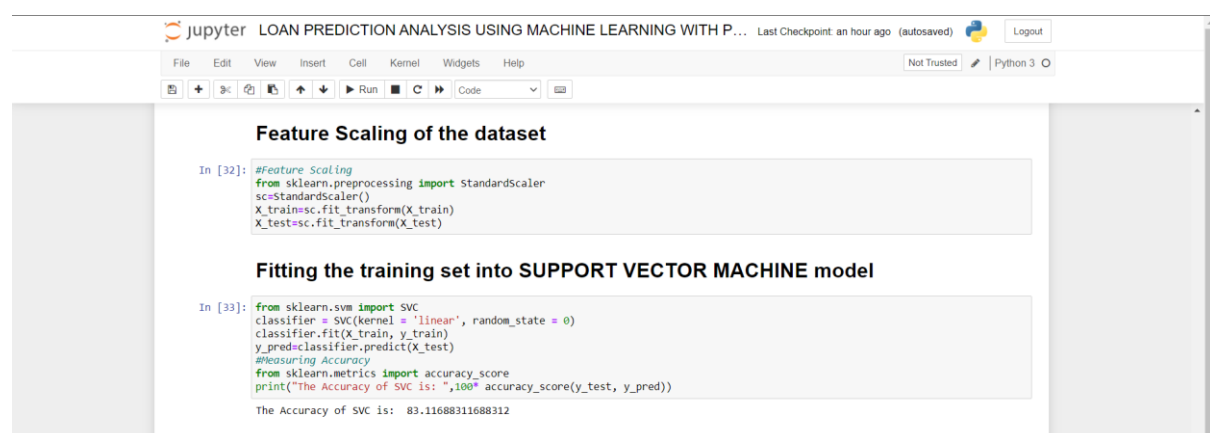


The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook is running on Python 3. The current cell is titled "Fitting Naive Bayes to the Training set" and contains the following code:

```
In [31]: # Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
#Measuring Accuracy
from sklearn.metrics import accuracy_score
print("The accuracy of Nb is: ", 100* accuracy_score(y_pred, y_test))

The accuracy of Nb is: 83.11688311688312
```

Explanation:After the feature scaling of the dataset we used Naïve Bayes model to predict the accuracy of the model and got an accuracy of 83.116.



The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook is running on Python 3. The current cell is titled "Feature Scaling of the dataset" and contains the following code:

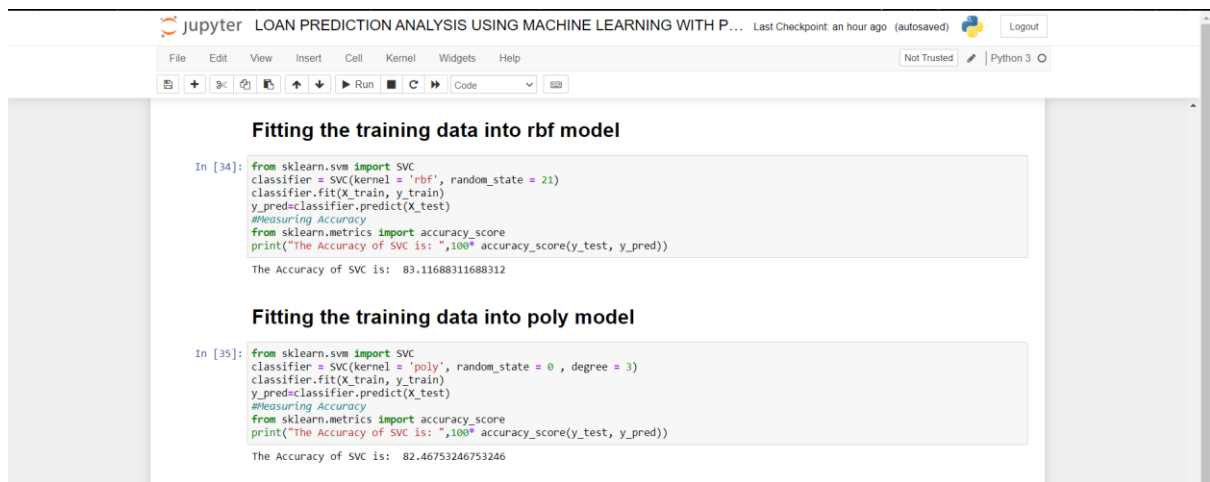
```
In [32]: #Feature Scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

The next cell is titled "Fitting the training set into SUPPORT VECTOR MACHINE model" and contains the following code:

