4 WEEKS OF INDUSTRIAL TRAINING REPORT

At



### ASPEXX Health Solution Pvt. Ltd.

**29th August 2021 - 27th September 2021**

**SUBMITTED BY:**

**Program Name: “PYTHON, AI, ML, DS, APP”**

**CERTIFICATION**

This is to certify that this project report entitled “**PYTHON, AI, ML, DS, APP**” submitted *by* ***Archi Agrawal*** is approved for Industrial Internship Programme from

29th August, 2021 to 29th September, 2021 by CUREYA (ASPEXX Health Solution Private Ltd.). During this period of internship, it’s a bonafide record of work carried out under my guidance and supervision.

(Signature of Director & Founder)

Ms. Shivani Mishra

**DECLARATION**

I hereby declare that I have completed my 4 weeks’ industrial training at Cureya

(ASPEXX Health Services Private Ltd.) From 29th August 2021 to 27th September 2021. I declare that I have worked with full dedication throughout the training duration and my learning outcomes fulfil the requirements for the award of certificate in Artificial Intelligence with Python.

**ACKNOWLEDGEMENT**

It’s a privilege to have two months of internship experience at Cureya. It is with a sense of gratitude. I acknowledge the efforts of entire hosts of well-wishers who have in some way or other contributed to the successful completion of the Industrial Internship. Completing any technology training requires help from several people. All that I have done is only due to such supervision and assistance. Thus, I would not forget to thank them. The success of learning Data Science required patience, guidance, and a helping hand from many people. I am privileged to have got this all along with completing this course and a few projects.

First, I express my sense of gratitude and indebtedness to our Founder, **Director, and CEO of Cureya- Ms. Shivani Mishra**. I thank her from the bottom-of-my-heart, for her immensesupport and guidance throughout the training. Without her kind direction and properguidance, this study would have been a little successful. In every phase of the internship, hersupervision and guidance shaped this internship for the perfect completion.Special thanks should go to **Mrs. Anita Mishra, Senior Director of Cureya**, for theenormous support and motivation provided during the internship period. Her guidance wasvaluable to develop me to make better decisions and focus on goals while encounteringproblems to complete the internship.

Finally, I thank and appreciate **Dr. Bajarang Mishra, Coordinator, Supervisor at CureYa**, who’s also the **Editor-in-chief at IEEE Explore and a Professor at JSS Academy of** **Technical Education, Noida**. He conducted weekly meetings superbly and presented weekly report details submitted by interns. He also guided students about research paper publishing to let them achieve higher levels in their careers. I learned a lot from him during a one-to-one discussion on research papers and about publishing research papers and thus feel a great sense of gratitude towards him.

I must say that the **Cureya team and Cureya community** are so down-to-earth, kind, and always ready to help people solve their problems.

I am grateful to all the staff members of **CureYa** and my fellow interns who directly or indirectly contributed to the successful completion of the internship.

**ABSTRACT**

Industrial training is an important phase in the life of career aspirants. A well planned, properly executed and evaluated industrial training helps a lot in developing a professional attitude. It develops an awareness of industrial approach to problem solving, based on a broad understanding of process and mode of operation of organization. The aim and motivation of this industrial training is to receive discipline, skills, teamwork and technical knowledge through a proper training environment, which will help me in the field of **Artificial Intelligence**, to develop a responsiveness of the self-disciplinary nature of problems in data industry. During a period of two months training at **CureYa**, I was trained on different aspects of data science field so I can help the companies advance by helping them to develop strategic plan based on predictive modelling and findings.

The training gave me good experience from the view of implementing my theoretical knowledge in practical aspects. It gave me first-hand experience of working as an engineering professional. It helped me in improving my technical, interpersonal and communication skills, both oral and written. Overall, it is a great experience to have industrial training in such a reputed firm and I believe that it will help me in building a successful career.

**CONTENTS**

| **S. NO.** | **TOPICS** | **PAGE NO.** |
| --- | --- | --- |
| 1. | Certification | 2 |
| 2. | Candidate Declaration | 3 |
| 3. | Acknowledgement | 4 |
| 4. | Abstract | 5 |
| 5. | **CHAPTER 1 - INTRODUCTION** | 9 |
| 6. | * ABOUT CUREYA | 9 |
| 7. | * OBJECTIVE | 9 |
| 8. | * MISSION | 9 |
| 9. | * COMPANY INFORMATION | 9 |
| 10. |  |  |
| 11. | **CHAPTER 2 – INTERNSHIP SCHEDULE** | 10 |
| 12. | * INTERNSHIP DURATION | 10 |
| 13. | * JOB SCOPE | 10 |
| 14. |  |  |
| 15. | **Chapter 3** -**Topics covered in Week 1** | 11 |
| 16. | * AI, ML, DL | 11 |
| 17. | * AI VS ML VS DL VS Data Science | 11 |
| 18. | * Learn Python using Jupyter Notebook | 11 |
| 19. |  |  |

| **S. NO.** | **TOPICS** | **PAGE NO.** |
| --- | --- | --- |
| 20. | **Chapter 4 – Data Pre-Processing** | 14 |
| 21. | * Steps for Data Pre-Processing | 14 |
| 22. | * Get the Dataset | 14 |
| 23. | * Importing Libraries | 15 |
| 24. | * Importing Datasets | 15 |
| 25. | * Finding Missing Data | 18 |
| 26. | * Encoding Categorical Data | 18 |
| 27. | * Splitting dataset into training and test set | 20 |
| 28. | * Feature Scaling | 21 |
| 29. |  |  |
| 30. | **Chapter 5 – Data Visualization** | 26 |
| 31. | * Common general types of data visualization | 26 |
| 32. | * Interactive data visualization | 28 |
| 33. |  |  |
| 34. | **Chapter 6 – Data Mapping** | 29 |
| 35. | * Feature Scaling | 29 |
| 36. | * Why machine learning is important to data mapping | 30 |
| 37. | * Uses of Data Map | 31 |
| 38. | **Chapter 7 – Basics of Neural Networks** | 32 |
| 39. | * Application of Neural Networks | 33 |
| 40. | * Conventional Neural Network | 34 |
| 41. | * Types of CNN | 35 |
| 42. | **Chapter 8 – Image Classification** | 36 |
| 43. | * Supervised Machine learning | 37 |
| 44. | * Model Construction | 38 |
| 45. | * Model Building | 39 |
| 46. | * Conclusion | 47 |
| 47. |  |  |
| 48. |  |  |
| 49. |  |  |
| 50. |  |  |
| 51. |  |  |
| 52. |  |  |
| 53. |  |  |

**CHAPTER 1**

**INTRODUCTION**

* **ABOUT CUREYA**

CUREYA (registered as Aspexx Health solutions Pvt Ltd, under MCA) is DPIIT recognized start up, registered under 'STARTUP INDIA SCHEME'. CUREYA collaborated with stakeholders include – World Yoga associations, Flag bits technologies and many more.

