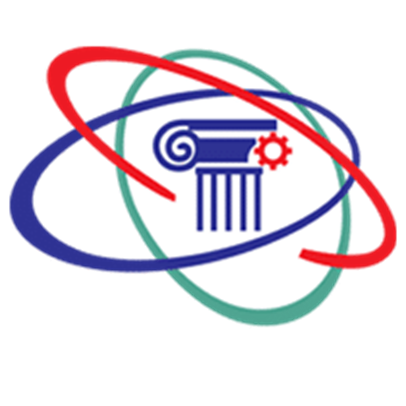
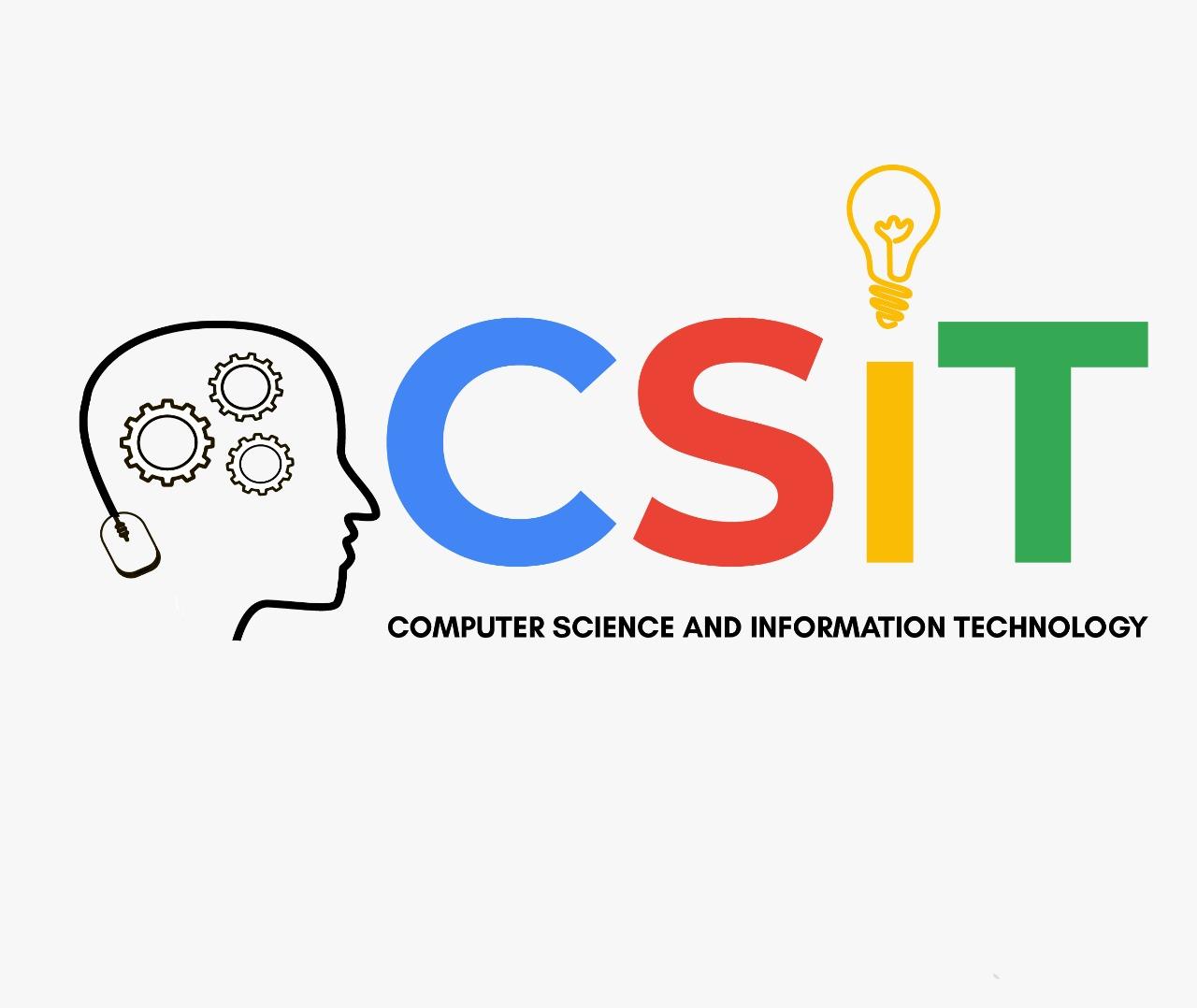
**Acropolis Institute of Technology and Research, Indore**



