INTERNSHIP REPORT
AT PANTECH E-LEARNING

Attended by

B. Sucharitha (18121072)

in partial fulfilment for the award of the degree of

Bachelor of Technology

In

ELECTRONICS AND COMMUNICATION

from 10.11.2021 to 16.12.2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL SCIENCES

HINDUSTAN INSTITUTE OF TECHNOLOGY AND SCIENCE

PADUR 603 103

HINDUSTAN
INSTITUTE OF TECHNOLOGY & SCIENCE
(DEEMED TO BE UNIVERSITY)
CHENNAI

BONAFIDE CERTIFICATE

This is to certify that the "Internship report" submitted by B. Sucharitha is the work done by him and submitted during 2021–2022 academic year, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in ELECTRONICS AND COMMUNICATION, at PANTECH E-LEARNING.

HEAD OF THE DEPARTMENT
Dr.A.L.Vallikannu

INTERNSHIP COORDINATOR
Ms.K.Thenkumari

CERTIFICATION

TABLE OF CONTENT

1.Introduction

Started modestly in 2004, Pantech E Learning strives to be a leader in the area of Technical Training's, Products & Research Initiatives. Having developed and established a comprehensive set of sustainable business initiatives that facilitate our bottom-line approach to operating our business.

Services: Internship matching, online training

Founder: Malaiyappan maruthamuthu

Industry: Education, Employment

Headquarters; Chennai, India

Pantech E Learning Visions for promoting technical excellence & offering fantastic opportunities to the students and faculty of educational institutes & corporates, enabling them to pursue insight into transformative technical trends with a view to empower & build a knowledge society.

The Top Pantech e-learning Courses

- Data science masterclass.
- IoT Master class.
- Internship on IoT with Machine learning.

- Machine Learning Master class.
- Electric vehicle Design Master class.
- AI Masterclass.
- Complete Mat lab programming.

In this Internship, I had learnt about Master class on ML and AI. Such that 40 Days of internship has 32 topics and did mini projects and final assessment in the end of internship i.e from 32nd day.

2. AGENDA

Day 1 - Introduction to ML and AI
Day 2 - Python code in google colab, Jupyter notebook Supervised Learning Classification and Regression
Day 3&4 - Python Data Types, Numbers and Casting
Day 5&6 - Python Strings, Booleans, and Operators
Day 7&8 - Lists and Tuples
Day 9 - Sets and Dictionaries
Day 10 - Python String Python If…. else and While loop statements
Day 11 - Python For loop Statements
Day 12&13 - Python functions
Day 14 - Python Arrays
Day 15 - Python Classes and objects
Day 16 - Python Modules
Day 17&18 - Formatting
Day 19&20 - Python Oops Concepts
Day 21 & 22 - Machine learning vs deep learning
Day 23 - Data science library - NumPy
Day 24 - Data science library - pandas
Day 25 - Pandas Data forms

Day 26 - Pandas Data Frames-Data Science Library, Pandas Slicing

Pandas Aggregate functions

Day 27&28 - Python GUI Tkiner

Day 29 - Basics of Pygame Python Library

Day 30 - Credit card Fraud detection (Random Forest)

Day 31 - Advertisement and Sales Prediction (Multiple Linear Regression)

Day 32 - Agriculture Price Prediction - (Random Forest Algorithm)

Day 33 - KNN Programming

Day 34 - Assignment

Day 35 - Introduction Data science

Day 36 - programming

Day 37 - Types of Data Science

3. Topics

3. Introduction to ML and AI

Artificial intelligence generally refers to processes and algorithms that can simulate human intelligence, including mimicking cognitive functions such as perception, learning and problem solving. Machine learning and deep learning (DL) are subsets of AI.

Specific practical applications of AI include modern web search engines, personal assistant programs that understand spoken language, self-driving vehicles, and recommendation engines, such as those used by Spotify and Netflix.

There are four levels or types of AI—two of which we have achieved, and two which remain theoretical at this stage.

4 types of AI

In order from simplest to most advanced, the four types of AI include reactive machines, limited memory, theory of mind and self-awareness.

Reactive machines are able to perform basic operations based on some form of input. At this level of AI, no "learning" happens—the system is trained to do a particular task or set of tasks and never deviates from that. These are purely reactive machines that do not store inputs, have any ability to function outside of a particular context, or have the ability to evolve over time.

Examples of reactive machines include most recommendation engines, IBM’s Deep Blue chess AI, and Google’s AlphaGo AI (arguably the best Go player in the world).

Limited memory AI systems are able to store incoming data and data about any actions or decisions it makes, and then analyse that stored data in order to improve over time. This is where "machine learning" really begins, as limited memory is required in order for learning to happen.

Since limited memory AIs are able to improve over time, these are the most advanced AIs we have developed to date. Examples include self-driving vehicles, virtual voice assistants and chatbots.

