ANDHRA UNIVERSITY, Visakhapatnam

Industrial Training report on "PREDICTION OF SALARY BASED ON EXPERIENCE OF EMPLOYEES USING MACHINE LEARNING"

NAME: N.Sailavanya REG NO :317126511096

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Sangivalasa-531162, Bheemunipatnam Mandal, Visakhapatnam Dist .)

(2020–2021)

Department of Information Technology

Anil Neerukonda Institute Of Technology & Sciences

Sangivalasa – 531162

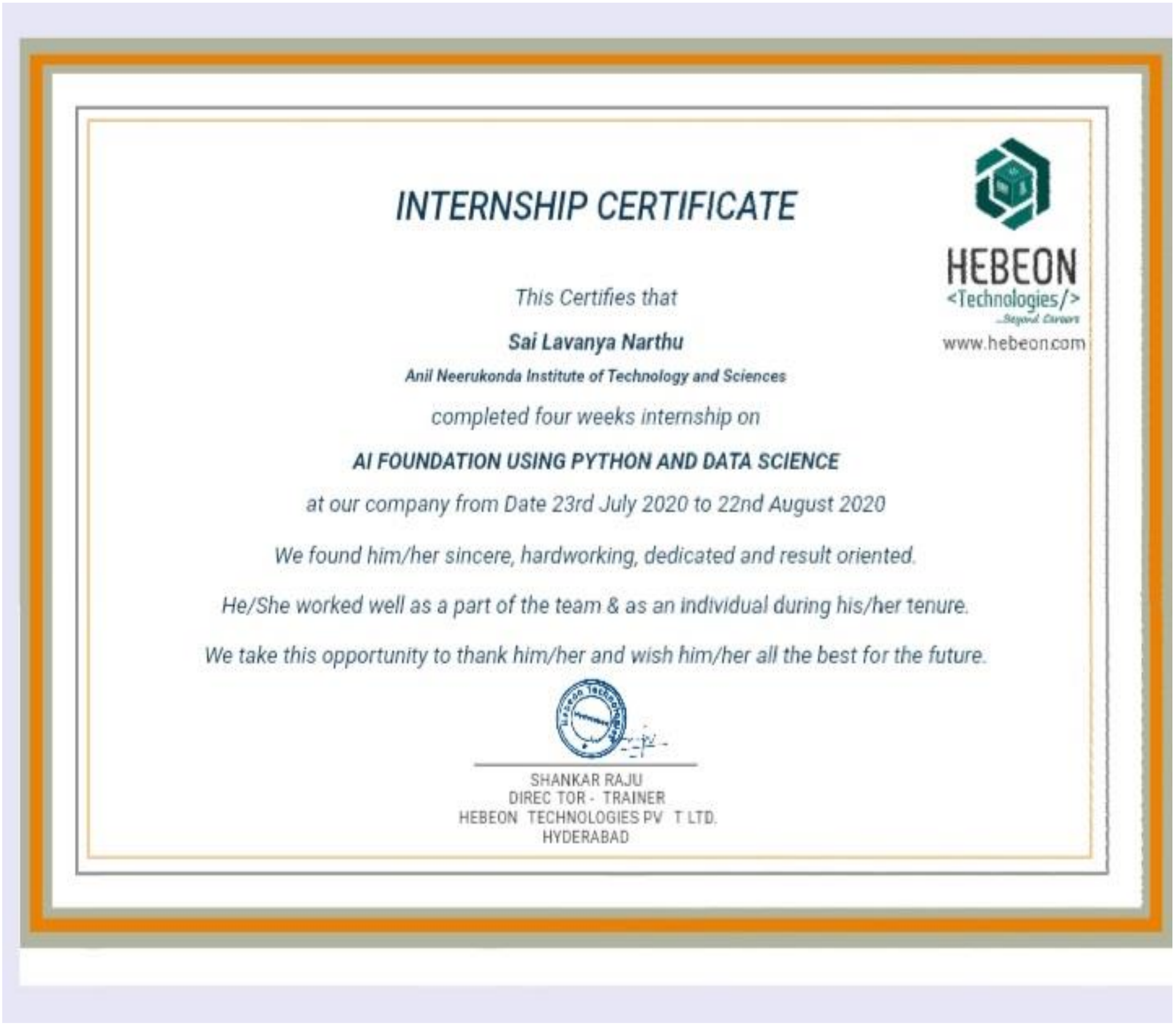
CERTIFICATE

Certified that the industrial training on "**MACHINE LEARNING" and** mini project work entitled **PREDICTION OF SALARY BASED ON EXPERIENCE OF EMPLOYEES** which is a bonafide work carried out by **N.Sailavanya** bearing Register No **317126511096** respectively in fulfillment of Industrial Training in IV/IV B.Tech,1st semester in **Information Technology** of the Andhra University, Visakhapatnam during the year 2020-2021. It is certified that all corrections indicated for internal assessment have been incorporated to account.

HEAD OF THE DEPARTMENT

Poosapati Padmaja

(Information Technology)



INTERNSHIP CERTIFICATE

HEBEON
<Technologies/>
...Beyond Careers
www.hebeon.com

This Certifies that

Sai Lavanya Narthu

Anil Neerukonda Institute of Technology and Sciences

completed four weeks internship on

AI FOUNDATION USING PYTHON AND DATA SCIENCE

at our company from Date 23rd July 2020 to 22nd August 2020

We found him/her sincere, hardworking, dedicated and result oriented.

He/She worked well as a part of the team & as an individual during his/her tenure.

We take this opportunity to thank him/her and wish him/her all the best for the future.

