Hi there, welcome to our machine learning course.  In this course we will be learning fundamental of machine learning and will dive in the details in one of the domain of ML (machine learning), which is supervised machine learning.

Programming language used in this course: **Python**

**Introduction**

This book is extracted from machine learning course on Udemy. That course contains exactly same written material as used in this book. Our Udemy course contain explanative video lecture of each topic as well. But we hope that even following this book, you will grab all topics without videos. And if you want better experience along with video lectures, you can navigate to this link:

https://www.udemy.com/course/supervised-machine-learning-in-python-o/

I would suggest you to follow this book thoroughly and if you feel need of video lectures, you can take course on official udemy website (link is shared in beginning.

There are 7 sections in this course with total 52 lectures (topics to discuss).

**In first section** we have discussed about machine learning and types of machine learning. There are comprehensive videos explanations well. We have tried our best to explain the concept in simple and understandable language.

**In second section** we trained our first machine learning model. This section contains 20 lectures and we assure that if you take those all 20 lectures, you will have no confusion in rudimentary machine learning concepts. There are coding, video lectures, written lectures and quizzes for you in this section.

We also have covered all basic steps of data science in this section for your clarity. Such as:

Exploring the dataset, making your own dataset, and understanding datasets for machine learning etc.

**In third section** we have trained model on another machine learning algorithm (linear regression). Working of linear regression, graphical understanding and basic math behind this algorithm is well explained in video as well in text lectures. There are 8 lectures in this section containing 3 video lectures. We also have added a short assignment for you to increase your confidence on your understandings.

**In fourth section** we have trained a model on famous dataset, which is an iris dataset containing feature of flowers. We have trained a model which predict flower category based on some features of flower (petal and sepal dimensions). Things are explained in text as well in video lecture.

**In fifth section** we have discussed technique to check accuracy of our models. Until now, accuracy of model was not known, we were not having any idea how well our model works. We have explained very comprehensively how to evaluate performance of any of our machine learning model. There are 7 lectures in this section and for sample, we have checked the accuracy of our iris dataset machine learning model (which we trained in previous lecture), which is amazingly 97%.

**In sixth and last section** we have discussed another widely used machine learning algorithm (logistic regression). Using logistic regression we have trained a model to predict whether a patient is suffering from diabetes or not. Within 8 lectures we have created our model which predict diabetes in a patient with accuracy of almost 79%.

**In seventh section** we have assembled all codes of this course in a single place where you can easily access code of any section of this course.

**How much time you need to complete this course?**

Take this course slow and steady. If you give one hour a day, you can complete this course within 20-30 days. When you start this course, make sure that you are consistent with your learning.

**Time per day      |      time to complete this course**

*0.5 - 1 hour                20 - 35 days*

*1 - 2 hours                 15 - 25 days*

*2 - 3 hours                 7 - 15 days*

Spending more than 2 hours is not suggested if this is beginning of your machine learning track.

There are 52 lectures in this lecture containing more almost 20 video lectures.

We hope you will find this course very helpful in your learning.

**Suggestion:** Practice all codes of this course in your own compiler along with.

**Lecture 1: Welcome**

Hi there, I Harman and my sister Sharmeen are going be your instructor in this course.

Contact us if you need any help.

harmanwaheed@gmail.com

**Good luck for your journey!**

**Lecture 2: What is Machine learning**

**Machine Learning...** as the name says, machine is learning something. What does it mean?

In traditional programming, humans write instructions for a computer to follow, but in machine learning, the computer learns to perform a task itself.  
  
Humans still write instructions. **Let suppose** a property dealer have given these instructions to his website:

* If the house is in area, where gas and electricity is in sufficient amount, then the price of house will be 50,000 rupees
* If house is in area where gas and electricity is short, then the price of house will be 30,000 rupees.

One day you are exploring his website and tried to check price of house where gas is sufficient, but electricity is always gone, boom! his website wont be able to show you the price for this case because he didn't added this scenario.

Till now, his code was just some control statements to show a particular price for different areas. His website can't handle the unseen data (different area scenarios).

To make his website more robust that can deal well with unseen data, we will use machine learning tools.   
  
I hope this will make sense now:

*'In Machine learning, machine learns to deal with unseen data'*

I hope you got the basic idea of machine learning.

*point to ponder: to explain the concept, I have given a very simple example, real world problems are a bit different. We will deal with those soon!*

**Lecture 3: How Machine learns**

Think of it as a 2-9 years kid, who observe his/her surrounding and learn things.

**For example my 2 years old nephew** last month saw a dog in ground, he ran to me, I told him not to afraid of this being, and told him that he is 'dogi'.

Now whenever I take him to ground, he look for dog and when he sees the dog, he says to me 'dugi..dugi..'   
Even if different dog appear, he recognize that as 'dugi..dugi..'  
  
Human’s machine is pretty fast and accurate. He just saw a dog for the first time and now he know that a dog can appear in any size (1-3 feet maybe) and color and he recognize any dog immediately.

**Let’s get back to machines now,** as a said in the first line, take machine learning as a kid, who learns whatever from surrounding and parents. And he tries to copy his elders sometimes as well.

Similarly machines require something from which they can learn. That thing is data.

**For Example**

Let say you are making a machine learning model to tell you whether an image have cat in it or not.  For this, we have to give our model some images of real cats and we will tell our model that: 'see this! this is how a Cat looks like!'

After this, we will give this model another image of cat and we will ask our model whether there is a cat in this image or not?

We provide our system some data (images of cats for upper example) from which it learns. From given data, machine learns how to make predictions (whether there is a cat in image or not). That's why to make a good workable model, we have to provide bunch of situations (large data).

**Why?**

Let say I have given my model an image of orange cat. Model can learn that everything orange is cat. After that it will tell me that orange(fruit) is a cat as well. rip model.

To overcome this,

We will upload bunch of cat images of different colors and size and we will tell our model that 'These all are cats! cats looks like these'

Now my system will see that cat can be of different colors and size. Model will try to figure out something common in all images. these can be:

* 4 legs of each cat
* 2 ears
* Small height
* One tail of each cat

Now these things will be common in each image, so model can learn that anything having 4 legs, 2 ears, a tail and small height is a cat.

And this model is a lot better than before which was classifying orange(fruit) as a cat.

That's why we have to train our model with large data.

**Lecture 4: Two types of Machine learns**

In this section, We will be discussing two types of machine learning.

Consider previous example in which we provided our model with some images of cat. We told our model that 'these are the cats! learn from these!'

We are performing two tasks here.

1 - Giving images of cats.

2 - Telling our model that this is a cat image (labeling our data).

Such scenario is **'Supervised Machine learning'**.

Now lets make it a little robust.

Let say you have 2000 images of different animals. Your task is to separate each animal in a folder, for example, all monkeys images should be in one folder, and each lion images should be in other, and so on.

It can be a tedious task for you to classify separate 2000 images in different folders.

You as a data scientist, feed all these images in a machine and machine will learn something from it. **How?**

**Well in this case, as we see that step 2 is missing, so how and what machine will learn?**

As we see that there are 2000 images of different animals. Let say there are 500 images of elephant**,** 500 images of lion**,** 500 images of monkey and 500 images of cats.

All elephants will look same in there images,

All lion will look same,

All monkeys also,

And all cats will also look same.

On these bases, our model may learn that there are 4 types of things in our data:

* First one is large, fat, trunk, grey (mostly).
* Second one is normal size, yellow (mostly) in color, hairy fair around neck
* Third one is thick, two feet, two hands, prominant bones
* Fourth one is small, four legs, small face, different colors.

Our model will separate all the images on these basis. Once separated, you can see which category contains which animal.

such scenario is **'Unsupervised Machine learning'**.

**In supervised machine learning, data is already labeled.**

**In unsupervised machine learning, data isn't labeled.**

Have a cup of water, understand upper concept, here's the so called definition:

* In unsupervised learning, the algorithm is given unlabeled data and model find patterns, structures, or relationships within the data
* In supervised learning, the algorithm is trained on a labeled dataset. (relate to example from previous chapter, How machine learns)

I hope things are making sense.

**Lecture 5: Video explanation of Machine learning and upper lectures**

**Lecture 6: Terms we will be using**

Till now, I have tried to maintain language as simple as possible. I will try this for the rest of my course as well.

Here are some of the technical terms we will be using in this course:

**Classification:**

Recall the example from previous chapter, where I explained an example of unsupervised learning. I wrote sufficient text to deliver the concept that each animal (elephant, lion, monkey, and cat) photo will be separated by our model.

Term 'classification' makes it easier to deliver such concepts.

In upper case, I will say that:

'Our model will classify images of animals'

Classification can be explained as making classes. Each class contains students of a particular class.

For our example, there will be 4 classes, (any number can be possible)

First class contains elephants.

Second class contains lions.

Third class contains monkeys.

Fourth class contains cats.

'Male and female images are classified through machine learning.' (I hope this makes sense now, we will discuss it later).

**Features**

Well, features sound not very technical but in terms of machine learning, it is!

What features are?

Features are something on the basis of which our model gives predictions.

You remember the example where we had 2000 images (section 1, lecture 4).

We discussed four categories of animals:

* First one is large, fat, trunk, grey (mostly).
* Second one is normal size, yellow (mostly) in color, hairy fair around neck
* Third one is thick, two feet, two hands, permanent bones
* Fourth one is small, four legs, small face, and different colors.

We see that each category have some specifications of each animal. These specifications are:

Size, color, thickness and number of feet.

These are specifications of our model (for simplicity I haven't discussed other attributes for now)

These specifications are Features of our model! Try to understand this sentence:

'In the task of classification of cats and sparrows, our features are: size, wings, tail, color etc.'

**Labels**

In terms of machine learning, labels are the different objects that we predict.

In the upper example, our labels were:

Elephant, Lion, Monkey and cat. (Example is explained in section 1 lecture 4)

Now try to understand this sentence:

'In task of classification, our labels are cats and sparrow.'

Now this:

'In task of classification, our labels are cats and sparrow. And for addition, our features are: color, tail, wings, size'

These terms will be used in this course.

Solve the next quiz; let’s see how much you grabbed.