CUREYA provide natural healthcare solutions that aims to reduce the medical health expenditure and eliminate the information asymmetry. Medical practitioners are available at CUREYA via video call, chat, email, or phone call system. CUREYA provide all kinds of medical services related to Ayurveda, Yoga & Naturopathy, Unani, Siddha, and Homoeopathy.

* **OBJECTIVE**

CUREYA’s primary objective is ‘HEALTH FOR ALL’, by reducing the medical expenditure, eliminating the information asymmetry, promoting health awareness and achieving inclusive & holistic approach for healthcare treatments. The objective is to eliminate the information asymmetry, language barrier, and emphasize to achieve global standards of healthcare delivery systems based on access, equity, affordability and quality, efficiency, sustainability.

* **MISSION**

CUREYA’s mission is to achieve the right to "Health for All" and improve the healthcare indicators by dissemination of health education that focuses on health promotion, health prevention, and self- medication.

* **COMPANY INFORMATION**

Company Name: **CureYa**

Founder & CEO: Shivani Mishra

Industry: Healthcare & Lifesciences

Location: Greater Noida, Uttar Pradesh, India

E-mail: info@cureya.in

Phone: 9990375133

* **Useful Links:**

⮚ Website: https://cureya.in

⮚ Facebook: https://www.facebook.com/cureya7/

⮚ LinkedIn: https://www.linkedin.com/company/cureya/

⮚ Instagram: <https://www.instagram.com/cureya.in/?hl=en>

⮚ YouTube: <https://www.youtube.com/channel/UCjsRwGm--mr1ADln5CB5Siw>

**CHAPTER 2**

**INTERNSHIP SCHEDULE**

* **INTERNSHIP DURATION**

It was a 4 weeks’ work from home (WFH) internship. The internship started on 25th August, 2021 and completed on 27th September, 2021. During this period of 4 weeks of internship, firstly, was introduced to basic concepts of artificial intelligence, machine learning, deep learning and data science. And then implement the learning into projects.

* **JOB SCOPE**

I was assigned various tasks and projects during this internship. I completed all the tasks successfully under the guidance of supervisors and mentors. This internship enables me to get relevant jobs in the data industry such as data analyst, data engineer, data scientist, and machine learning engineer.

**CHAPTER 3**

5 **essential skills every well-rounded programmer should know:**

**1.** Version Control (GIT)

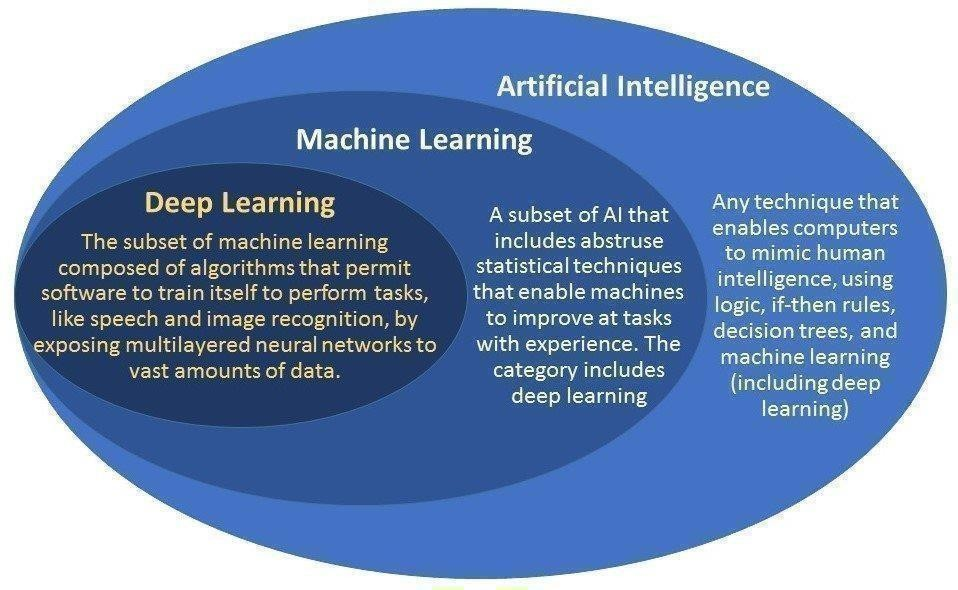
**2.** Databases (SQL) (NO-SQL) (POSTGRES)

**3.** Command Line (Terminal)

**4.** Unit Testing (Bonus: Continuous Integration/Delivery)

**5.** Learn Multiple Languages

* **Difference between AI, ML, Deep Learning:**



* **What is Artificial Intelligence or AI?**

Artificial Intelligence describes machines that can perform tasks resembling those of humans. So AI implies machines that artificially model human intelligence. AI systems help us manage, model, and analyse complex systems. It is the superset which has Machine Learning & Deep Learning as subset.

* **What is Machine Learning or ML?**

Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned

* **What is Deep Learning or DL?**

Deep learning structures algorithms in layers to create an “artificial neural network” that can learn and make intelligent decisions on its own. Deep learning is a subfield of machine learning. While both fall under the broad category of artificial intelligence, deep learning is what powers the most human-like artificial intelligence.

* **What is Data Science?**

Data science is a broad field that spans the collection, management, analysis and interpretation of large amounts of data with a wide range of applications.

It integrates all the terms above and summarizes or extract insights from data (exploratory data analysis) and make predictions from large datasets (predictive analytics).

* **Learn Python using Jupyter Notebook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

**CHAPTER 4**

**Data Pre-processing**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding Missing Data
* Encoding Categorical Data
* Splitting dataset into training and test set
* Feature scaling

**1. Get the Dataset:**

To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the **dataset**.

Dataset may be of different formats for different purposes, such as, if we want to create a machine learning model for business purpose, then dataset will be different with the dataset required for a liver patient. So each dataset is different from another dataset. To use the dataset in our code, we usually put it into a CSV **file**. However, sometimes, we may also need to use an HTML or xlsx file.

CSV File:

CSV stands for "**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs.

We can also create our dataset by gathering data using various API with Python and put that data into a .csv file.