Hebeon Technologies

[illegible]

SHANKAR RAJU
DIREC TOR - TRAINER
HEBEON TECHNOLOGIES PV T LTD.
HYDERABAD

COMPANY DETAILS

Hebeon Technologies Private Limited is a Private incorporated on 17 September 2019. It is classified as Non-govt company and is registered at Registrar of Companies, Hyderabad. Its authorized share capital is Rs. 1,000,000 and its paid up capital is Rs. 415,200. It is involved in Other computer related activities [for example maintenance of websites of other firms/ creation of multimedia presentations for other firms etc.]
Hebeon Technologies Private Limited's Annual General Meeting (AGM) was last held on N/A and as per records from Ministry of Corporate Affairs (MCA), its balance sheet was last filed on N/A.
Directors of Hebeon Technologies Private Limited are Lavanya Nandimandalam and Sankara Raju Chamarthi.
Hebeon Technologies Private Limited's Corporate Identification Number is (CIN) U72900TG2019PTC135479 and its registration number is 135479.Its Email address is csraju@hebeon.com and its registered address is Plot No. 196, Beside of Kidzee School, Prasanthi Hills Colony, Raidurgam HYDERABAD Hyderabad TG 500032 INDIA. Current status of Hebeon Technologies Private Limited is – Active.

DECLARATION

I **N.Sailavanya** bearing University Register No. **317126511096** do here by declare that this industrial training on entitled “**PROJECT ON PREDICTION OF SALARY BASED ON EXPERIENCE OF EMPLOYEES USING MACHINE LEARNING”**, is
carried out by under the guidance of “HEBEON TECHNOLOGIES”

I hereby declare that the original industrial training and mini project is done by me as per requirements of training.

N.Sailavanya 317126511096

Station: Sangivalasa

Date: 24-02-2021

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this mini project work a grand success.
On successful completion of our Industrial Training, we are bound to convey our sincere thanks to “HEBEON TECHNOLOGIES” for giving support and guidance with valuable during training days

N.Sailavanya 317126511096

CONTENTS

ABSTRACT

Organisations all around the world are using data to predict behaviours and extract valuable real-world insights to inform decisions. Managing and analysing big data has become an essential part of modern finance, retail, marketing, social science, development and research, medicine and government.

The objective is to focus on the basic mathematics, statistics and programming skills that are necessary for typical data analysis tasks and Linear regression is an important technique. Its basis is illustrated here, and various derived values such as the standard deviation from regression and the slope of the relationship between two variables are shown. The purpose of this project is to use data transformation and machine learning to create a model that will predict a salary when given years of experience. For this purpose we use a machine learning Model called Regression.The project flow goes initially with loading the data set and then splitting the data set into training set and testing set ,fitting the model to training set,predicting the test set,visualising the training set and test set and finally we would get salary prediction of employees. Information Used to Predict Salaries is the number of Years of Experience and salaries.

CHAPTER 1 – INTRODUCTION

Industrial Training Objective

Foundations of Data Science invites submissions focusing on advances in mathematical, statistical, and computational methods for data science. Results should significantly advance current understanding of data science, by algorithm development, analysis, and/or computational implementation which demonstrates behavior and applicability of the algorithm.

- Linear regression is a linear approach to modelling the relationship between a scalar response and one or moreexplanatory variables. The case of one explanatory variable is called simple linear regression for more than one, the process is called multiple linear regression.This term is distinct from multivariate linear regression where multiple correlated dependent variables are predicted, rather than a single scalar variable.In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

Industrial Training Outline

Data science is very powerful, but it is not a magic wand. It has limitations and it cannot solve all the problems in all settings. The quality of the data analysis depends heavily on the quality and quantity of the data. Data science is a "concept to unify statistics, data analysis and their related methods" in order to "understand and analyze actual phenomena" with data. It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, domain knowledge and information science.Focuses on the basic mathematics, statistics and programming skills that are necessary for typical data analysis tasks.

CHAPTER 2 – FOUNDATIONS OF DATA SCIENCE

INTRODUCTION

Data science is all about extracting intelligence from data. Data science draws on methods from statistics and computer science to process the vast amounts of data that we generate in our daily lives and to turn this into something meaningful and valuable. Whenever you buy a coffee with your debit or credit card, whenever you visit a website, whenever you go for a run wearing a wrist-worn heart rate monitor, whenever and whatever you do in today's world, you generate data. With our lives been increasingly digitised, constantly on smartphones, electronic payments, we've all become data-generating machines.

This data says a lot about you. Your purchase information, browsing preferences, and heart rate, for example, are all valuable sources of information on you as an individual. But when gathered from large populations, this data becomes even more valuable. It gives the companies and governments collecting it the ability to analyze not only what we've done, but also the ability to predict what you're about to do in future. Already this has had a massive impact in the world of business and politics, in just about every aspect of our lives. Most of us have no idea about exactly how our data is being used. The algorithms used by social media companies, banks, and health insurers can seem like mysterious black boxes. This mystery is part of what keeps all the profits and power that come from this knowledge about us, within those organisations. But the details of what actually goes on the data science is often not as sophisticated as we may imagine. Deep down, the algorithms used by the multinational organisations are often fairly simple and can be implemented in only a few lines of computer code. The true source of all the value, the knowledge is contained within the data itself, is contained within our data. Part of the motivation for this course is to help you open up the black box of data science, to break open the mystery and learn what's going on inside.

What is Data?

The word "data" is often used interchangeably with information. From an information theoretic perspective, the degree to which data is informative depends on how unexpected it is to the person receiving it. However, in this course, we won't get ourselves bogged down on that difference. Data as a collection of factual information that can be used by a computer as a basis for calculation. Data can be represented either as a number such as age or a price, or as a category such as hair color or type of fruit. Data becomes particularly interesting when we're dealing with large volumes of it. Take information on the hair color and age of a handful of people. With only a few examples, not much can be said about how using only these two features gives you any information. But if this is scaled up to 10,000 people, then patterns emerge in the data which can be useful to, for example, a marketing company that's trying to market new hair care products. They might use this to decide which age group they should market their products to. This also shows the usefulness of having more than one dimension or data feature about whatever object it is we're looking at. Our two dimensions here are hair and age. Using only a single dimension such as hair, would only be of limited use. Data science allows us to explore the interactions between different data features and to find patterns across large populations, is what computer science and statistics joined together to create a powerful combination.