**Quiz 1: Fundamentals about ML, multiple choice questions**

**What is the main characteristic of supervised learning?**

Unlabeled data

Learning without guidance

Labeled data

**What is main purpose of machine learning**

To make computer work with unseen data

To make computer work with already seen data in more robust way

**My model can classify patients of blood pressure and patients of diabetes.  What does it mean?**

My model can differ between blood pressure patients and diabetes patients and can separate them.

My can make a single class for all patients suffering from blood pressure and diabetes.

**What are features in term of machine learning?**

Features are different elements through which we make our model perform good.

Features are the object that our model predict. For previous example, 'blood pressure patient' and 'diabetes patients' are two features.

Features are the attributes or specifications, from which our model learn to predict something.

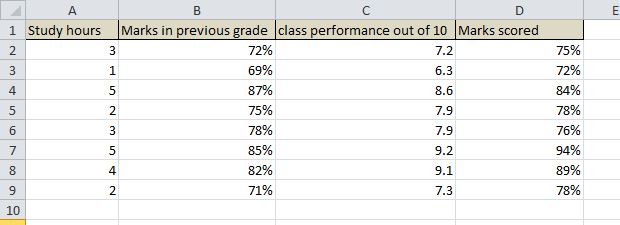
**What are labels in term of machine learning?**

The features or attributes of a dataset are called labels.

The output or target variable that the model predicts.

**Lecture 7: Exploring the dataset**

Let us see how a dataset looks like.



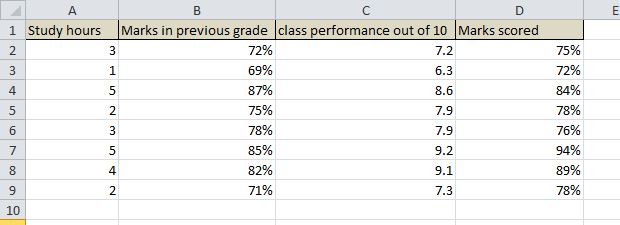
* This data is in excel format, and is added in the resources of this lecture.
* There are 4 columns in this data. Each column has appropriate name.
* There are 9 rows in this dataset.
* Each row represents data of one student.
* **According to 2nd row in our dataset there is a student who study 3 hours, has taken 72% in his previous grade, his performance in class is 7.2 out of 10 and he scored 75% marks in his annual exam.**
* We see that each row will give detailed information about each student.
* Have a look at row 6th and write on your notebook what information you gain from that row.
* Each row contains information about one student, so each row is a data point (this will make sense soon).

**Lecture 8: Understanding data for machine learning**

Hi, Welcome here. Let’s now understand our data for machine learning.

**What does it mean?**

Have a look at our dataset:



In supervised machine learning, we try to predict something....that is called *'label'.*

What is our label in this dataset? Well now this depend on us....what you want your computer to predict from upper data.

Let us make a model to predict the annual marks scored (last column). Our model will learn from this dataset, and will become able to predict on unseen data. What will be the features in our data?

For our case, features or attributes are first three columns. (*Study hours, previous grade marks, class performance*) Fourth one is not a feature because we have selected that column as label.

Don't worry if you didn't get previous 6-7 lines. Let us have a breakdown:

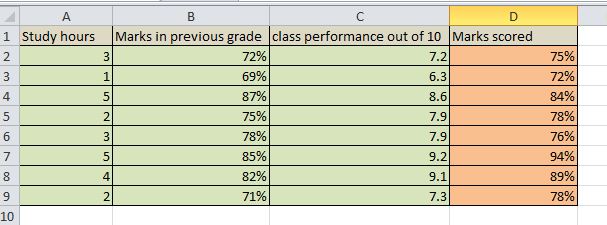
**What we expect from our model to do:**

Once we have trained our model, we expect it to predict marks of a student. Those predictions will be based on the features.

We will tell our model that there exist a student who study 8 hours a day, scored 95% in previous exams and performed 9.6 out of 10 in class. After telling these things, we expect our model to tell us how much marks that student will score in annual exams.

**How to train our model:**

We will feed our data in a model.



In this image, I have colored different columns for good understanding. Last column (orange one) is the label. This is what we want our model to predict later. Green columns (first three columns) are features. Our model will learn from this whole.

In short:

* We want to make a model to predict students annual exam marks
* We will predict this on the basis of study hours, marks in previous grade, and class performance.

Video coming next will give further clarity.

**Lecture 9: Video explanation**

**Lecture 10: Two types supervised machine learning**

Before going any further, let me introduce you to the types of supervised machine learning.

These two types are based on the labels that we have.

Let suppose that we are trying to predict whether the image is of cat or of dog. Now there are two predictions possible. Whether it can make prediction of dog or a cat. This type of model in which model's prediction are based on particular labels, is called **Classification**. Note that classification problems not only consist of 2 labels, it can have 3, 4, even 100 of labels. Main thing is that, there is fixed number of labels. Model will give predictions within those labels.

In simpler terms, you're trying to categorize or classify input data into predefined classes.

**Regression** is another type of machine learning task where the goal is to predict a continuous numerical value based on input features. In this case, you're not predicting classes; instead, you're predicting a quantity.

For example if you are constructing a model to predict price of house, that price will obviously be varying. Prices in that case are continuous values, they have no range.  And hence your model should predict continuous values.

There are different algorithms for Classification problems and different algorithms for Regression problems.

In both cases, the machine learning model learns from data to make predictions on new, unseen data

**Lecture 11: Video explanation of two types of supervised Machine learning**

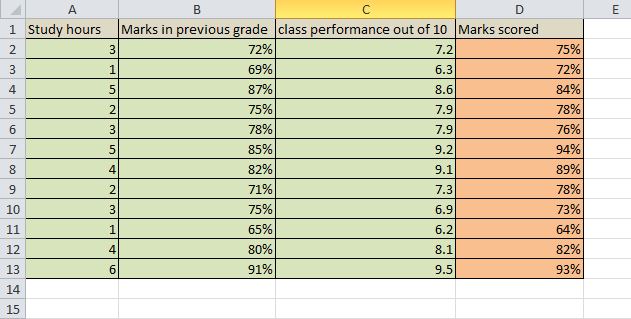
**Lecture 12: Building our model**

Now finally you know how our dataset looks like, labels and features. Now it’s time to make our first machine learning model.

Let suppose *Professor Ahmad* teaches math to 10th grade and he want to predict how much each student will score in his/her annual exams. *Professor Ahmad* contacted you, a data scientist, and requested you to make a machine learning model for his task.

As you know we always need some data to train our train, you will ask *Professor Ahmad* to give you previous data of students. He will take some time and gather data of students.

Let say he provide us give data:



As we see, there are four columns, one of them is label, and other three are features. Overall there are 12 data points.

**Previous line requires your attention!**

This is data that professor has provided us, so we have to construct our model based on this data!

Let’s open up your Python.

**Lecture 13: Building our model part 2**

Now that we know our data, label and features, you are all done with the requirements.

**Loading our data in python environment:**

To load our data in python, we will use Pandas.

import pandas as pd

data = pd.read\_excel('path to your data.xlsx')

For example, if you have saved your data in local disk D, you will write:

data = pd.read\_excel('D://file\_name.xlsx')

'file\_name' will be replaced with the name of excel file that contain our data

**For my case:** To make the path and format more clearly to you, I will show you where I have saved my data and how I loaded my data in python.

I have local disk D and within that, I have a folder named 'datasets' within that folder, I have saved my excel file. And my excel file is named 'professor\_ahmad\_class'.

So my line of code will look like:

data = pd.read\_excel('D://datasets/professor\_ahmad\_class.xlsx')

.xlsx is common in all, it is the format of our excel file which we are loading (all excel file of format xlsx)

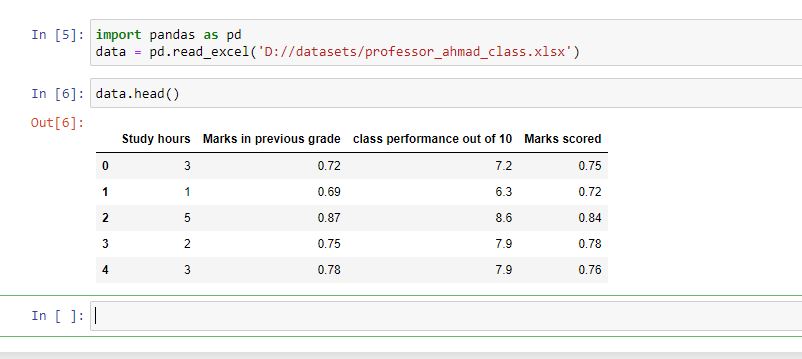
Now as we have loaded the data, let’s see how it looks like in python.

For that, we can use command

data.head()

This command will show first 5 rows of data (data points).

See this code of mine and its output:



You may have noticed that python have automatically removed '%' sign from our data and calculated percentages of those values.  (See column number 2 and 4). Don't worry, we will deal with this while constructing our model.

Great, now we have loaded our dataset in python environment.

**Lecture 14: Video lecture explanation of upper lectures**

**Lecture 15: Building our model part 3, separating features and labels**

Now as we have loaded our data in python, we have to tell our model what are the labels and what are the features. Model can't decide it automatically of course.

As we decided earlier, our **features** are: first 3 columns of dataset. And our **labels** are: last column of dataset. (We want to predict annual exam marks of students). If you are getting confusion in features and labels, it is advised to read section 2, lecture 5.

Now let’s separate out features and labels from our data.

We will store features and labels in two different variables:

**Features:**

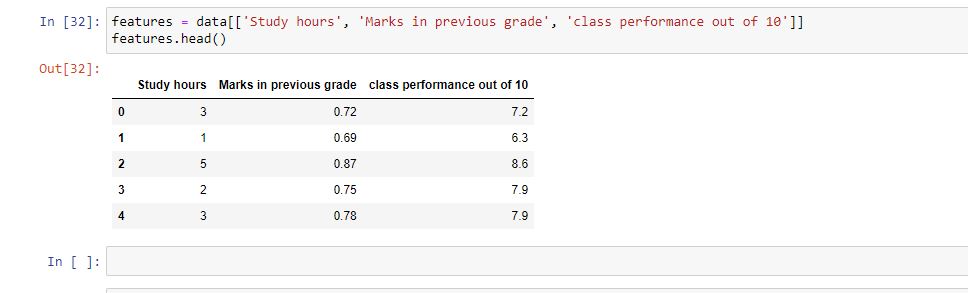
features = data[['Study hours', 'Marks in previous grade', 'class performance out of 10']]

This line of code is specifying features as first three columns of our data.

