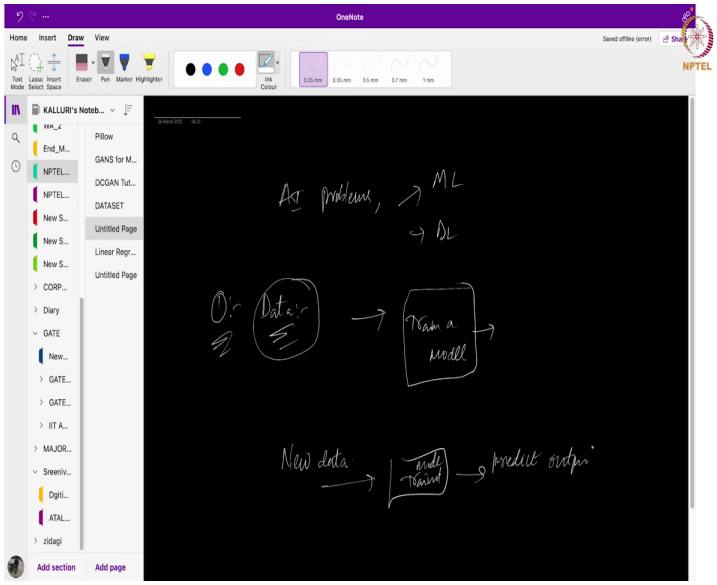
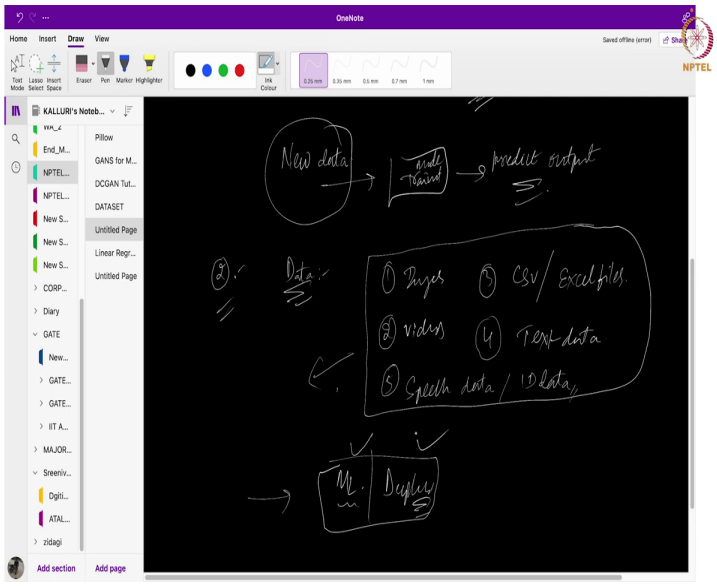
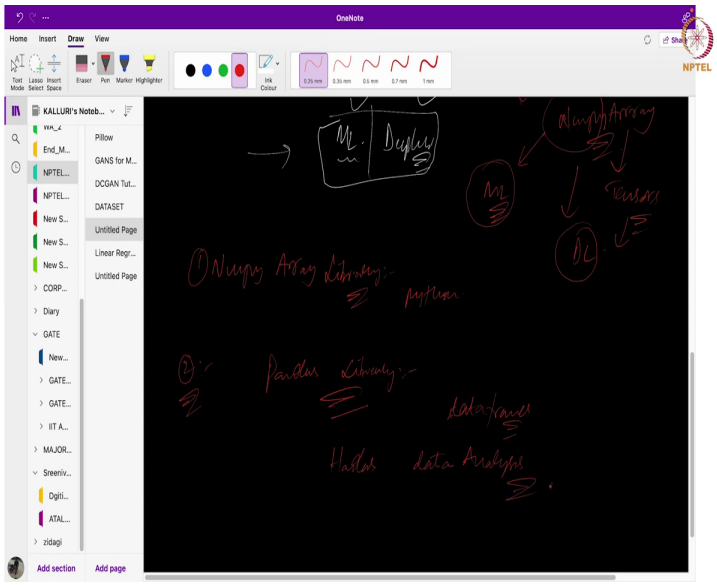
Medical Image Analysis
Professor Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology, Madras
Lecture 40

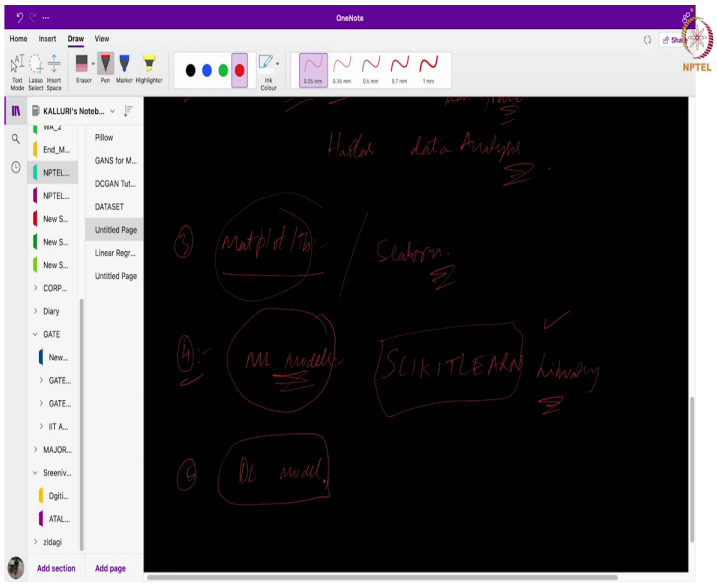
Linear Regression Demo

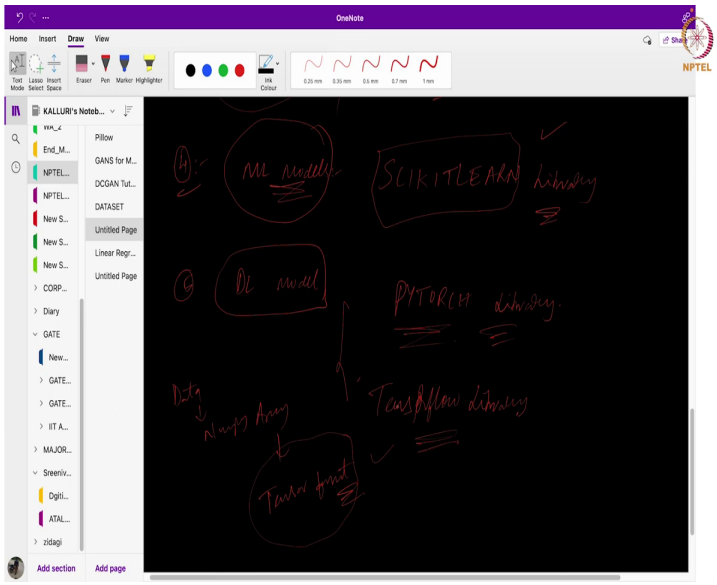
(Refer Slide Time: 00:15)

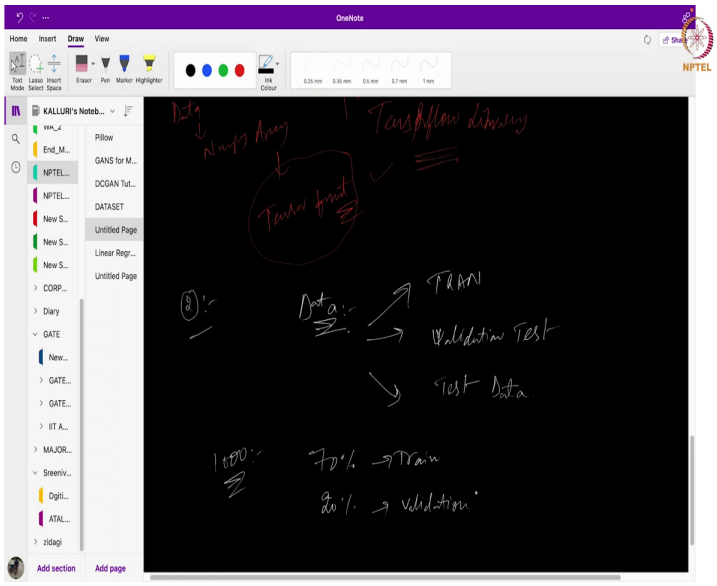


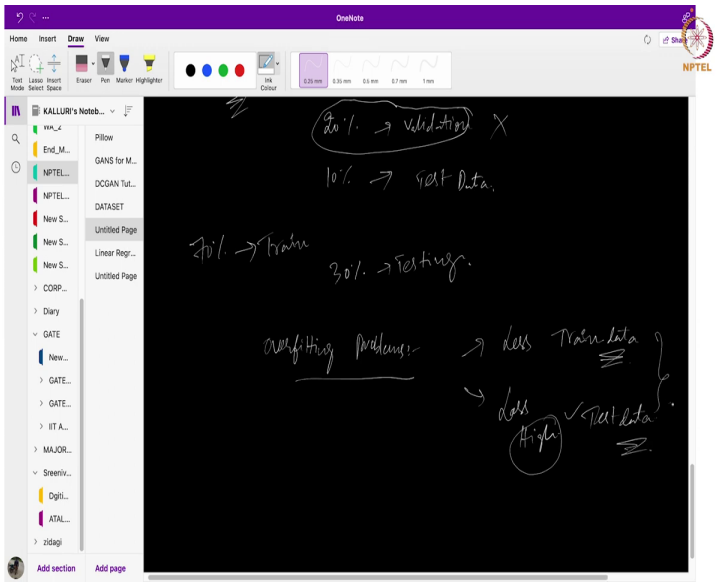


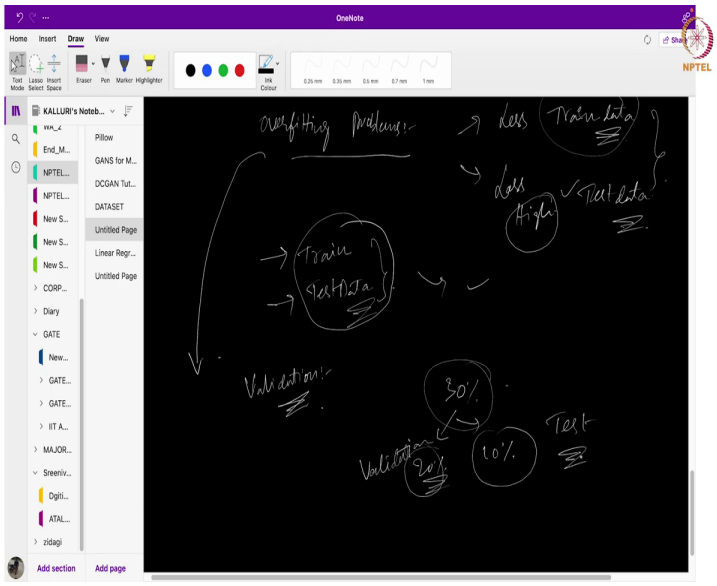

OneNote
Home
Insert
Draw
View
KALLURI's Noteb...
Pillow
GANS for M...
DCGAN Tut...
DATASET
Untitled Page
Linear Regr...
Untitled Page
Add section
Add page
NPTEL

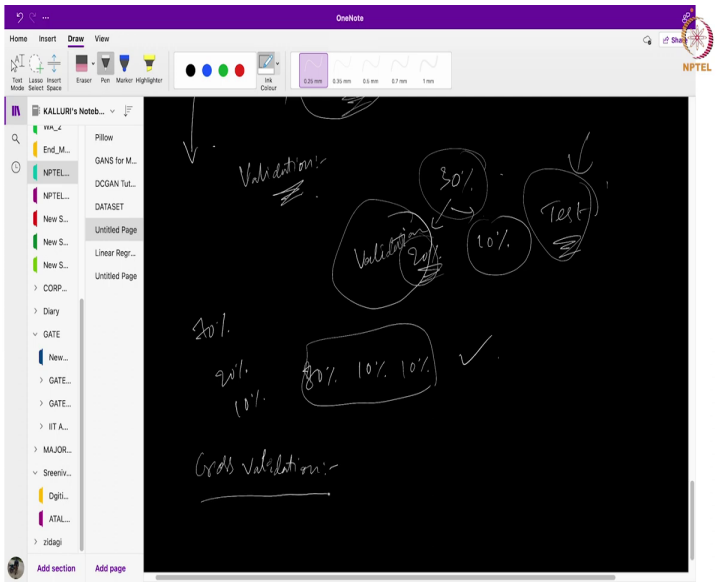

OneNote
Home
Insert
Draw
View
KALLURI's Noteb...
Pillow
GANS for M...
DCGAN Tut...
DATASET
Untitled Page
Linear Regr...
Untitled Page
Add section
Add page
NPTEL

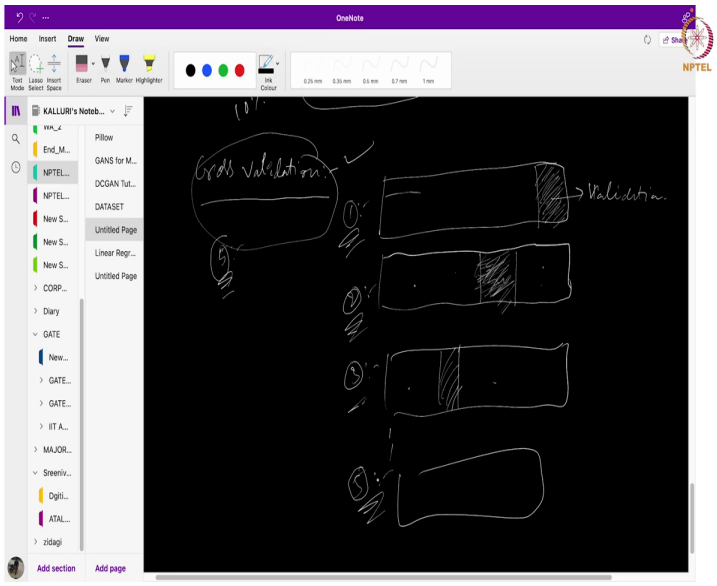

OneNote
Home
Insert
Draw
View
KALLURI's Noteb...
Pillow
GANS for M...
DCGAN Tut...
DATASET
Untitled Page
Linear Regr...
Untitled Page
Add section
Add page
NPTEL


OneNote
Home
Insert
Draw
View
KALLURI's Noteb...
Pillow
GANS for M...
DCGAN Tut...
DATASET
Untitled Page
Linear Regr...
Untitled Page
Add section
Add page
NPTEL


OneNote
Home
Insert
Draw
View
KALLURI's Noteb...
Pillow
GANS for M...
DCGAN Tut...
DATASET
Untitled Page
Linear Regr...
Untitled Page
Add section
Add page
20% → Validation
10% → Test Data.
70% → Train
30% → Testing.
Overfitting Problems:
→ less Train data
→ less High Test data.


OneNote
Home
Insert
Draw
View
KALLURI's Noteb...
Pillow
GANS for M...
DCGAN Tut...
DATASET
Untitled Page
Linear Regr...
Untitled Page
Add section
Add page
Overfitting Problems:
→ less Train data
→ less High Test data.
→ Train
→ Test Data
Validation:
30%
Validation
20%
10%
Test





Hi, everyone, I am Krishna Khalori, PA of the Ganapathy Krishnamurty Sir. In all artificial intelligence related problems, in all artificial intelligence related problems, be it either ML problem or a DL problem, like machine learning problem or deep learning problems, to solve any kind of problem, fundamental thing that is necessary is data. Once we have data, using this particular data, we are going to train a particular model, train a model, using this particular data we are going to train a model.

Once the, that model is trained, we will you use some new data to feed as input to the trained model. So, it will predict some output. The main goal is we should train this

particular model in such a way that whenever new data comes, it should predict in a desired way. This is the whole idea of this entire training model. What sort of data that is available in digital domain?

First thing is, images you can expect, second thing is videos, you can expect, third thing is like CSV or Excel files, mostly the numerical values or tabular datas will be stored in CSV or Excel files, videos, similarly, we can have text data. Let us assume that these are the four sorts of data. One more thing is more or less like speech kind of data, speech data or we can say one dimensional data. These are the several ways in which datas are available in digital domain.

So, whenever you want to feed the machine learning model or deep learning model data, data should be sent in a particular format to both machine learning and deep learning problems or deep learning models irrespective of the data, irrespective of the data, like images, video, speech data, one-dimensional text data or CSV, Excel files, all these data should be converted into NumPy Array, NumPy Array. This is the fundamental thing one should be noted while dealing with machine learning and deep learning problems.