## **2. Importing Libraries:**

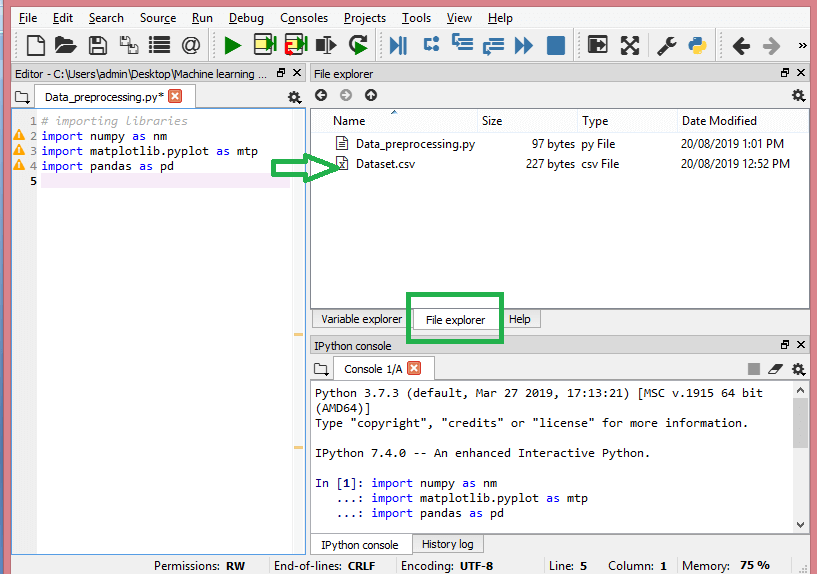
Since Python is the most extensively used and also the most preferred library by Data Scientists around the world, we’ll show you how to import Python libraries for data pre-processing in Machine Learning. Read more about [Python libraries for Data Science here.](https://www.upgrad.com/blog/python-libraries-for-data-science/) The predefined Python libraries can perform specific data pre-processing jobs. Importing all the crucial libraries is the second step in data pre-processing in machine learning. The three core Python libraries used for this data pre-processing in Machine Learning are:

* **NumPy** – NumPy is the fundamental package for scientific calculation in Python. Hence, it is used for inserting any type of mathematical operation in the code. Using NumPy, you can also add large multidimensional arrays and matrices in your code.
* **Pandas** – Pandas is an excellent open-source Python library for data manipulation and analysis. It is extensively used for importing and managing the datasets. It packs in high-performance, easy-to-use data structures and data analysis tools for Python.
* **Matplotlib** – Matplotlib is a Python 2D plotting library that is used to plot any type of charts in Python. It can deliver publication-quality figures in numerous hard copy formats and interactive environments across platforms (IPython shells, Jupyter notebook, web application servers, etc.).

**3. Importing Datasets:**

In this step, you need to import the dataset/s that you have gathered for the ML project at hand. Importing the dataset is one of the important steps in data preprocessing in machine learning. However, before you can import the dataset/s, you must set the current directory as the working directory. You can set the working directory in Spyder IDE in three simple steps:

1. Save your Python file in the directory containing the dataset.
2. Go to File Explorer option in Spyder IDE and choose the required directory.
3. Now, click on the F5 button or Run option to execute the file.



Once you’ve set the working directory containing the relevant dataset, you can import the dataset using the “read\_csv()” function of the Pandas library. This function can read a CSV file (either locally or through a URL) and also perform various operations on it. The read\_csv() is written as:

data\_set= pd.read\_csv(‘Dataset.csv’)

In this line of code, “data\_set” denotes the name of the variable wherein you stored the dataset. The function contains the name of the dataset as well. Once you execute this code, the dataset will be successfully imported.

During the dataset importing process, there’s another essential thing you must do – extracting dependent and independent variables. For every Machine Learning model, it is necessary to separate the independent variables (matrix of features) and dependent variables in a dataset.

**Extracting Independent Variables:**

To extract the independent variables, you can use “iloc[ ]” function of the Pandas library. This function can extract selected rows and columns from the dataset.

x= data\_set.iloc[:,:-1].values

In the line of code above, the first colon(:) considers all the rows and the second colon(:) considers all the columns. The code contains “:-1” since you have to leave out the last column containing the dependent variable. By executing this code, you will obtain the matrix of features, like this –

[[‘India’ 38.0 68000.0]

 [‘France’ 43.0 45000.0]

 [‘Germany’ 30.0 54000.0]

 [‘France’ 48.0 65000.0]

 [‘Germany’ 40.0 nan]

 [‘India’ 35.0 58000.0]

 [‘Germany’ nan 53000.0]

 [‘France’ 49.0 79000.0]

 [‘India’ 50.0 88000.0]

 [‘France’ 37.0 77000.0]]

**Extracting Dependent Variable:**

We can use the “iloc[ ]” function to extract the dependent variable as well. Here’s how you write it:

y= data\_set.iloc[:,3].values

This line of code considers all the rows with the last column only. By executing the above code, you will get the array of dependent variables, like so –

array([‘No’, ‘Yes’, ‘No’, ‘No’, ‘Yes’, ‘Yes’, ‘No’, ‘Yes’, ‘No’, ‘Yes’],

      dtype=object)

**4. Finding Missing Data:**

In data pre-processing, it is pivotal to identify and correctly handle the missing values, failing to do this, you might draw inaccurate and faulty conclusions and inferences from the data. Needless to say, this will hamper your ML project.

Basically, there are two ways to handle missing data:

* **Deleting a particular row** – In this method, you remove a specific row that has a null value for a feature or a particular column where more than 75% of the values are missing. However, this method is not 100% efficient, and it is recommended that you use it only when the dataset has adequate samples. You must ensure that after deleting the data, there remains no addition of bias.
* **Calculating the mean** – This method is useful for features having numeric data like age, salary, year, etc. Here, you can calculate the mean, median, or mode of a particular feature or column or row that contains a missing value and replace the result for the missing value. This method can add variance to the dataset, and any loss of data can be efficiently negated. Hence, it yields better results compared to the first method (omission of rows/columns). Another way of approximation is through the deviation of neighbouring values. However, this works best for linear data.

**5.Encoding Categorical Data:**

Categorical data refers to the information that has specific categories within the dataset. In the dataset cited above, there are two categorical variables – country and purchased.

Machine Learning models are primarily based on mathematical equations. Thus, you can intuitively understand that keeping the categorical data in the equation will cause certain issues since you would only need numbers in the equations.