**Department of Computer Science and Engineering**



**B. Tech. VI Semester(CT/CO)**

**Jan - June 2022**

**Lab Manual** **On** **Machine Learning [CS-601]**

**Submitted to- Submitted By-**

Prof. Vandana Kate Sahil Dubey

Associate Professor, Head ,CSIT 0827CO191049

**LAB WORK**

Programs to be uploaded on Github

**Github Link: https://github.com/sahildubey28/ML**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No | Program | Commit date ( in Github) | Sign of faculty |
| 1 | Basic Python commands |  |  |
| 2 | List Comprehension |  |  |
| 3 | List Comprehension 2 |  |  |
| 4 | Basic of Numpy |  |  |
| 5 | Basic of Pandas |  |  |
| 6 | Program of Linear regression from Scratch |  |  |
| 7 | Program of Linear Regression using Python library (for any given CSV dataset ) |  |  |
| 8 | ANN implementation (Diabetes Dataset) |  |  |
| 9 | ANN implementation use of batch normalization, early stopping and drop out (For Image Dataset) |  |  |
| 10 | Implementation of CNN (using MNIST data Set) |  |  |
| 11 | Implementation of ANN /CNN (using Covid DataSet) |  |  |
| 12 | Implementation of Transfer Learning (VGG 16) |  |  |
| 13 | Implementation of RNN |  |  |
| 14 | Program of Linear regression from Sklearn |  |  |
| 15 | Perceptron |  |  |
| 16 | Convolutional Neural Network |  |  |

**Assignment 1**

**Q1. Explain the 10 function of Numpy**

**Ans -**

**1. Numpy.linspace-**

The numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0) function returns evenly spaced numbers over a specified interval defined by the first two arguments of the function (start and stop — required arguments). The number of samples generated is specified by the third argument num. If omitted, 50 samples are generated. One important thing to bear in mind while working with this function is that the stop element is provided in the returned array (by default endpoint=True), unlike in the built-in python function range.

**2. Numpy.digitize-**

Maybe you have never heard about this function, but it can be really useful working with continuous spaces in reinforcement learning. The numpy.digitize(x, bins, right=False) function has two arguments: (1) an input array x, and (2) an array of bins, returning the indices of the bins to which each value in input array belongs.

**3. Numpy.repeat-**

The numpy.repeat(a, repeats, axis=None) function repeats the elements of an array. The number of repetitions is specified by the second argument repeats.

**4. numpy.random-**

**4.1. Numpy.random.randint-**

The numpy.random.randint(low, high=None, size=None, dtype=’l’) function returns random integers from the interval [low,high). If high parameter is missing (None), the random numbers are selected from the interval [0,low). By default, a single random number(int) is returned. To generate a narray of random integers, the shape of the array is provided in the parameter size.

**4.2. Numpy.random.choice-**

The numpy.random.choice(a, size=None, replace=True, p=None) returns a random sample from a given array. By default, a single value is returned. To return more elements, the output shape can be specified in the parameter size as we did before with the numpy.random.randint function.

**4.3. Numpy.random.binomial-**

We can simulate a wide variety of statistical distributions by using numpy such as normal, beta, binomial, uniform, gamma, or poisson distributions.

**5. Numpy.polyfit-**

The numpy.polyfit(x, y, deg, rcond=None, full=False, w=None, cov=False) function outputs a polynomial of degree deg that fits the points (x,y), minimizing the square error.

This function can be very useful in linear regression problems. Linear regression models the relationship between a dependent variable and an independent variable, obtaining a line that best fits the data.

y =a+bx

where x is the independent variable, y is the dependent variable, b is the slope, and a is the intercept. To obtain both coefficients a and b, we can use the numpy.polyfit function as follows.

**6. Numpy.polyval-**

The numpy.polyval(p, x) function evaluates a polynomial at specific values. Previously, we have obtained a linear model to predict the weight of a man (weight=5.96\*height-224.50) by using the numpy.polyfit function. Now, we use this model to make predictions with the numpy.polyval function. Let’s say we want to predict the weight of a men 70 inches tall. As arguments, we provide the polynomial coefficients (obtained with polyfit) from highest degree to the constant term (p=[5.96,-224.49]), and a number at which to evaluate p (x=70).