Types of Data:

There are two broad types of data, categorical and numeric. Categorical is when a textual description or label is used to represent specific categories of objects. Categories include things like primary colors: red, green, and blue; or fruit: banana, apple, orange. Numeric data can be continuous. Something that is measured on a continuous scale, like air temperatures at the North Pole or the weight of a cow. Numeric data can also be discreet. Something that is countable like the number of beans in a jar, or the number of people who survived the sinking of the Titanic. Ultimately, all data is stored on a computer as a discrete binary number. This means that when we're working with categorical data or continuous numeric data, the computer is actually mapping this to some discrete value behind the scenes. But we don't need to concern ourselves with that here. For practical purposes, the data we will be dealing with in our examples will be continuous numeric data.

Machine learning

Machine learning is the term used to describe a series of processes in which a computer learns from evidence or learns from lots of examples of data to help it to certain data-based tasks. Common to all machine learning algorithms is a training step. Training is where the computer learns something about the world or a particular problem, based on data drawn from that world. This may allow the computer to build some internal representation, or model, about that world. Alternatively, the computer can be trained to search for patterns in the data to help structure the world. The outcome of training is an algorithm that can be used for a variety of tasks such as predicting future events, automatically recognising objects, or structuring data in a manageable way. Here are a few examples of how machine learning maybe me used.

Machine learning can be used to predict the future. Take earthquake prediction. This is important applications in predicting natural disasters and helping people plan to minimise the impact. When trained on information such as time, location, and magnitude of historical earthquakes in a region, a machine learning algorithm may be used by geologists to work out the probability of another earthquake occurring at a certain time and place in the future. This sort of machine learning is known as regression. Where continuous data such as time or magnitude is used to train a model that outputs a continuous valued output such as the probability of earthquake. Machine learning can also be used to recognise and classify previously unseen objects. Given a large database of possible road signs, a machine learning algorithm in a self-driving car may be able to correctly recognise a stop sign in an unfamiliar location or recognise a stop sign that is drawn slightly different from the others it's seen. This information can be used to direct the car to take appropriate action, such as stop. This sort of machine learning where the output is a discrete label, a stop sign, is known as classification. Machine learning can also be used to structure lots of data into manageable chunks or clusters. Information from thousands of people's shopping habits for example, can be used to link some groups of those people into specific clusters for more targeted marketing. Clustering is an example of unsupervised learning, which we'll talk more about in a moment. It allows us to find patterns of behavior in data, based on similarities in that data. The same approach may make use of data drawn from social media posts to automatically cluster people into groups of similar political affiliation, or use sequences of information from samples of DNA to cluster people into groups with similar genetics. The algorithms themselves may be the same but depending on the data used, the uses can vary enormously, as can the impact of their use. In summary, machine learning is really just about building algorithms that can help a computer to learn from a body of data so that may make sense or make predictions about new and previously unseen data. Machine learning is when a machine learns from examples.

Supervised vs Unsupervised Learning

Machine learning algorithms fall into two categories, supervised learning and unsupervised learning. What do I mean by supervised? Consider a parent teaching an infant the difference between dogs and cats. Every time the child sees a dog, the parent will point out "dog", and similarly for a cat. After seeing lots of instances of the two animals, the child will have learned to distinguish between them. All that is needed are lots of examples of data, images of animals that had been pre-labelled as either cat or dog by the child supervisor or parent. Supervised learning is essentially this, providing the computer with lots of training data images of dogs and cats in this example, alongside the class labels that we have assigned either dog or cat. When completely new data is then presented, the trained algorithm can infer the most likely corresponding class or value. For example, given a completely unknown animal our trained algorithm, will make a decision on whether it's either a dog or a cat. This is an example of classification, where a single label is output. Alternatively, the algorithm can give some measure indicating the degree of dogishness, or catishness. This is an example of regression, or a continuous valued output or a probability is given. Unsupervised learning is when we don't know beforehand the structure of the data. We don't use any labels. In our child learning example, the parent never tells the child what it is they're looking at. Perhaps they don't know themselves. The child may be exposed to lots of dogs and cats, but they have to learn the similarities and differences between examples of the creatures based on some other criteria. Over time, the child may well learn traditional dichotomy dog versus cat. But equally, they could form a different clustering. For example, degree of furriness versus non furry. They might not even be limited to just two classes or clusters. Perhaps, the data suggests that three or more clusters may be appropriate. For example, if the similarity criteria was colour of fur. This division of data into groups based on some measure of similarity is why this type of unsupervised learning is referred to as data clustering. Data clustering is a very powerful tool and exemplifies many of the most important aspects of machine learning and data science.

CHAPTER 3 - Linear Regression

Linear Regression

- Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis.
- Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression.
- Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.
- The linear regression model provides a sloped straight line representing the relationship between the variables.

Types of Linear Regression:

Linear regression can be further divided into two types of the algorithm

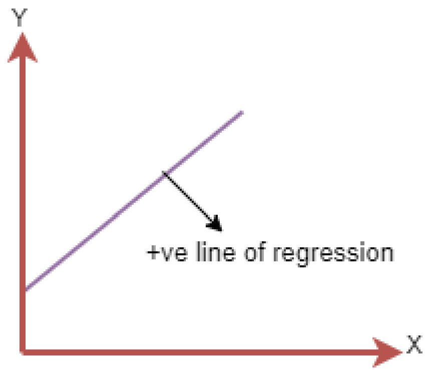
- **Simple Linear Regression:**
  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

- **Multiple Linear Regression**:
  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

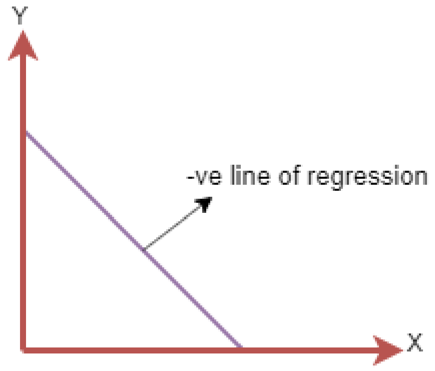
A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

- **Positive Linear Relationship:**
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $\mathbf{Y= a_0+a_1x}$

- **Negative Linear Relationship:**
  If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: $\mathbf{Y= -a_0+a_1x}$

Finding the best fit line:

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.
The different values for weights or the coefficient of lines (a0, a1) gives a different line of regression, so we need to calculate the best values for a0 and a1 to find the best fit line, so to calculate this we use cost function.