Let’s see how our features variable looks like. For that, write:

features.head()

And our output will be:



Great, now we see that '*features'* variable contain only first three columns. This is what we wanted. We wanted these three columns as features for our model.

**Label:**

For label, we want only one column, the last one.

label = data[['Marks scored']]

final\_label = label \* 100

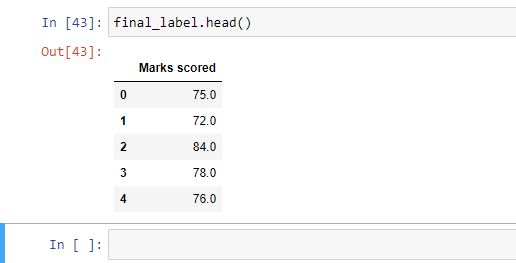
First line of code is specifying the last column of our data as label. As we saw earlier that this column was converted into percent values automatically,  (last lines of previous lecture).

Each value was divided by 100 automatically due to '%' sign. Now we will multiply each value by 100 to make it normal. That’s what we have done in second line of code.

Now our labels are stored in a variable *'final\_label'.*

This is what it looks like:

final\_label.head()



Now we see that our variable *'final\_label'* contain only last column of our dataset that Professor Ahmad provided us.

Now features and labels are ready, train our model.

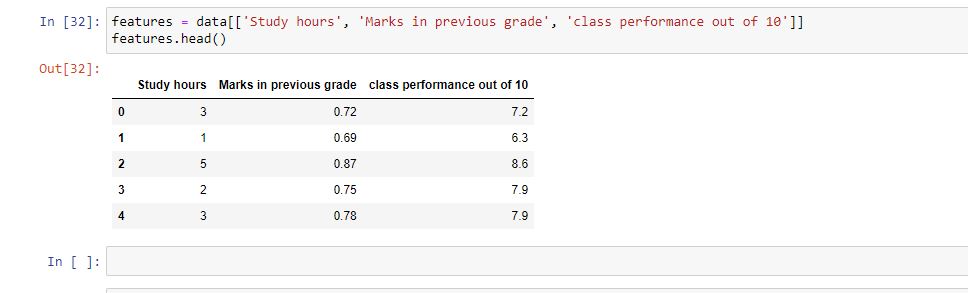
Coding done in this lecture:

features = data[['Study hours', 'Marks in previous grade', 'class performance out of 10']]

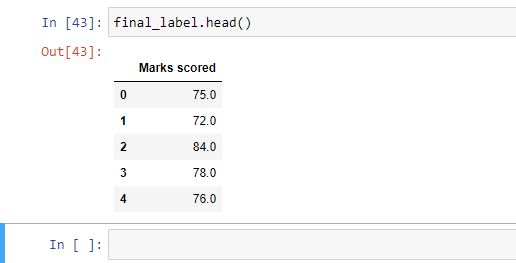
label = data[['Marks scored']]

final\_label = label \* 100

*'features'* Looks like:



*'final\_label'*Looks like:



*'features'* Contain features of our data.

*'final\_label'* Contain labels of our data.

**Lecture 16: Video lecture of separating features and labels**

**Lecture 17: Building our model part 4**

Till now, we have loaded our dataset in python environment, separated features and labels from our dataset. We are all ready to construct a machine learning model. You may be thinking that this example is very simple....This is how we start learning complex things.

There are different machine learning algorithms, and for now we will be using '*nearest neighbors classifier'.*

This is one of the widely used ml(machine learning) algorithm.

Later on we will also discuss how this algorithm will make predictions.

Okay, only thing we need is scikit-learn for this task. **scikit-learn** is a Python module for machine learning

To install this, we open our command prompt and write:

pip install -U scikit-learn

You can read official documentations [here](https://pypi.org/project/scikit-learn/)

Now let’s continue our coding. To make everything clear, I am writing whole code we done till now:

import pandas as pd

data = pd.read\_excel('D://datasets/professor\_ahmad\_class.xlsx')

features = data[['Study hours', 'Marks in previous grade', 'class performance out of 10']]

label = data[['Marks scored']]

final\_label = label \* 100

To construct our model, we will have to import our '*nearest neighbors classifier'*

**Step 1**

We will import our classifier this way:

from sklearn.neighbors import KNeighborsClassifier

**Step 2**

Initialize our model. Its means we have to make a model first, we will train it afterwards.

To create model:

mdl = KNeighborsClassifier()

Here we have created our model which is stored in variable *'mdl'.*

**Step 3**

Here, we will train our model. And it’s pretty very simple!

Recall that we have to train our model on features and labels, and our features are stored in variable *'features'*

And our labels are stored in variable *'final\_label'* (coding is done in lecture 10).

To train our model on features and labels:

mdl.fit(features, final\_label)

We pass features as first argument and labels as second argument. And everything is done.

Now our model is ready. Yes, it’s very simple. Our model is ready to make predictions.

We will give this model to Professor Ahmad and for sure he will ask how can I use this thing? How to make prediction from this model?

For that, we need to give three pieces of information to our model, and that were features.

Let suppose there is another student of Professor Ahmad and he want to know how much marks that student can perform according to our model. For that, we must provide the model with these pieces of information:

1 - Study hours of that student.

2 - Marks he scored in previous exams.

3 - Class performance of that student out of 10.

Let suppose, the new student study 7 hours a day, have scored 83% in his previous exams, and class performance of that student is 8.0 out of 10. We will use *'predict'* method.

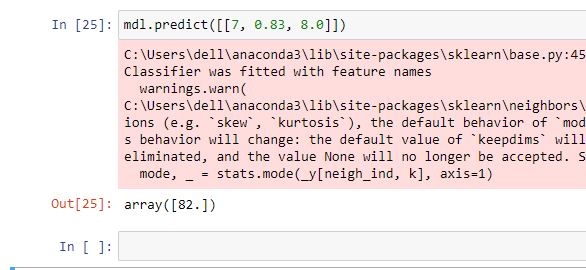
Let’s feed this data in the model:

mdl.predict([[7, 0.83, 8.0]])

This is how we feed our data. You may have noticed that second argument (marks) is 0.83 instead of 83.

Well, while training, we used these values so that's why to make prediction, we will give our model data in same format in which model was trained.

What will be the output of upper code? (In which we made prediction)



**Greaaat,** our model has predicted that the student will score 82% marks in his annual exams.

That pink box in image is not part of code, we can ignore this.

Let’s talk about model performance...

We had a student who study 7 hours a day (too much). He was performed 83% in previous exams, and class performance is 8.0 out of 10.

It is clear from this data that this student is extraordinary and will score well. And the same thing is predicted by our model.

This is what Professor Ahmad needed, now he will input data of student and explore how much each student can perform.

Overall coding in this lecture (Training our model):

from sklearn.neighbors import KNeighborsClassifier

#initializing our model

mdl = KNeighborsClassifier()

#training our model

mdl.fit(features,final\_label)

#making predcitions

mdl.predict([[7, 0.83, 8.0]])

Note that the lines containing # are not parts of code; these are placed for instructions purpose.

Now one thing Professor may request you is to make your model user friendly. For that, I can design inputs statements that will take inputs from Professor Ahmad and then I will take Professor's answers and pass these to the model.

This way:

study\_hours = int(input('How many hours your student study per day: '))

marks\_previous = int(input('How much he scored in his previous grade give your answer in percentage: '))

mp = marks\_previous / 100

class\_performance = int(input('What is class performance of student out of 10: '))

prediction = mdl.predict([[study\_hours, mp, class\_performance]])

print('Predicted marks: ')

print(prediction)

After we have our KNieghborsClassifier on training dataset, we can now make predictions on test data. Professor wants us to make the model as much user friendly as it can be. One consideration you should have in your mind is that this model predicts marks on the basis of given criteria.

Now there are two ways of giving test data, either you can give list containing features which you can feed into model or you can define your variables. In the above code, I am giving this privilege to the professor that the model will one by one ask him about the features.

**Defining variables in Python**

**1)** Variables are like containers that hold values, and you can assign different types of data to them.

**2)** user can define the value of variable by writing it in input form

   study\_hours = int(input('How many hours your student study per day: '))

Here study\_hours is our variable and by giving **input** as a key word, when the code is run, it asks the user the question asked in string  **'How many hours your student study per day: '**

**3)** Outside the key word input, we wrote int, this is because we want data type of our feature to be integer not string!

mp = marks\_previous / 100

Here, we are dividing previous marks by 100, because they are in percentage form. For example: 50%, 85% and so on but we want our data in the floats (as it was in the training data) It will make valid predictions when test data and training data are of the same type

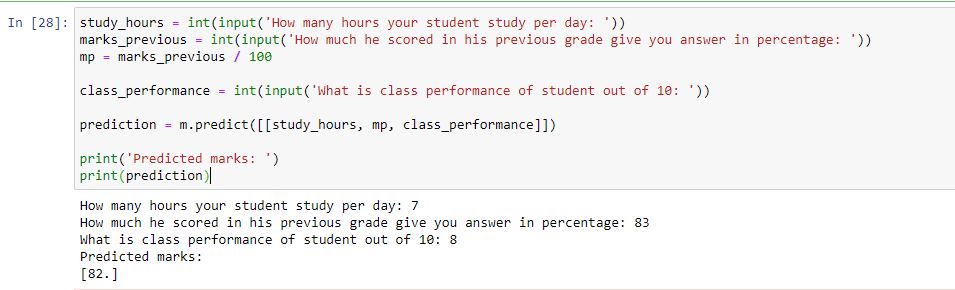
**Predicting on variables!**

After we have defined our variables, we now make predictions with our variables. The code will ask previous marks, study hours and class performance of a student. Professor Ahmad will answer values one by one. And then model will predict the score based on these variables (answers from professor).

The benefit of defining variables is that the model now becomes a lot user friendly and easy to understand than it was before!

--

Test run:



Congratulations! Now our model is ready for Professor Ahmad.