**7. Numpy.nan-**

Numpy library includes several constants such as not a number (Nan), infinity (inf) or pi. In computing, not a number is a numeric data type that can be interpreted as a value that is undefined. We can use not a number to represent missing or null values in Pandas. Unfortunately, dirty data sets contain null values with other denominations (e.g. Unknown, — , and n/a), making difficult to detect and drop them.

**8. Numpy.argmax-**

The numpy.argmax(a, axis=None, out=None) function returns the indices of the maximum values along an axis.

**9. Numpy.squeeze-**

The numpy.squeeze(a, axis=None) removes single-dimensional entries from the shape of an array. The argument axis specifies the axis we want to squeeze out. If the shape of the selected axis is greater than 1 a ValueError is raised.

**10. Numpy.histogram-**

The numpy.histogram(a, bins=10, range=None, normed=None, weights=None, density=None) computes the histogram of a set of data. The function returns 2 values: (1) the frequency count, and (2) the bin edges.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q2. Explain concept of Data frame in ML and write 10 commands used in ML from Pandas**

**Ans-**

A DataFrame is a data structure that organizes data into a 2-dimensional table of rows and columns, much like a spreadsheet. DataFrames are one of the most common data structures used in modern data analytics because they are a flexible and intuitive way of storing and working with data.

Every DataFrame contains a blueprint, known as a *schema*, that defines the name and data type of each column. Spark DataFrames can contain universal data types like StringType and IntegerType, as well as data types that are specific to Spark, such as StructType. Missing or incomplete values are stored as *null* values in the DataFrame.

A simple analogy is that a DataFrame is like a spreadsheet with named columns. However, the difference between them is that while a spreadsheet sits on one computer in one specific location, a DataFrame can span thousands of computers. In this way, DataFrames make it possible to do analytics on big data, using distributed computing clusters.

The reason for putting the data on more than one computer should be intuitive: either the data is too large to fit on one machine or it would simply take too long to perform that computation on one machine.

**10 commands used in ML from Pandas -**

## **Pandas import convention**

import pandas as pd

Pandas is now accessible with the acronym pd. You can also install Pandas using the built-in Python tool pip and run the following command.

$ pip install pandas

## **Create and name a Series** - Create one-dimensional array to hold any data type. Invoke the pd.Series() method and then pass a list of values. Pandas will default count index from 0.

series1 = pd.Series([1,2,3,4]), index=['a', 'b', 'c', 'd'])

Set the Series name

srs.name = "Insert name"

Set index name.

srs.index.name = "Index name"

## **3. Create a DataFrame -** Create a two-dimensional data structure with columns. Create and print a df.

df = pd.DataFrame(

{"a" : [1 ,2, 3],

"b" : [7, 8, 9],

"c" : [10, 11, 12]}, index = [1, 2, 3])

## **4. Specify values in DataFrame columns** - Specify how you want to organize your DataFrame by columns.

df = pd.DataFrame(

[[1, 2, 3],

[4, 6, 8],

[10, 11, 12]],

index=[1, 2, 3],

columns=['a', 'b', 'c'])

## **5. Read and Write to CSV fill**

## Open the CSV file, copy the data, paste it in our Notepad, and save it in the same directory that houses your Python scripts. Use read\_csv function build into Pandas and index it the way we want.

import pandas as pd

data = pd.read\_csv('file.csv')

data = pd.read\_csv("data.csv", index\_col=0)

## **6. Get the first 5 elements of a Series**

## Use ser[:n] to get the first *n* elements of a Series.

## **import pandas as pd**

## **df = pd.read\_csv**

## **df['Name'].head(10)**

## **ser[:5]**

## **7. Get the last 5 elements in a Series**

## Use ser[-n:] to get the last *n* elements of a Series.

## **import pandas as pd**

## **df = pd.read\_csv**

## **df['Name'].head(10)**

## **ser[-5:]**

## **8. Drop values from rows**

s.drop(['a', 'c'])

## **9. Drop values from columns**

df.drop('Value', axis=1)

## **10. Select a single value position**

df.iloc[[0],[0]] 'Name'

df.iat([0],[0]) 'Name'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q3. Explain command dataset splitting in python.**

**Ans-**

To train any machine learning model irrespective what type of dataset is being used you have to split the dataset into training data and testing data. So, let us look into how it can be done?