Theory of mind is the first of the two more advanced and (currently) theoretical types of AI that we haven't yet achieved. At this level, AIs would begin to understand human thoughts and emotions, and start to interact with us in a meaningful way. Here, the relationship between human and AI becomes reciprocal, rather than the simple one-way relationship humans have with various less advanced AIs now.

The "theory of mind" terminology comes from psychology, and in this case refers to an AI understanding that humans have thoughts and emotions which then, in turn, affect the AI's behaviour.

What is ML?

In a nutshell, machine learning is a subset of AI that falls within the "limited memory" category in which the AI (machine) can learn and develop over time.

There are a variety of different machine learning algorithms, with the three primary types being supervised learning, unsupervised learning, and reinforcement learning.

3 types of machine learning algorithms

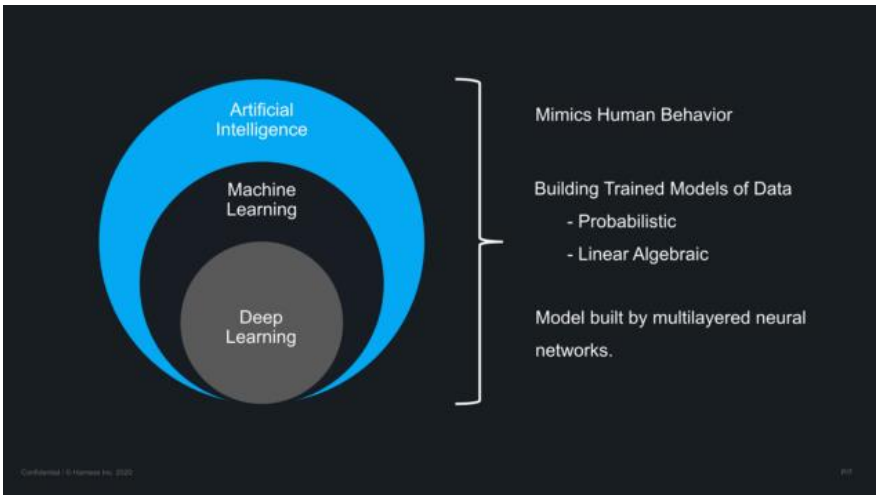
As with the different types of AI, these different types of machine learning cover a range of complexity. And while there are several other types of machine learning algorithms, most are a combination of—or based on—these primary three.

Supervised learning is the simplest of these, and, like it says on the box, is when an AI is actively supervised throughout the learning process. Researchers or data scientists will provide the machine with a quantity of data to process and learn from, as well as some example results of what that data should produce (more formally referred to as inputs and desired outputs).

The result of supervised learning is an agent that can predict results based on new input data. The machine may continue to refine its learning by storing and continually re-analyzing these predictions, improving its accuracy over time.

Unsupervised learning involves no help from humans during the learning process. The agent is given a quantity of data to analyse, and independently identifies patterns in that data. This type of analysis can be extremely helpful, because machines can recognize more and different patterns in any given set of data than humans. Like supervised machine learning, unsupervised ML can learn and improve over time.

Reinforcement learning is the most complex of these three algorithms in that there is no data set provided to train the machine. Instead, the agent learns by interacting with the environment in which it is placed. It receives positive or negative rewards based on the actions it takes and improves over time by refining its responses to maximize positive rewards.



3.1 Numpy

1. Introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects.

2. provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance.

3. many other python libraries are built on NumPy

Arrays

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

3.2 Pandas

DATA FRAMES

Pandas DataFrame is two-dimensional, size-mutable, heterogeneous tabular data structure with labeled rows and columns (axes). Can be thought of a dictionary-like container to store python Series objects.

Python objects have attribute and methods

df.attribute	description
columns	List the column names
axes	List the raw labels and column names
ndim	Number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy repersentation of data

dtypes	List the type of columns

3.3 Data Frame methods

df.methods	Discription
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(),min()	return max/min values for all numeric columns
mean(),median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Why Pandas are used in Python? Is pandas built into Python?

- Pandas used mostly for Data Analysis, provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data.

What does pandas stand for Python?

Pandas is a python package that deals mostly with

- Series (1d homogeneous array)

- Data Frame (2d labeled heterogeneous array)

- Panel (general 3d array)