**Encoding Country Variable:**

As seen in our dataset example, the country column will cause problems, so you must convert it into numerical values. To do so, you can use the LabelEncoder() class from the sci-kit learn library. The code will be as follows –

#Catgorical data

#for Country Variable

from sklearn.preprocessing import LabelEncoder

label\_encoder\_x= LabelEncoder()

x[:, 0]= label\_encoder\_x.fit\_transform(x[:, 0])

**Output:**

Out[15]:

  array([[2, 38.0, 68000.0],

            [0, 43.0, 45000.0],

         [1, 30.0, 54000.0],

         [0, 48.0, 65000.0],

         [1, 40.0, 65222.22222222222],

         [2, 35.0, 58000.0],

         [1, 41.111111111111114, 53000.0],

         [0, 49.0, 79000.0],

         [2, 50.0, 88000.0],

        [0, 37.0, 77000.0]], dtype=object)

Here we can see that the LabelEncoder class has successfully encoded the variables into digits. However, there are country variables that are encoded as 0, 1, and 2 in the output shown above. So, the ML model may assume that there is come some correlation between the three variables, thereby producing faulty output. To eliminate this issue, we will now use Dummy Encoding.

Dummy variables are those that take the values 0 or 1 to indicate the absence or presence of a specific categorical effect that can shift the outcome. In this case, the value 1 indicates the presence of that variable in a particular column while the other variables become of value 0. In dummy encoding, the number of columns equals the number of categories.

Since our dataset has three categories, it will produce three columns having the values 0 and 1. For Dummy Encoding, we will use OneHotEncoder class of the scikit-learn library. The input code will be as follows –

#for Country Variable

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

label\_encoder\_x= LabelEncoder()

x[:, 0]= label\_encoder\_x.fit\_transform(x[:, 0])

#Encoding for dummy variables

onehot\_encoder= OneHotEncoder(categorical\_features= [0])

x= onehot\_encoder.fit\_transform(x).toarray()

**Output:**

array ([[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.80000000e+01,

        6.80000000e+04],

       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.30000000e+01,

        4.50000000e+04],

       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 3.00000000e+01,

        5.40000000e+04],

       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.80000000e+01,

        6.50000000e+04],

       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 4.00000000e+01,

        6.52222222e+04],

       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.50000000e+01,

        5.80000000e+04],

       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 4.11111111e+01,

        5.30000000e+04],

       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.90000000e+01,

        7.90000000e+04],

       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 5.00000000e+01,

        8.80000000e+04],

       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.70000000e+01,

        7.70000000e+04]])

 In the output shown above, all the variables are divided into three columns and encoded into the values 0 and 1.

**Encoding purchased variable:**

For the second categorical variable, that is, purchased, you can use the “labelencoder” object of the LableEncoder class. We are not using the OneHotEncoder class since the purchased variable only has two categories yes or no, both of which are encoded into 0 and 1.

**Input code:**

labelencoder\_y= LabelEncoder()

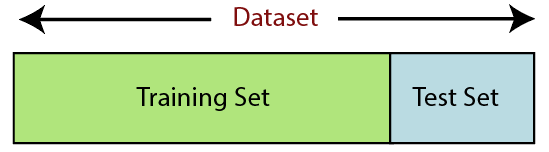
y= labelencoder\_y.fit\_transform(y)

**Output:**

Out[17]: array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1])

**6.** **Splitting Dataset into Training and Test set:**

Splitting the dataset is the next step in data preprocessing in machine learning. Every dataset for Machine Learning model must be split into two separate sets – training set and test set.



Training set denotes the subset of a dataset that is used for training the machine learning model. Here, you are already aware of the output. A test set, on the other hand, is the subset of the dataset that is used for testing the machine learning model. The ML model uses the test set to predict outcomes.

Usually, the dataset is split into 70:30 ratio or 80:20 ratio. This means that you either take 70% or 80% of the data for training the model while leaving out the rest 30% or 20%. The splitting process varies according to the shape and size of the dataset in question.

 To split the dataset, you have to write the following line of code:

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

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.2, random\_state=0)

Here, the first line splits the arrays of the dataset into random train and test subsets. The second line of code includes four variables:

* x\_train – features for the training data
* x\_test – features for the test data
* y\_train – dependent variables for training data
* y\_test – independent variable for testing data

Thus, the train\_test\_split() function includes four parameters, the first two of which are for arrays of data. The test\_size function specifies the size of the test set. The test\_size maybe .5, .3, or .2 – this specifies the dividing ratio between the training and test sets. The last parameter, “random\_state” sets seed for a random generator so that the output is always the same.

7. **Feature Scaling:**

Feature scaling marks the end of thedata pre-processing in Machine Learning**.** It is a method to standardize the independent variables of a dataset within a specific range. In other words, feature scaling limits the range of variables so that you can compare them on common grounds.

Consider this dataset for example –

Graphical user interface

Description automatically generated with medium confidence

In the dataset, you can notice that the age and salary columns do not have the same scale. In such a scenario, if you compute any two values from the age and salary columns, the salary values will dominate the age values and deliver incorrect results. Thus, you must remove this issue by performing feature scaling for Machine Learning.

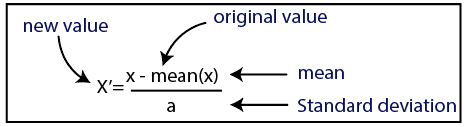
Most ML models are based on Euclidean Distance, which is represented as:

Text

Description automatically generated with medium confidence

We can perform feature scaling in Machine Learning in two ways:

**Standardization:**



**Normalization:**

Diagram

Description automatically generated

For our dataset, we will use the standardization method. To do so, we will import StandardScaler class of the sci-kit-learn library using the following line of code:

from sklearn.preprocessing import StandardScaler

The next step will be to create the object of StandardScaler class for independent variables. After this, you can fit and transform the training dataset using the following code:

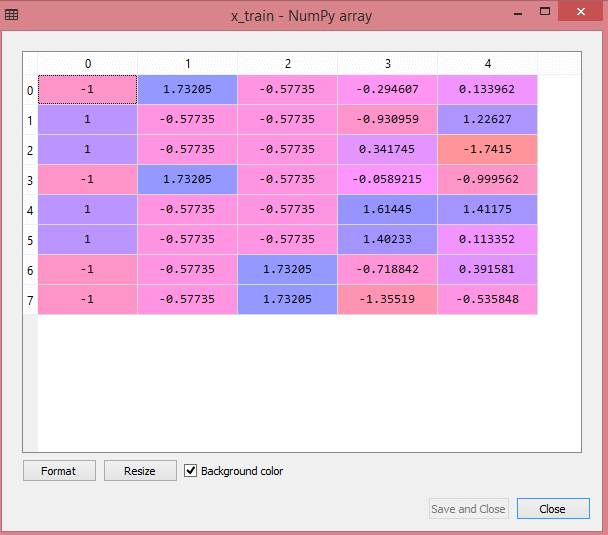
st\_x= StandardScaler()

x\_train= st\_x.fit\_transform(x\_train)

For the test dataset, you can directly apply transform() function (you need not use the fit\_transform() function because it is already done in training set). The code will be as follows –

x\_test= st\_x.transform(x\_test)

The output for the test dataset will show the scaled values for x\_train and x\_test as:



Graphical user interface, application, table

Description automatically generated

All the variables in the output are scaled between the values -1 and 1.