Summary is, irrespective of the data, we should convert that kind of a data into NumPy Array format. Suppose if you take as ML problem, machine learning problem, NumPy Array can be directly fed into the machine learning models, whereas into the deep learning models, we should convert this again, convert this NumPy Arrays into tensors before feeding the data into deep learning models. This is the usual way. So, one should be familiar with NumPy Array, NumPy Array like, there is in Python.

Now, coming to second thing, in order that, to load files like CSV files or Excel files, we need pandas. Pandas library, where, in this pandas library we are going to feed the CSV files as data frames, data frames. Once you convert this CSV files or Excel files into data frames, further it will be easy for us to handle the data to do our data analysis, analysis. So, to work, just I am explaining whatever are libraries that are very much important to start off working with either machine learning problems or deep learning problems.

Fundamental thing that is necessary is NumPy Array, second thing is pandas library, third thing is Matplotlib, Matplotlib. While dealing with the data, you always try to visualize

the data, like suppose if I have Excel sheet data I just want to see a column of data as a graph. I can use that particular column of data and I can use this Matplotlib for the visualization of the particular column however that data has been distributed. To get some visual picture, we go for Matplotlib library.

One more library that is also, mostly people use, that is like Seaborn, s-e-a-b-o-r-n. So, one should be familiar with both, Matplotlib and Seaborn is an added advantage. So, if you know Matplotlib alone, that is also fine.

Next thing is machine learning models, machine learning models. To get familiarized with the machine learning models, or how to use this particular machine learning models, one should be familiar with scikit-learn library, one should be familiar with scikit-learn library. Irrespective of the data, whatever you have data, you should convert that particular data into NumPy Arrays. After converting into NumPy Array, we should use the data directly into the scikit-learn library.

Now, coming back to deep learning models, and after that to train your, to train the models of deep learning models, we just have to be familiarized with the either PyTorch library provided by Facebook or TensorFlow library, TensorFlow library. So, make sure that get familiarized with either PyTorch library or TensorFlow library. Before feeding data into this particular PyTorch or TensorFlow libraries, our PyTorch or TensorFlow models, first, we convert the data into NumPy Arrays, NumPy Arrays.

After converting into NumPy Array, we should convert that into basic tensor format. These deep learning models expect data in terms of tensor format. This is the whole idea. The summary is to start off your work in machine learning problems or with the deep learning problems you should be familiar with these kind of libraries.

First and foremost, first and foremost is for NumPy library, second thing is pandas library, third thing is Matplotlib library, fourth thing is machine learning models related, that is scikit-learn level library, next thing is deep learning models related models that is PyTorch library or TensorFlow library. TensorFlow library is provided by Google, PyTorch library provided by Facebook, that is it.

So, to get your hands on using Python to solve any kind of AI related problems, you should be aware of this stuff. Once we obtained the data, the usual procedure to solve any ML problem is to please split this data into training data. Second thing is validation data, third thing is test data. Suppose if you have some 1,000 data points, you can divide either it like 70 percent is for train, 20 percent is for validation, and third one is test data.

But why, that is the question? Suppose, I have used this, instead of using this validation data in between, what I can do is directly, I can use 70 percent is for training and the whole 30 percent is for testing. Mostly, ML problems suffer from overfitting problems, problems, that is, the matrix, let us assume that the matrix is loss value, or error value. That particular loss value is less for trained data.

After training, we are going to get very less error or loss value for the trained data, whereas if you test the, that, if you use the test data on this trained model, you will get loss value as high, loss value as high for the test data. This is nothing but overfitting problem. So, our goal is to always try to maintain the balance of both trained data and test data, loss values of both trained data and tested data should be more or less same. That is how we should design the models or train the models.

So, what happens, suppose this, on this particular test data, model is not giving less value just like for the training, just as training error. That means, again, you have to train the model. So, suppose second time also I have done and I have used the test data, same thing happened. Again, third time again, I have trained. Now, I used my test data, again happened that we are not reaching the error value or loss value more or less same to trained data.

So, while training the model itself, instead of after training and doing up, instead of doing the validation after the entire compression of training, we, we came up with validation part. We came up with validation part while the model is training. So, as the, instead of, for, from those 30 percentage, we are converting into 20 percentage into 10 percentage. This 20 indicates validation, this one indicates test.

So, when the model is training, validation loss is not improving or I mean validation loss is not decreasing. Again, the model will train in such a way that validation loss also

should be low. So, the advantage is here. While the, while training the model itself, we are just testing whether that particular model is performing well or not. So, once this entire training is done, only once we can use the test data whether this model is performing well or not. That is the basic idea behind using validation data set.

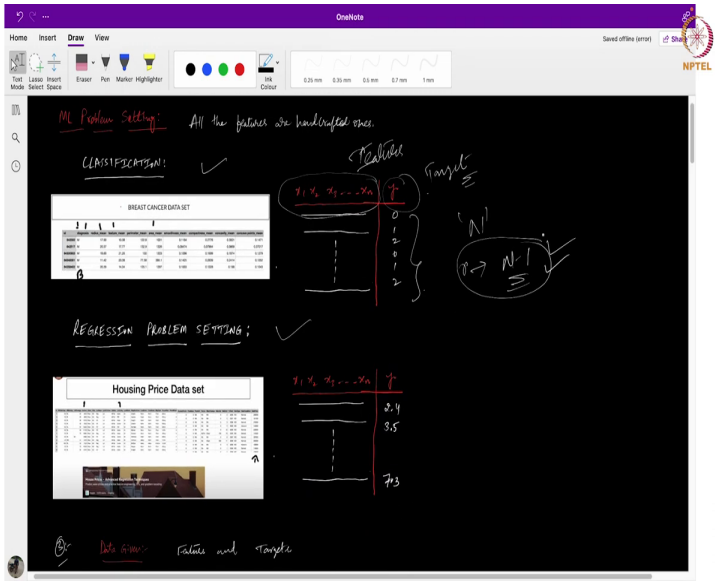
The summary is, validation data will help you to better, to train the model in a better way while that, while doing the training itself. So, the usual procedure is either you can split the entire data into a 70 percentage, 20 percentage or 10 percentage or 80, 10 percentage or 10 percentage. This is how you should use this one. Next thing is cross validation, one should be familiar with.

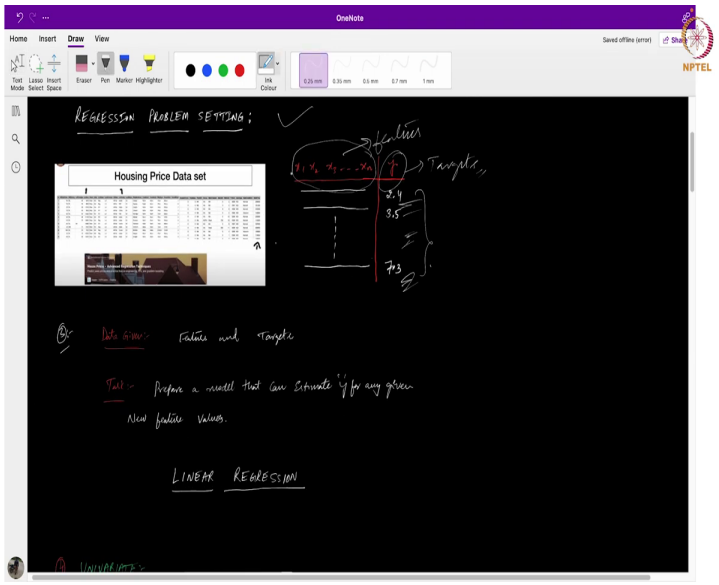
Cross validation is nothing but, since I am using validation, assume that I am training my model like this. Suppose this is my trained data, enter trained data. One iteration is done. In the second iteration, I will use this part for validation and remaining part for training. Here, I am using this part for, suppose this entire data is like, whole data which we are going to train. While training I am assuming that in the first iteration, all these points, I am taking as trained, and these points for validation.

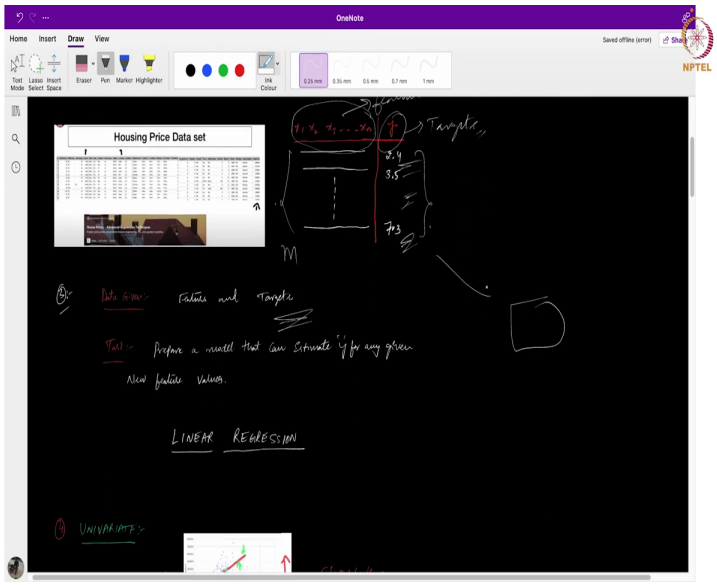
Let us assume that I am doing this cross validation like 5 times. In the second iteration, what I do, instead of doing, taking these same points as validation I am taking some other points and remaining as again, they act as training points only. Similarly, in the third iteration, we can choose these points as validation part, remaining as training points. Likewise, I will do for all the five iterations. This is nothing but cross validation concept.

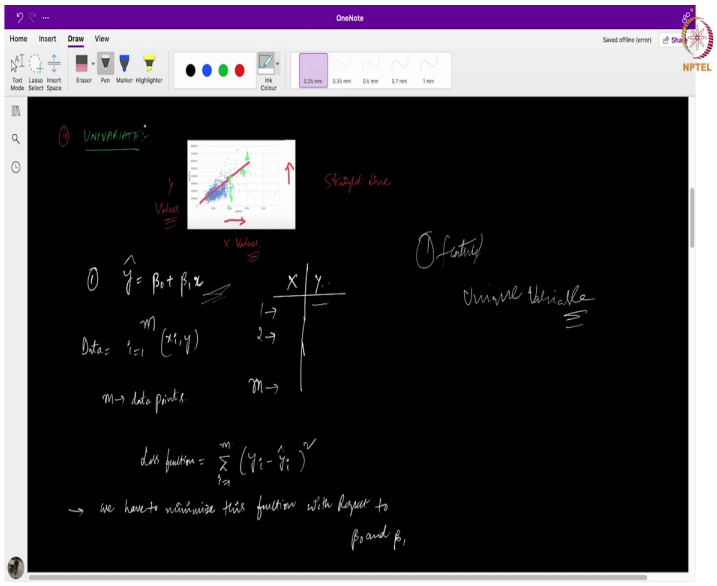
To do all these tasks, to automate all these tasks, whatever we have, something like cross validation, and all the implementing of this model, everything, we should be familiar with the libraries. That is it. There is nothing else.

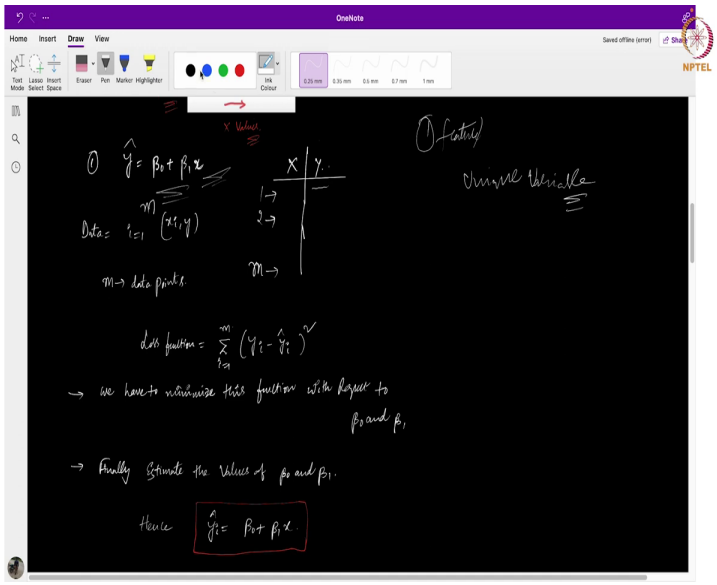
(Refer Slide Time: 15:10)

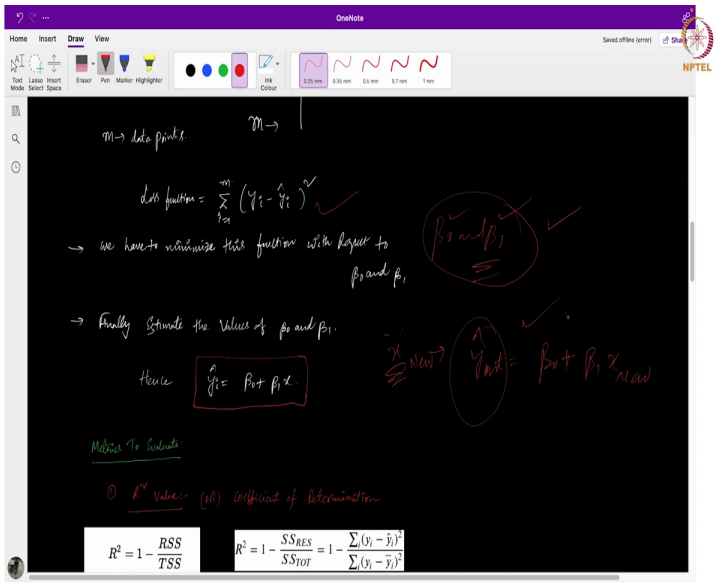


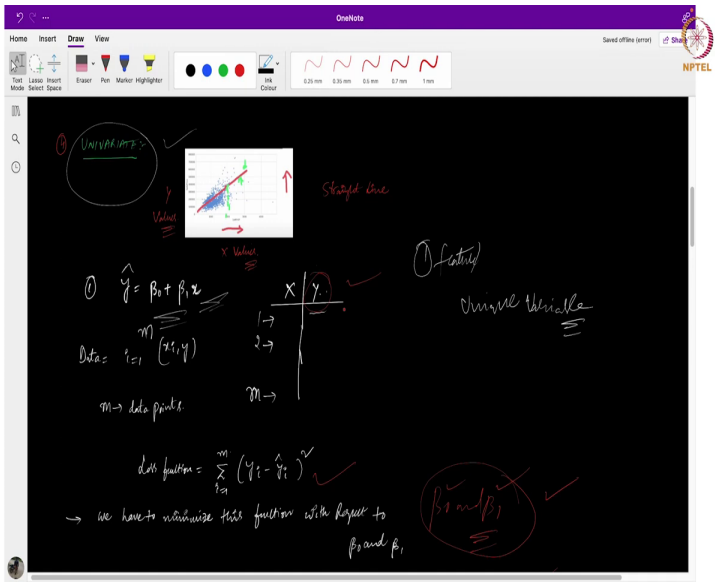


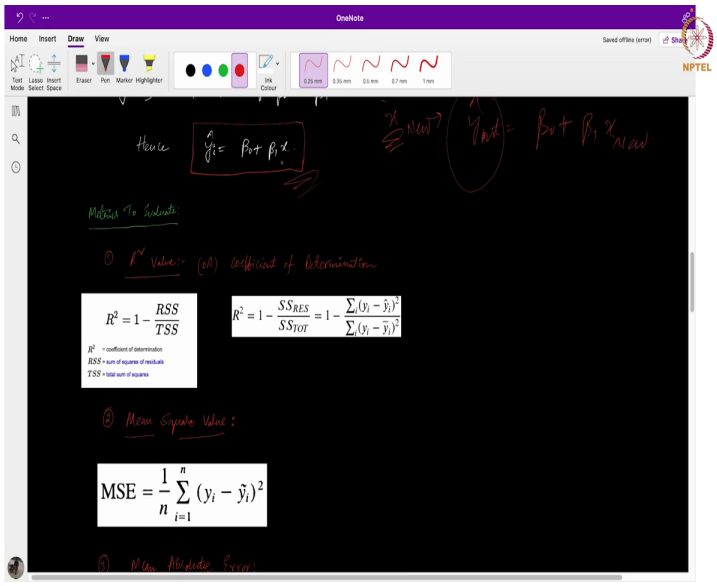

OneNote
Home
Insert
Draw
View
Housing Price Data set
Target
Data set:- Features and Target
Task:- Prepare a model that can estimate ŷ for any given New feature values.
LINEAR REGRESSION
1) UNIVARIATE:-