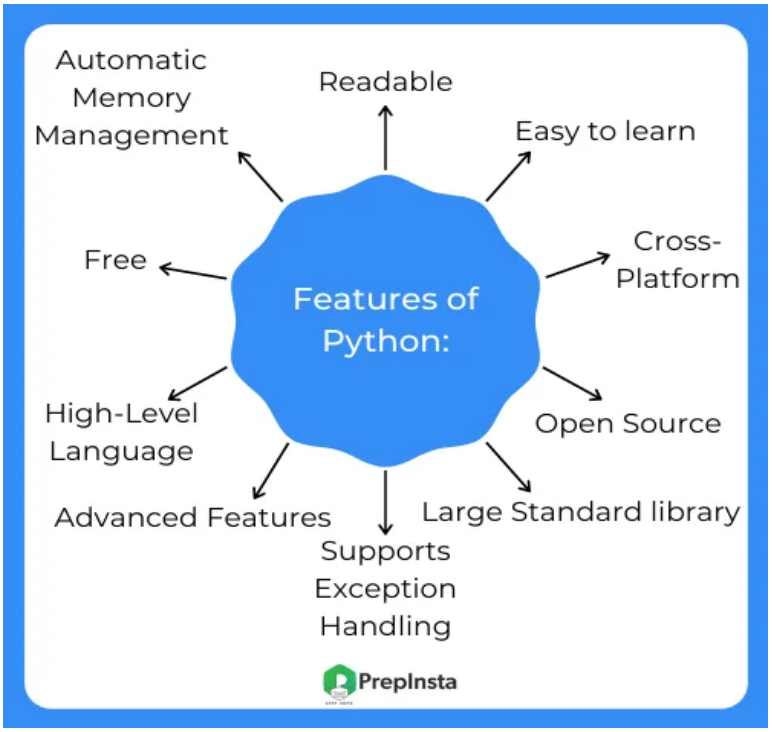
Popular Attributes

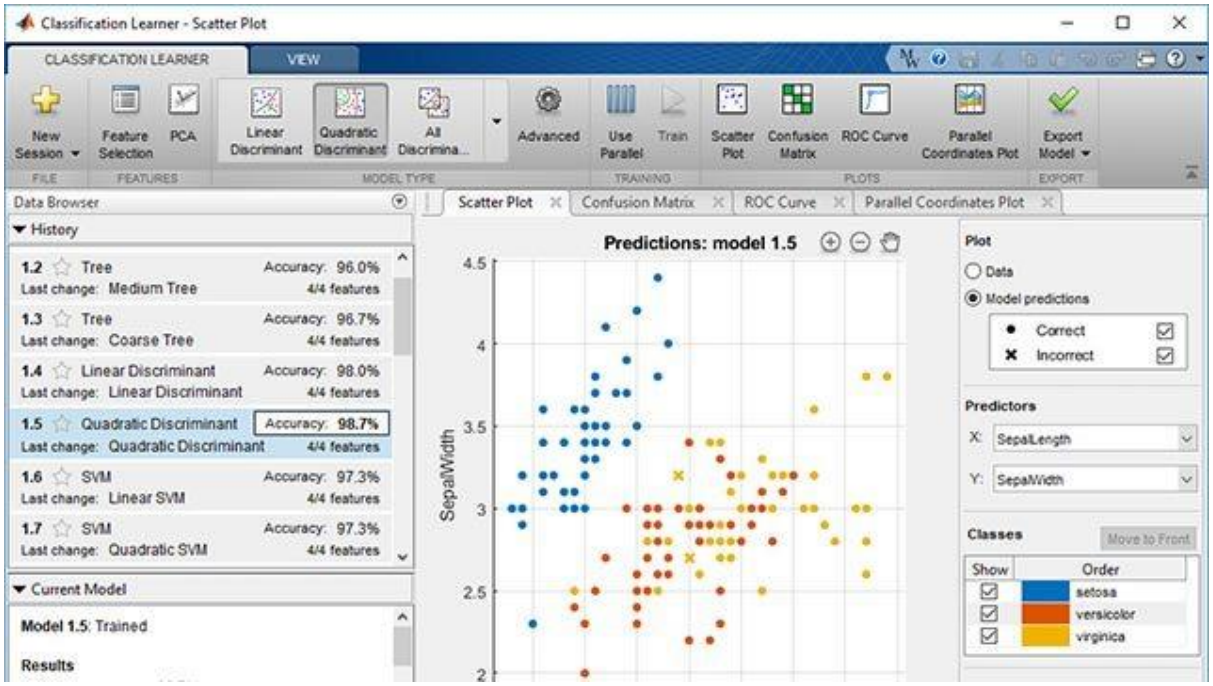
Attribute/Method	***Description***
dtype	data type of values in series
empty	True if series is empty
size	number of elements
values	Returns values as ndarray
head()	First n elements
tail()	Last n elements

4.1 Python History

Python is a widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. The two of the most used versions have to do with Python 2.x & 3.x. There is a lot of competition between the two and both of them seem to have quite a number of different fanbase.

For various purposes such as developing, scripting, generation and software testing, this language is utilised. Due to its elegance and simplicity, top technology organisations like Dropbox, Google, Quora, Mozilla, Hewlett-Packard, Qualcomm, IBM and Cisco have implemented Python.





4.2.Applied Machine Learning

Find the best machine learning models.

Whether you're a beginner looking for some help getting started with machine learning, or an expert looking to quickly assess many different types of models, apps for classification and regression provide quick results. Choose from a wide variety of the most popular classification and regression algorithms, compare models based on standard metrics, and export promising models for further analysis and integration. If writing code is more your style, you can use hyperparameter optimization built into model training functions, so you can quickly find the best parameters to tune your model.