Combining all the steps we get:

# importing libraries

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

#importing datasets

data\_set= pd.read\_csv(‘Dataset.csv’)

#Extracting Independent Variable

x= data\_set.iloc[:, :-1].values

#Extracting Dependent variable

y= data\_set.iloc[:, 3].values

#handling missing data(Replacing missing data with the mean value)

from sklearn.preprocessing import Imputer

imputer= Imputer(missing\_values =’NaN’, strategy=’mean’, axis = 0)

#Fitting imputer object to the independent varibles x.

imputerimputer= imputer.fit(x[:, 1:3])

#Replacing missing data with the calculated mean value

x[:, 1:3]= imputer.transform(x[:, 1:3])

#for Country Variable

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

label\_encoder\_x= LabelEncoder()

x[:, 0]= label\_encoder\_x.fit\_transform(x[:, 0])

#Encoding for dummy variables

onehot\_encoder= OneHotEncoder(categorical\_features= [0])

x= onehot\_encoder.fit\_transform(x).toarray()

#encoding for purchased variable

labelencoder\_y= LabelEncoder()

y= labelencoder\_y.fit\_transform(y)

# Splitting the dataset into training and test set.

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

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.2, random\_state=0)

#Feature Scaling of datasets

from sklearn.preprocessing import StandardScaler

st\_x= StandardScaler()

x\_train= st\_x.fit\_transform(x\_train)

x\_test= st\_x.transform(x\_test)

**CHAPTER 5**

**Data Visualization**

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

In the world of Big Data, data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions.

Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It’s storytelling with a purpose. If you’ve ever stared at a massive spreadsheet of data and couldn’t see a trend, you know how much more effective a visualization can be.

**Common general types of data visualization:**

* + - Charts
    - Tables
    - Graphs
    - Maps
    - Infographics
    - Dashboards

### More specific examples of methods to visualize data:

### Area Chart

### Bar Chart

### Box-and-whisker Plots

### Bubble Cloud

### Bullet Graph

### Cartogram

* + Circle View
  + Dot Distribution Map
  + Gantt Chart
  + Heat Map
  + Highlight Table
  + Histogram
  + Matrix
  + Network
  + Polar Area
  + Radial Tree
  + Scatter Plot (2D or 3D)
  + Streamgraph
  + Text Tables
  + Timeline
  + Treemap
  + Wedge Stack Graph

**Interactive data visualization** enables direct actions on a graphical [plot](https://en.wikipedia.org/wiki/Plot_(graphics)) to change elements and link between multiple plots.[[33]](https://en.wikipedia.org/wiki/Data_visualization#cite_note-33)

Interactive data visualization has been a pursuit of [statisticians](https://en.wikipedia.org/wiki/Statisticians) since the late 1960s. Examples of the developments can be found on the [American Statistical](https://en.wikipedia.org/wiki/American_Statistical_Association) [Association v](https://en.wikipedia.org/wiki/American_Statistical_Association)ideo lending library.[[34]](https://en.wikipedia.org/wiki/Data_visualization#cite_note-34) Common interactions include:

* [**Brushing**:](https://en.wikipedia.org/wiki/Brushing_and_linking) works by using the [mouse](https://en.wikipedia.org/wiki/Computer_mouse) to control a paintbrush, directly changing the color or glyph of elements of a plot. The paintbrush is sometimes a pointer and sometimes works by drawing an outline of sorts around points; the outline is sometimes irregularly shaped, like a lasso. Brushing is most commonly used when multiple plots are visible and some linking mechanism exists between the plots. There are several different conceptual models for brushing and a number of common linking mechanisms. Brushing [scatterplots c](https://en.wikipedia.org/wiki/Scatterplots)an be a transient operation, in which points in the active plot only retain their new characteristics while they are enclosed or intersected by the brush, or it can be a persistent operation, so that points retain their new appearance after the brush has been moved away. Transient brushing is usually chosen for linked brushing, as we have just described.
* **Painting**: Persistent brushing is useful when we want to group the points into clusters and then proceed to use other operations, such as the tour, to compare the groups. It is becoming common terminology to call the persistent operation painting,
* **Identification**: which could also be called labeling or label brushing, is another plot manipulation that can be linked. Bringing the cursor near a point or edge in a scatterplot, or a bar in a [barchart,](https://en.wikipedia.org/wiki/Barchart) causes a label to appear that identifies the plot element. It is widely available in many interactive graphics, and is sometimes called mouseover.

**CHAPTER 6**

**Data Mapping**

Data mapping is a way to [organize various bits of data](https://dzone.com/articles/what-is-data-mapping) into a manageable and easy-to- understand system. This system matches data fields with target fields while in storage. Simply put, not all data goes by the same organizational standards. They may refer to a phone number in as many different ways as you can think of. Data mapping recognizes phone numbers for what they are and puts them all in the same field rather than having them drift around by other names.

With this technique, we're able to take the organized data and put a bigger picture together. You can find out where most of your target audience lives, learn what sorts of things they have in common and even figure out a few controversies that you shouldn't touch on. Armed with this information, your business can make smarter decisions and spend less money while spinning your products and services to your audience.

* **Data Mapping and Machine Learning**

The earlier example of recognizing phone numbers has a lot to do with something called [unification and data cleaning.](https://towardsdatascience.com/machine-learning-for-data-cleaning-and-unification-b3213bbd18e) These processes are often powered by machine learning, which is not to be confused with artificial intelligence.

Machine learning uses patterns and inference to offer predictions rather than perform a single task, which is more of a subset of AI technology than anything. In the earlier example, machine learning is used to recognize a phone number and assign it to its proper category for organizational purposes.

Machine learning goes a step beyond just recognizing phone numbers though. The technology can recognize errors like missing values or typos and group information from the same source together

That's what data cleaning and unification really means — to clean up all of the data without any human input and present the information in its most perfect and precise form. This process saves time and is also more effective in regard to how correct the information will be.