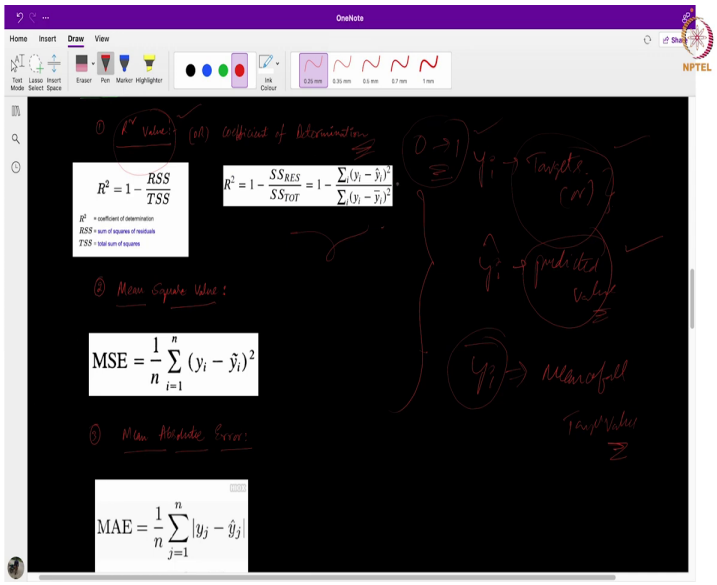

OneNote
Home
Insert
Draw
View
1) UNIVARIATE:-
Y Values
X Values
Straight line
Data =
m → data points
loss function =
→ we have to minimise this function with respect to β and β1

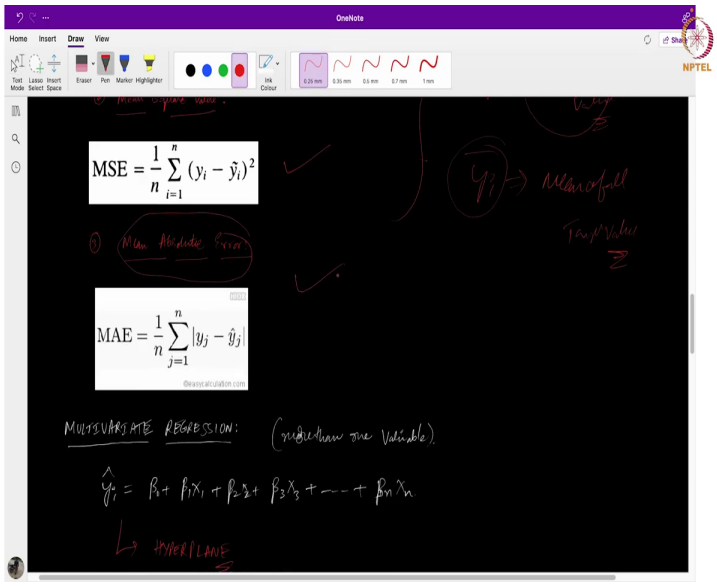

OneNote
Home
Insert
Draw
View
Saved offline (error)
Share
Text Mode
Lasso Select
Insert Space
Eraser
Pen
Marker
Highlighter
Ink Colour
0.25 mm
0.35 mm
0.5 mm
0.7 mm
1 mm
X Value
① $\hat{y} = \beta_0 + \beta_1 x$
Data = $\{(x_i, y_i)\}_{i=1}^{m}$
X | y_i
1 →
2 →
m →
m → data points.
① Input
Unique Variable
loss function = $\sum_{i=1}^{m} (y_i - \hat{y}_i)^2$
→ we have to minimise this function with Respect to β_0 and β_1.
→ Finally Estimate the Values of β_0 and β_1.
Hence $\hat{y}_i = \beta_0 + \beta_1 x$.

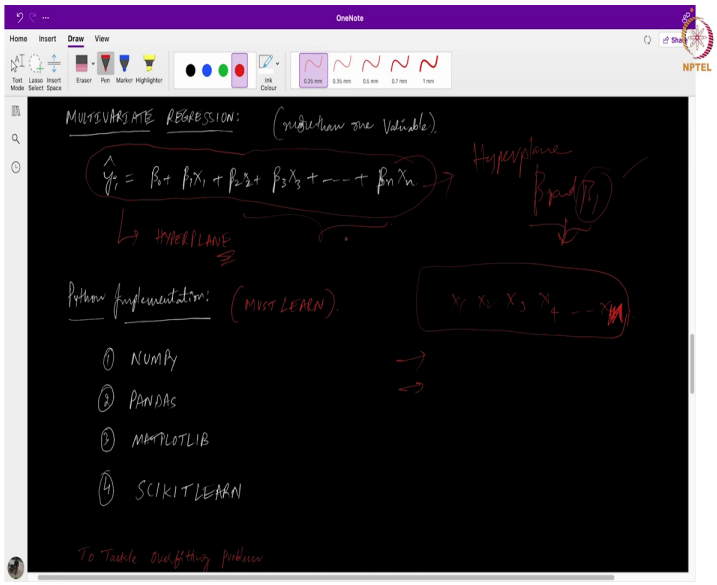

OneNote
Home
Insert
Draw
View
Saved offline (error)
Share
Text Mode
Lasso Select
Insert Space
Eraser
Pen
Marker
Highlighter
Ink Colour
0.25 mm
0.35 mm
0.5 mm
0.7 mm
1 mm
m → data points.
m →
loss function = $\sum_{i=1}^{m} (y_i - \hat{y}_i)^2$
→ we have to minimise this function with Respect to β_0 and β_1.
β_0 and β_1
→ Finally Estimate the Values of β_0 and β_1.
Hence $\hat{y}_i = \beta_0 + \beta_1 x$.
x_{new}
$\hat{y}_{out} = \beta_0 + \beta_1 x_{new}$
Methods To Evaluate:
① R^2 Value:- (OR) Coefficient of Determination
$R^2 = 1 - \frac{RSS}{TSS}$
$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$

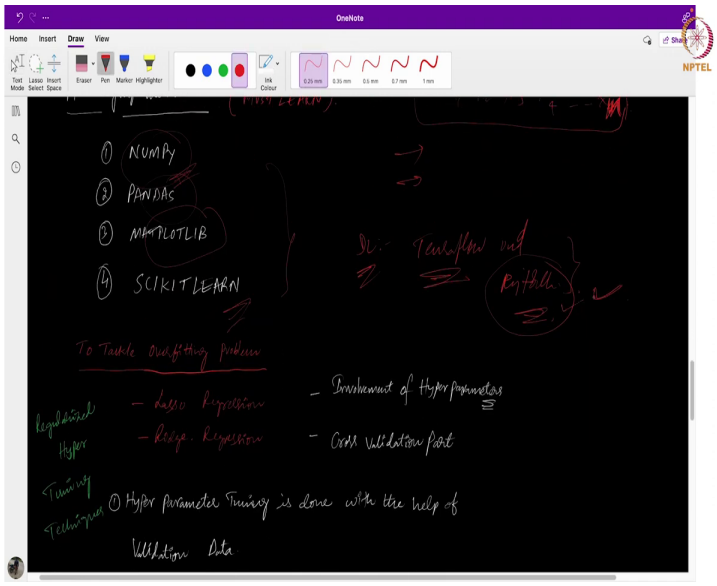

OneNote
Home
Insert
Draw
View
Univariate
Y Value
X Value
Straight line
Univariate Variable
Data =
m → data points
Cost function =
we have to minimize this function with respect to
and

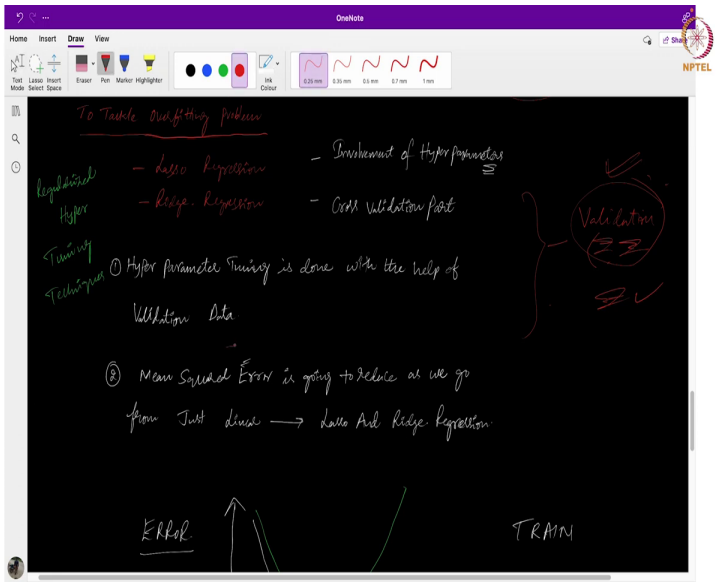

OneNote
Home
Insert
Draw
View
Hence
Metrics To Evaluate:
① R^2 Value: (OR) Coefficient of determination
$R^2 = 1 - \frac{RSS}{TSS}$
R^2 = coefficient of determination
RSS = sum of squares of residuals
TSS = total sum of squares
$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$
② Mean Square Value:
$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
③ Mean Absolute Error:

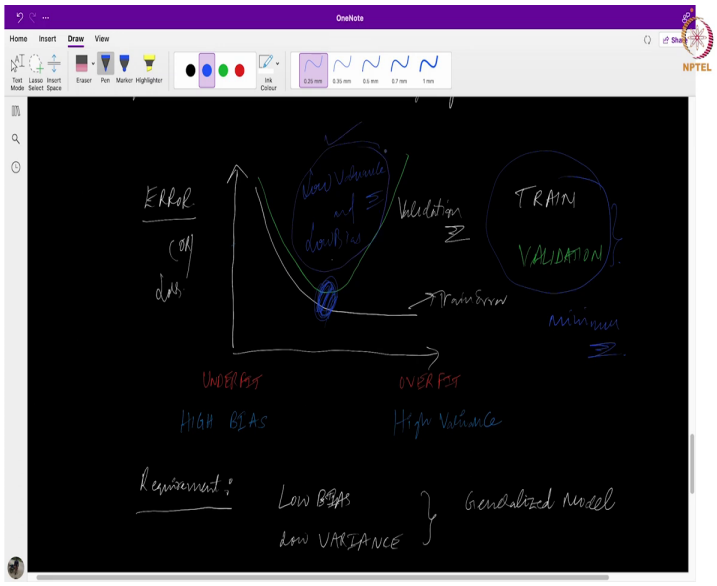

OneNote
Home
Insert
Draw
View
NPTEL
① R^2 Value: (OR) Coefficient of Determination
$R^2 = 1 - \frac{RSS}{TSS}$
R^2 = coefficient of determination
RSS = sum of squares of residuals
TSS = total sum of squares
$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$
y_i → Target
$\hat{y}_i$ → predicted value
② Mean Square Value:
$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$
③ Mean Absolute Error:
$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$

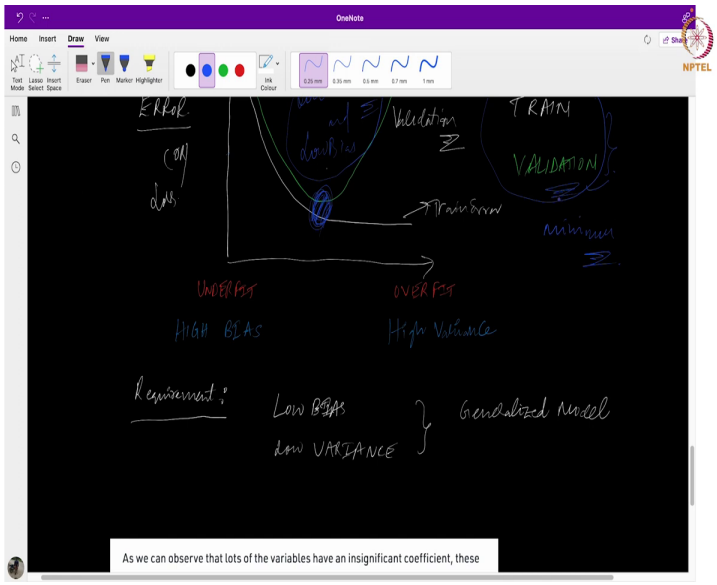

OneNote
Home
Insert
Draw
View
NPTEL
$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$
③ Mean Absolute Error:
$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$
MULTIVARIATE REGRESSION: (more than one Variable)
$\hat{y}_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n$
↳ HYPERPLANE


OneNote
Home
Insert
Draw
View
MULTIVARIATE REGRESSION:
(more than one Variable).
HYPERPLANE
Hyperplane
Python Implementation:
(MUST LEARN).
1 NUMPY
2 PANDAS
3 MATPLOTLIB
4 SCIKITLEARN
To Tackle Overfitting Problem


OneNote
Home
Insert
Draw
View
1 NUMPY
2 PANDAS
3 MATPLOTLIB
4 SCIKITLEARN
DL: Tensorflow
Pytorch
To Tackle Overfitting Problem
Regularized Hyper
- Lasso Regression
- Ridge Regression
- Involvement of hyper parameters
- Cross Validation part
Tuning Techniques
1. Hyper Parameter Tuning is done with the help of
Validation Data


OneNote
Home
Insert
Draw
View
To Tackle Overfitting Problem
Regularized Hyper Tuning Techniques
– Lasso Regression
– Ridge Regression
– Involvement of Hyperparameters
– Cross Validation Part
1) Hyper Parameter Tuning is done with the help of Validation Data
2) Mean Squared Error is going to Reduce as we go from Just Linear → Lasso And Ridge Regression
Validation
ERROR
TRAIN


OneNote
Home
Insert
Draw
View
ERROR
(OR) Loss
Low Variance and Low Bias
Validation
TRAIN
VALIDATION
Train Error
Minimum
UNDER FIT
HIGH BIAS
OVER FIT
High Variance
Requirement:
Low BIAS
Low VARIANCE
Generalized Model



Now coming to ML problem settings. Most of the times, you can see that either classification problems are regression problems mostly.

Suppose if you observe this breast cancer data set which Sir has discussed in the video also. Here, this is how the data set will be given. Like we have some x1, x2, x3, like that some features we will have, and this one is called targets. These are called features, these are nothing but targets. This is how the data has been provided.

Suppose, in terms of classification, the output values, like target variables are given as depending on the number of classes. Suppose I have some n number of classes. So, I will give the digits from 0 to (n-1), that is the usual criteria. Even you can give some distinct numbers also, that is up to you, but mostly this is how the people will give in problem setting with respect to classification.

Now, coming to the regression problems, we will given, we will be given same like, that CSV file where we have x features. These are nothing but the features and these are nothing but the targets. Regression means continuous. Here, if you observe the outputs here, you can see they are all continuous values, like 2.5, 3.5 or so many other kind of any real number.

So, in summary, in, see, most of the times we are dealing with the regression problems, we will be given data in terms of either CSV files or Excel sheets where you will be

given the, suppose we have n number of data points, for all the n number of data points, we have some m features, and correspondingly, we will have target variable also. This is how the data has been given.

Just here I have discussed like this, where features and targets are given. Our task is to, using this particular data like suppose you have some m data points, m data points you have, I have to prepare a particular model or train a model by using this kind of data set. That is my task. Here, especially the linear regression problems, where we will be given data like this, and we have to generate a function like this.

Like $\hat{y} = \beta_0 + \beta_1 x$ where suppose, assume that for all the data points, we have only two dimensionals, only, the data is having only one dimension, like, suppose I have only x, well x has only one column. My target anyway, it has only one column. Suppose, you have only one feature, one feature or in terms of, you can say unique variable or unique variable, unique variable you have.

At the point of time, we will go for univariate linear regression, where you will try to generate this function $\hat{y} = \beta_0 + \beta_1 x$ using this particular loss function, we are going to estimate the β_0 and β_1. This will be taken care by the library, what I have mentioned as a scikit-learn library, it will take care of to estimating this stuff.

After fitting the model, we, in theoretical part, we say that we will estimate β_0 and β_1. Once we know the β_0 and β_1 values, for any new kind of data point comes, suppose x value, new value comes, like X_{NEW}, now, I will substitute here. $\hat{Y}_{out} = \beta_0 + \beta_1 X_{NEW}$ I will give, β_0, I already know from the model, β_1, I know already from the model, and I will substitute this new value and I will predict this particular Y_{out}. This is how the criteria goes.