There are many features in Python, some of which are discussed below –

1. Easy to code:

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

2. Free and Open Source:

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword.

Download Python

Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

3. Object-Oriented Language:

One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

4. GUI Programming Support:

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python.

PyQt5 is the most popular option for creating graphical apps with Python.

5. High-Level Language:

Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. Extensible feature:

Python is a Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

7. Python is Portable language:

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

8. Python is Integrated language:

Python is also an Integrated language because we can easily integrate python with other languages like c, c++, etc.

9. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. Unlike other languages C, C++, Java, etc. there is no need to compile python code; this makes it easier to debug our code. The source code of python is converted into an immediate form called **bytecode**.

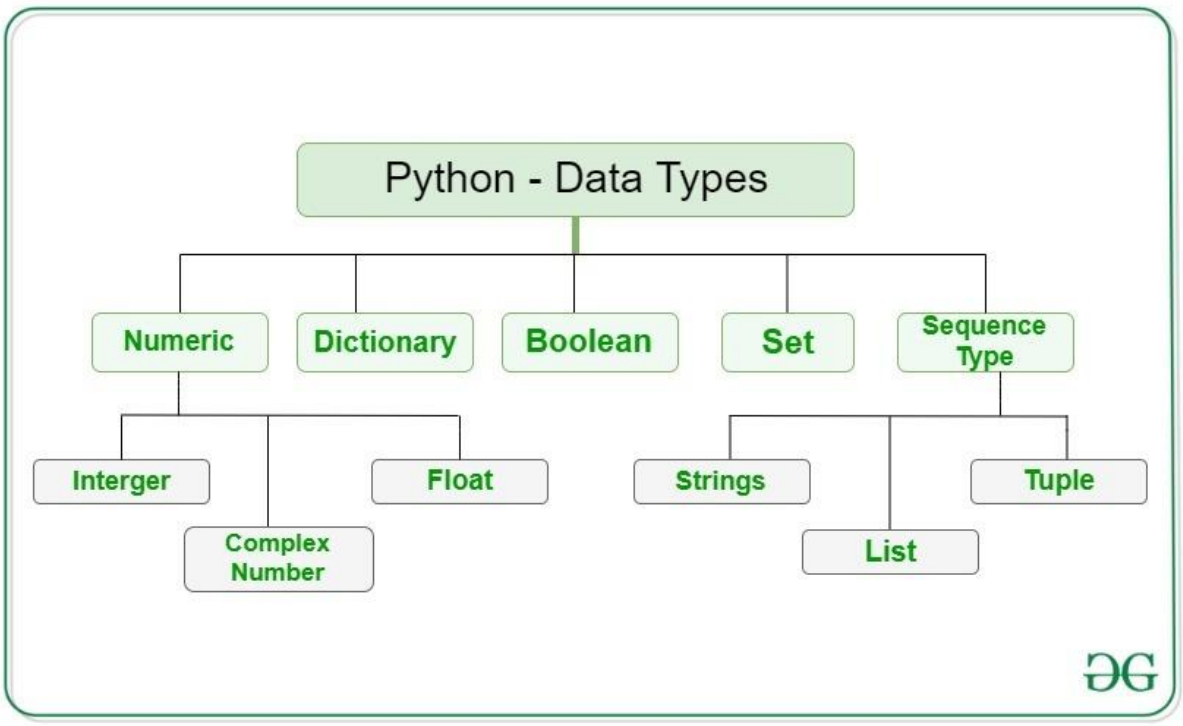
10. Large Standard Library

Python has a large standard library which provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in python such as regular expressions, unit-testing, web browsers, etc.

11. Dynamically Typed Language:

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

4.3 Data Types in Python



4.3.1 Numeric

In Python, numeric data types represent the data which has numeric value. Numeric values can be integers, floating numbers or even complex numbers. These values are defined as int, float and complex classes in Python.

- **Integers** – This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.

- **Float** – This value is represented by the float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
- **Complex Numbers** – Complex numbers are represented by complex classes. It is specified as (real part) + (imaginary part)j. For example – 2+3j.

4.3.2 Sequence Type

In Python, sequence is the ordered collection of similar or different data types. Sequences allow you to store multiple values in an organised and efficient fashion. There are several sequence types in Python –

- String
- List
- Tuple

1) String

In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by the str class.

Creating String

Strings in Python can be created using single quotes or double quotes or even triple quotes.

2) List

Lists are just like the arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

Creating List

Lists in Python can be created by just placing the sequence inside the square brackets[].

3) Tuple

Just like list, tuple is also an ordered collection of Python objects. The only difference between tuple and list is that tuples are immutable i.e. tuples cannot be modified after it is created. It is represented by tuple class.

Creating Tuple

In Python, tuples are created by placing a sequence of values separated by 'comma' with or without the use of parentheses for grouping of the data sequence. Tuples can contain any number of elements and of any datatype (like strings, integers, list, etc.).