Cost function-

- The different values for weights or coefficient of lines (a0, a1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.
- Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.
- We can use the cost function to find the accuracy of the mapping function, which maps the input variable to the output variable. This mapping function is also known as Hypothesis function.
- For Linear Regression, we use the Mean Squared Error (MSE) cost function, which is the average of squared error occurred between the predicted values and actual values. It can be written as:

For the above linear equation, MSE can be calculated as:

$$\boxed{\text{MSE}= 1\frac{1}{N}\sum_{i=1}^{n}(y_i \ - (a_1x_i + a_0))^2}$$

Where,
N=Total number of observation
Yi = Actual value
(a1xi+a0)= Predicted value.

Residuals: The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

Gradient Descent:

Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

Model Performance:

The Goodness of fit determines how the line of regression fits the set of observations. The process of finding the best model out of various models is called optimization. It can be achieved by below method:

R-squared method:

- R-squared is a statistical method that determines the goodness of fit.
- It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
- The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
- It is also called a coefficient of determination, or coefficient of multiple determination for multiple regression.

It can be calculated from the below formula:

$$\boxed{\text{R-squared}=\frac{\text{Explained variation}}{\text{Total Variation}}}$$

CHAPTER 4 – PYTHON

Python Lists and Variables

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

List: Lists are used to store multiple items in a single variable.Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage. Lists are created using square brackets.

Eg: mylist = ["apple", "banana", "cherry"]

Variables: Variables are containers for storing data values.Python has no command for declaring a variable.A variable is created the moment you first assign a value to it.

Example

```
x=5
y = "John"
```

Computing the Mean

We had a list of numbers all stored in different variables, and then we calculated the mean like this, by adding them altogether, and then doing a calculation.

Code:

```
age1 = 40
age2 = 26
age3 = 27
age4 = 100
age5 = 67
summed_ages = age1+age2+age3+age4+age5
mean_age = summed_ages/5
print(mean_age)
```

Output: 52.0

Using some of the built-in functions in Python, we can make list code really quite neat and efficient, and also you can make it more robust, so that if we decide we want to add more numbers or more variables into our list there, then the code still works, exactly the same code still computes it, by using the len() function and the sum() function.

Code:

```
ages = [40,26,27,100,67,57]
summed_ages = sum(ages)
mean_age = summed_ages/len(ages)
print(mean_age)
```

Output:52.83333333

Better Lists: NumPy

NumPy stands for Numerical Python, and it is an extension or a set of package for Python which gives you extra functionality. Because you can see that we're really hitting the limits of what Python can do because we're having to calculate the mean by hand, for example, and we have to calculate, sum it up and then divide it by the length. So, you cansee that we're at the limits, to some extent, of what Python can do numerically and were already writing our own code to do number processing. When we were data programming, we're not necessarily interested in just software engineering. We don't want to build everything from scratch and know exactly what everything's doing. We're better off building a data pipeline out of pre-existing functions that exist in well- established libraries like NumPy and SciPy because those functions are going to be really well-tested and they'll probably be a bit faster than anything that you can write as well, because it might well have some native system underneath that makes it run faster. So, this is our motivation for learning more advanced data structure to get us to the next level of simplicity in our code so we can write really lean data processing code. So, let's go in and start using a NumPy array and see what it looks like. So, the first thing we need to do is to import the NumPy library. So, we type import. So, what we need to get access to NumPy we need to import it and then an extra little thing we can do here is we can give it a friendly name or a short name so that when we're referring to it we can use a less verbose name. Okay, and np is very common.

Code:

```
Import numpy as np
ages = np.array(ages)
ages.mean()
```

Output:52.833333333

Computing the Standard Deviation

The standard deviation gives us a sense of how much variation there is in our array. So, for example, if we're talking about ages, if we've got lots of very young people and lots of very old people, it means we've got a big variation in our dataset. So, we would expect to have a high standard deviation. Now, if there's lots of people who have a very similar age in our dataset, then we would say that would come with a lower standard deviation. So, it's really a measure of the variation in our dataset. Now, it's calculated by calculating the sum squared error, if you like, between the values in our dataset and the mean point to the dataset. So, first of all, let's store the mean somewhere, and we'll do ages.mean(). So, I've got a variable called mean, and it's going to store the mean of the ages. Then the next step is to work out the distances between all of the values and the mean. Because it's a NumPy array, I can do that fairly easily. I can simply do distsequals ages minus mean. Now, because we're going to add these up, we need to make sure they're all positive so we know exactly how much variation there is, not whether something is above or below because we are not interested in above or below. We're just interested in the absolute variation. So, before we do that, we actually square it, which will turn them all into positive values. So, dists equals diststimesdists. So, you see they're all squared now into positive values. So, the next step is to then take the sum of all the distances. So, we can do summed_dists equals np.sum(dists). Now, we can't use the regular some in Python because it doesn't work with NumPy arrays. Finally, what we do is we divide the sum. So, we take the average distance now. So, it's the average distance between a thing in the dataset and the mean. So, the average distance is basically dists equals summed_dists divided by the length of the array, which is len(ages). So the final thing is to do np... So, the std_dev equals np.sqrt(avg_dists). It makes sense to take the square root because remember earlier, when we wanted to turn them all into positive values, we square them up.