In summary, suppose you have only one variable in terms of independent variables or in terms of features, we can say, where y is nothing but the target, at the time, we call it as univariate linear regression. There, we are assuming just this particular function, $\beta_0 + \beta_1 x$. It will take care of. So, we are saying that the training is going on. But while the training is going on, we should be in a position to evaluate, while the training itself is going on.

Like, we need some metric, while the training is going on, we have to see some metric. So, if you are able to find that, that metric is good enough, that means, the training is going well. Like that, in linear regression problems, we have three matrix. Most of the times people will use these three matrix only. Like R^2 value, it is given by here, this kind of a formula, where y_i is nothing but the original target, targets or the data that is given.

Like do not put data when it comes to $\hat{y}_i$. This is nothing but the predicted value. This is nothing but the original target value, this is nothing but the predicted value. Next, $\bar{y}_i$ is nothing but mean of all the target values, mean of all target values. This is one sort of a matrix. We expect this value to be 0 to 1, 0 to 1 in general, and I need a value of 1.

Suppose, after fitting the model, if you are getting 1 value, that means we have developed a very good number, that is very good. Next thing is mean square value. This is the matrix, similarly, you can always, you can also say that it is mean absolute error. These are the three metrics, we can see, in designing a linear regression problem.

Till now, I have discussed only matrix related to linear regression, at the same time I have discussed only with respect to univariate linear regression only. That means independent variables are features, we have only one, targets we have. Now, coming to multivariate regression. It is just another extension of this univariate regression.

Instead of only one β_0 and β_1, we are extending these coefficients. Now, suppose in case I have x1, x2, x3, x4, likewise I have xn, like n features I have, for every data point I have n features, n data points, n features, we just have to build this particular line. Usually, since it is not a two dimensional plane, that is why we call this one as a hyper plane, not a line, hyper plane we call.

Here we, instead of estimating just β_0 and β_1, we are extending our task and we are estimating remaining things also. I have already told you that is the basic difference between multivariate linear regression and univariate regression. That is, univariate regression, you will have only one variable as features, whereas in multivariate regression, you will have several variables or several features.

Now coming to Python implementation, here, I am going to use the scikit-learn library only. So, most of the times, whatever the task you do with respect to machine learning, you must be focusing about NumPy, pandas, matplotlib and scikit-learn. I request everyone to please be familiar with these four libraries. Suppose if you are dealing with deep learning you should be familiar with also TensorFlow and PyTorch.

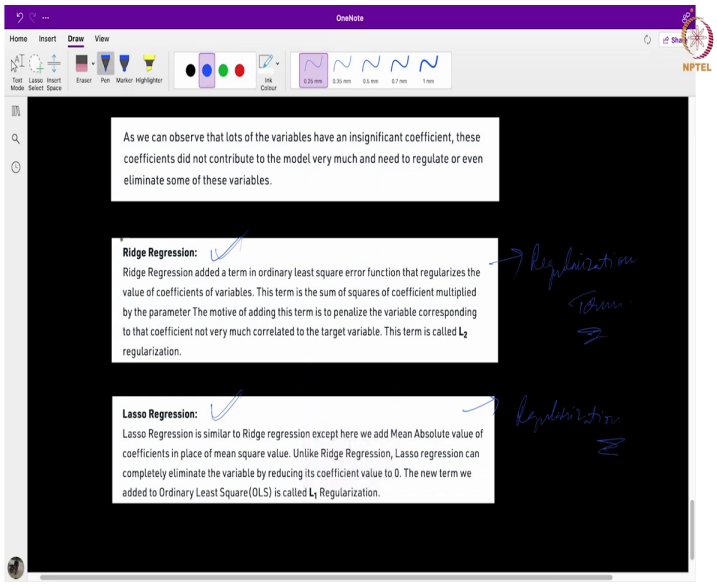
People who are in academic domain or research domain, most of the people I request you guys to please go with the TensorFlow. Whenever a new research paper comes, people are releasing their code mostly in terms of PyTorch. So, you will have an added advantage. So, I request those who are in academic and research domain, please go with the PyTorch. Even TensorFlow is fine, but I request you to go with the PyTorch because it is helpful for you.

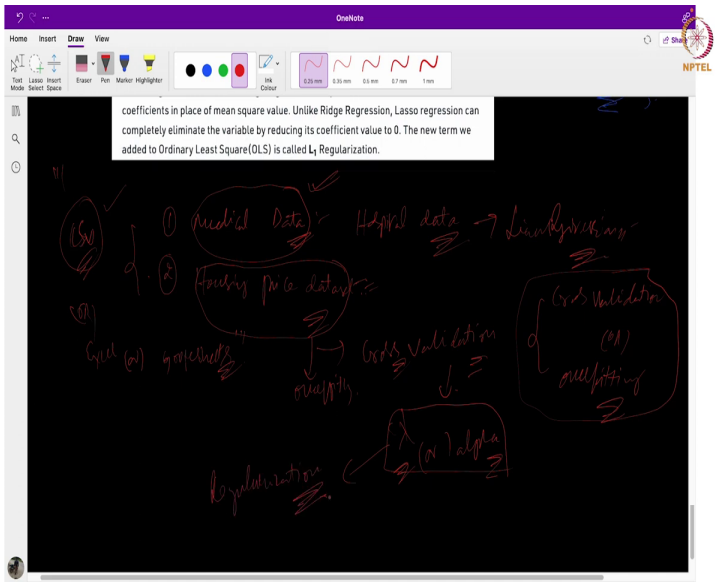
I have already discussed this overfitting problems. To deal the overfitting problem, I have already told you, we use the validation data set, validation data set while the training is going on, where we will tune the, all the hyper parameters according to this validation data set. It is the key, or we can say this is the center. Hyper parameter tuning is done with the validation data. Like here, I have already explained this graph.

Suppose the white color indicates, white indicates that training error or loss, error or loss, whereas the green color indicates this validation loss. We have to train our model and save our particular model exactly at this point. This is our region of interest, or we can say where we have training error and testing error, the, the difference or the gap between training error and validation error should be minimum. The difference between, or the gap between them is minimum. This is what we need.

We can even say this point is nothing but low bias and low variance. This is the, our interest, point of interest. This is nothing but the entire theoretical part, I just have discussed. In general, to deal these overfitting problems as I told you, validation data set we will use to tune the hyper parameters, but how, how, which kind of hyper parameters, we will discuss.

(Refer Slide Time: 25:00)





Usually, to deal this overfitting problem, we are going to have two other linear regressions, like lasso regression and regular regression, sorry, ridge regression, where both the terms will involve regularization term, regularization term. Even this one also indicates regularization term. This is not that difficult. Just for the sake of understanding, assume that using these ridge regression, lasso regression, we can deal the linear regression problems without getting affected by overfitting, that is the summary point.

Now, coming to what this ridge regression and lasso regression. You can go through them, it is not that difficult, theoretical part. By the implementation, I will explain how

exactly to use this stuff. This particular, apart from the loss function which we have designed for simple linear regression, we will have some λ term or we can even say some α term. According to your comfort, you can use any term.

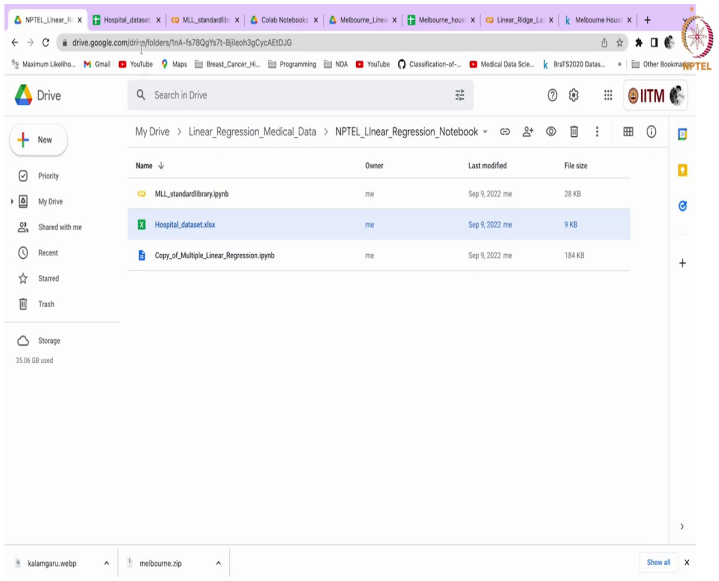
We can have weight square or otherwise you can say just weight itself. So, theoretically this much is not that much necessary while implementing the things. For the sake of understanding I am just intimating you that you guys just have to learn what is meant by ridge regression and how to use this one, what is this lasso regression and how to use this one.

In the implementation part, I am going to show you two examples, one with respect to the medical data, one is related to medical data, data that is available in medical domain, that means some, with respect to hospital data, hospital data. And the second thing is something like housing price data set, housing price data set. Both are given in CSV format only, comma separated values, we call CSV format. Or we can even go for Excel format also or Google Sheets, Google Sheets. These are the usual things you can found.

Two problems, I will explain. In the medical domain, medical data related problem, I will just discuss only the simple linear regression or multivariate linear regression, where it does not involve any kind of cross validation or any sort of overfitting. I do not discuss all this stuff. I will just let you know how to apply this entire training data, whatever that has been given to you. After fitting that particular, after fitting the particular model, I will just give the test data to that particular model and I will generate the output. That is first example.

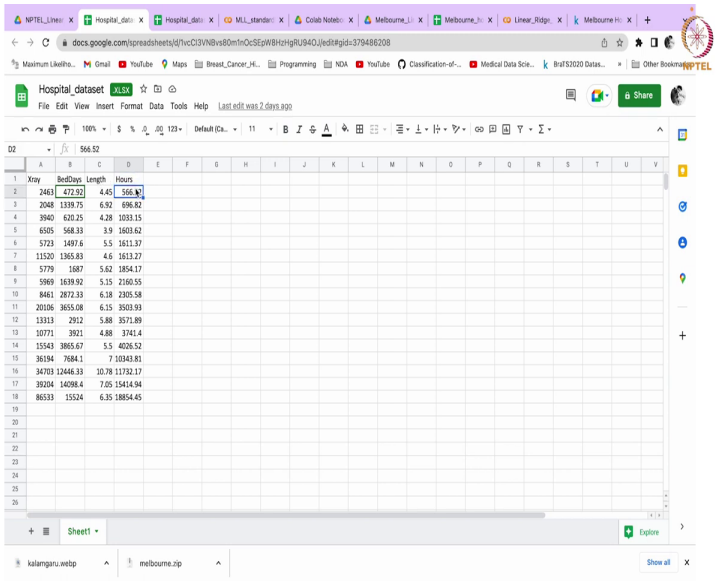
I am not going to discuss anything related to overfitting and all that issues in this particular data set. Now, coming to house pricing data set, I will discuss that stuff where it involves even overfitting problem. And it will even discuss the cross validation stuff, using cross validation, we are going to hyper, tune the hyper parameter as sigma, sorry, λ, you can even call α, the term that is belonging to regularization.

(Refer Slide Time: 28:37)



Now, coming to the implementation part. I have just shared this entire documents or the folder, entire folder I have given to you. Just upload this particular folder onto the Google Drive. After uploading the particular folder to the Google Drive, you will get to see these three things, where you can see that this is nothing but the hospital data set, where you can see.

(Refer Slide Time: 29:10)

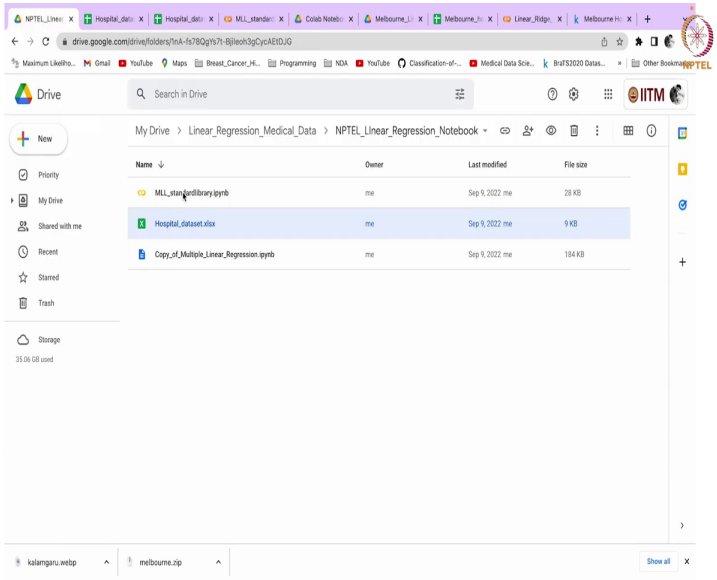


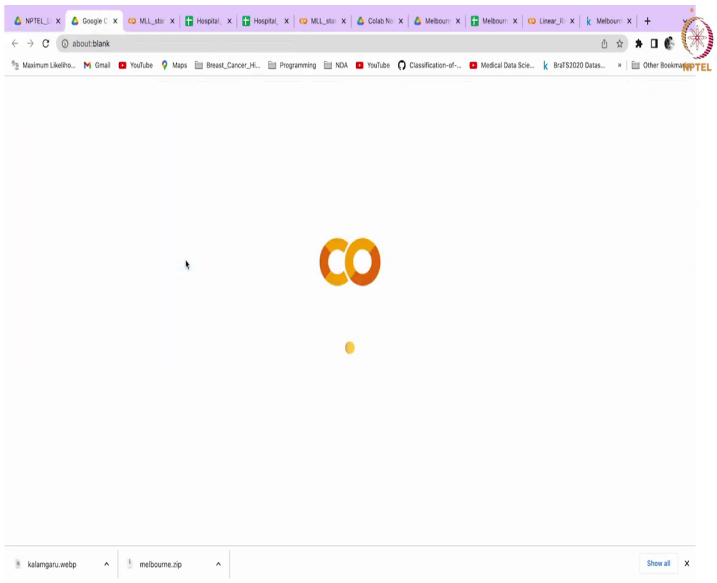
Once you click on that one, you will get to see this kind of a data, where it indicates number of access in a particular month, at the same time like bed days means how many beds are occupied in a single month.

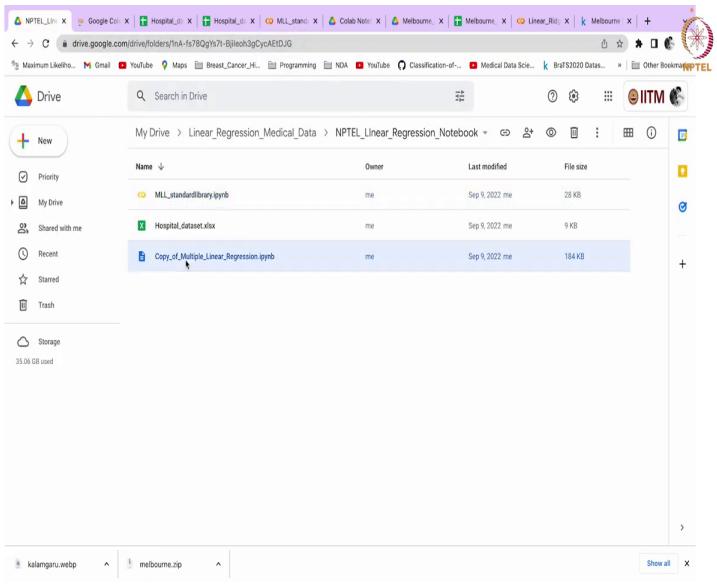
Similarly, length of the, length indicates how many hours a particular patient has been stayed, has stayed in that particular hospital. So, if the number of beds are taken are being, taken more that means hospital is functioning those many number of hours. I mean increase in this number will indicate more the number of operating hours. Similarly, as the number of beds are occupied in a particular single month is very high, that means hours also will be more.

Similarly, as the stay of a particular patient is more, that means his, operating hours of the hospital also should be more. So, all these three variables can be treated as an independent variable, and hours here, which will be called as a target variable.