Here I am going to use the iris dataset and split it using the ‘train\_test\_split’ library from sklearn

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_iris

Then I load the iris dataset into a variable.

iris = load\_iris()

Which I then use to store the data and target value into two separate variables

x, y = iris.data, iris.target

Here I have used the ‘train\_test\_split’ to split the data in 80:20 ratio i.e. 80% of the data will be used for training the model while 20% will be used for testing the model that is built out of it.

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=123)

As you can see here I have passed the following parameters in ‘train\_test\_split’:

1. x and y that we had previously defined
2. test\_size: This is set 0.2 thus defining the test size will be 20% of the dataset
3. random\_state: it controls the shuffling applied to the data before applying the split. Setting random\_state a fixed value will guarantee that the same sequence of random numbers are generated each time you run the code.

When splitting a dataset there are two competing concerns:

-If you have less training data, your parameter estimates have greater variance.

-And if you have less testing data, your performance statistic will have greater variance.

The data should be divided in such a way that neither of them is too high, which is more dependent on the amount of data you have. If your data is too small then no split will give you satisfactory variance so you will have to do cross-validation but if your data is huge then it doesn’t really matter whether you choose an 80:20 split or a 90:10 split (indeed you may choose to use less training data as otherwise, it might be more computationally intensive).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

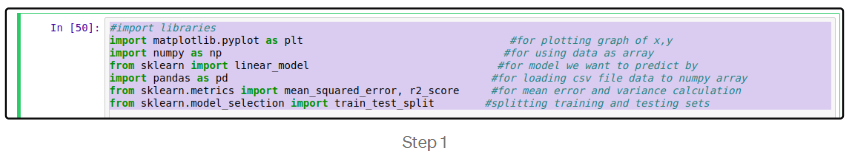
**Q4. Write the command for model creation in linear regression with detail of each parameter**

**Ans -**

**y=mx+c**

where y is the dependent variable, m is slope, x is the independent variable and c is the intercept for a given line.We also have multiple regression model where multiple independent variables are used to calculate one dependent variable. I have used Jupyter Notebook for implementation. Any Python IDE can be used of your choice. So let’s get rolling..

Step 1: Importing libraries

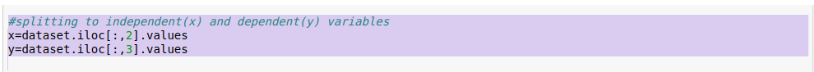
There are already developed libraries in Python for implementation of Machine Learning models. First library called matplotlib is used to plot the graph in last step. ”plt” is used as variable name for using this library in code ahead. sklearn is official machine learning library in python for various model implementation. numpy is used to convert data into arrays for actual use by sklearn library. pandas is used to access .csv file of our dataset.

Step 2: Loading dataset



Our dataset is in a .csv file type. Pandas variable pd is used to access the dataset with read\_csv() function.

Step 3: Split to independent and dependent variables



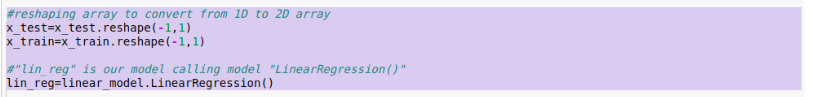
We define x as the independent variable in dataset by iloc(index location) value. [] is used to define array elements. “:” inside [] indicates consider all rows in dataset and separating by using “,” we specify the number of column which we want to use as independent or dependent variable values starting the count from zero in dataset.

Step 4: Splitting data into training and testing data



Now, entire dataset is divided into training and testing set so that prediction does not overfit or underfit and correct values are obtained. train\_test\_split() is inbuilt function from scikit learn for splitting x and y variables data. “test\_size” parameter is used to divide (1/3)rd of entire dataset(30%) into test data and remaining as training data.Setting random\_state as null would not allow random values to be taken from dataset.

Step 5: Choosing the Model



We reshape our independent variable as sklearn expects a 2D array as input.

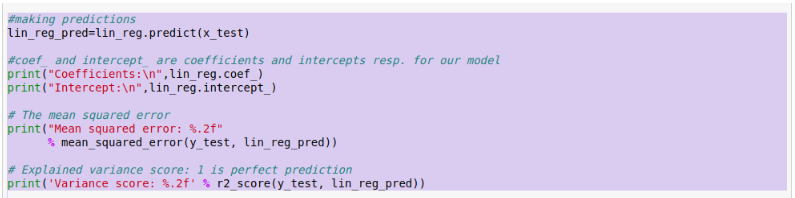
Linear Regression is our model here with variable name of our model as “lin\_reg”. We can try the same dataset with many other models as well. This part varies for any model otherwise all other steps are similar as described here.