Code:

```
mean = ages.mean()
dists = ages-mean
dists = dists*dists
summed_dists = np.sum(dists)
avg_dist = summed_dists/len(ages)
std_dev = np.sqrt(avg_dist)
print(std_dev)
```

Mathematics for Multidimensional data

Multidimensional Data Points and Features

A matrix is a rectangular array of numbers, a way of packing numbers, or arrays of numbers (vectors) together, like so:

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{pmatrix}$$

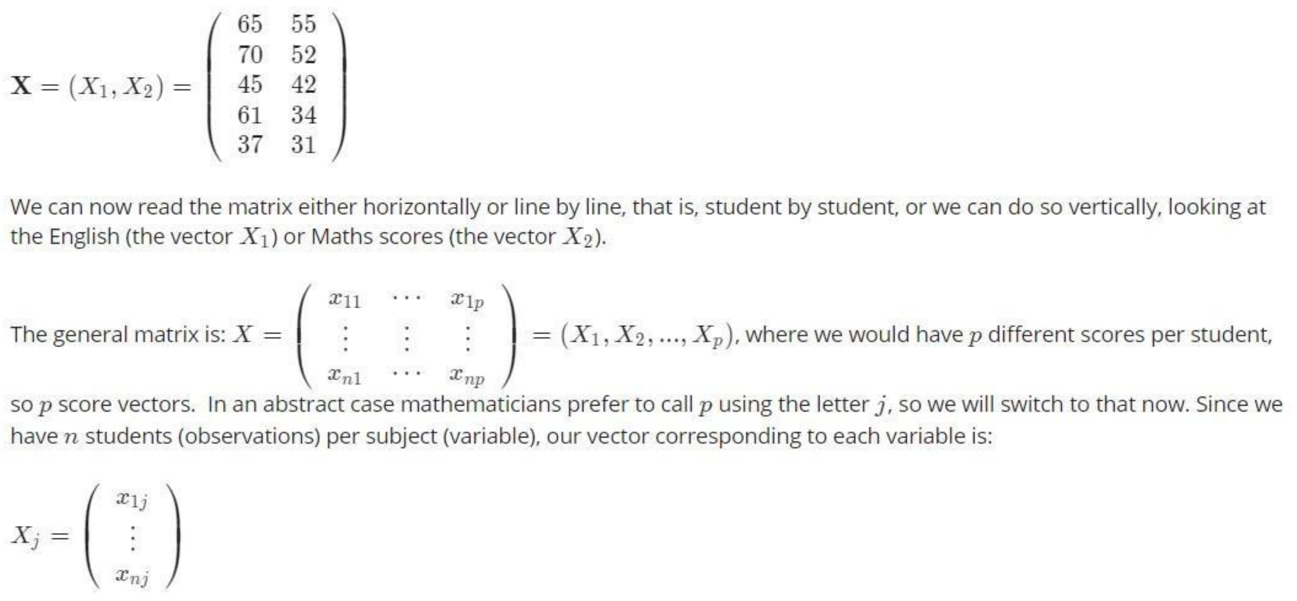
Fig. 4.5.1.1 Matrix Representation

A vector can be seen as a matrix with either one row, or one column:

$$\mathbf{Y} = \begin{pmatrix} 4 \\ 6 \\ 2 \\ 4 \\ 2 \end{pmatrix},$$

$$\mathbf{Z} = \begin{pmatrix} 4 & 6 & 2 & 4 & 2 \end{pmatrix}.$$

In the example presented, we had the marks of our students packed in one matrix, the first column being the English score, the second the Maths score:



$$\mathbf{X} = (X_1, X_2) = \begin{pmatrix} 65 & 55 \\ 70 & 52 \\ 45 & 42 \\ 61 & 34 \\ 37 & 31 \end{pmatrix}$$

We can now read the matrix either horizontally or line by line, that is, student by student, or we can do so vertically, looking at the English (the vector X_1) or Maths scores (the vector X_2).

The general matrix is: $X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} = (X_1, X_2, ..., X_p)$, where we would have p different scores per student, so p score vectors. In an abstract case mathematicians prefer to call p using the letter j, so we will switch to that now. Since we have n students (observations) per subject (variable), our vector corresponding to each variable is:

$$X_j = \begin{pmatrix} x_{1j} \\ \vdots \\ x_{nj} \end{pmatrix}$$

Outliers

An outlier is any data value that lies a great distance from other data values in a dataset. Being an outlier in itself is not an indicator that a data value is invalid or erroneous. Outliers may be the result of experimental errors or variability in measurement. When outliers are the result of bad data, then the mean would be a poor estimate of location. In order to avoid the influence of outliers, a trimmed mean is widely used as it is robust to, or not affected by, extreme values in the dataset. It is important to identify outliers and if possible, investigate them further. In anomaly detection, outliers are the objects of interest. Here, the great mass of data serves primarily to define the normal against which anomalies are measured. In order to identify outliers, it is necessary to characterise normal observations or data values.

Matplotlib in python

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general- purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.

Matplotlib.pyplot.scatter

matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin =None, vmax=None, alpha=None, linewidths=None, verts=<deprecatedparameter>, edgecolors= None, *, plotnonfinite=False, data=None, **kwarg)

A scatter plot of *y* vs. *x* with varying marker size and/or color.

Storing 2D Coordinates in a Single Data Structure

How we can store two-dimensional data points in a single data structureSo, here's the code and you see that we had two variables, one for the x values and one for the y values. Then we assume that each they're aligned so that zero in xs is connected to zero in ys, and then one is connected to one and so on. For me that seems like a bit of a clunky way of doing it. Wouldn't it be better if we could kind of somehow store these as a single data structure? Because, imagine, we're doing something a bit more complicated like storing data about cars. So, let's say we had the number of cylinders, the engine size, the speed, and the amount of pollution it produces. That's four data points and then we would need four different variables using this model to store that and then it could become really confusing having to work with these four different variables and things like that. So, what we're going to look at now is, some NumPy data structure code which allows us to store two-dimensional vectors in a single data structure. So, I'm going to take this x and y here, and I stick that into a new cell. I'll just make sure I've imported NumPy first. So, import NumPy as np.