(Refer Slide Time: 30:06)

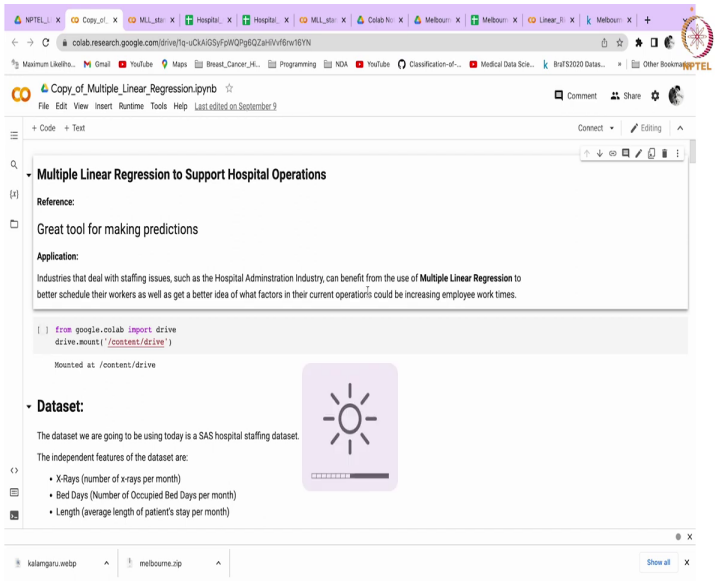


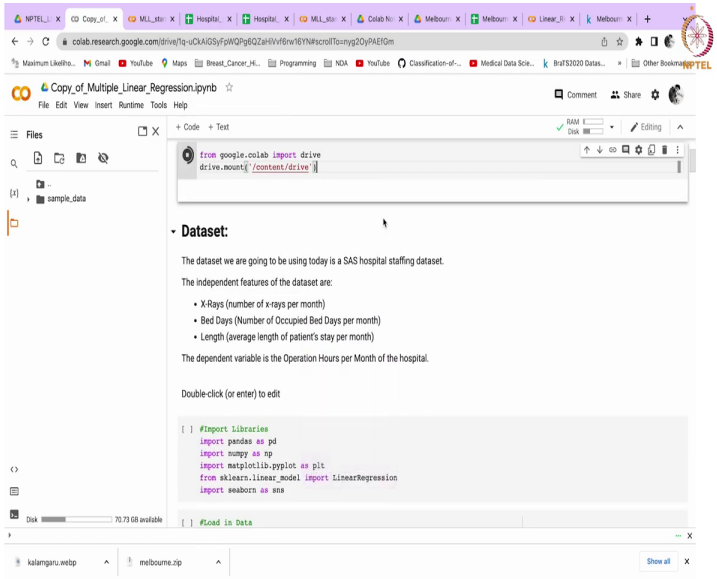


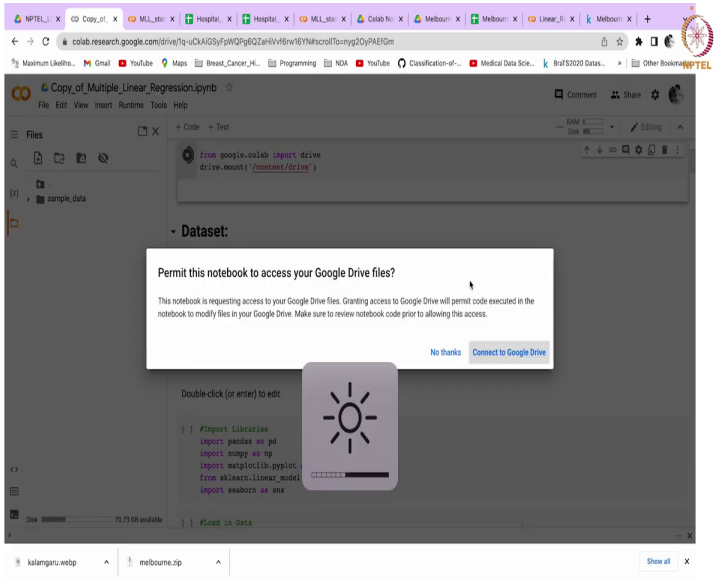


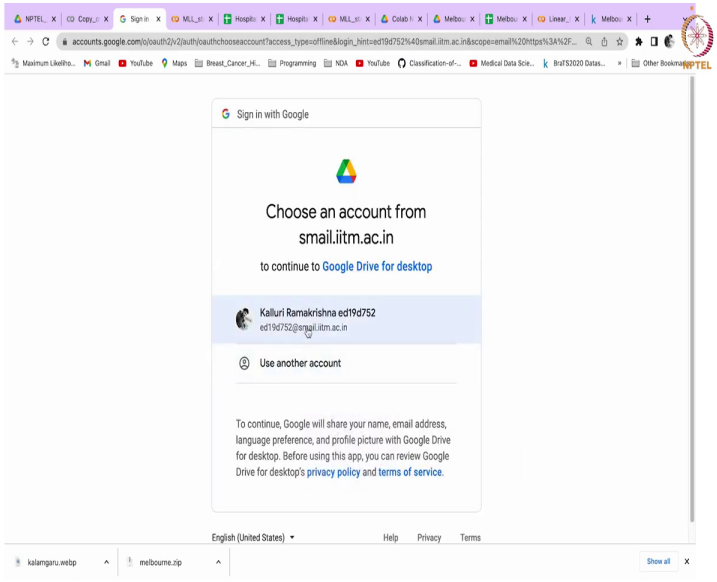
Now, coming to implementation part. This is a collab notebook, sorry, jupiter notebook, that I am opening in Google Collab.

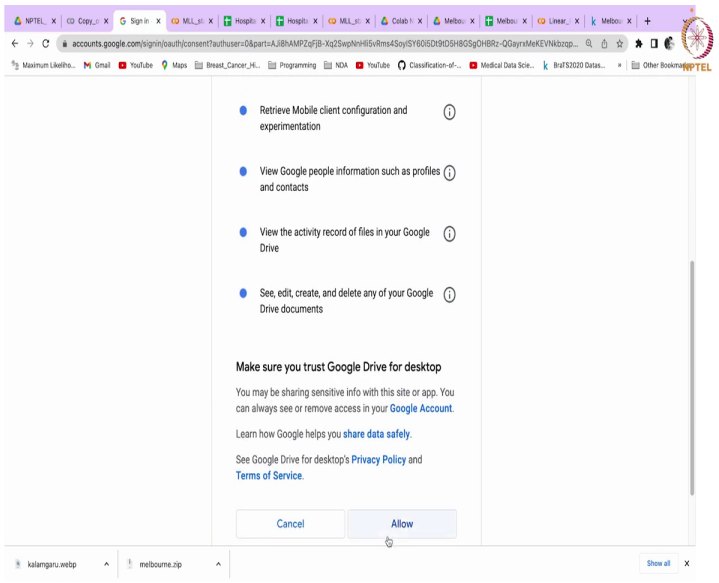
(Refer Slide Time: 30:21)





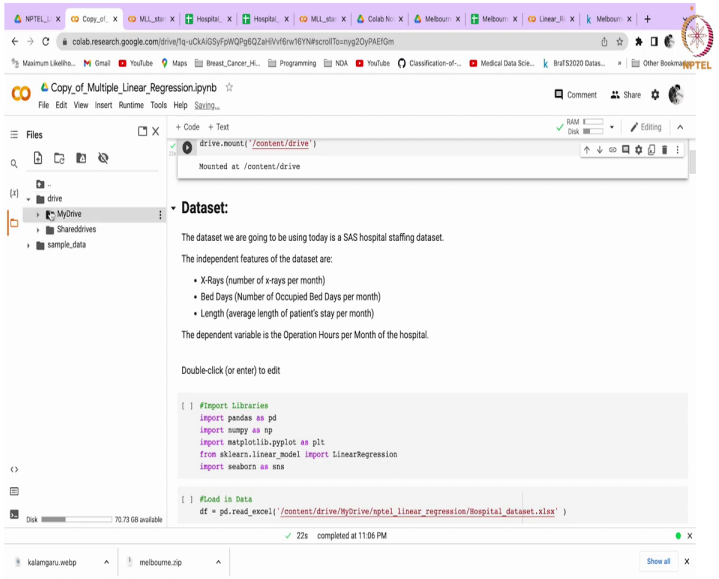

Copy_of_Multiple_Linear_Regression.ipynb
Files
sample_data
from google.colab import drive
drive.mount('/content/drive')
Dataset:
Permit this notebook to access your Google Drive files?
This notebook is requesting access to your Google Drive files. Granting access to Google Drive will permit code executed in the notebook to modify files in your Google Drive. Make sure to review notebook code prior to allowing this access.
No thanks
Connect to Google Drive
Double-click (or enter) to edit
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot
from sklearn.linear_model
import seaborn as sns
#Load in Data
kalamgaru.webp
melbourne.zip
Show all

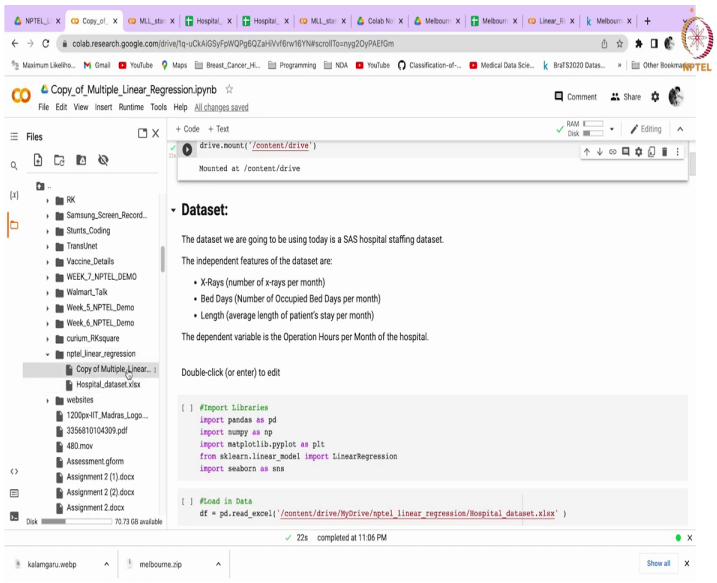

Sign in with Google
Choose an account from
smail.iitm.ac.in
to continue to Google Drive for desktop
Kalluri Ramakrishna ed19d752
ed19d752@smail.iitm.ac.in
Use another account
To continue, Google will share your name, email address, language preference, and profile picture with Google Drive for desktop. Before using this app, you can review Google Drive for desktop's privacy policy and terms of service.
English (United States)
Help
Privacy
Terms
kalamgaru.webp
melbourne.zip
Show all

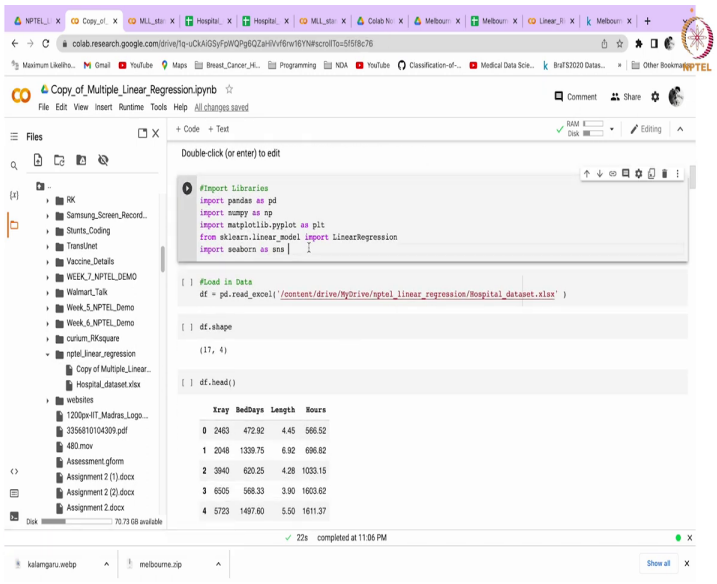


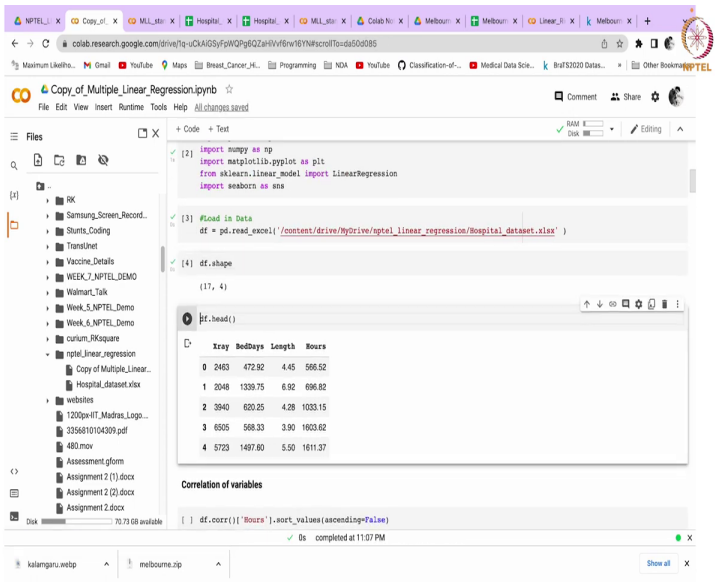
If you observe clearly, here we are just noting down the hospital number of hours, just open this Google collab notebook after doing that one, just connect, shift enter so that particular, connect to the Google drive, it will ask you. Suppose it is mine. We are mounted at this particular drive, just refresh this one.

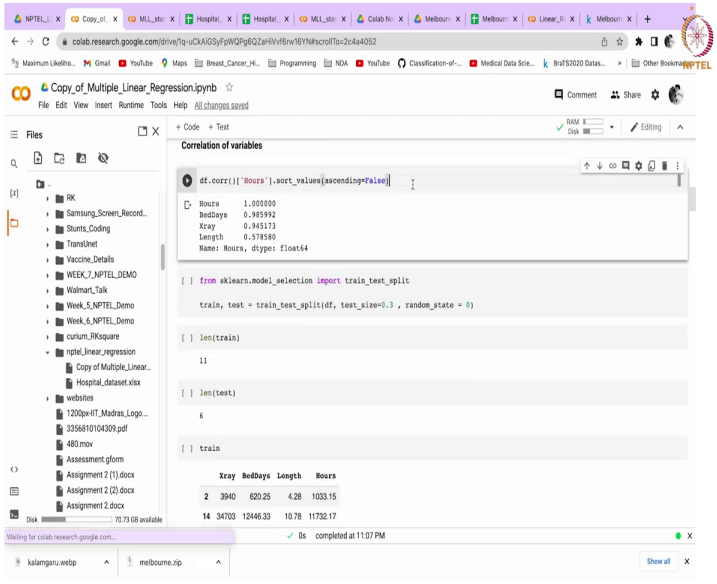
(Refer Slide Time: 31:21)

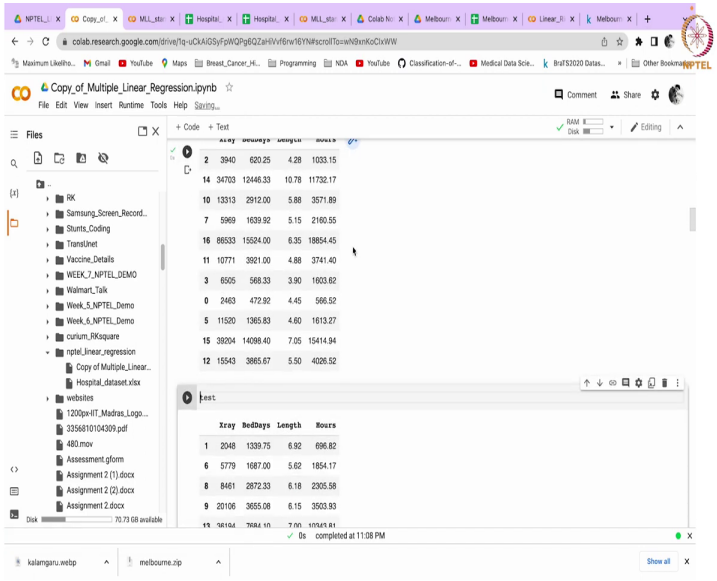


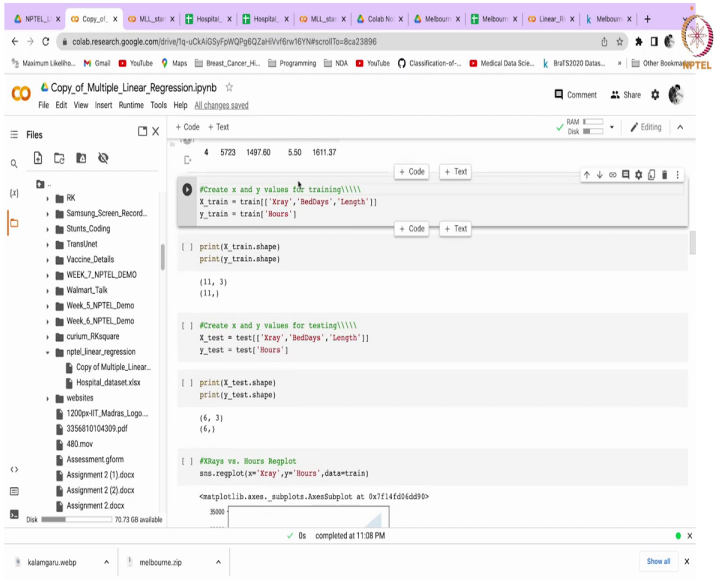

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
drive.mount('/content/drive')
Mounted at /content/drive
Dataset:
The dataset we are going to be using today is a SAS hospital staffing dataset.
The independent features of the dataset are:
X-Rays (number of x-rays per month)
Bed Days (Number of Occupied Bed Days per month)
Length (average length of patient's stay per month)
The dependent variable is the Operation Hours per Month of the hospital.
Double-click (or enter) to edit
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
#Load in Data
df = pd.read_excel('/content/drive/MyDrive/nptel_linear_regression/Hospital_dataset.xlsx')
22s completed at 11:06 PM
kalamgaru.webp
melbourne.zip
Show all