Step 6: Fit our model



We now fit our model to the linear regression model by training the model with our independent variable and dependent variables.

Step 7: Predict the output



Finally our model predicts the dependent variable “lin\_reg\_pred” using the test values of independent variable.

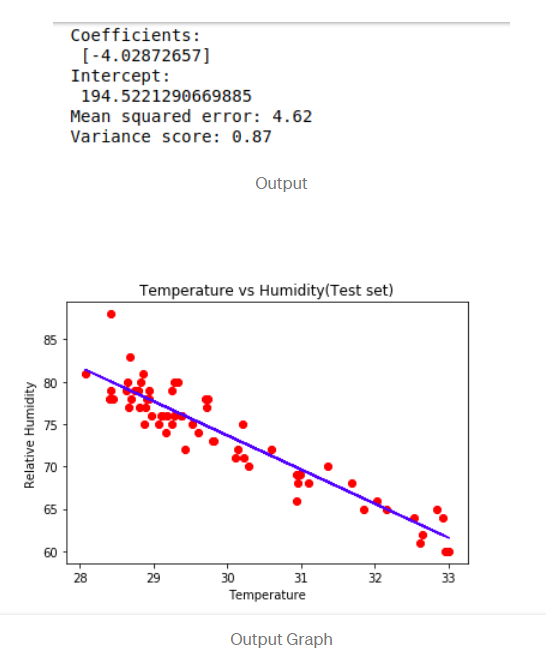
We can see the coefficient,intercept values for our outlier and also the mean squared error and variance for the predicted values(lin\_reg\_pred) and actual test value of dependent variable(y\_test). Inbuilt methods does the math with the predefined formulae for each value.

Step 8: Plot the graph



We ultimately want to visualize the actual data values and predicted data values in a graphical format. “plt”, matplotlib variable, is used to plot points using “scatter()” and outlier using “plot()” functions.

Output might vary depending on various system features. Output I got is as follows:



This is it! Our first linear regression model is implemented! Below is the link to the entire code for the entire model that we implemented here.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q5. What is model.fit() and model.predict()? Explain with example**

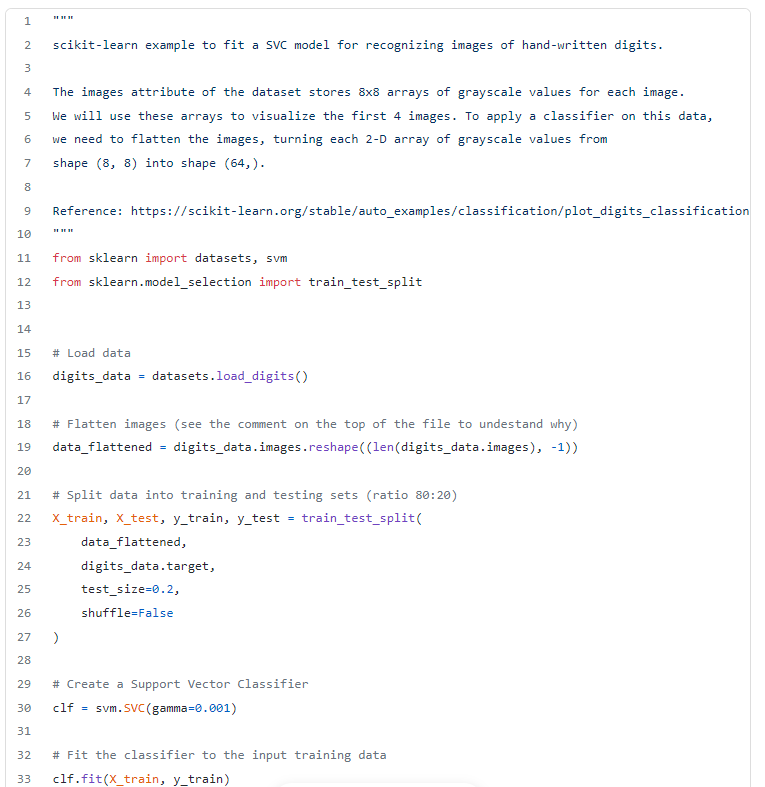
**Ans-**

**model.fit() -**

fit() is implemented by every estimator and it accepts an input for the sample data (X) and for supervised models it also accepts an argument for labels (i.e. target data y ). Optionally, it can also accept additional sample properties such as weights etc.