Code:

```
Import numpy as np
xs = [10,100,25,67,74]
ys = [125,56,66,1,10]
xys = [[10,125], [100,26], [25,66], [67,1], [74,10]]
xys = np.arrays(xys)
print(xys)
```

Output:

```
[[10  125]
 [100  26]
 [25  66]
 [67  1]
 [74  10]]
```

what I actually want to do is I want to send the first column, which is the xs, and the second column, which is the ys. Now, in data programming, it's very common for us to want to kind of rip certain bits out of our data and present it somewhere or to strip out one column or the other column and NumPy does provide quite a lot of functionality for doing this to allow you to selectively pull out bits of a data structure. So, what we're going to do now is see the NumPy syntax which allows you to pull out just that column and the second column. So, if instead of printing that I just do this: xys[:, 0].

Code:

```
importnumpy as np
xs = [10,100,25,67,74]
ys = [125,56,66,1,10]
xys = [[10,125], [100,26], [25,66], [67,1], [74,10]]
xys = np.arrays(xys)
print(xys[:, 0])
```

Output:

```
[10, 100, 25, 67, 74]
```

4.8.3Matplotlib.patches

Classes

Arc(xy, width, height[, angle, theta1, theta2])	An elliptical arc, i.e.
Arrow(x, y, dx, dy[, width])	An arrow patch.
ArrowStyle(stylename, **kw)	ArrowStyle is a container class which defines several arrowstyle classes, which is used to create an arrow path along a given path.
BoxStyle(stylename, **kw)	BoxStyle is a container class which defines several

	boxstyleclasses, which are used for FancyBboxPatch.
Circle(xy[, radius])	A circle patch.
CirclePolygon(xy[, radius, resolution])	A polygon-approximation of a circle patch.
ConnectionPatch(xyA, xyB, coordsA[, ...])	A patch that connects two points (possibly in different axes).
ConnectionStyle(stylename, **kw)	ConnectionStyle is a container class which defines several connectionstyle classes, which is used to create a path between two points.
Ellipse(xy, width, height[, angle])	A scale-free ellipse.
FancyArrow(x, y, dx, dy[, width, ...])	Like Arrow, but lets you set head width and head height independently.
FancyArrowPatch([posA, posB, path, ...])	A fancy arrow patch.
FancyBboxPatch(xy, width, height[, ...])	A fancy box around a rectangle with lower left at $xy = (x, y)$ with specified width and height.
Patch([edgecolor, facecolor, color, ...])	A patch is a 2D artist with a face color and an edge color.
PathPatch(path, **kwargs)	A general polycurve path patch.
Polygon(xy[, closed])	A general polygon patch.

Rectangle(xy, width, height[, angle])	A rectangle defined via an anchor point *xy* and its *width* and *height*.
RegularPolygon(xy, numVertices[, radius, ...])	A regular polygon patch.
Shadow(patch, ox, oy[, props])	Create a shadow of the given *patch*.
Wedge(center, r, theta1, theta2[, width])	Wedge shaped patch.

Functions

bbox_artist(artist, renderer[, props, fill])	A debug function to draw a rectangle around the bounding box returned by an artist's Artist.get_window_extent to test whether the artist is returning the correct bbox.
draw_bbox(bbox, renderer[, color, trans])	A debug function to draw a rectangle around the bounding box returned by an artist's Artist.get_window_extent to test whether the artist is returning the correct bbox.

Table 4.8.3Matplotlib patches classes and functions

4.8.4 List Comprehensions

List comprehensions provide a concise way to create lists. Common applications are to make new lists where each element is the result of some operations applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.

For example, assume we want to create a list of squares, like:

```
>>>squares=[]
>>>forxinrange(10):
... squares.append(x**2)
...
```

```
>>>squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Note that this creates (or overwrites) a variable named x that still exists after the loop completes. We can calculate the list of squares without any side effects using:

```
squares=list(map(lambdax:x**2,range(10)))
```

or, equivalently:

```
squares=[x**2forxinrange(10)]
```

which is more concise and readable.

A list comprehension consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses. The result will be a new list resulting from evaluating the expression in the context of the for and if clauses which follow it. For example, this listcomp combines the elements of two lists if they are not equal:

```
>>>[(x,y)forxin[1,2,3]foryin[3,1,4]ifx!=y]
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

4.8.5 Pandas

we import Pandas as follows:

```
In [1]: importnumpyasnp

In [2]: importpandasaspd
```

Object creation

Creating a **Series** by passing a list of values, letting pandas create a default integer index:

```
In [3]: s=pd.Series([1,3,5,np.nan,6,8])

In [4]: s
Out[4]:
0    1.0
1    3.0
```

```
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

Creating a **DataFrame** by passing a NumPy array, with a datetime index and labeled columns:

```
In [5]: dates=pd.date_range("20130101",periods=6)


In [6]: dates
Out[6]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
dtype='datetime64[ns]', freq='D')


In [7]: df=pd.DataFrame(np.random.randn(6,4),index=dates,columns=list("ABCD"))


In [8]: df
Out[8]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
```

Sorting by values:

```
In [22]: df.sort_values(by="B")
Out[22]:
                   A         B         C         D
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
```