The data can then be displayed in almost any way a person or company needs to see it. For instance, geospatial data is one route machine learning can automatically take and create without input. Geospatial data is basically [translating data into a map a](https://www.mapbusinessonline.com/Whitepaper.aspx/How-Businesses-Use-Geo-Spatial-Data)nd plotting out physical locations and routes that your target audience takes every day. This technique can provide a unique aid to your next advertising campaign.

* **Why Machine Learning Is Important to Data Mapping**

Machine learning allows data mapping to be more precise. Without that technology, data mapping would be either very rudimentary or have to be done completely manually. Assuming we go the rudimentary route, a simple spreadsheet would be able to take information and plug into its best guess of a proper category. Typos wouldn't be fixed, missing values would remain missing and some information would just be scattered in random places.

Trying to complete data mapping manually would be worse. For one, a person would never be able to keep up with the flow of information, not to mention the backlog of information already hiding and in need of sorting in the Internet of Things. Assuming someone could keep up with the flow, there would still be errors as the sheer amount of data would lead to the human being unable to notice connections like a machine could.

* **Why Data Mapping Is Important**

The use of data is an extremely important part of modern-day marketing. Knowing the best possible place and time to reach customers will allow you to target your audience more efficiently. Even large industries that can afford to splay their names in all possible media outlets use data mapping to save money and appear more loyal to their customer base.

Big or small, you can use this information and get ahead of everyone else vying for your customers' attention. The competition is dense these days, so getting ahead of the curve and staying ahead is an art everyone is trying to perfect. Data mapping can help you get there as early as possible.

* **Uses of Data Map**

**Population Distribution**

According to demographic data such as age, gender, income, education level, etc., analyze and classify customers in different regions or communities on the map. Data can help us figure out their lifestyle, interests and shopping habits.

**Market Capacity Forecast**

Analyse the resource investments, sales revenue, and product sales of each outlet on the map, and predict the capacity of the entire market, so that the resources can be scientifically allocated to the region with the greatest market potential.

**CHAPTER 7**

**Basics of Neural Networks**

Neural networks, in the world of finance, assist in the development of processes such as time-series forecasting, [algorithmic trading,](https://www.investopedia.com/terms/a/algorithmictrading.asp) securities classification, credit risk modeling and constructing proprietary indicators and price [derivatives.](https://www.investopedia.com/terms/d/derivative.asp)

A neural network works similarly to the human brain’s neural network. A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a [multiple linear regression.](https://www.investopedia.com/terms/m/mlr.asp) The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptron are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. For instance, the patterns may comprise a list of quantities for [technical indicators](https://www.investopedia.com/terms/t/technicalindicator.asp) about a security; potential outputs could be “buy”,” hold” or “sell.”

Hidden layers fine-tune the input weightings until the neural network’s margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

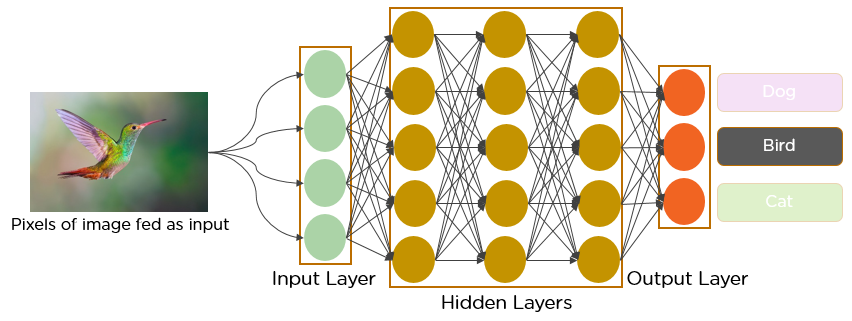
* **Application of Neural Networks**

**N**eural networks are broadly used, with applications for financial operations, enterprise planning, trading, business analytics and product maintenance. Neural networks have also gained widespread adoption in business applications such as forecasting and marketing research solutions, fraud detection and risk assessment.

A neural network evaluates price data and unearths opportunities for making trade decisions based on the data analysis. The networks can distinguish subtle nonlinear interdependencies and patterns other methods of technical analysis cannot. According to research, the accuracy of neural networks in making price predictions for stocks differs. Some models predict the correct stock prices 50 to 60 percent of the time while others are accurate in 70 percent of all instances. Some have posited that a 10 percent improvement in efficiency is all an investor can ask for from a neural network.

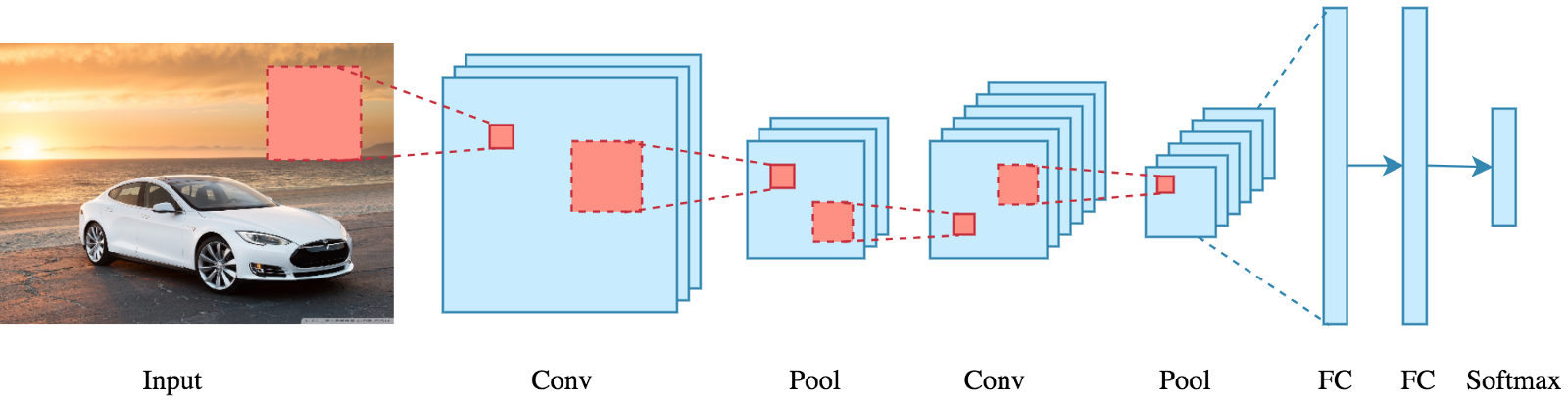
## **Convolutional Neural Networks (CNN)**

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks



* **What exactly is a CNN?**

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.