**Lecture 18: Video explanation of upper lectures**

**Quiz 3:**

**What keyword is used to train model?**

model.train()

model.fit()

**What keyword is used to make prediction from our model?**

model.predict()

model.guess()

**int(input('write your age: '))**

**What is role *'int'* in this code?**

Takes input from user

Specifies that our answer should be in form of integer.

**Lecture 19: Predictions at once**

Now let’s pass a bunch of students in our model and let’s see how well our model predict.

Note that we can also pass data of more than one student in our model. For example Professor Ahmad won’t have to input data of each student one by one, he can also input data of as many students as he wants in one step.

Recall that preciously we were having predictions using this line of code:

mdl.predict([[7, 0.83, 8.0]])

Note that there are two [ ] brackets. This mean it is a list within list.  
Square brackets at the edges can be referred as parent list, and square bracket inside can be referred as kid list.

We always give data points in the kid list. What will happen if we increase our kid lists? Each kid list contains data of one student so more kid lists will contain data of more students. This way we can have prediction of whole class or more number of students as once.

Have a look at upper code,

First argument is number of study hours, 7.

Second argument is percentage of previous class, 0.83.

Third argument is performance in class out of 10, which is 8.

We will have to give data in the same arrangement.

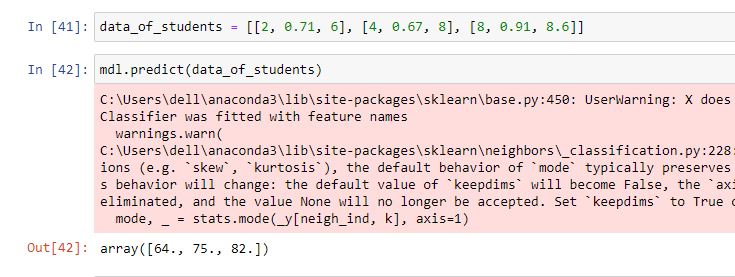
data\_of\_students =  [[2, 0.71, 6], [4, 0.67, 8], [8, 0.91, 8.6]]

Here I have defined a variable *'data\_of\_students'* in which there are lists within list. Each list contains data of a specific student.

We already have a model, let’s now pass this data into our model and let’s see how predictions are made.

mdl.predict(data\_of\_students)

Output is:



We can give list within lists for prediction of relatively large data at once as shown in the above code.

There are three outputs for our three data points (lists of data). And these make good sense.

Have a look at data of first student, who study 2 hours a day, scored 71% in previous exams and class performance is 6. Our model is saying that this student will score 64% marks in annual exams.

And for the third student, who is a book freak, studies 8 hours a day, have scored 91% in previous exams and class performance is 8.6 out of 10. And our model predict that he will score 82% in annual exams, which is pretty good prediction. Note that these predictions are not considered as exact values, these are approximate values.

**Lecture 20: How our model works**

We have the model which is making good predictions yet. But we don't know how this model works and made predictions. Let's discuss that now. As mentioned before and seen in the code, we are using **KNeighbors Classifier.**

Kneighbors classifier or nearest neighbors classifier is one of the major machine learning algorithm and also considered as simplest one.

**To make a prediction for a new data point, the algorithm finds the closest data points in the training dataset.**

*Quick-Revision: Data point is formed after combining all the features. In previous lecture, the lists of data that we provided our model were data points*

Now read the bold line again.

Let’s have a breakdown of this algorithm.

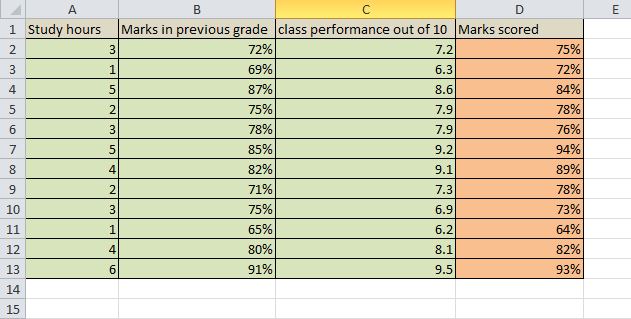
Let suppose I am a bad person you are a good person. And then comes another person who is your best friend. That new person is more close to you than to me. Now any other person may think that this new person, who is your friend, is a good person. Well the logic is clear, you are a good person so your friend will be considered good and likewise my friend may considered as a bad person because I am a bad person.

Now take you and me as data points. You are a data point and your label is *'good person'* and I am another data point and my label is *'bad person'.* After this, take that new person as the data point that we want to predict. As that new person is more close to you, we will give classify him as *'good person'.* This is how nearest neighbors work. It finds the nearest data point to the new data point and classify new data point as that nearest data point. In our case, it finds you more close to that new friend of yours and hence classified that friend as a good person.

This is how nearest neighbors classifier works. 'neighbors' refer to the data point.

Let’s now see how it implements on our dataset.

Our dataset was:



This was the data on which we trained our model. Recall that column 2 and 4 was converted into point values. Each value was divided by 100. For example: 72% will be considered as 0.72 and same way 69% will be considered as 0.69.

We already have trained a model on this data and have made predictions through it. This was one of the code we used in lecture 17 to do prediction.

mdl.predict([[7, 0.83, 8.0]])

Now let’s see any closest data point to this data. Have a look at dataset and try to figure out any student who has approximately same features as this new student on which we are doing prediction.

We have to find any student from our data who approximately:

Study 7 hours a day

Have scored 83% marks in previous class

Class performance is 8

Take some time and find that data point from upper image of data

.

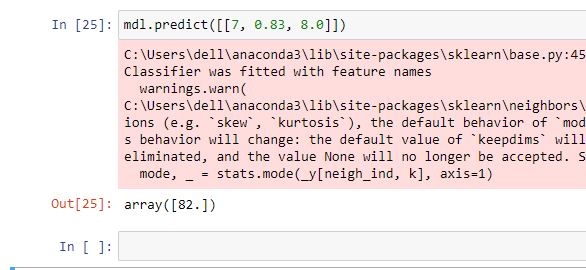
.

.

We may find many students close to this new data.

For this case I think we can consider two students, *row 12* and *row 4*. These two students have data close to our new student. This is not very close, but closer than others. Well python won't have this confusion of *'approximately'.* Python will measure exact distance between data points mathematically and will have exact one answer.

Based on our observation, row 4 and row 12 students can be considered as *'nearest neighbor'*. Row 4 student have scored 84% marks and row 12 student have scored 82% marks. New student will be predicted one of these marks.



As expected! 82% marks is one of the marks we thought model will predict.

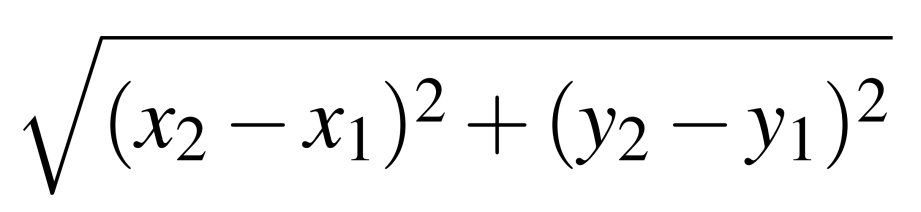
What model has done? Model have calculated mathematical distances of new data point to every data point in our training dataset, the most least distant data point is nearest data point, for our case it looks like 12th row has least distance  to our new data point and hence its label (82) is given to new data point (Predicted value).

Now go at the top of this lecture and again read that bold statement.

I hope you got the concept. Video lecture give further clearance.

We haven't discussed how our model calculates the mathematical distances. Here is the brief information about how those calculations are made:

* We plot data points in 3 dimensional planes (as there are three features).
* Then we find coordinated of every training point.
* When a new data point is added (test data point), we plot that plot in the same coordinate system.
* Now we have coordinates of test point and training points. We find the distance of test point to every training point, and the point with least distance is selected as nearest neighbors.



Model calculates distance using this formula which you may remember from your high school.

**Lecture 21: Video explanation, How Knn model works**

**Lecture 22: How our model works, understanding the concept**

**Lecture 23: Training model on more than one neighbor**

In next video we have explained how model with more than one neighbors work. Note that until now we didn't specify number of neighbors (by default model automatically select 1 neighbors) but we will do that in next example of nearest neighbor classifier.

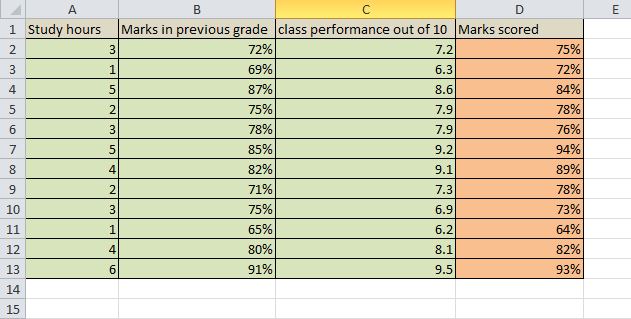
It’s not mandatory that you should understand things in the first attempt. It can take time and that's the boring part of learning something. Take it slow (1 hour a day is more than enough in beginning) and steady.

**Lecture 24: Video explanations, training model on more than one neighbor**

**Lecture 25: All in once**

We have done basics of machine learning. Let’s have a look at the code we write yet. I will not be explaining this code now. You will have to take a look at each line of code and make sure that you understand everything.

Dataset:



You can download this dataset from resources of lecture 12.

Coding to train a model based on this data:

import pandas as pd

data = pd.read\_excel('D://Professor ahmad class.xlsx')

features = data[['Study hours', 'Marks in previous grade', 'class performance out of 10']]

label = data[['Marks scored']]

final\_label = label \* 100

from sklearn.neighbors import KNeighborsClassifier

mdl = KNeighborsClassifier()

mdl.fit(features,final\_label)

mdl.predict([[7, 0.83, 8.0]])

Last line of code is to make prediction. For more predictions at once, we used this code:

data\_of\_students = [[2, 0.71, 6], [4, 0.67, 8], [8, 0.91, 8.6]]

mdl.predict(data\_of\_students)

To make this code more user friendly, we used this code:

study\_hours = int(input('How many hours your student study per day: '))

marks\_previous = int(input('How much he scored in his previous grade give you answer in percentage: '))

mp = marks\_previous / 100

class\_performance = int(input('What is class performance of student out of 10: '))

prediction = mdl.predict([[study\_hours, mp, class\_performance]])

print('Predicted marks: ')

print(prediction)

Have a grip on this lecture and previous one.