```
In [33]: from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
y_pred=classifier.predict(X_test)
#Measuring Accuracy
from sklearn.metrics import accuracy_score
print("The Accuracy of SVC is: ",100* accuracy_score(y_test, y_pred))

The Accuracy of SVC is: 83.11688311688312
```

Explanation:We usedStandardScaler feature scaling to normalize the range of independent variables or features of data. After the feature scaling of the dataset we used Support Vector Machine model to predict the accuracy of the model and got an accuracy of 83.116.



The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook contains two code cells. The first cell, titled "Fitting the training data into rbf model", shows the fitting of an SVM with kernel 'rbf' and random_state = 21. The output is "The Accuracy of SVC is: 83.11688311688312". The second cell, titled "Fitting the training data into poly model", shows the fitting of an SVM with kernel 'poly' and degree = 3. The output is "The Accuracy of SVC is: 82.46753246753246".

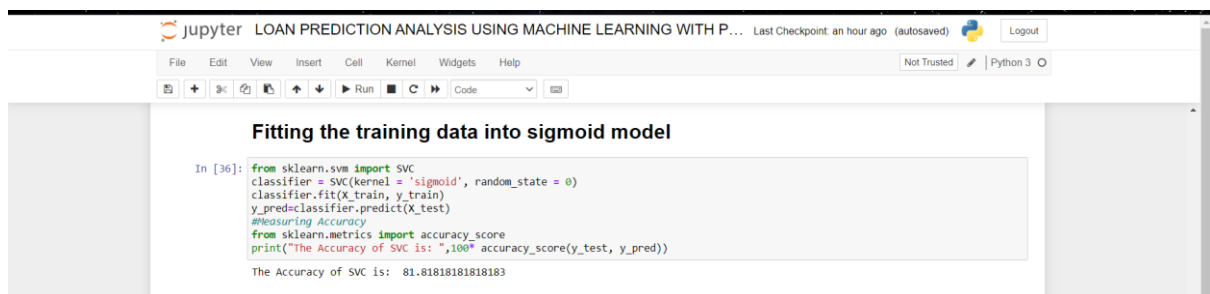
```
In [34]: from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 21)
classifier.fit(X_train, y_train)
y_pred=classifier.predict(X_test)
#Measuring Accuracy
from sklearn.metrics import accuracy_score
print("The Accuracy of SVC is: ",100* accuracy_score(y_test, y_pred))

The Accuracy of SVC is: 83.11688311688312
```

```
In [35]: from sklearn.svm import SVC
classifier = SVC(kernel = 'poly', random_state = 0 , degree = 3)
classifier.fit(X_train, y_train)
y_pred=classifier.predict(X_test)
#Measuring Accuracy
from sklearn.metrics import accuracy_score
print("The Accuracy of SVC is: ",100* accuracy_score(y_test, y_pred))

The Accuracy of SVC is: 82.46753246753246
```

Explanation: We used StandardScaler feature scaling to normalize the range of independent variables or features of data. After the feature scaling of the dataset we used Support Vector Machine model with kernel tricks rbf and poly to predict the accuracy of the model and got an accuracy of 83.116 & 82.467

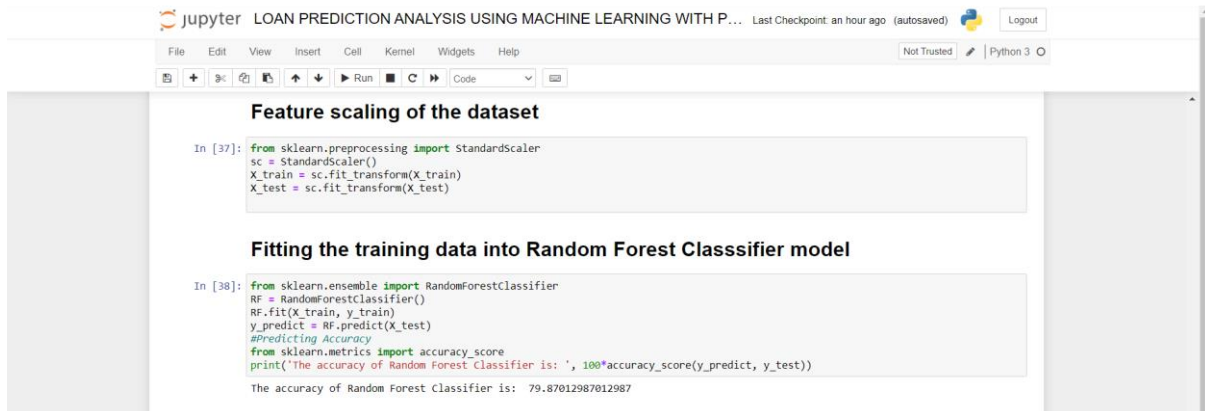


The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook contains one code cell titled "Fitting the training data into sigmoid model". The code shows the fitting of an SVM with kernel 'sigmoid' and random_state = 0. The output is "The Accuracy of SVC is: 81.81818181818183".