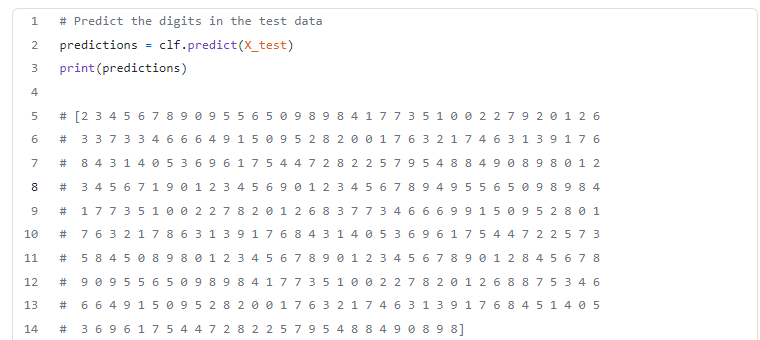
fit methods are usually responsible for numerous operations. Typically, they should start by clearing any attributes already stored on the estimator and then perform parameter and data validation. They are also responsible for estimating the attributes out of the input data and store the model attributes and finally return the fitted estimator.

Now as an example, let’s consider a classification problem where we need to train a SVC model to recognise hand-written images. In the code below, we first load our data and then split it into training and testing sets. Then we instantiate a SVC classifier and finally call fit() to train the model using the input training and data.



**model.predict() -**

Now that we have trained our model, the next step typically involves predictions over the testing set. To do so, we need to call the method predict() that will essentially use the learned parameters by fit() in order to perform predictions on new, unseen test data points.

Essentially, predict() will perform a prediction for each test instance and it usually accepts only a single input (X). For classifiers and regressors, the predicted value will be in the same space as the one seen in training set. In clustering estimators, the predicted value will be an integer. The predicted values of the provided test instances will be returned in a form of an output of an array or sparse matrix.

fit() method will fit the model to the input training instances while predict() will perform predictions on the testing instances, based on the learned parameters during fit.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q6. Explain any pre-trained model and how it is useful in transfer Learning**

**Ans -**

Transfer Learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task.

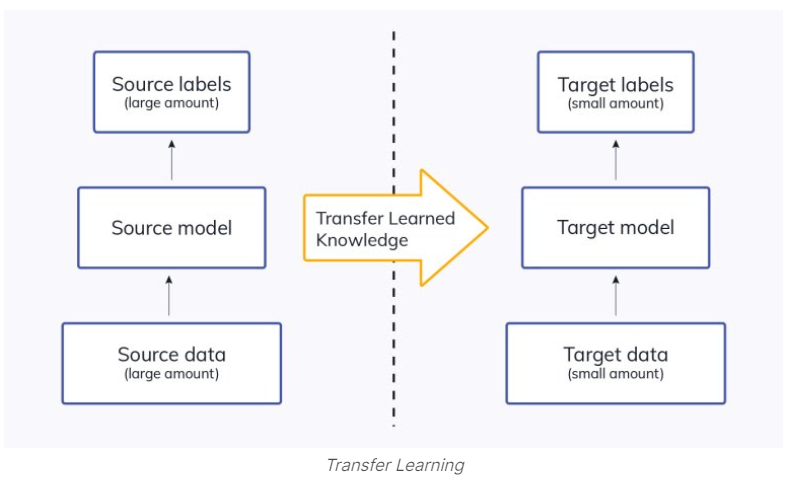
To put it simply—a model trained on one task is repurposed on a second, related task as an optimization that allows rapid progress when modeling the second task.

By applying transfer learning to a new task, one can achieve significantly higher performance than training with only a small amount of data.

Transfer learning is so common that it is rare to train a model for an image or natural language processing-related tasks from scratch.

Instead, researchers and [data scientists](https://www.v7labs.com/blog/data-science-interview-questions-and-answers) prefer to start from a pre-trained model that already knows how to [classify objects](https://www.v7labs.com/blog/image-classification-guide) and has learned general features like edges, shapes in images.

ImageNet, AlexNet, and Inception are typical examples of models that have the basis of Transfer learning.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_