**Assignment 1: Machine learning assignment**

**Make neat notes of nearest neighbors classifier on your notebook. This practice gives us great confidence. Discuss how it works, don't focus on decorations. Make it in less time and more valuable. At least make it neat so that you can have benefit from it later.**

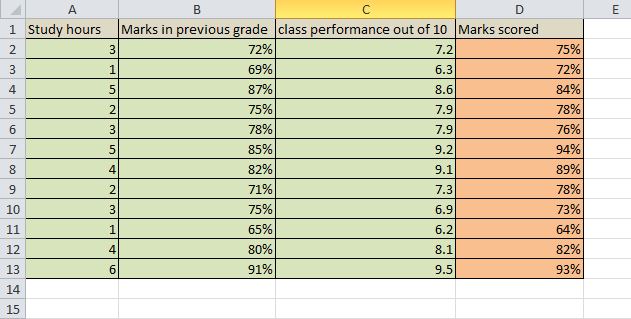
**Also try to add short code snippets.**

Assignment Instructions

Get your paper and pen and create notes. This may sound childish but this will make your concept clearer. Also try to find someone like your sibling or friend and explain to them how neighbors classifier works and show them your model. If you don’t fine anybody, create a video of yours explaining how this model works, and send that video to me. I will reply. (This is optional but try to do this)

Also discuss how neighbor’s classifier is so good for our predictions?

**Quiz 4: Predictions of our model**



This is our data. A new student have data point = (3, 0.74, 6.8).

To which data point, our new data point is more closer? Answer in terms of row number.

3 13

6 10

What do you think our model will predict if I pass it the data points of new students same as in previous mcq question.

mdl.predict([[3, 0.74, 6.8]])

73 78

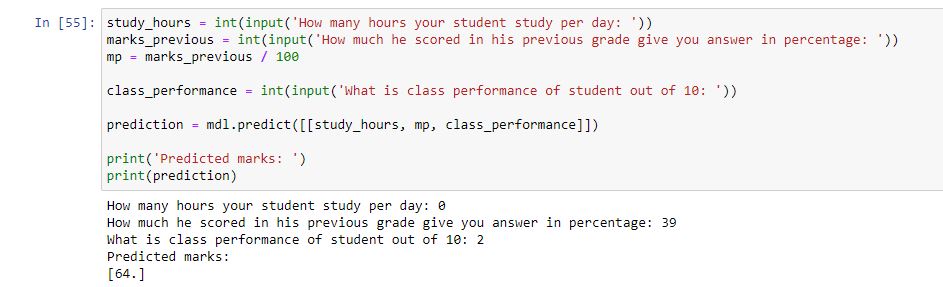
Correct answers: 10, 73

**Lecture 26: Any drawbacks of neighbors Classifier**

As now as we have discussed much about Kneighbors classifier, let’s explore if there are any drawbacks in our model.

*Quick-Revision: To make prediction on a new data point nearest neighbors classifier find nearest data point to the new data point and then make prediction based on label of that nearest data point.*

Let us consider a student who plays video games all day and study 0 hours a day.



**Drawbacks of KNeighborsClassifier:**

kNN is sensitive to the scale of features. It means our kNN model can make relatively accurate predictions if and only if the test data we are feeding into the model falls in the range of training data! It is important for you to understand this statement.

Let’s relate this statement to the model we have trained. The range within which our students studied in the training data is 1-6, previous class scores ranges between 0.65 and 0.91 .

Now if we introduce a new student as test entry and his study hours are 16 and previous class score is 100, we might assume that the predicted score should be 100 in this case as well but our trained model will give 0.91 because it is the last entity/range till which our model is trained. Similarly, observe the above given image, we gave input of 0 study hours, 39 percent previous score and 2/10 in class performance, here the study hours and class participation deviate a lot from the training data, that’s why the predicted score of 64 might sound false to us and it actually is weird for someone with 0 study hours to score 64!

In order to overcome this issue, another model is introduces to you which is Linear Regression. We'll train Linear Regression model on the same dataset so that benefits and drawbacks of each can be discussed simultaneously!

**Lecture 27: K in Kneighbors Classifier**

K in neighbors classifier deflect the number of neighbors on which we train our model. We will explore further about changing number of neighbors in next lectures (Section 4). For now we will be exploring another important model of machine learning.

**Lecture 28: What is Linear Regression and why use this**

As we discussed before, there are many algorithms of machine learning, Linear Regression is one of those algorithm.

As the name says, it is some kind of linearity and it is a regression model. If you want an idea of regression and classification, read the section 2 lecture 10 of this course.

Quick Revision: When we know the number of classes that we want to predict, it is called Classification and when the predictions can be any number, there is no defined class, then it will be called Regression.

Linear Regression is the model that is designed to predict on continuous data (regression problems). Neighbors classifier was having the defect that it could make predictions only from the labels that it is trained on (section 2 lecture 26).

We hope you remember the defects from our nearest neighbors model, now to overcome those defects we are going to use linear regression model.

Math used behind linear regression models is not part of this course, but here is brief idea:

* When training, model tries to find a straight line that is closest to all the data points. (take distance of each data point to that line, take square of each distance, and add all answers, that is mean squared error.)
* In other words, that line is closest to all the training data points.
* Based on that straight line, our model will make predictions for the coming unseen data.

**Lecture 29: Making our Linear Regression model**

**Step 1 - Preparing data**

As we already are familiar how we load our data in python, separate features and labels, now it will be very easy to code linear regression model.

# loading the dataset

data = pd.read\_excel('D://Professor ahmad class.xlsx')

# specifying features and labels

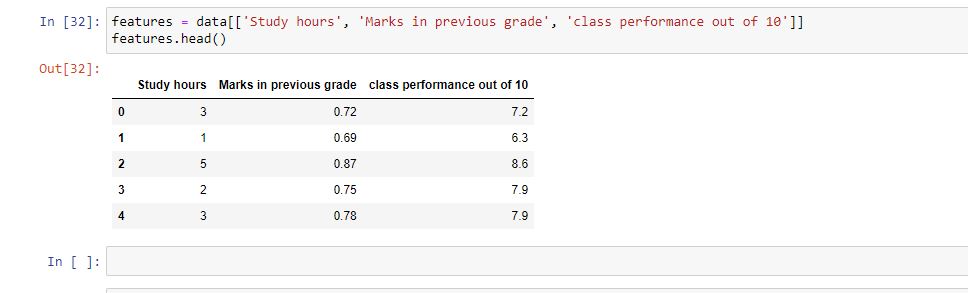
features = data[['Study hours', 'Marks in previous grade', 'class performance out of 10']]

label = data[['Marks scored']]

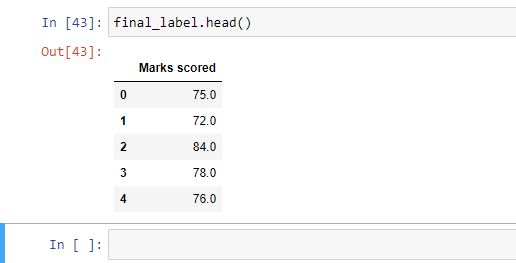
final\_label = label \* 100

Lines containing *'#'* in the start are not considered as part of code, they are added your instructions. This code is same as we used before. If anything is not clear, you can look at section 2 lecture 13 and 15. This code is explained there step by step.

This is how our features look like:



And this is how our labels look like:



As we have our data loaded and made variables on which we will train our model, let’s train our linear regression model.

**Step 2 - Importing linear regression**

from sklearn.linear\_model import LinearRegression

This line of code will import linear regression in our python environment

**Step 3 - Initializing model**

Once imported, we first have to define a model which we will train later on.

model = LinearRegression()

The same way we initialized model in nearest neighbors, here we are initializing linear regression model.

**Step 4 - Training our model**

The same we trained our neighbors classifier model, here we will define our linear regression model.

*Quick-revision: While training we pass features as first argument and labels as second argument*

model.fit(features, final\_label)

That's it! Our model is ready to use.

**Lecture 30: Video explanation of linear regression model**

**Lecture 31: Performance of our model**

By the end of previous lecture, we have trained our linear regression model. It’s time to check to see if there is any improvement so we can give our modified model to Professor Ahmad.

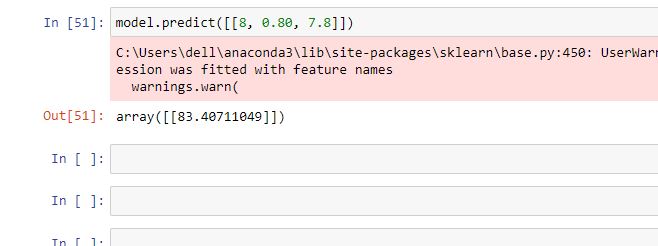
To make a prediction, syntax is same as of neighbors classifier.

We use this line of code:

model.predict([[8, 0.80, 7.8]])

In the new data I have specified a student who study 8 hours a day, has scored 80% marks in previous exams, and class performance is 7.8 out of 10.

The predicted answer (annual exam marks prediction) by our linear regression model is: **83.40.**



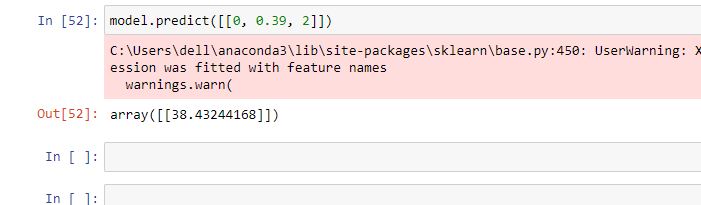
There is not any single student is our training data who performed 83.40. So our model is finally able to make predictions that are not present in our training data. Next prediction will give further clarity.