4.3.3 Boolean

Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

Note – True and False with capital 'T' and 'F' are valid booleans otherwise python will throw an error.

4.3.4 Set

In Python, Set is an unordered collection of data types that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

Creating Sets

Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by 'comma'. Type of elements in a set need not be the same, various mixed-up data type values can also be passed to the set.

4.3.5 Dictionary

Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only a single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimised. Each key-value pair in a Dictionary is separated by a colon:, whereas each key is separated by a 'comma'.

Creating Dictionary

In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by 'comma'. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated and must be immutable. Dictionary can also be created by the built-in function dict(). An empty dictionary can be created by just placing it in curly braces{}.

5. K-Nearest Neighbor (KNN)

- KNN is simple supervised learning algorithm used for both regression and classification problems.
- KNN is basically store all available cases and classify new cases based on similarities with stored cases.

What is K in KNN

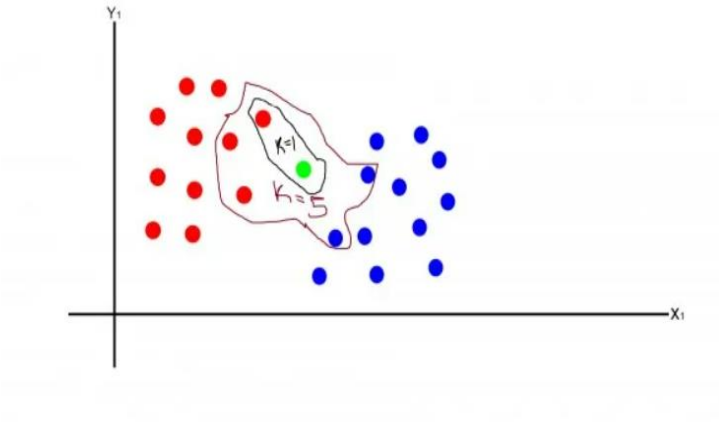
What happens in KNN, we trained the model and after that we want to test our model, means we want to classify our new data (test-data),for that we will check **some (K)** classes around it and assign the most common class to the test-data.

K- Number of nearest neighbours

K=1 means the testing data are given the same level as the closet example in training set.

K=4 means the labels of the four closet classes are check and most common class is assigned to the testing data.

How does KNN is work?



Let's understand it with the above given diagram

1. In this diagram we have 2 classes one blue class one red class

2. Now we have a new green point, we have to find out whether this point is in class red or blue
3. For this, we will define the value of K
4. At K= 1, we will see the distance from the green point to the nearest points, and select the point with lowest distance and classify the green point in that class, here it red.
5. At K=5 We will calculate the distance from the green point to the nearest points and select the five points with the lowest distance and classify the green point to the most common class, that is red here.
6. **How to choose the value of K?** The value of k is not defined, it depends on the cases.

Lazy Learner

1. KNN is simple algorithm for classification but that's not the reason
2. KNN is lazy learner because it doesn't learn a discriminative function from the training data but **memorizes** the training dataset instead.

KNN Algorithm

let's understand the concept of KNN algorithm with **iris flower problem**

Data: This data consist of total 150 instances (samples) , 4 features , and three classes (targets).
Problem: Using four features we must classify which flower belongs to which category.

Importing Dataset

```
import sklearn
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()
iris.keys()
df=pd.DataFrame(iris['data'])
print(df)
print(iris['target_names'])
iris['feature_names']
```

Output

Note:

1. Now we need a target and data so that we can train the model
2. As we know that we must find out the class from the features we have
3. With this logic, our target is classes (0,1,2) and data is in df.

```
X=df
y=iris['target']
```

Splitting Data

1. The data is split so that with some data we can train the model and from the remaining data we can test the model and can check how well our model is

2. To do this we have an inbuilt function in sklearn

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Note: It will split our 33% data into testing data and remaining data is our training data

KNN Classifier and Training of the Model

```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=3)
```

Note:

1. It implements the concepts of KNN. Here we have taken number of neighbors (K)= 3.
2. First, it will calculate the distance with all the training points to the test point and then select the three lowest distance points.
3. And test data point is classify to the class most common in among three.

```
knn.fit(X_train,y_train)
```

Note:- Training the model with features values (data) and target values (target)

Prediction and Accuracy

Demo:

1. Here I want to show you just by taking one data point
2. we have a data point x_new

```
import numpy as np
x_new=np.array([[5,2.9,1,0.2]])
```

Now we want to see the class or category of this point

```
prediction=knn.predict(x_new)
iris['target_names'][prediction]
```

Output

Note: As we can see that our point belongs to class (0 or setosa class), this demo is just for understanding