```
In [36]: from sklearn.svm import SVC
classifier = SVC(kernel = 'sigmoid', random_state = 0)
classifier.fit(X_train, y_train)
y_pred=classifier.predict(X_test)
#Measuring Accuracy
from sklearn.metrics import accuracy_score
print("The Accuracy of SVC is: ",100* accuracy_score(y_test, y_pred))

The Accuracy of SVC is: 81.81818181818183
```

Explanation: After the feature scaling of the dataset we used Support Vector Machine model with kernel tricks sigmoid to predict the accuracy of the model and got an accuracy of 81.818



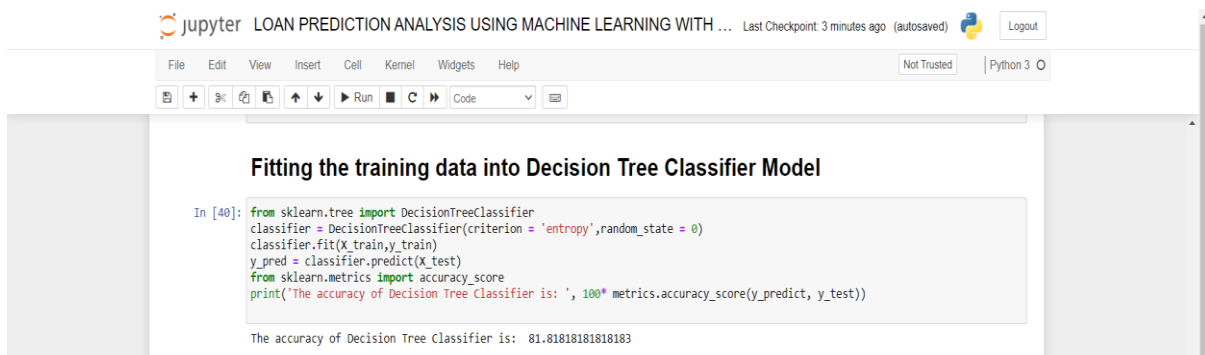
The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH P...". The notebook contains two code cells. The first cell, titled "Feature scaling of the dataset", imports StandardScaler from sklearn.preprocessing and applies it to X_train and X_test. The second cell, titled "Fitting the training data into Random Forest Classifier model", imports RandomForestClassifier from sklearn.ensemble, fits the model on X_train and y_train, predicts on X_test, and prints the accuracy score. The output shows an accuracy of 79.87012987012987.

```
In [37]: from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)

In [38]: from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()
RF.fit(X_train, y_train)
y_predict = RF.predict(X_test)
#Predicting Accuracy
from sklearn.metrics import accuracy_score
print('The accuracy of Random Forest Classifier is: ', 100*accuracy_score(y_predict, y_test))