```
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215 0.119209 -1.044236
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
```

4.8.6 Pandas Read_CSV Function

`pandas.read_csv`*(*filepath_or_buffer, sep=<object object>, delimiter=None, header='infer', names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, skipfooter=0, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, cache_dates=True, iterator=False, chunksize=None, compression='infer', thousands=None, decimal='.', lineterminator=None, quotechar='"', quoting=0, doublequote=True, escapechar=None, comment=None, encoding=None, dialect=None, error_bad_lines=True, warn_bad_lines=True, delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None, storage_options=None*)*

Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file into chunks.

CHAPTER 5 – PROJECT

PROBLEM STATEMENT:

Prediction of salary of an employee based on their experience using Machine Learning.

DESCRIPTION:

The purpose of this project is to use data transformation and machine learning to create a model that will predict a salary when given years of experience. For this purpose we use a machine learning Model called Regression.The project flow goes initially with loading the data set and then splitting the data set into training set and testing set ,fitting the model to training set,predicting the test set,visualising the training set and test set and finally we would get salary prediction of employees. Information Used to Predict Salaries is the number of Years of Experience and salaries

PROCEDURE:

- **sklearn.model_selection** : Split arrays or matrices into random train and test subsets.
- **Train and Test:** Train/Test is a method to measure the accuracy of your model. It is called Train/Test because you split the the data set into two sets: a training set and a testing set.

-80% for training, and 20% for testing.
-You train the model using the training set.
-You test the model using the testing set.
-Train the model means create the model.
-Test the model means test the accuracy of the model.

SOURCE CODE

```
import numpy as np

import matplotlib.pyplot as

plt import pandas as pd

from google.colab import drive

drive.mount("/content/drive")

df=pd.read_csv('/content/drive/My Drive/Salary_Data.csv')

X =

df.iloc[:,0:1].values y =

df.iloc[:, 1].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

from sklearn.linear_model import LinearRegression
```

```
regressor = LinearRegression() regressor.fit(X_train, y_train, y_pred = regressor.predict(X_test)
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color =
'blue') plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

NEW PREDICTIONS:

```
regressor.predict([[9]])

regressor.predict([[5]])
```

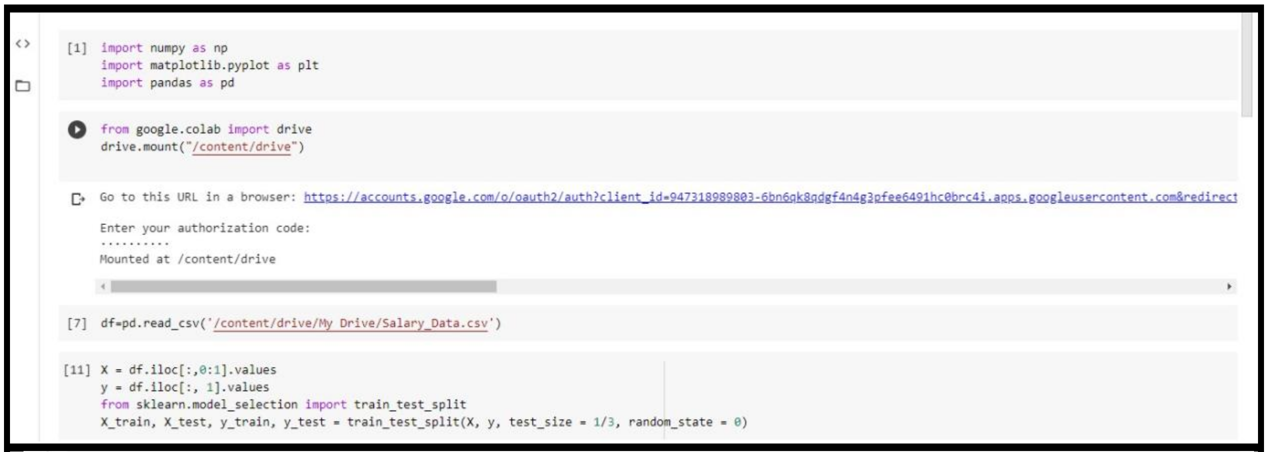
- **DATASET INFERRED IN THE PROJECT:**

https://drive.google.com/file/d/1V3ZrcekeLMIRUxMOSYIgZVB-Hzegc7F_/view?usp=sharing

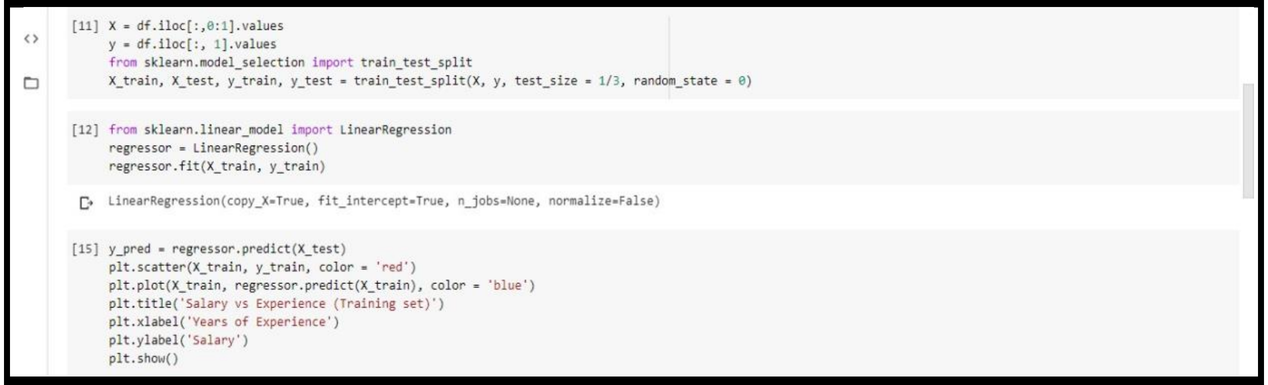
The above data has the salaries and experience of employees which we can use to construct a model for predicting the salary of a person based on number of year of experience.