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from sklearn.metrics import classification_report

y_pred=knn.predict(X_test)

cm=confusion_matrix(y_test,y_pred)

print(cm)

print(" correct predicition",accuracy_score(y_test,y_pred))

print(" worng predicition",(1-accuracy_score(y_test,y_pred)))
```

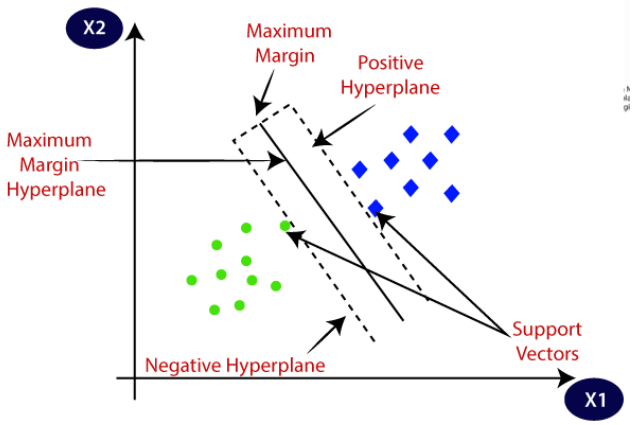
6.SUPPORT VECTOR MACHINE(SVM):

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that

we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



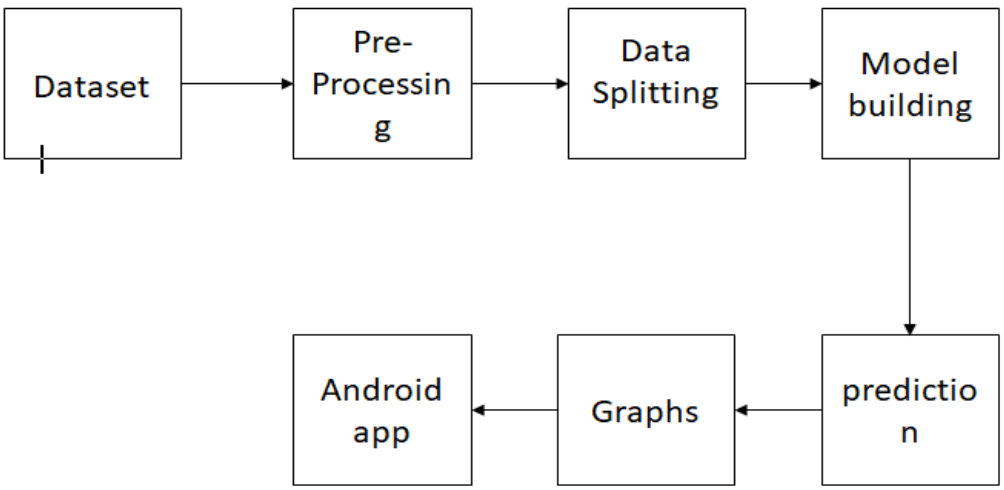
PROJECT

7.Project

Project is done for this Case Study.

Agricultural Price Prediction and Visualization on Android App and Machine Learning AI

System Architecture:



Project code and output:

```
# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Handle table-like data and matrices
import numpy as np
import pandas as pd

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from math import sqrt

import pprint
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split


# Visualisation
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import seaborn as sns
```

Figure 1 Required Libraries

```
df = pd.read_csv("total.csv")
```

```
df.head()
```

	timestamp	state	district	market	commodity	variety	arrival_date	min_price	max_price	modal_price
0	1523025126	Assam	Barpeta	Bohorihat	Black Gram (Urd Beans)	Black Gram (Whole)	06/04/2018	2800.0	3000.0	2900.0
1	1523025126	Assam	Barpeta	Bohorihat	Corriander seed	Coriander Seed	06/04/2018	3500.0	4000.0	3825.0
2	1523025126	Assam	Barpeta	Bohorihat	Jute	TD-5	06/04/2018	3200.0	3875.0	3825.0
3	1523025126	Assam	Barpeta	Bohorihat	Lentil(Masur)	Masoor Gola	06/04/2018	3400.0	3600.0	3500.0
4	1523025126	Assam	Barpeta	Bohorihat	Mustard	Mustard	06/04/2018	3550.0	3650.0	3600.0
...	...	...	...	...	...	...	...	...	...	...
7413	1523025127	West Bengal	Uttar Dinajpur	Islampur	Jute	TD-5	06/04/2018	3700.0	3900.0	3800.0
7414	1523025127	West Bengal	Uttar Dinajpur	Islampur	Potato	Jyoti	06/04/2018	830.0	870.0	850.0
7415	1523025127	West Bengal	Uttar Dinajpur	Islampur	Rice	Other	06/04/2018	3250.0	3350.0	3300.0
7416	1523025127	West Bengal	Uttar Dinajpur	Raiganj	Jute	TD-5	06/04/2018	3600.0	3800.0	3700.0
7417	1523025127	West Bengal	Uttar Dinajpur	Raiganj	Rice	Other	06/04/2018	3150.0	3250.0	3200.0

7418 rows × 10 columns

Figure 2 Preview of Dataset