## **How do convolutional neural networks work?**

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

* Convolutional layer
* Pooling layer
* Fully-connected (FC) layer
  + **Convolutional Layer**

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let’s assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also

determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

* + - **Pooling Layer**

Pooling layers, also known as down sampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling:

* **Max pooling:** As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.
* **Average pooling:** As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.
  + - **Fully-Connected Layer**

The name of the full-connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully-connected layer, each node in the output layer connects directly to a node in the previous layer.

This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

**CHAPTER 8**

**IMAGE CLASSIFICATION**

CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google — for search through among users’ photos.

Let us consider the use of CNN for image classification in more detail. The main task of image classification is acceptance of the input image and the following definition of its class. This is a skill that people learn from their birth and are able to easily determine that the image in the picture is an elephant. But the computer sees the pictures quite differently:

A picture containing text

Description automatically generated

Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.

To solve this problem the computer looks for the characteristics of the base level. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.

In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

**The** **Convolution layer** is always the first. The image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of image. Next the software selects a smaller matrix there, which is called a **filter**(or neuron, or core). Then the filter produces convolution, i.e. moves along the input image. The filter’s task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller then a input matrix.

Text

Description automatically generated with low confidence

This operation, from a human perspective, is analogous to identifying boundaries and simple colours on the image. But in order to recognize the properties of a higher level such as the trunk or large ears the whole network is needed.

The network will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.

**The nonlinear layer**is added after each convolution operation. It has an activation function, which brings nonlinear property. Without this property a network would not be sufficiently intense and will not be able to model the response variable (as a class label).

**The pooling layer** follows the nonlinear layer. It works with width and height of the image and performs a downsampling operation on them. As a result the image volume is reduced. This means that if some features (as for example boundaries) have already been identified in the previous convolution operation, than a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach **a** **fully connected layer**. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the amount of classes from which the model selects the desired class.

A fragment of the code of this model written in Python will be considered further in the practical part.

**Process carried out:**

**Supervised machine learning**

It is one of the ways of machine learning where the model is trained by input data and expected output data.

Тo create such model, it is necessary to go through the following phases:

1. model construction
2. model training
3. model testing
4. model evaluation

**Model construction** depends on machine learning algorithms. In this projects case, it was neural networks.

Such an algorithm looks like:

1. begin with its object: model = Sequential()
2. then consist of layers with their types: model.add(*type\_of\_layer()*)
3. after adding a sufficient number of layers the model is compiled. At this moment Keras communicates with TensorFlow for construction of the model. During model compilation it is important to write a loss function and an optimizer algorithm. It looks like: model.comile(loss= ‘name\_of\_loss\_function’, optimizer= ‘name\_of\_opimazer\_alg’ ) The loss function shows the accuracy of each prediction made by the model.

Before model training it is important to scale data for their further use.

After model construction it is time for **model training.**In this phase, the model is trained using training data and expected output for this data.

It’s look this way: model.fit(training\_data, expected\_output).

Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model.

Once the model has been trained it is possible to carry out **model testing.**During this phase a second set of data is loaded. This data set has never been seen by the model and therefore it’s true accuracy will be verified.

After the model training is complete, and it is understood that the model shows the right result, it can be saved by: model.save(“name\_of\_file.h5”).

Finally, the saved model can be used in the real world. The name of this phase is **model evaluation**. This means that the model can be used to evaluate new data.

**Classification Model (elephants vs cars)**

Here I would like to describe the code that was taken as the basis of this project. It is considered that a deep learning model needs a large amount of data. But the model given in this script is excellent for training with a small amount of data. Because of that I took only 200 photos per class for training and 80 photos per class for expected output during training.

Using little data is possible when the image is pre-processing with Keras Image Data Generator class. Тhis class can create a number of random transformations, which helps to increase the number of images when it is needed.

**from** keras.preprocessing.image **import** ImageDataGenerator, array\_to\_img, img\_to\_array, load\_img  
  
datagen = ImageDataGenerator(  
 rotation\_range=40,  
 width\_shift\_range=0.2,  
 height\_shift\_range=0.2,  
 shear\_range=0.2,  
 zoom\_range=0.2,  
 horizontal\_flip=**True**,  
 fill\_mode=**'nearest'**)img = load\_img(**'train/elephants/adventure-1822636\_640.jpg'**) *# this is a PIL image*x = img\_to\_array(img) *# this is a Numpy array with shape (300, 300, 3)*x = x.reshape((1,) + x.shape) *# this is a Numpy array with shape (1, 300, 300, 3)*x.shape  
  
*# the .flow() command below generates batches of randomly transformed images  
# and saves the results to the `preview/` directory*i = 0  
**for** batch **in** datagen.flow(x, batch\_size=1,  
 save\_to\_dir=**'preview'**, save\_prefix=**'el'**, save\_format=**'jpeg'**):  
 i += 1  
 **if** i > 20:  
 **break** *# otherwise the generator would loop indefinitely*

ImageDataGenerator has the following arguments:

1. **rotation\_range** — which is used for random rotations, given in degrees in the range from 0 to 180
2. **width\_shift\_range**— which is shown in fraction of total width, used for random horizontal shifts
3. **height\_shift\_range** — which is the same as width\_shift\_range, but with height
4. **shear\_range** — shear intensity, used for linear mapping that displaces each point in a fixed direction
5. **zoom\_range** — use for random zooming
6. **horizontal\_flip** — unlike other arguments has boolean type, used for randomly flipping inputs horizontally
7. **fill\_mode** — can be “constant”, “reflect”, “wrap” or “nearest” as in this case; indicates the method of filling the newly formed pixels
8. These are not all the arguments that could be used, the further ones can be found [here](https://keras.io/preprocessing/image/).

To specify the input directory **load\_image**is used. Also load\_imagemeans that image will load to PIL format.

**Image\_to\_array** means that image in PIL format returns a 3D Numpy array, which will be reshaped on further.

Then in the loop with **flow(x,y)** method, the image transformation takes place. Random transformations are stored in the “preview” folder and look like:

A picture containing text, different

Description automatically generated

The following code fragment will describe construction of the model. Here the layers begin to be added. This architecture was made on the principle of convolutional neural networks. It consists of 3 groups of layers, where the convolution layers (Conv 2D) alternate with the nonlinear layers (Relu) and the pooling layers (Max Pooling 2D). It then follows 2 tightly bound layers (Dense). Consider their structure in more detail.