Thus, **The predicted salary of a person with 5 years experience is [73545.90445964] .**

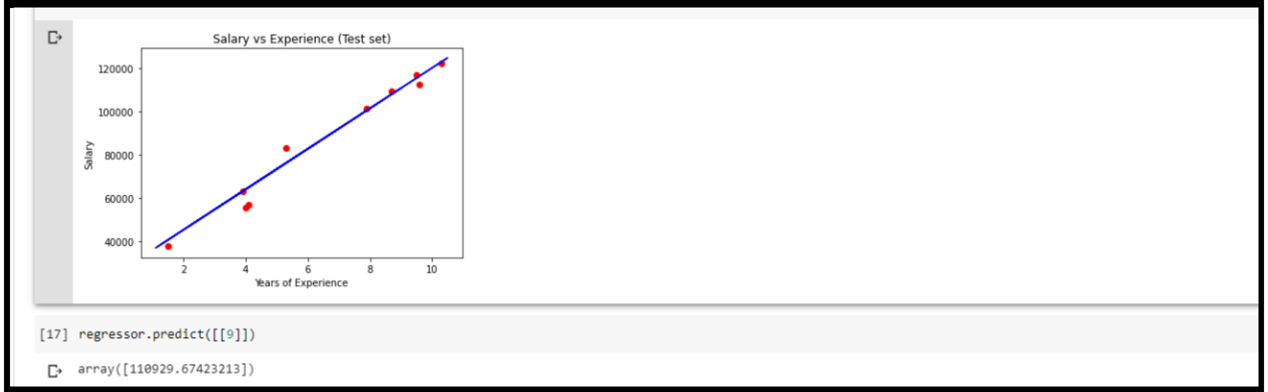
OUTPUT:



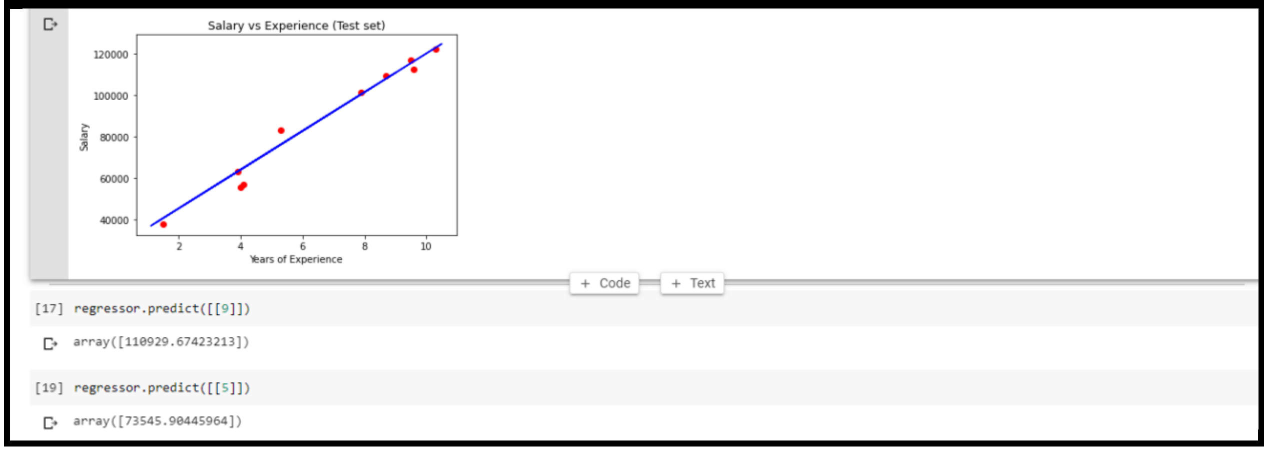
```
[1] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd

    from google.colab import drive
    drive.mount("/content/drive")

    Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect

    Enter your authorization code:
    ··········
    Mounted at /content/drive

[7] df=pd.read_csv('/content/drive/My Drive/Salary_Data.csv')

[11] X = df.iloc[:,0:1].values
     y = df.iloc[:, 1].values
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```



```
[11] X = df.iloc[:,0:1].values
     y = df.iloc[:, 1].values
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

[12] from sklearn.linear_model import LinearRegression
     regressor = LinearRegression()
     regressor.fit(X_train, y_train)

     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

[15] y_pred = regressor.predict(X_test)
     plt.scatter(X_train, y_train, color = 'red')
     plt.plot(X_train, regressor.predict(X_train), color = 'blue')
     plt.title('Salary vs Experience (Training set)')
     plt.xlabel('Years of Experience')
     plt.ylabel('Salary')
     plt.show()
```



```
[17] regressor.predict([[9]])

     array([110929.67423213])
```



```
[17] regressor.predict([[9]])

     array([110929.67423213])

[19] regressor.predict([[5]])

     array([73545.90445964])
```

CHAPTER-6 CONCLUSIONS

Data science education is well into its formative stages of development; it is evolving into a self-supporting discipline and producing professionals with distinct and complementary skills relative to professionals in the computer, information, and statistical sciences. However, regardless of its potential eventual disciplinary status, the evidence points to robust growth of data science education that will indelibly shape the undergraduate students of the future. It will develop

- Mathematical foundations
- Computational foundations
- Statistical foundations
- Data management and curation
- Data description and visualization
- Data modeling and assessment
- Workflow and reproducibility
- Communication and teamwork
- Domain-specific considerations
- Ethical problem solving.

REFERENCES

1. APA Citation Provost, F., & Fawcett, T. (2013). *Data science for business: [what you need to know about data mining and data-analytic thinking]*. Sebastopol, Calif.: O'Reilly.
2. Chicago / Turabian - Author Date Citation Provost, Foster, 1964- and Tom. Fawcett. 2013. *Data Science for Business: [what You Need to Know About Data Mining and Data-analytic Thinking]*. Sebastopol, Calif.: O'Reilly.
3. Judith Bell and Stephen Waters *Doing Your Research Project: A Guide For First-Time Researchers.*Open University Press Hill Education, 2014.
4. Wes McKinney *Python for Data Analysis*, 2nd Edition. O'Reilly Media, 2017.