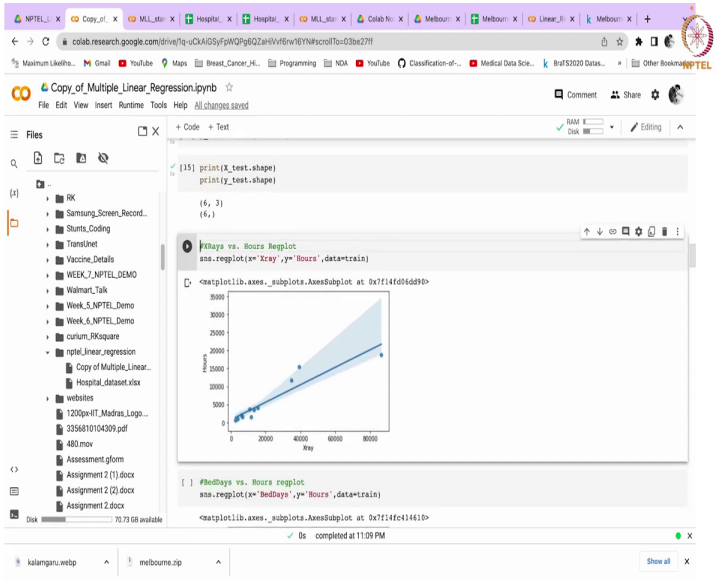

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
Double-click (or enter) to edit
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
#Load in Data
df = pd.read_excel('/content/drive/MyDrive/nptel_linear_regression/Hospital_dataset.xlsx')
df.shape
(17, 4)
df.head()
Xray BedDays Length Hours
0 2463 472.92 4.45 566.52
1 2048 1339.75 6.92 696.82
2 3940 620.25 4.28 1033.15
3 6505 568.33 3.90 1603.62
4 5723 1497.60 5.50 1611.37
22s completed at 11:06 PM
kalamgaru.webp
melbourne.zip
Show all

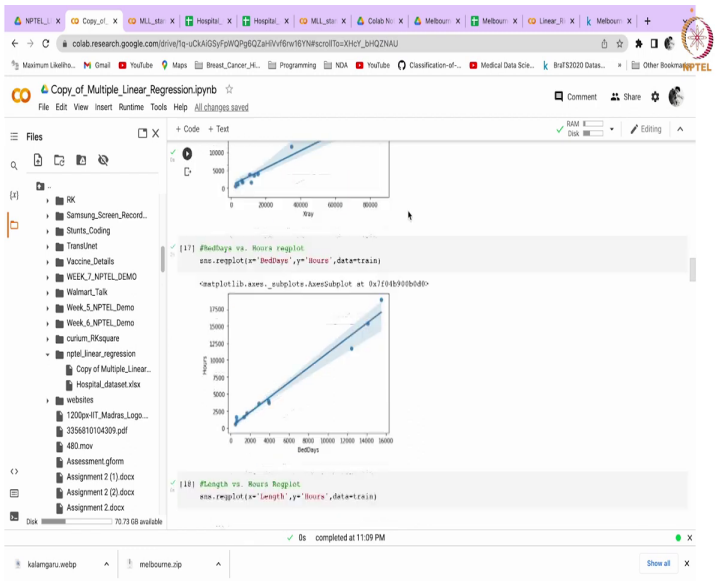

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
#Load in Data
df = pd.read_excel('/content/drive/MyDrive/nptel_linear_regression/Hospital_dataset.xlsx')
df.shape
(17, 4)
df.head()
Xray BedDays Length Hours
0 2463 472.92 4.45 566.52
1 2048 1339.75 6.92 696.82
2 3940 620.25 4.28 1033.15
3 6505 568.33 3.90 1603.62
4 5723 1497.60 5.50 1611.37
Correlation of variables
df.corr()['Hours'].sort_values(ascending=False)
0s completed at 11:07 PM
kalamgaru.webp
melbourne.zip
Show all

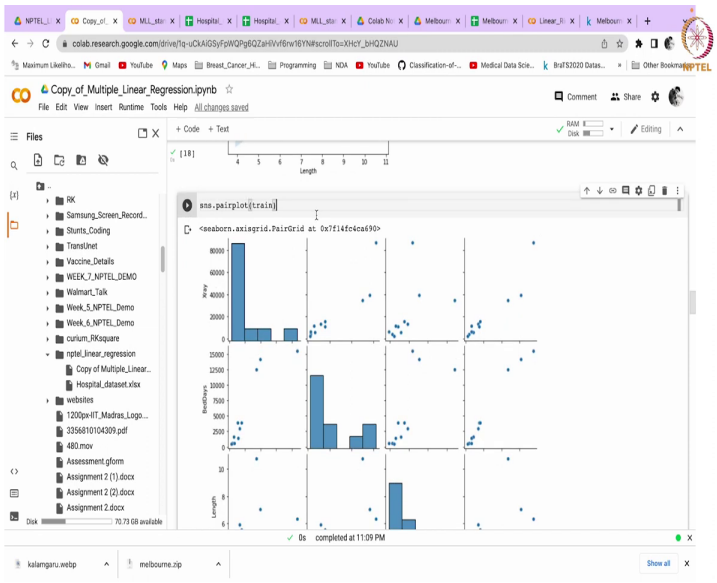

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
Correlation of variables
df.corr()['Hours'].sort_values(ascending=False)
Hours 1.000000
BedDays 0.985992
Xray 0.945173
Length 0.578580
Name: Hours, dtype: float64
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.3 , random_state = 0)
len(train)
11
len(test)
6
train
Xray BedDays Length Hours
2 3940 620.25 4.28 1033.15
14 34703 12446.33 10.78 11732.17
Waiting for colab.research.google.com...
0s completed at 11:07 PM
kalamgaru.webp
melbourne.zip
Show all

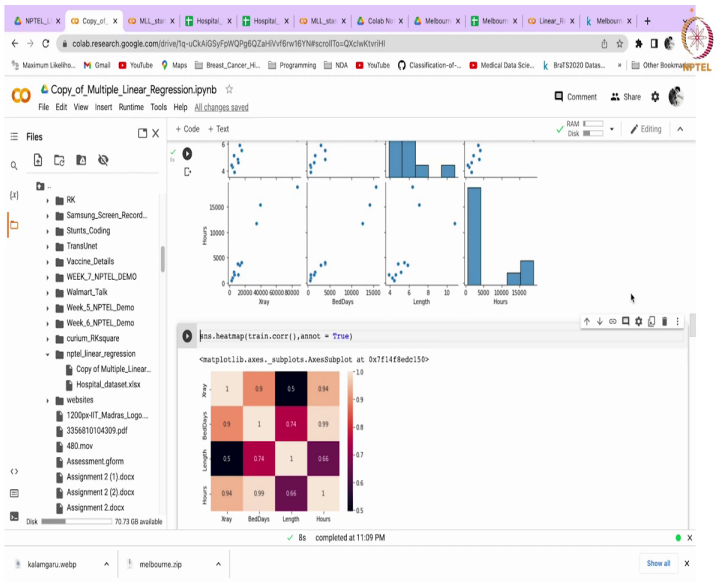

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
Editing
Files
+ Code
+ Text
2 3940 620.25 4.28 1033.15
14 34703 12446.33 10.78 11732.17
10 13313 2912.00 5.88 3571.89
7 5969 1639.92 5.15 2160.55
16 86533 15524.00 6.35 18854.45
11 10771 3921.00 4.88 3741.40
3 6505 568.33 3.90 1603.62
0 2463 472.92 4.45 566.52
5 11520 1365.83 4.60 1613.27
15 39204 14098.40 7.05 15414.94
12 15543 3865.67 5.50 4026.52
test
Xray BedDays Length Hours
1 2048 1339.75 6.92 696.82
6 5779 1687.00 5.62 1854.17
8 8461 2872.33 6.18 2305.58
9 20106 3655.08 6.15 3503.93
0s completed at 11:08 PM
kalamgaru.webp
melbourne.zip
Show all

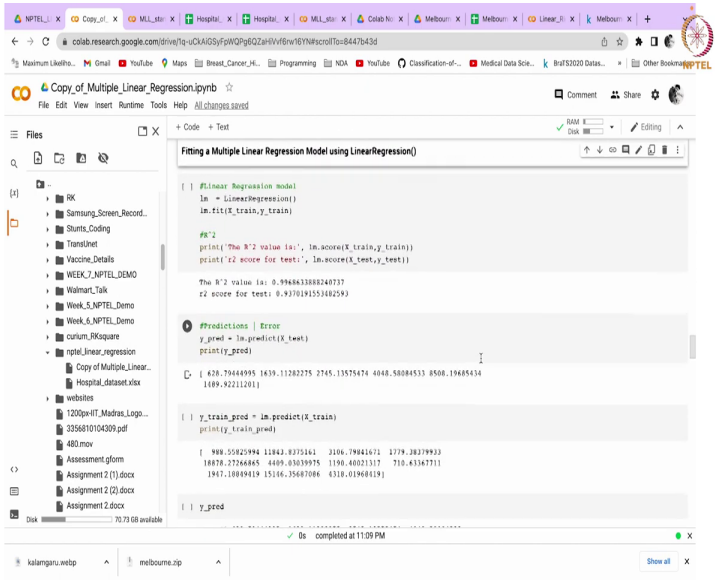

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Editing
Files
4 5723 1497.60 5.50 1611.37
#Create x and y values for training\\\\
X_train = train[['Xray','BedDays','Length']]
y_train = train['Hours']
print(X_train.shape)
print(y_train.shape)
(11, 3)
(11,)
#Create x and y values for testing\\\\
X_test = test[['Xray','BedDays','Length']]
y_test = test['Hours']
print(X_test.shape)
print(y_test.shape)
(6, 3)
(6,)
#XRays vs. Hours Regplot
sns.regplot(x='Xray',y='Hours',data=train)
<matplotlib.axes._subplots.AxesSubplot at 0x7f14fd06dd90>
0s completed at 11:08 PM
kalamgaru.webp
melbourne.zip
Show all

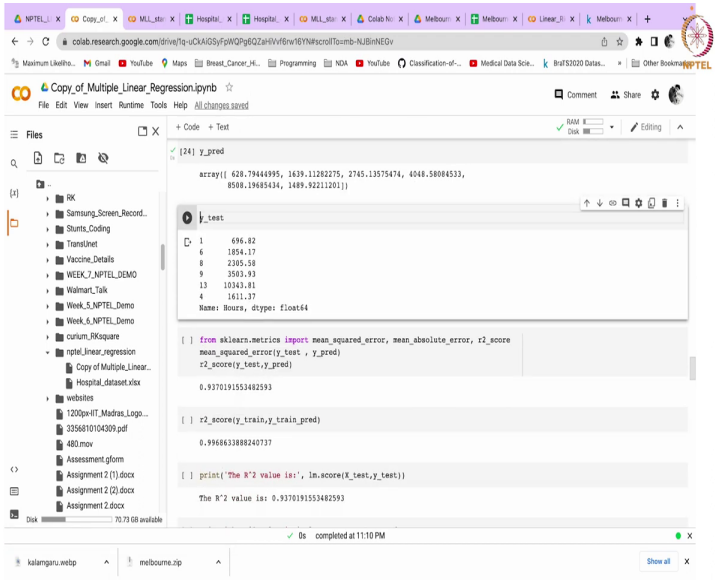

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
[15] print(X_test.shape)
print(y_test.shape)
(6, 3)
(6,)
#XRays vs. Hours Regplot
sns.regplot(x='Xray',y='Hours',data=train)
<matplotlib.axes._subplots.AxesSubplot at 0x7f14fd06dd90>
Hours
Xray
[ ] #BedDays vs. Hours regplot
sns.regplot(x='BedDays',y='Hours',data=train)
<matplotlib.axes._subplots.AxesSubplot at 0x7f14fc414610>
0s completed at 11:09 PM
kalamgaru.webp
melbourne.zip
Show all

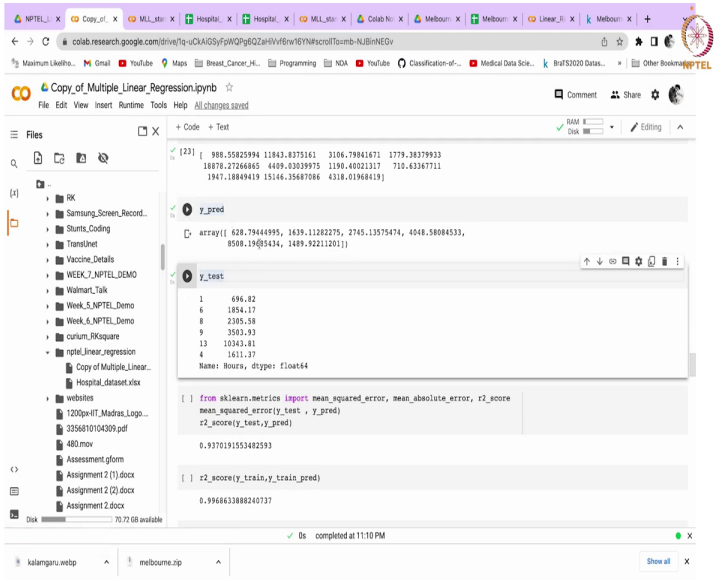

colab.research.google.com/drive/1q-uCkAlGSyFpWQPg6QZaHiVvf6rw16YN#scrollTo=XHcY_bHQZNAU
Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
Xray
[17] #BedDays vs. Hours regplot
sns.regplot(x='BedDays',y='Hours',data=train)
<matplotlib.axes._subplots.AxesSubplot at 0x7f04b90b0d0>
Hours
BedDays
[18] #Length vs. Hours Regplot
sns.regplot(x='Length',y='Hours',data=train)
0s completed at 11:09 PM
kalamgaru.webp
melbourne.zip
Show all

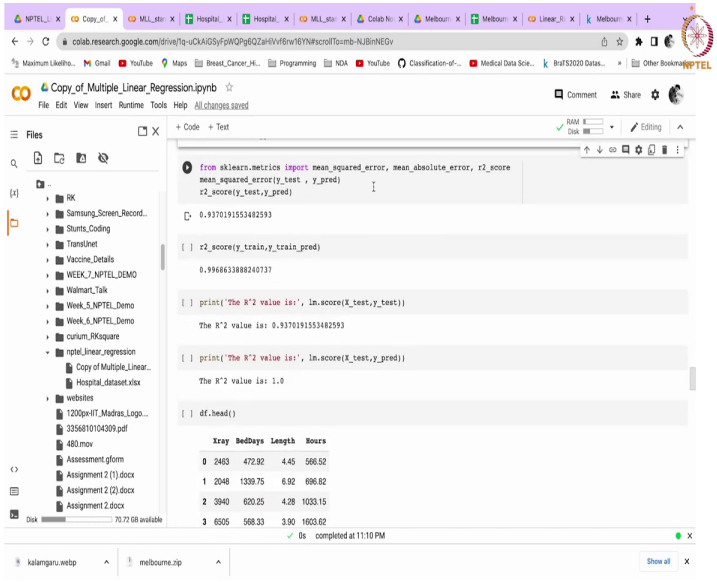

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Editing
Files
sns.pairplot(train)
<seaborn.axisgrid.PairGrid at 0x7f14fc4ca690>
Xray
BedDays
Length
Hours
0s completed at 11:09 PM
kalamgaru.webp
melbourne.zip
Show all