We discussed a student in lecture 26 who study 0 hours a day, have scored 39% in previous exams and class performance is 2. When we passed this data to neighbors classifier, our prediction was 64% and this is pretty unrealistic that student with this record will score 64% in annual exams (still possible). The reason that our model predicted 64% was: The data point with lowest marks in our training set was with marks 64%. The features of this least marks student were closest to the new student that why our model (KNN model) predicted 64%.

Now let’s use linear regression model on same data and let’s see the predicted answer.

model.predict([[0, 0.39, 2]])

Output:



There is a great improvement in our model!

Now for the same case our model is predicting that the student will score 38.43 % in annual exams. (Based on the predictions made, this model sound more accurate to me than the nearest neighbors model)

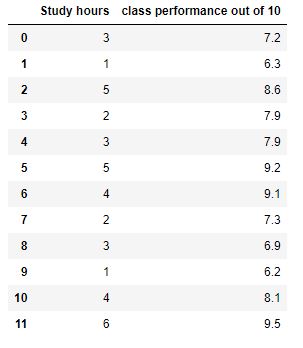
Furthermore we can make our model more user friendly by the technique we used in nearest neighbors model.

**Lecture 32: Line of best fit**

To understand how predictions are made through linear regression model, lets first understand how to plot data points in 2d plane.

**Assignment 2: Plotting data points**

**Take your notebook out and plot the data. See the assignment instructions for details.**

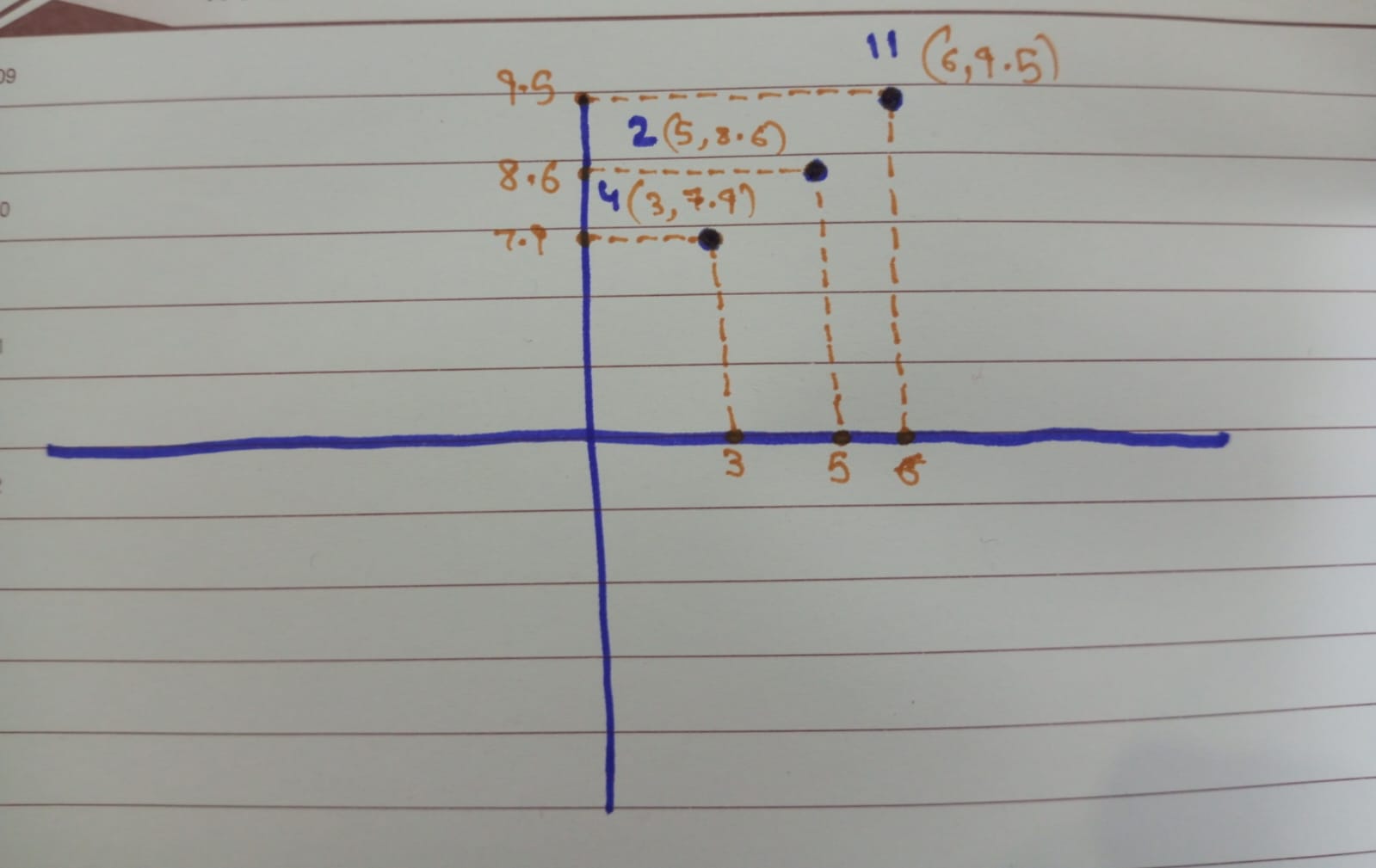


**How to plot this?**You may remember from your primary math book how we plotted x and y in a two dimensional plane. We draw two lines cutting each each other (x and y axis). Then we mark number on those axis and plot our data.

Now consider study hours as x and class performance as y.

For the first data point, we will plot (3, 7.2) in our xy axis and so on.

For your help I will plot few data points (2nd, 4th and 11th) from our dataset.



Orange dotted lines are just for instructions. Data points are marked with marker. With each data point I have mentioned there coordinate and number of data point that I plotted from dataset.

You have to mark each data point at right place. Furthermore you don't have to to draw those dotted lines if you don't want to, and you also don't have to mention coordinates. Just make a rough graph for your understanding.

Question 1

How you plotted the data? What is x and y from our dataset?

x is study hours in our dataset and y is class performance. You may have taken y as study hours and x as class performance. More suitable idea is first one.

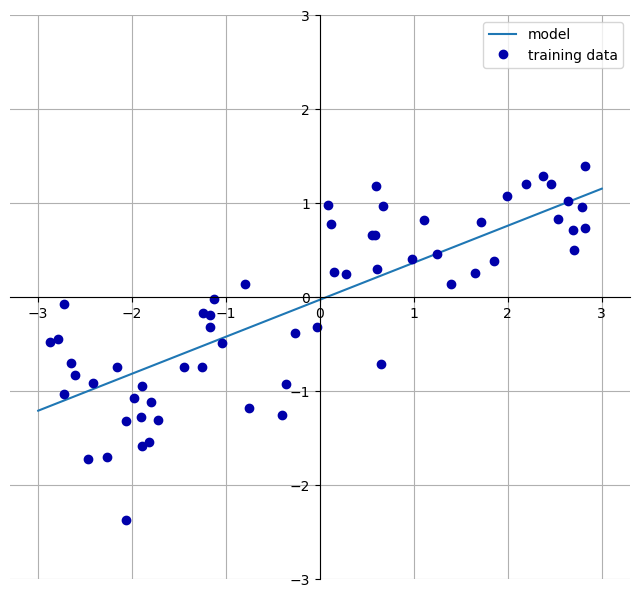
**Lecture 33: Video explanation of how linear regression works, explaining line of best fit**

**Lecture 34: Video explanation of line of best fit, part 2**

**Lecture 35: Line of best fit part 2**

As you now understand how to plot data,

Have a look at this graph:



There are many data points scattered in this graph in blue color. To mark the data points, same logic is used as used in the assignment. Note that there is a blue straight line in the graph. This line is plotted in such a way that it is closest to all the data points. This is what our linear regression model do, it plot all the data points and then plot a line that is closest to all the data points, and that line is known as line of best fit.

Bottom of Form

**Lecture 36: When to use nearest neighbors classifier**

As we saw in the Professor Ahmad's model that linear regression was working more efficient than nearest neighbors. But that's not the case always. It depend on the dataset you are working with. There are many cases in which nearest neighbors work with more accuracy than the linear regression model.

**Why?**

As discussed in lecture 10, there are two kinds of machine learning problems and there are different algorithms for those. Those two kinds are: **Classification problem and regression problem.**

Nearest neighbors classifier is widely used for classification problems and linear regression is used for regression problems. What type of problem we were dealing with?

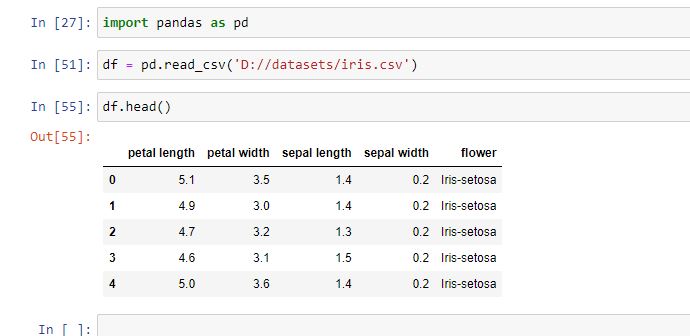
It was a regression problem! That’s why our regression model (linear regression) performed better.

It’s important to recognize the type of problem you are dealing with.

**Lecture 37: Iris data, classification problem**

Now we will be working on a classification problem dataset. You can download that dataset [here](https://www.kaggle.com/datasets/vikrishnan/iris-dataset)

Iris dataset is very famous dataset for working with classification models in python. You can see four columns which are petal length, petal width, and sepal length and sepal width.



As you can see in the above image after loading the dataset we print its head by .head() attribute in python. It will print first five rows of our dataset and now we can observe our dataset and see what kind of predictions we want to make!

A botanist did this research and measure these features of 3 categories of flowers which are **setosa, virginica and versicolor**(labels). By using .shape attribute we see that there are 150 rows.



That means there are 150 data points for which the botanist researched about the three species of flowers!

**Now what we'll be doing is that we are going to train a classification model on this dataset and that classification model will predict the specie of flower when new data points are given other than the given dataset.**

**Let's begin building our 1st extensive classification model!**

Importing nearest neighbors classifier:

from sklearn.neighbors import KNeighborsClassifier

After importing KNeighborsClassifier, we now have to define our features and labels. Features are the data we give our model as input and label is the output we want. In this case, the features will be the petal length, petal width, sepal length, sepal width. Our label will be flower column which contain our three wanted species names.

features = df[['petal length', 'petal width', 'sepal length', 'sepal width']]

label = df['flower']

The syntax for defining features and label is pretty simple. We are simply using square brackets to extract the wanted columns from our data set. The variable we used "features" and "label" is obviously changeable, you can use whatever you want. By convention mostly used variables are X for features and y for label but here let’s go along with features and labels as we defined them.