*# MODEL***from** keras.models **import** Sequential  
**from** keras.layers **import** Conv2D, MaxPooling2D  
**from** keras.layers **import** Activation, Dropout, Flatten, Dense  
  
model = Sequential()  
model.add(Conv2D(32, (3, 3), input\_shape=(300, 300, 3)))  
model.add(Activation(**'relu'**))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
  
model.add(Conv2D(32, (3, 3)))  
model.add(Activation(**'relu'**))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
  
model.add(Conv2D(64, (3, 3)))  
model.add(Activation(**'relu'**))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
  
*# the model so far outputs 3D feature maps (height, width, features)*model.add(Flatten()) *# this converts our 3D feature maps to 1D feature vectors*model.add(Dense(64))  
model.add(Activation(**'relu'**))  
model.add(Dropout(0.5))  
model.add(Dense(1))  
model.add(Activation(**'sigmoid'**))  
*# COMPILE*model.compile(loss=**'binary\_crossentropy'**,  
optimizer=**'rmsprop'**,  
metrics=[**'accuracy'**])

Let us look at the first convolution layer **Conv 2D.**The number 32 shows the amount of output filter in the convolution. Numbers 3, 3 correspond to the kernel size, which determinate the width and height of the 2D convolution window. An important component of the first convolution layer is an input shape, which is the input array of pixels. Further convolution layers are constructed in the same way, but do not include the input shape.

The activation function of this model is **Relu**. This function setts the zero threshold and looks like: f(x) = max(0,x). If x > 0 — the volume of the array of pixels remains the same, and if x < 0 — it cuts off unnecessary details in the channel.

**Max Pooling 2D** layer is pooling operation for spatial data. Numbers 2, 2 denote the pool size, which halves the input in both spatial dimension.

After three groups of layers there are two **fully connected layers**. Flatten performs the input role. Next is Dense — densely connected layer with the value of the output space (64) and Relu activation function. It follows Dropout, which is preventing overfitting. **Overfitting** is the phenomenon when the constructed model recognizes the examples from the training sample, but works relatively poorly on the examples of the test sample. Dropout takes value between 0 and 1. Тhe last fully connected layer has 1 output and Sigmoid activation function.

Next step is model **compiling.** It has a binary cross entropy loss function, which will show the sum of all individual losses. The optimizer algorithm is RMSprop, which is good for recurrent neural networks. The accuracy metrics shows the performance of the model.

The following code fragment prepares the model for training:

batch\_size = 16  
  
*# this is the augmentation configuration we will use for training*train\_datagen = ImageDataGenerator(  
 rescale=1./255,  
 shear\_range=0.2,  
 zoom\_range=0.2,  
 horizontal\_flip=**True**)  
  
*# this is the augmentation configuration we will use for testing:  
# only rescaling*test\_datagen = ImageDataGenerator(rescale=1./255)  
  
*# this is a generator that will read pictures found in subfolers of 'data/train', and indefinitely generate  
# batches of augmented image data*train\_generator = train\_datagen.flow\_from\_directory(  
 **'train'**, *# this is the target directory* target\_size=(300, 300), *# all images will be resized to 300x300* batch\_size=batch\_size,  
 class\_mode=**'binary'**) *# since we use binary\_crossentropy loss, we need binary labels  
  
# this is a similar generator, for validation data*validation\_generator = test\_datagen.flow\_from\_directory(  
 **'validation'**,  
 target\_size=(300, 300),  
 batch\_size=batch\_size,  
 class\_mode=**'binary'**)

**Batch size** the number of training examples in one forward/backward pass (or for 1 epoch, which is expected).

Then the already described Image Data Generator is added for training and tasting samples. But it has a new transformation, which is called **rescale**. It multiplies the data by the given value.

The **flow\_from\_directory(directory)** method is added for training and testing data. First, the path to the folders is specified. Further, the target size follows. It shows width and height to which images will be resized. Next, the batch size is added. Finally binary class mode is set.

When the preparation is complete, the code fragment of the training follows:

*# TRAINING*model.fit\_generator(  
 train\_generator,  
 steps\_per\_epoch=400 // batch\_size,  
 epochs=50,  
 validation\_data=validation\_generator,  
 validation\_steps=160 // batch\_size)  
  
model.save\_weights(**'50\_epochs.h5'**) *# always save your weights after training or during training*

Training is possible with the help of **the** **fit\_generator**. Here it is important to indicate a number of **epochs**, which defines for how many times the training will repeat. **1 epoch** is 1 forward pass and 1 backward pass over all the training examples.

Also, in this section steps\_per\_epoch and validation\_steps are set. **Steps\_per\_epoch** (or **number of iterations**) shows total number of steps, which is used to declare one epoch finished and begin the next. Typically this number is equal to the number of samples for training (in my case it is 400: 200 photos of cars and 200 photos of elephants) divided by the batch size (16). It means that the number of iterations: 200 / 16 = 25. **Validation\_steps** is total number of steps (batches of samples) to validate before stopping.

When the model is trained it should be saved with **save\_weights**.

Now, when the model is dissembled it can be run. Running takes some time. At the end of the program shows this result here:

Table

Description automatically generated

It can be seen that after 50 epochs the **validation accuracy** is 0.9375, it shows the ability of the model to generalize to new data.

After running the code and saving the model it’s time to check its accuracy on the new testing photos. It is possible through Scoring code. After running this code with the new 400 photos of elephants and cars, We can get a classification accuracy of 96% (383 photos correct).

We build 2 plots here. The first shows the dependence of the evaluation accuracy on the number of epochs. The evaluation accuracy was calculated using additional dataset of 400 pictures. The second plot shows the dependence of accuracy and validation accuracy on the number of epochs during the testing.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

On the first plot it can be seen that the high accuracy (96%) is achieved after 10 epoch. In subsequent epochs on the plot the accuracy does not improve (and even decreases in interval 10–25 epochs).

The second graph shows the intersection of accuracy and validation accuracy. Validation accuracy sows the ability of the model to generalize to new data. Validation dataset contains only the data that the model never sees during the training and therefor cannot just memorize. If your training data accuracy (“acc”) keeps improving while your validation data accuracy (“val\_acc”) gets worse, you are likely in an overfitting situation, i.e. your model starts to basically just memorize the data.

This means that after the 10th epoch the model can show the same result, but it will not be better. Consequently, this model is be sufficient to train on 10 epochs.

**Conclusions:**

By this way we can assemble and train the CNN model to classify photographs of cars and elephants. We have measured how the accuracy depends on the number of epochs in order to detect potential overfitting problem. Here, we have determined that 10 epochs are enough for a successful training of the model.