```
correlation_map = df[df.columns].corr()
obj = np.array(correlation_map)
obj[np.tril_indices_from(obj)] = False
fig,ax= plt.subplots()
fig.set_size_inches(15,10)
sns.heatmap(correlation_map, mask=obj,vmax=.7, square=True,annot=True)
```

```
<AxesSubplot:>
```

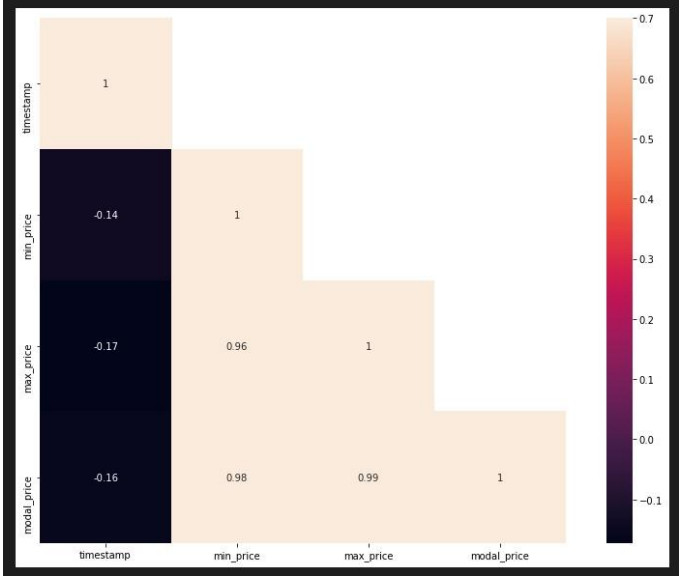


Figure 3 Correlation Heatmap

```
(parameter) commodity: Any
def get_val(state , district , commodity):
    temp = df.loc[(df.state == state) & (df.district == district) & (df.commodity == commodity), : ]
    fig, axes = plt.subplots(6,1)
    fig.set_size_inches(20, 30)
    sns.barplot(data=temp ,y="min_price",x="market" ,orient="v" , ax = axes[0] )
    sns.pointplot(data=temp ,y="min_price",x="market" ,orient="v" , ax = axes[1] )
    #sns.stripplot(x="market", y="min_price", data=temp , jitter=True ,  ax = axes[1])
    #sns.distplot(temp.modal_price , ax = axes[3])
    sns.barplot(data=temp ,y="modal_price",x="market" ,orient="v" , ax = axes[2] )
    sns.pointplot(data=temp ,y="modal_price",x="market" ,orient="v" , ax = axes[3] )
    sns.barplot(data=temp ,y="max_price",x="market" ,orient="v" , ax = axes[4] )
    sns.pointplot(data=temp ,y="max_price",x="market" ,orient="v" , ax = axes[5] )