**kNN Classification**

We define the model.

model = KNeighborsClassifier()

Fit in the data.

model.fit(features, label)

Now we can easily make predictions by giving new features as integers. Let suppose you have a flower at your home with petal length of 2cm, petal width of 3cm, sepal length of 1cm and sepal width of 2cm. We can make prediction on our flower data this way:

model.predict([[2, 3, 1, 2]])

**OUTPUT:  SETOSA**

But do we know the science behind all this?

We hope you remember lecture 20, which was all about how nearest neighbors classifier make predictions. It is advised to read that lecture again.

Quick-revision: KNeighborsClassifier classifies on the basis of Neighbor's label as the name suggest but what does that actually mean. Let's understand the crux working of kNN without going any further!

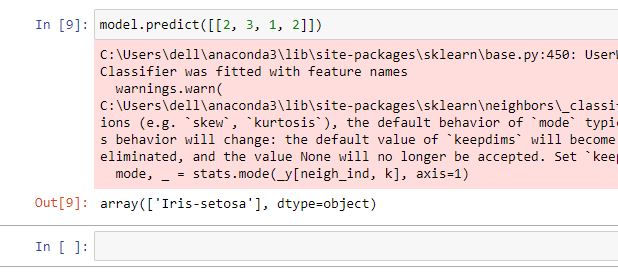
While training the model, we can also play around with the number of neighbors. We will discuss those details in the video lecture.

**Lecture 38: Video explanation of how to make KNN model on iris dataset**

**Lecture 39: How well our model works?**

In this section, we will discuss how to evaluate the performance of our models. Remember from section 4 where we created the model to have predictions on category of flower. We provided a data point to our model and it gave us a prediction. Now how do I know that the model is predicting good or not?

This was one of the predictions we made:



Maybe flower wasn't setosa and our model has made a mistake... Let’s explore how to know whether the predictions are correct or not.

**Lecture 40: Splitting our data**

Splitting the data into two chunks is most widely way to evaluate the model performance. For example, in the iris dataset we were having 150 labeled data points.

Let say we have divided the data into 100 data points and 50 data points. We have trained our model on 100 data points. After that we will make a prediction on those 50 separated data points. As that data was already labeled so we know the original labels of those 50 data points.

Now we will just see whether the predicted label matches with original label or not. Hence we can analyze how many times the model gives accurate predictions and how many times wrong prediction on those 50 data points.

**Training data**

The data on which we train our model is called training data.

**Test data**

The data on which we check our model performance is called test data.

Normally we train our model on 70% of our dataset and test our model on 30% of our dataset.

Let's apply this technique to iris dataset and analyze how well predictions are made by our model.

**Lecture 41: Checking accuracy of model**

To split our data into we have a module known as *train\_test\_split.*

We suppose that dataset is already loaded in python environment.

**1 - Import module**

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

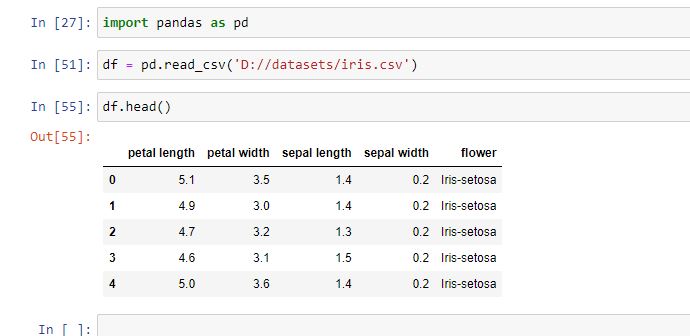
**2 - Specifying features and labels**

Preview section 4 for the details. We used this code to specify labels and features.

features = df[['petal length', 'petal width', 'sepal length', 'sepal width']]

label = df['flower']

*Quick-revision: Iris dataset was a flowers categorical dataset with 4 features of a flower, those are: petal length, petal width, sepal length, sepal width. 5th column specifies from which category the flower belongs to.*



**3 - Train-test-split**

As we have specified features and labels, let’s split those.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, label)

There are four variables created here.

1. X\_train: These will contain training data points (features only).
2. X\_test : These will contain test data points (features only).
3. y\_train: These will contain training labels (labels only).
4. y\_test: These will contain test labels (labels only).

We have to train our model on training data points (features) and training labels.

So:

**4 - Training**

model = KNeighborsClassifier()

model.fit(X\_train, y\_train)

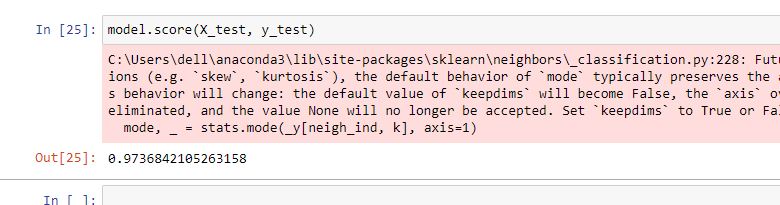
Notice that we have trained our model on training data points.

**5 - Accuracy of our model**

Now to check the accuracy on test data, we use this line of code:

model.score(X\_test, y\_test)

Output:



Great! Our model have achieved an amazing accuracy of 97%

**How this accuracy is measured?**

Note that X\_test contain all data points we reserved for test purpose. y\_test contain all labels for that data points.

To calculate the accuracy, we use .score method of model, and then it will automatically do some math.

It will predict on each of the data point of X\_test and then will check whether the predicted label matches with actual label or not. Hence we will have number of correctly made predictions and number of incorrectly made predictions.

To calculate the accuracy we use this formula:

*accuracy = correct predictions / Total number of predictions*

Let say there were total 30 test data points and our model predicted 20 with correct label and 10 with incorrect. To calculate the accuracy of this model, put the values in upper formula:

*accuracy = 20 / 30  = 0.66*

Note that this score is out of 1, to get the percentage out of hundred, multiply this value by 100. So we can say that the model accuracy is: 0.66 \* 100 =  66%.

We don't have to do this calculation manually; we can just use .score method as done in the upper image.

**Lecture 42: Video lecture, Understanding technique for model accuracy**

**Lecture 43: Explanation of calculating model accuracy on iris dataset**

**Quiz 5: How to calculate model accuracy, MCQ**

**How do we analyze how well our model is working?**

Divide data into two chunks, train model on one chunk and test it on other chunk.

Test the model with data on which it is trained

There is no such technical way to do that

**Lecture 44: What is Logistic Regression**

Despite its name, Logistic Regression is a classification algorithm and not a regression algorithm, and it should not be confused with Linear Regression.

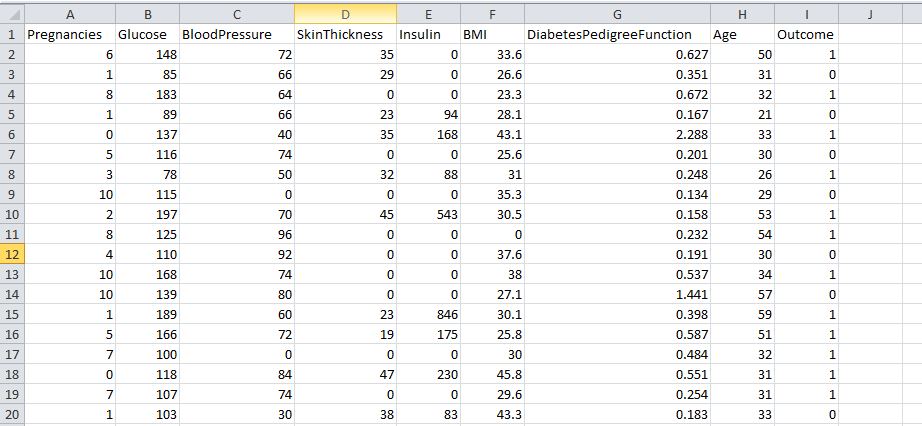
Prediction made by logistic regression models is between 0 and 1. If prediction in less than 0.5, we consider it as 0 and if greater than 0.5, we consider it as 1. Don't worry if you don't get the logic right now, sooner after the example, these things will became clear.

**Lecture 45: Video explanation of Logistic regression**

**Lecture 46: Working with diabetes dataset**

**Dataset we will be working with:**

For this model, we have selected a famous machine learning dataset that is diabetes dataset. This is how that dataset looks like:



This is a small upper portion of our dataset. Each row of this dataset gives data about one costumer. There are 8 features in this dataset which are:

1. Number of pregnancies.
2. Glucose level.
3. Blood pressure.
4. Skin thickness.
5. Insulin level
6. BMI (body mass index)
7. DiabetesPedgreeFunction (probability of diabetes in family).
8. Age.

The last column is the Label, this is in binary format. Here 1 means that patient is suffering with diabetes and 0 means that patient is not suffering with diabetes.

**Analyzing a costumer (12th row):**

She had 4 pregnancies,

Her glucose level is 110,

Blood pressure is 92,

Skin thickness and insulin level is 0,

BMI is 37.6,

Probability is 0.191 (very low),

And her age is 30.

**Based on this data, her label is 0, which means she isn't suffering with diabetes.**

Now it’s your time. Try to analyze row 20 and write down your observation on a paper.