The accuracy of Random Forest Classifier is: 79.87012987012987
```

Explanation: We used StandardScaler feature scaling to normalize the range of independent variables or features of data. After the feature scaling of the dataset we used Random Forest Classifier model to predict the accuracy of the model and got an accuracy of 79.870.

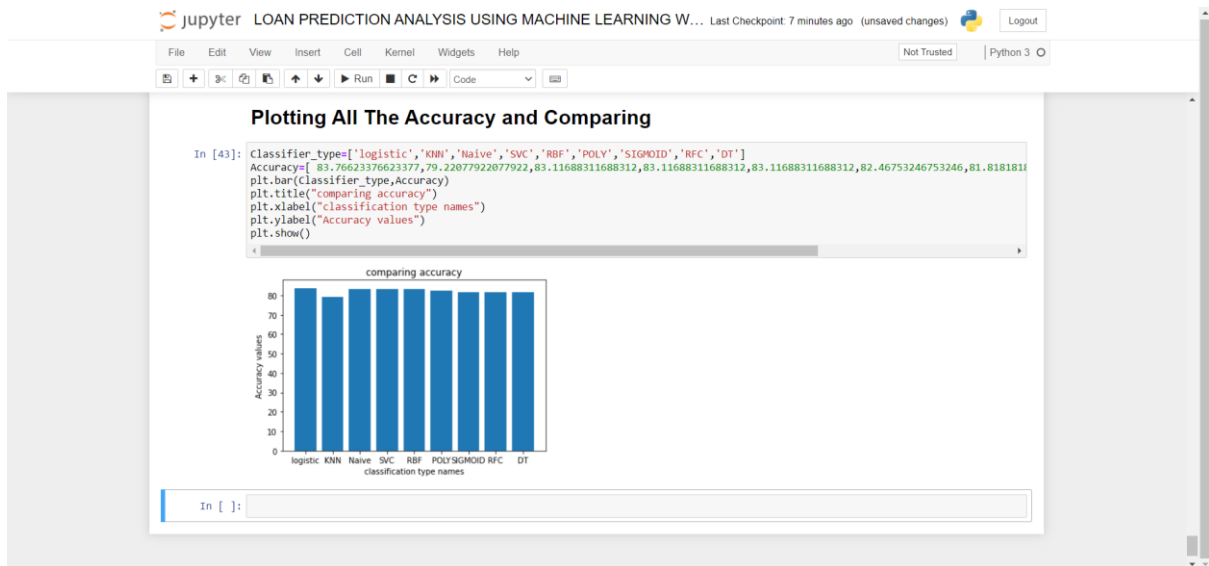


The screenshot shows a Jupyter Notebook interface with the title "LOAN PREDICTION ANALYSIS USING MACHINE LEARNING WITH ...". The notebook contains one code cell titled "Fitting the training data into Decision Tree Classifier Model". It imports DecisionTreeClassifier from sklearn.tree, fits the model on X_train and y_train, predicts on X_test, and prints the accuracy score. The output shows an accuracy of 81.81818181818183.

```
In [40]: from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import accuracy_score
print('The accuracy of Decision Tree Classifier is: ', 100*metrics.accuracy_score(y_predict, y_test))

The accuracy of Decision Tree Classifier is: 81.81818181818183
```

Explanation: We used StandardScaler feature scaling to normalize the range of independent variables or features of data. After the feature scaling of the dataset we used Decision Tree Classifier model to predict the accuracy of the model and got an accuracy of 81.818.



Explanation: Now here we plotted all the accuracy with corresponding classifier types and comparing them to see which model shows the best accuracy and we see Logistic Regression Model shows the best accuracy among all.

CONCLUSION

We have collected the raw data from online sources. Then we take this raw data and format it.

Now we have selected few models for error detection. We have used six models namely logistic regression model, K-Nearest-Neighbour Model, Naïve Bayes Model, Support Vector Machine model (Linear, polynomial, rbf, sigmoid), Random Forest Classification model and Decision Tree Classifier model.

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient component. This application is working properly and meeting to all Banker requirements. This component can be easily plugged in many other systems.

FUTURE SCOPE

The data taken was limited. In near future this module of prediction can be integrated with the module of automated processing system. The system is trained on old training dataset in future software can be made such that new testing data should also take part in training data after some fixed time.

In near future this module of prediction can be integrated with the module of automated processing system. The system is trained on old training dataset in future software can be made such that new testing data should also take part in training data after some fixed time.

Bibliography:

The dataset and the help about resources have been taken from the following websites for this project:-

- 1) <https://www.kaggle.com/>
- 2) <https://www.javatpoint.com/machine-learning>
- 3) <https://www.tutorialspoint.com/global-vs-local-variables-in-python>
- 4) <https://www.google.co.in/search?q=deep+learning>
- 5) <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>
- 6) <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
- 7) <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>

THANK YOU