get_val("Uttar Pradesh" , "Agra" , "Brinjal")
```

Figure 4 Function to plot min, modal and max price of commodity in a district

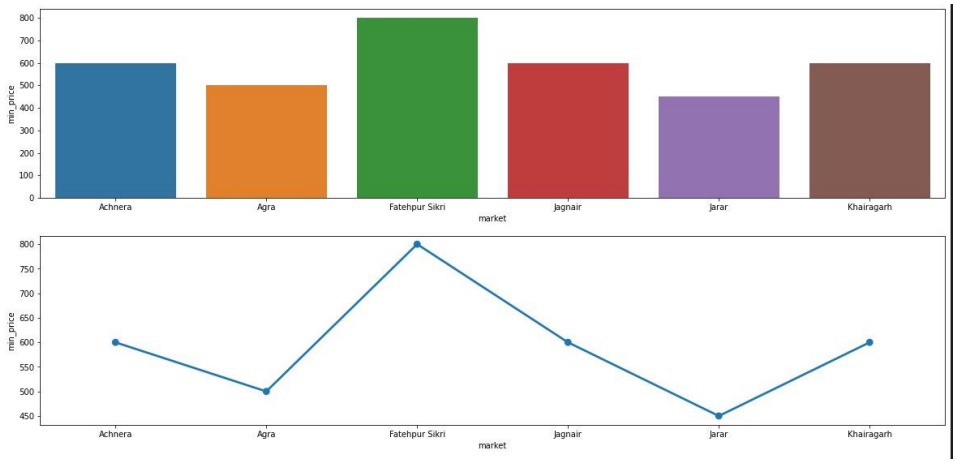


Figure 5 Minimum Price of Brinjal in Agra district

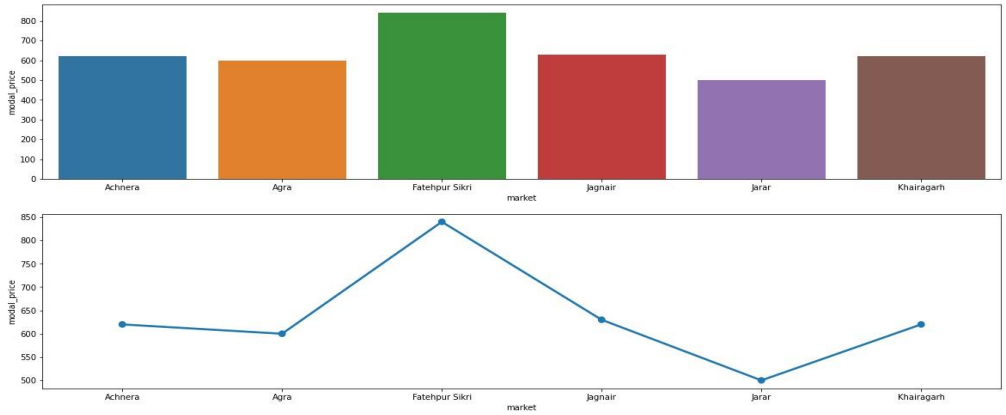


Figure 6 Model Price of Brinjal in Agra district

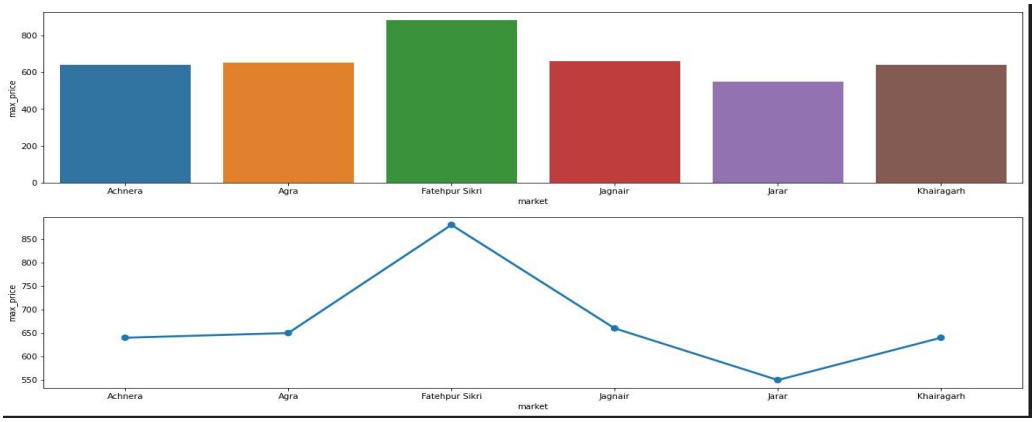


Figure 7 Maximum Price of Brinjal in Agra district

```
width=12
height=10
plt.figure(figsize=(width, height))


ax1 = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(y_predict, hist=False, color="b", label="Fitted Values" , ax=ax1)


plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price')
plt.ylabel('Proportion of Crops')

plt.show()
plt.close()
```

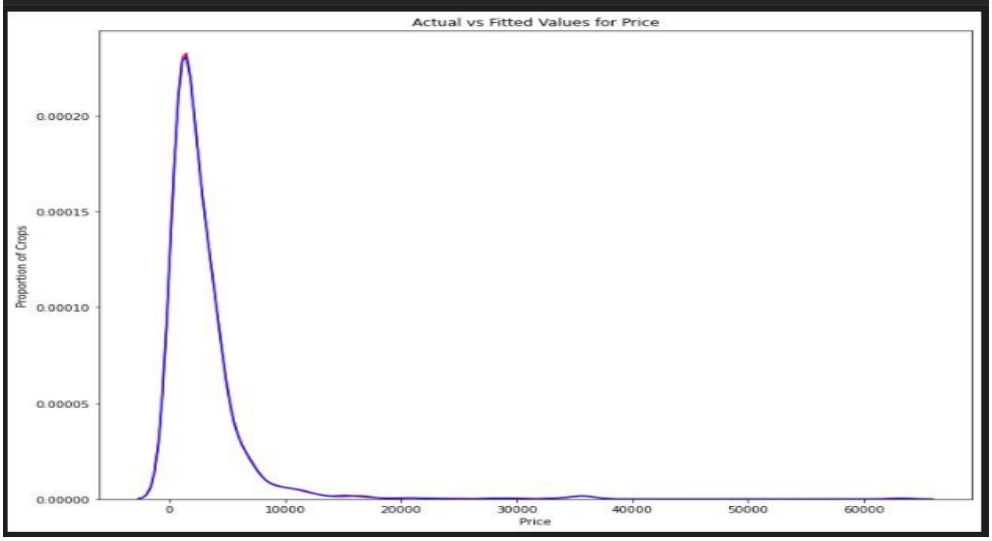


Figure 8 Actual vs Fitted Values for Price

```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
rfr.fit(X_train,y_train)
y_predict1 = rfr.predict(X_test)
rfr.score(X_test,y_test)
```

```
0.9920975126255703
```

Figure 9 Accuracy of model built

END