You can download whole dataset [here](https://www.kaggle.com/datasets/mathchi/diabetes-data-set)

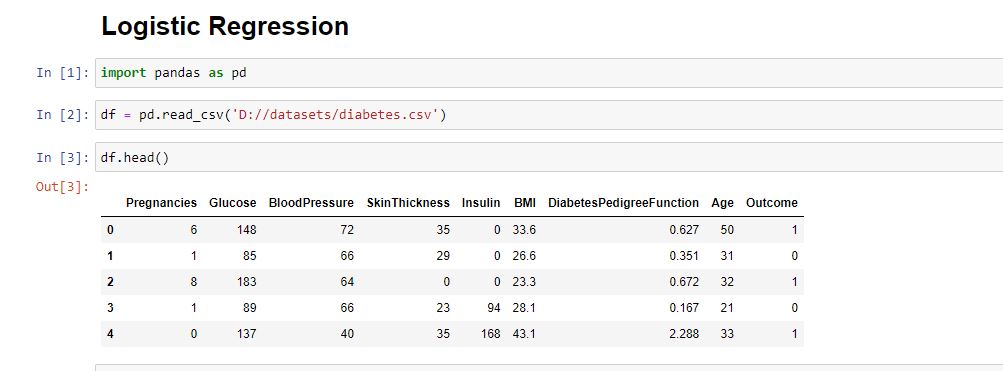
**Lecture 47: Making logistic regression model**

We have to load our data and separate features and labels from dataset.

This is how we load our dataset:

import pandas as pd

df = pd.read\_csv('D://datasets/diabetes.csv')



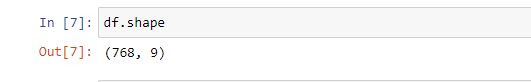
Note that this dataset is not in excel format, it is in csv format. That's why we have used read\_csv method and .csv path format in upper code.

.head() method of data frame is showing us first 5 rows of dataset.

To check the data shape, we can use .shape method of data frame this way:

df.shape

Output will look like this:



First element is number of rows in our dataset and second element is number of columns. It is clear from this output that our data set contain information of approximately 770 patients, which is a large number than before.

You can also explore this in the csv file.

As the dataset is loaded, its time to specify features and labels from this dataset. It is clear what we want as prediction from this data, and that is the last column which is telling us whether the patient has diabetes or not.

Hence our label is last column *'Outcome’.*

And all other columns are features.

To specify these variables, instead of writing name of each column, let’s use another method (writing 8 columns is not feasible)

That technique is:

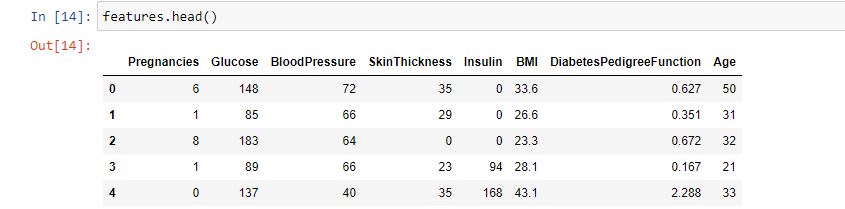
features = df.drop('Outcome', axis=1)

labels = df[['Outcome']]

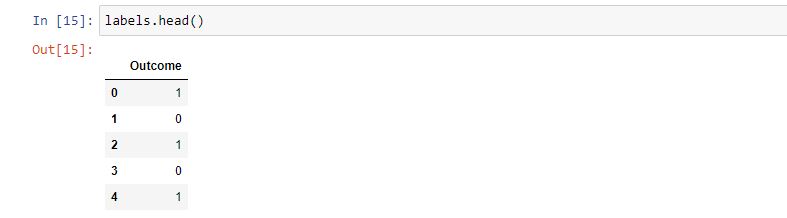
Instead of choosing 8 columns from 9 columns, I have deleted one column (label) from real data so that we are left with other 8 columns, which are features. *Axis=1* is to make sure that I want to delete a columns.

Labels are defined with the same method as done before.

Features look like this:



Labels look like this:



**Lecture 48: Making logistic regression part 2**

To import logistic regression, we use this line of code:

from sklearn.linear\_model import LogisticRegression

The way we train our logistic regression model is same as other algorithms discussed.

**Step - 1 : Initialization**

model = LogisticRegression()

**Step - 2 : Training/fitting**

model.fit(features, labels)

Features and labels are the variables that we made in previous lecture. Features contain first 8 columns and labels contain last column of data that is to be predicted later.

**We are all done! Let’s make predictions**

As discussed before, logistic regression returns the probability score.

To get that, we can use *predict\_proba* method like this:

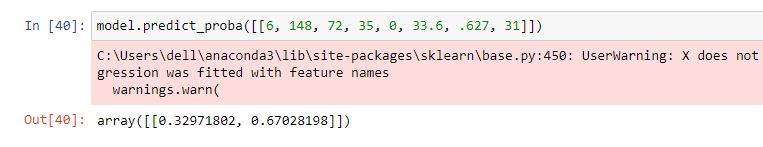
model.predict\_proba([[6, 148, 72, 35, 0, 33.6, .627, 31]])

I have given a new data point containing information of a patient that is:

* 6 pregnancies
* 148 glucose level
* 72 blood pressure
* 35 skin thickness
* 0 insulin
* 33.6 body mass index
* 0.62 probability of diabetes in family
* 31 age

Note that these are 8 pieces of information, same as the number of features.

Output is:

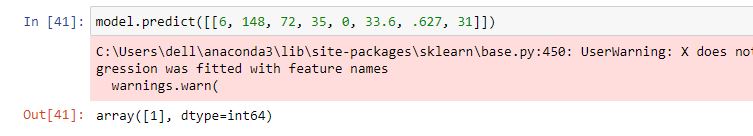


This prediction doesn't seem very familiar. How many outputs were we having? Those were just two, 0 and 1. As there are only two outputs available, hence we have 2 results in this prediction.

Each element in upper array gives probability of possible prediction.

* probability of 0:   *0.32971802*
* probability of 1:    *0.67028198*

As we see that probability of 1 is greater than 0, **hence our model thinks that this patient is suffering from diabetes.** Instead of these two results if you only want to show the predicted label, you can use .predict method like this:



As discussed in upper lines, our model thinks that this data point is suffering with diabetes.

Similarly you can make more predictions and see the probabilities.

Using *.predict\_proba* method we can also analyze how much confident our model is on its decision (more the probability, more is is confident).

**Lecture 49: Video explanation of making logistic regression model**

**Lecture 50: Accuracy of our model**

In our previous model, we didn't split our data into training and test chunks, that's why we can't calculate accuracy directly. To do so, we are going to train another model from beginning.

These lines of code load the dataset and specify features and labels from our dataset.

import pandas as pd

df = pd.read\_csv('D://datasets/diabetes.csv')

features = df.drop('Outcome', axis=1)

labels = df[['Outcome']]

These are the imports of modules that we need:

from sklearn.linear\_model import LogisticRegression

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

This line of code split the data into two chunks.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, labels)

We will train our model on X\_train and y\_train. After training we will test our model on X\_test and y\_test. Review section 5 for clarity.

Initializing and training our model:

model = LogisticRegression()

model.fit(X\_train, y\_train)

We have created our model, now let’s check its performance on test data and evaluate the accuracy. We don't have to worry about mathematical calculations; we can just use *.score* method like this:

model.score(X\_test, y\_test)

This line will give us the accuracy of our model.

Output of upper code:



Accuracy of our model is approximately 77% which is quite good.

**Lecture 51: Video explanation of accuracy of diabetes model**

**Lecture 52: All codes here**

**Section 2**

import pandas as pd

data = pd.read\_excel('D://Professor ahmad class.xlsx')

#separating features and labels

features = data[['Study hours', 'Marks in previous grade', 'class performance out of 10']]

label = data[['Marks scored']]

final\_label = label \* 100

#creating model

from sklearn.neighbors import KNeighborsClassifier

mdl = KNeighborsClassifier()

mdl.fit(features,final\_label)

#making predictions

mdl.predict([[2, 0.60, 7]])

#making user friendly

study\_hours = int(input('How many hours your student study per day: '))

marks\_previous = int(input('How much he scored in his previous grade give you answer in percentage: '))

mp = marks\_previous / 100

class\_performance = int(input('What is class performance of student out of 10: '))

prediction = mdl.predict([[study\_hours, mp, class\_performance]])

print('Predicted marks: ')

print(prediction)

Predictions at once:

data\_of\_students = [[2, 0.71, 6], [4, 0.67, 8], [8, 0.91, 8.6]]

mdl.predict(data\_of\_students)

**Section 3**

import pandas as pd

from sklearn.linear\_model import LinearRegression

#loading the dataset

data = pd.read\_excel('D://Professor ahmad class.xlsx')

#specifying features and labels

features = data[['Study hours', 'Marks in previous grade', 'class performance out of 10']]

label = data[['Marks scored']]

final\_label = label \* 100

#creating model

model = LinearRegression()

model.fit(features, final\_label)

Plotting line of best fit (which was explained in video lecture)

features = data['Study hours'].values

labels = data['class performance out of 10']

features = features.reshape(-1,1)

model\_.fit(features, labels)

plt.scatter(features, labels)

plt.plot(features, model.predict(features), c = 'y')

**Section 4**

import pandas as pd

df = pd.read\_csv('D://datasets/iris.csv')

#separataing features and labels

features = df[['petal length', 'petal width', 'sepal length', 'sepal width']]

label = df['flower']

#creating model

from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier()

model.fit(features, label)

#making prediction

model.predict([[2, 3, 1, 2]])

**Section 5**

import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

df = pd.read\_csv('D://datasets/iris.csv')

#separating features and labels

features = df[['petal length', 'petal width', 'sepal length', 'sepal width']]

label = df['flower']

#splitting data and creating model

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, label)

model = KNeighborsClassifier()

model.fit(X\_train, y\_train)

#testing model

model.score(X\_test, y\_test)

**Section 6**

import pandas as pd

df = pd.read\_csv('D://datasets/diabetes.csv')

#separating features and labels

features = df.drop('Outcome', axis=1)

labels = df[['Outcome']]

#making our model

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(features, labels)

#predictions of our model

model.predict\_proba([[2, 148, 52, 35, 0, 33.6, .627, 31]])

model.predict([[2, 148, 52, 35, 0, 33.6, .627, 31]])

Splitting data to calculate accuracy

import pandas as pd

df = pd.read\_csv('D://datasets/diabetes.csv')

#separating features and labels

features = df.drop('Outcome', axis=1)

labels = df[['Outcome']]

#splitting and training our model

from sklearn.linear\_model import LogisticRegression

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, labels)

model = LogisticRegression()

model.fit(X\_train, y\_train)

#checking accuracy

model.score(X\_test, y\_test)