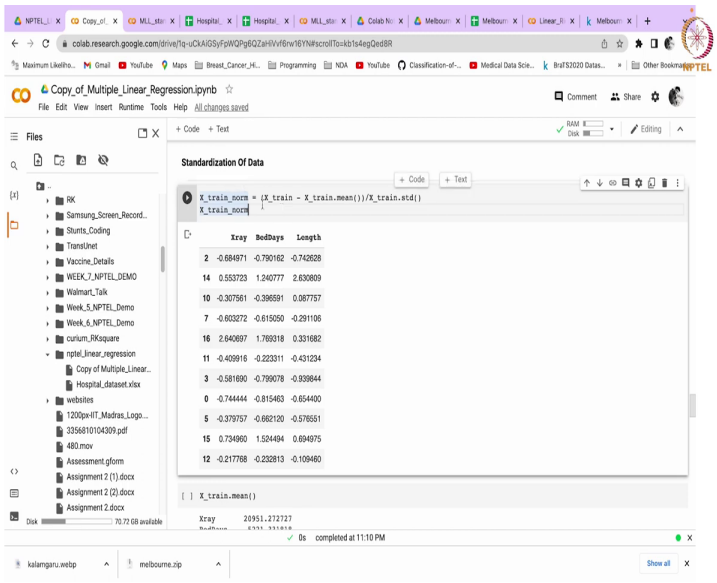

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Editing
Files
sns.heatmap(train.corr(),annot = True)
<matplotlib.axes._subplots.AxesSubplot at 0x7f14f8edc150>
Xray
BedDays
Length
Hours
8s completed at 11:09 PM
kalamgaru.webp
melbourne.zip
Show all

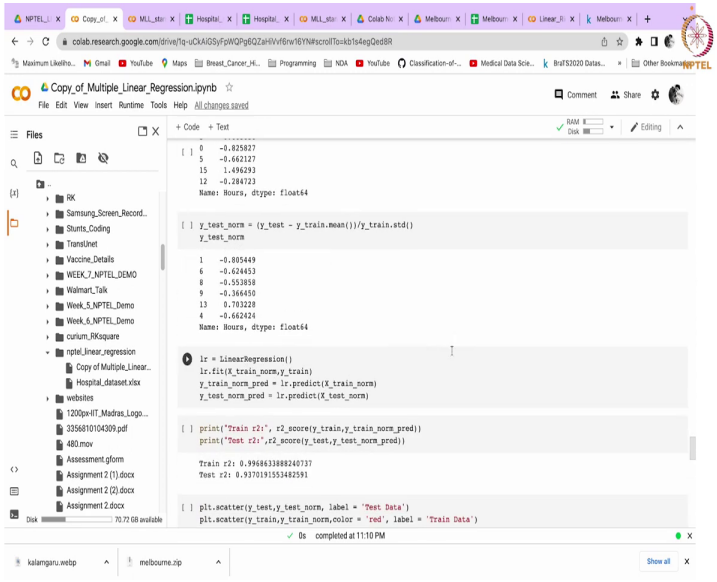

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
Fitting a Multiple Linear Regression Model using LinearRegression()
#Linear Regression model
lm = LinearRegression()
lm.fit(X_train,y_train)
#R^2
print('The R^2 value is:', lm.score(X_train,y_train))
print('r2 score for test:', lm.score(X_test,y_test))
The R^2 value is: 0.9968633888240737
r2 score for test: 0.9370191553482593
#Predictions | Error
y_pred = lm.predict(X_test)
print(y_pred)
[628.79444995 1639.11282275 2745.13575474 4048.58084533 8508.19685434
1489.92211201]
y_train_pred = lm.predict(X_train)
print(y_train_pred)
[988.55925994 11843.0375161 3106.79941671 1779.3837933
18878.27266865 4409.03039975 1190.40021317 710.63367711
1947.10089419 15146.35687086 6310.01960819]
y_pred
0s completed at 11:09 PM
kalamgaru.webp
melbourne.zip
Show all

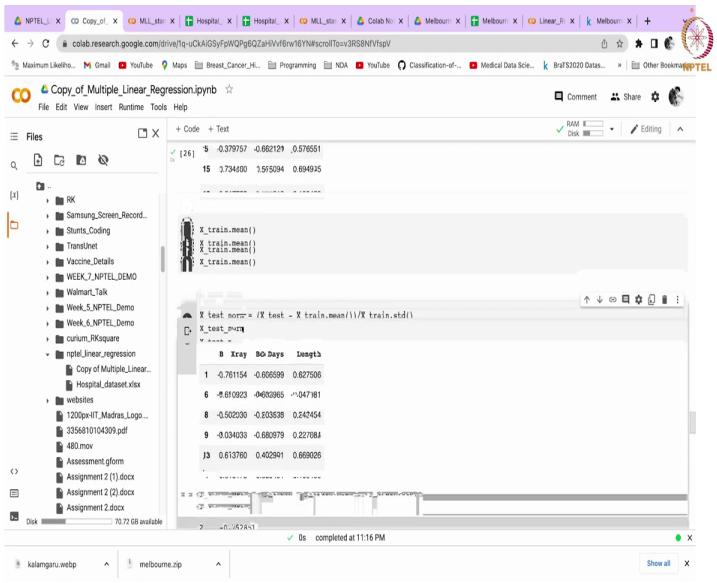

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
[24] y_pred
array([628.79444995, 1639.11282275, 2745.13575474, 4048.58084533,
8508.19685434, 1489.92211201])
y_test
1 696.82
6 1854.17
8 2305.58
9 3503.93
13 10343.81
4 1611.37
Name: Hours, dtype: float64
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
mean_squared_error(y_test , y_pred)
r2_score(y_test,y_pred)
0.9370191553482593
r2_score(y_train,y_train_pred)
0.9968633888240737
print('The R^2 value is:', lm.score(X_test,y_test))
The R^2 value is: 0.9370191553482593
0s completed at 11:10 PM
kalamgaru.webp
melbourne.zip
Show all

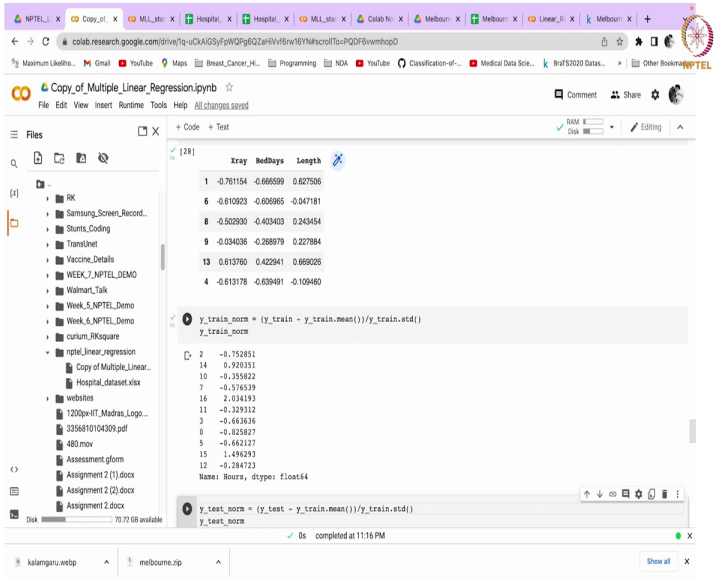

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
[23] [986.55825994 11843.8375161 3106.79841671 1779.38379933
18878.27266865 4409.03039975 1190.40021317 710.63367711
1947.18849419 15146.35687086 4318.01968419]
y_pred
array([628.79444995, 1639.11282275, 2745.13575474, 4048.58084533,
8508.19@85434, 1489.92211201])
y_test
1 696.82
6 1854.17
8 2305.58
9 3503.93
13 10343.81
4 1611.37
Name: Hours, dtype: float64
[ ] from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
mean_squared_error(y_test , y_pred)
r2_score(y_test,y_pred)
0.9370191553482593
[ ] r2_score(y_train,y_train_pred)
0.9968633888240737
0s completed at 11:10 PM
kalamgaru.webp
melbourne.zip
Show all

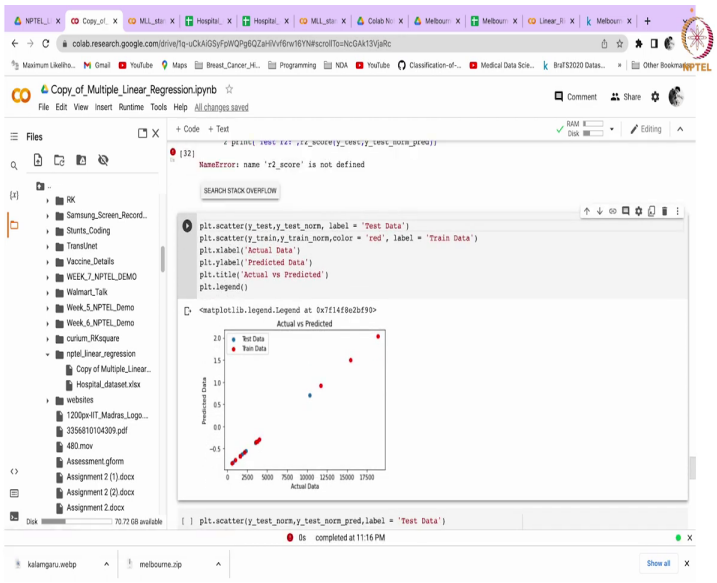

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
mean_squared_error(y_test , y_pred)
r2_score(y_test,y_pred)
0.9370191553482593
[ ] r2_score(y_train,y_train_pred)
0.9968633888240737
[ ] print('The R^2 value is:', lm.score(X_test,y_test))
The R^2 value is: 0.9370191553482593
[ ] print('The R^2 value is:', lm.score(X_test,y_pred))
The R^2 value is: 1.0
[ ] df.head()
Xray BedDays Length Hours
0 2463 472.92 4.45 566.52
1 2048 1339.75 6.92 696.82
2 3940 620.25 4.28 1033.15
3 6505 568.33 3.90 1603.62
0s completed at 11:10 PM
kalamgaru.webp
melbourne.zip
Show all

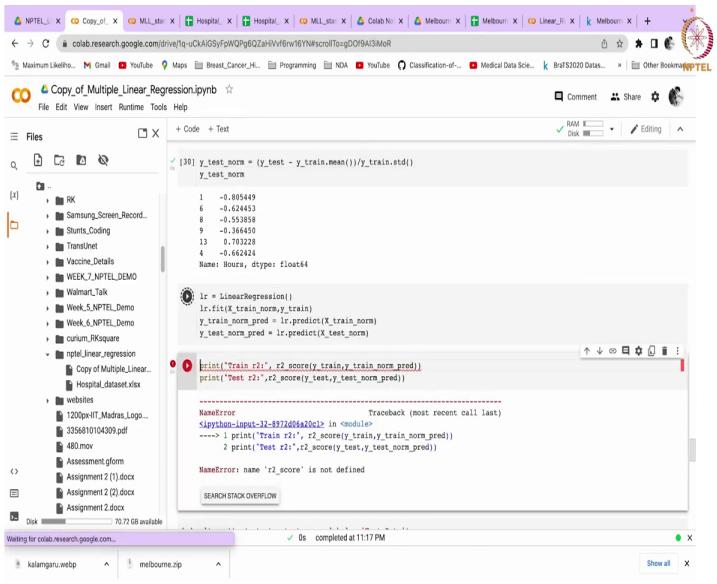

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
Editing
Standardization Of Data
X_train_norm = (X_train - X_train.mean())/X_train.std()
X_train_norm
Xray BodDays Length
2 -0.684971 -0.790162 -0.742628
14 0.553723 1.240777 2.630809
10 -0.307561 -0.396591 0.067757
7 -0.603272 -0.615050 -0.291106
16 2.640897 1.769318 0.331682
11 -0.409916 -0.223311 -0.431234
3 -0.581690 -0.799078 -0.939844
0 -0.744444 -0.815463 -0.654400
5 -0.379757 -0.662120 -0.576551
15 0.734960 1.524494 0.694975
12 -0.217768 -0.232813 -0.109460
X_train.mean()
Xray 20951.272727
0s completed at 11:10 PM
kalamgaru.webp
melbourne.zip
Show all


Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
Editing
0 -0.825827
5 -0.662127
15 1.496293
12 -0.284723
Name: Hours, dtype: float64
y_test_norm = (y_test - y_train.mean())/y_train.std()
y_test_norm
1 -0.805449
6 -0.624453
8 -0.553858
9 -0.366450
13 0.703228
4 -0.662424
Name: Hours, dtype: float64
lr = LinearRegression()
lr.fit(X_train_norm,y_train)
y_train_norm_pred = lr.predict(X_train_norm)
y_test_norm_pred = lr.predict(X_test_norm)
print("Train r2:", r2_score(y_train,y_train_norm_pred))
print("Test r2:",r2_score(y_test,y_test_norm_pred))
Train r2: 0.9968633888240737
Test r2: 0.9370191553482591
plt.scatter(y_test,y_test_norm, label = 'Test Data')
plt.scatter(y_train,y_train_norm,color = 'red', label = 'Train Data')
0s completed at 11:10 PM
kalamgaru.webp
melbourne.zip
Show all


Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Files
+ Code
+ Text
RAM
Disk
Editing
[26]
5 -0.379757 -0.662129 -0.576551
15 3.734600 1.595094 0.694935
X_train.mean()
X_train.mean()
X_train.mean()
X_train.mean()
X_test_norm = (X_test - X_train.mean())/X_train.std()
X_test_norm
B Xray BedDays Length
1 -0.761154 -0.666599 0.627506
6 -0.610923 -0.606965 -0.047181
8 -0.502030 -0.603638 0.242454
9 -0.034033 -0.680979 0.227084
13 0.613760 0.402941 0.669026
0s completed at 11:16 PM
kalamgaru.webp
melbourne.zip
Show all


Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code
+ Text
RAM
Disk
Editing
[28]
Xray BedDays Length
1 -0.761154 -0.666599 0.627506
6 -0.610923 -0.606965 -0.047181
8 -0.502930 -0.403403 0.243454
9 -0.034036 -0.268979 0.227884
13 0.613760 0.422941 0.669026
4 -0.613178 -0.639491 -0.109460
y_train_norm = (y_train - y_train.mean())/y_train.std()
y_train_norm
2 -0.752851
14 0.920351
10 -0.355822
7 -0.576539
16 2.034193
11 -0.329312
3 -0.663636
0 -0.825827
5 -0.662127
15 1.496293
12 -0.284723
Name: Hours, dtype: float64
y_test_norm = (y_test - y_train.mean())/y_train.std()
y_test_norm
0s completed at 11:16 PM
kalamgaru.webp
melbourne.zip
Show all


Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Editing
Files
RK
Samsung_Screen_Record...
Stunts_Coding
TransUnet
Vaccine_Details
WEEK_7_NPTEL_DEMO
Walmart_Talk
Week_5_NPTEL_Demo
Week_6_NPTEL_Demo
curium_RKsquare
nptel_linear_regression
Copy of Multiple_Linear...
Hospital_dataset.xlsx
websites
1200px-IIT_Madras_Logo...
3356810104309.pdf
480.mov
Assessment.gform
Assignment 2 (1).docx
Assignment 2 (2).docx
Assignment 2.docx
Disk 70.72 GB available
[32]
NameError: name 'r2_score' is not defined
SEARCH STACK OVERFLOW
plt.scatter(y_test,y_test_norm, label = 'Test Data')
plt.scatter(y_train,y_train_norm,color = 'red', label = 'Train Data')
plt.xlabel('Actual Data')
plt.ylabel('Predicted Data')
plt.title('Actual vs Predicted')
plt.legend()
<matplotlib.legend.Legend at 0x7f14f8e2bf90>
Actual vs Predicted
Test Data
Train Data
Predicted Data
Actual Data
plt.scatter(y_test_norm,y_test_norm_pred,label = 'Test Data')
0s completed at 11:16 PM
kalamganu.webp
melbourne.zip
Show all


Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
Editing
Files
RK
Samsung_Screen_Record...
Stunts_Coding
TransUnet
Vaccine_Details
WEEK_7_NPTEL_DEMO
Walmart_Talk
Week_5_NPTEL_Demo
Week_6_NPTEL_Demo
curium_RKsquare
nptel_linear_regression
Copy of Multiple_Linear...
Hospital_dataset.xlsx
websites
1200px-IIT_Madras_Logo...
3356810104309.pdf
480.mov
Assessment.gform
Assignment 2 (1).docx
Assignment 2 (2).docx
Assignment 2.docx
Disk 70.72 GB available
[30] y_test_norm = (y_test - y_train.mean())/y_train.std()
y_test_norm
1 -0.805449
6 -0.624453
8 -0.553858
9 -0.366450
13 0.703228
4 -0.662424
Name: Hours, dtype: float64
lr = LinearRegression()
lr.fit(X_train_norm,y_train)
y_train_norm_pred = lr.predict(X_train_norm)
y_test_norm_pred = lr.predict(X_test_norm)
print("Train r2:", r2_score(y_train,y_train_norm_pred))
print("Test r2:",r2_score(y_test,y_test_norm_pred))
NameError Traceback (most recent call last)
<ipython-input-32-8972d06a20c1> in <module>
----> 1 print("Train r2:", r2_score(y_train,y_train_norm_pred))
2 print("Test r2:",r2_score(y_test,y_test_norm_pred))
NameError: name 'r2_score' is not defined
SEARCH STACK OVERFLOW
Waiting for colab.research.google.com...
0s completed at 11:17 PM
kalamganu.webp
melbourne.zip
Show all

colab.research.google.com/drive/1q-uCkAIGSyFpWQPg6QZaHiVvf6rw16YN#scrollTo=NcGAk13VjaRc

Copy_of_Multiple_Linear_Regression.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

Files

RK
Samsung_Screen_Record...
Stunts_Coding
TransUnet
Vaccine_Details
WEEK_7_NPTEL_DEMO
Walmart_Talk
Week_5_NPTEL_Demo
Week_6_NPTEL_Demo
curium_RKsquare
nptel_linear_regression
Copy of Multiple_Linear...
Hospital_dataset.xlsx
websites
1200px-IIT_Madras_Logo...
3356810104309.pdf
480.mov
Assessment.gform
Assignment 2 (1).docx
Assignment 2 (2).docx
Assignment 2.docx

Disk 70.72 GB available

```
[30] 8    -0.553858
     9    -0.366450
     13    0.703228
     4    -0.662424
     Name: Hours, dtype: float64
```

```
[42] lr = LinearRegression()
     lr.fit(X_train_norm,y_train)
     y_train_norm_pred = lr.predict(X_train_norm)
     y_test_norm_pred = lr.predict(X_test_norm)
```

```
print("Train r2:", r2_score(y_train,y_train_norm_pred))
print("Test r2:",r2_score(y_test,y_test_norm_pred))
```

```
Train r2: 0.9968633888240737
Test r2: 0.9370191553482591
```

```
plt.scatter(y_test,y_test_norm, label = 'Test Data')
plt.scatter(y_train,y_train_norm,color = 'red', label = 'Train Data')
plt.xlabel('Actual Data')
plt.ylabel('Predicted Data')
plt.title('Actual vs Predicted')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f14f8e2bf90>
```

0s completed at 11:24 PM

kalamgaru.webp melbourne.zip Show all

colab.research.google.com/drive/1q-uCkAIGSyFpWQPg6QZaHiVvf6rw16YN#scrollTo=NcGAk13VjaRc

Copy_of_Multiple_Linear_Regression.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

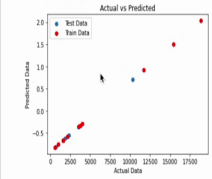
Files

RK
Samsung_Screen_Record...
Stunts_Coding
TransUnet
Vaccine_Details
WEEK_7_NPTEL_DEMO
Walmart_Talk
Week_5_NPTEL_Demo
Week_6_NPTEL_Demo
curium_RKsquare
nptel_linear_regression
Copy of Multiple_Linear...
Hospital_dataset.xlsx
websites
1200px-IIT_Madras_Logo...
3356810104309.pdf
480.mov
Assessment.gform
Assignment 2 (1).docx
Assignment 2 (2).docx
Assignment 2.docx

Disk 70.72 GB available

```
plt.scatter(y_train,y_train_norm,color = 'red', label = 'Train Data')
plt.xlabel('Actual Data')
plt.ylabel('Predicted Data')
plt.title('Actual vs Predicted')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f14f8e2bf90>
```

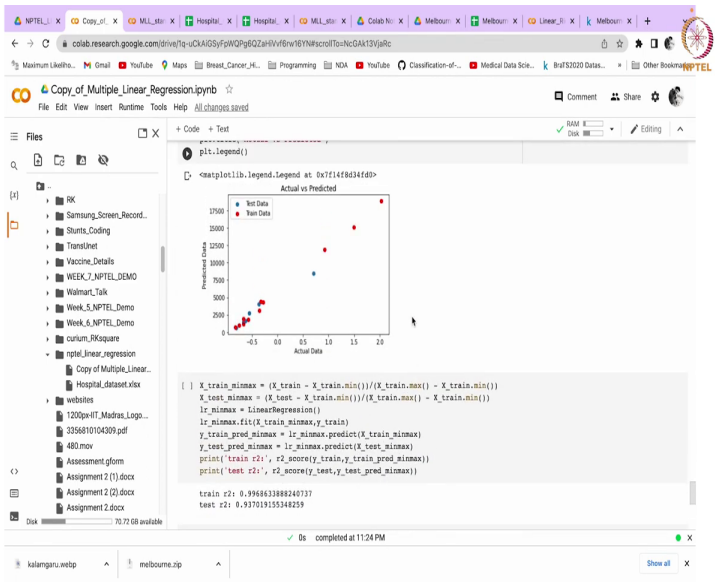


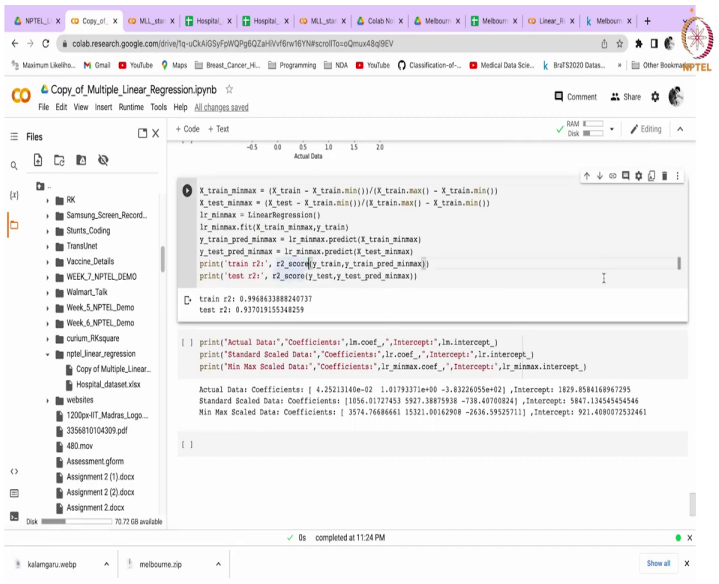

```
plt.scatter(y_test_norm,y_test_norm_pred,label = 'Test Data')
plt.scatter(y_train_norm,y_train_norm_pred,color = 'red',label = 'Train Data')
plt.xlabel('Actual Data')
plt.ylabel('Predicted Data')
plt.title('Actual vs Predicted')
plt.legend()
```

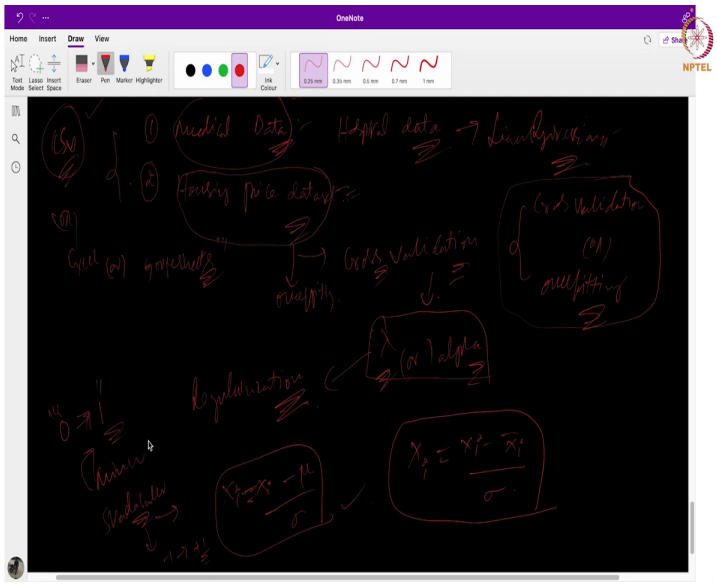
```
<matplotlib.legend.Legend at 0x7f14f8d34fd0>
```

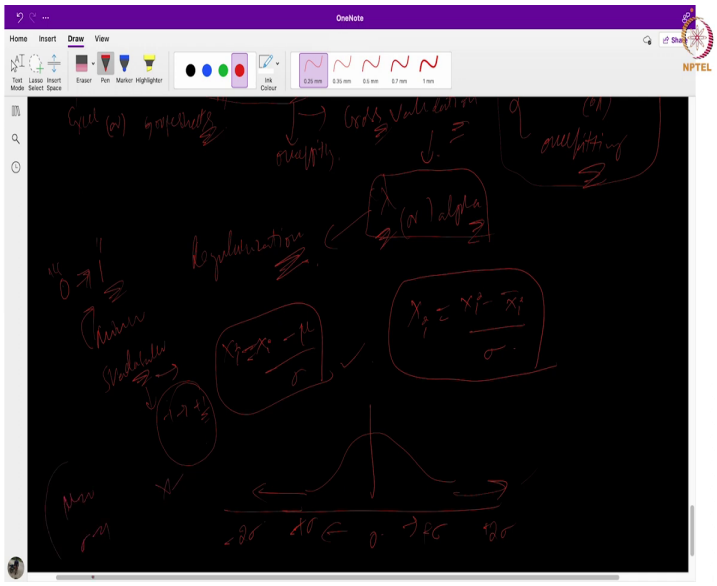
0s completed at 11:24 PM

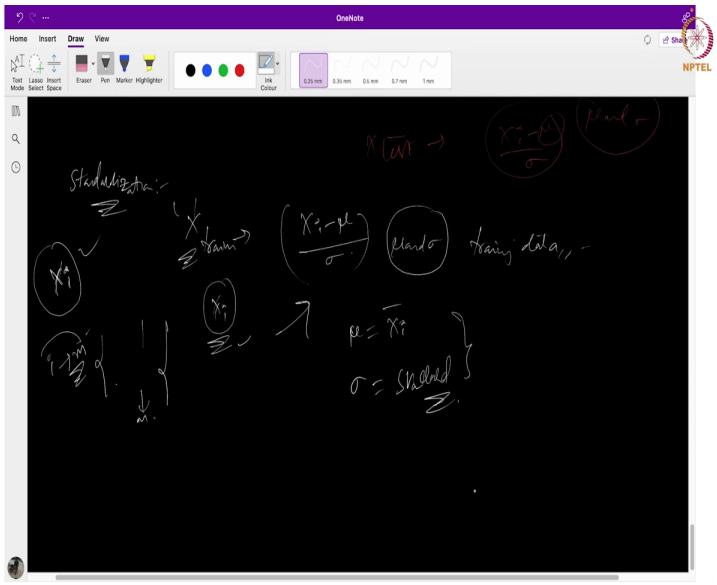
kalamgaru.webp melbourne.zip Show all

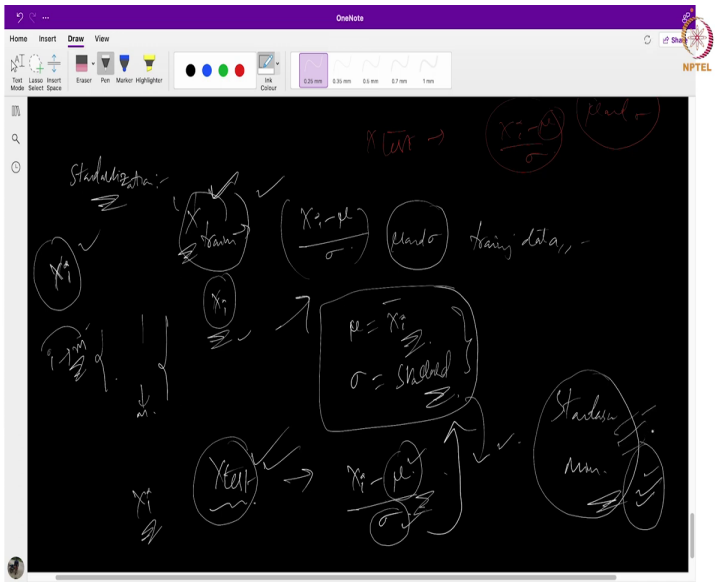

Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
plt.legend()
<matplotlib.legend.Legend at 0x7f14f8d34fd0>
Actual vs Predicted
Test Data
Train Data
Predicted Data
Actual Data
X_train_minmax = (X_train - X_train.min())/(X_train.max() - X_train.min())
X_test_minmax = (X_test - X_train.min())/(X_train.max() - X_train.min())
lr_minmax = LinearRegression()
lr_minmax.fit(X_train_minmax,y_train)
y_train_pred_minmax = lr_minmax.predict(X_train_minmax)
y_test_pred_minmax = lr_minmax.predict(X_test_minmax)
print('train r2:', r2_score(y_train,y_train_pred_minmax))
print('test r2:', r2_score(y_test,y_test_pred_minmax))
train r2: 0.9968633888240737
test r2: 0.9370191553482 59
0s completed at 11:24 PM
kalamgaru.webp
melbourne.zip
Show all


Copy_of_Multiple_Linear_Regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
Actual Data
X_train_minmax = (X_train - X_train.min())/(X_train.max() - X_train.min())
X_test_minmax = (X_test - X_train.min())/(X_train.max() - X_train.min())
lr_minmax = LinearRegression()
lr_minmax.fit(X_train_minmax,y_train)
y_train_pred_minmax = lr_minmax.predict(X_train_minmax)
y_test_pred_minmax = lr_minmax.predict(X_test_minmax)
print('train r2:', r2_score(y_train,y_train_pred_minmax))
print('test r2:', r2_score(y_test,y_test_pred_minmax))
train r2: 0.9968633888240737
test r2: 0.9370191553482 59
print("Actual Data:","Coefficients:",lm.coef_,"Intercept:",lm.intercept_)
print("Standard Scaled Data:","Coefficients:",lr.coef_,"Intercept:",lr.intercept_)
print("Min Max Scaled Data:","Coefficients:",lr_minmax.coef_,"Intercept:",lr_minmax.intercept_)
Actual Data: Coefficients: [4.25213140e-02 1.01793371e+00 -3.83226055e+02] ,Intercept: 1829.0584168967295
Standard Scaled Data: Coefficients: [1056.01727453 5927.38875938 -738.40700824] ,Intercept: 5847.134545454546
Min Max Scaled Data: Coefficients: [3574.76686661 15321.00162908 -2636.59525711] ,Intercept: 921.488007532461
0s completed at 11:24 PM
kalamgaru.webp
melbourne.zip
Show all


OneNote
Home
Insert
Draw
View
Text Mode
Lasso Select
Insert Space
Eraser
Pen
Marker
Highlighter
Ink Colour
NPTEL


OneNote
Home
Insert
Draw
View
Text Mode
Lasso Select
Insert Space
Eraser
Pen
Marker
Highlighter
Ink Colour
NPTEL





There you can see the entire trip where the folder name is NPTEL Linear Regression Book, NPTEL underscore linear underscore. From here, I just have to first import all these libraries where from the scalar, I am importing the linear regression model, Matplotlib for visualization, and this pandas will be for data frame analysis, Seaborn is also like more or less like a plotting library only.

Here, I have read the data. If you observe, I am just taking the shape of this particular file df.head, it will generate the first five points. The shape of this entire data frame is 17×4,

that means 17 rows and 4 columns. The last, Column 4 hours indicates the target variable. These first three variables indicate the independent variables or features.

Next, is just I am finding out how exactly this hours is related to hours, similarly, bed days is a correlation with this particular independent, each individual variable with the hours, I am just checking out. Here, I have imported the train test split, where as I have told you, I am just here splitting the data into training and testing only. There is no part of validation here in this particular problem.

After splitting, you can see the length of the train is 11, because I have used 30% is for training, sorry, 70% training and 30% is for testing, you can see here. This is nothing but the test data. Just, I am taking all these first three columns of into the x variables and last column into the y variable, x_train, y_train, just check out the shape.

Because I am dealing with the x_train, these are the things you have to focus. Similarly, I am just separating out the extract in x_test and y_test. This is nothing but the shape (6,3). They are just correlation plots, how exactly x beds are related to hours, similarly, how exactly bed days is, all this data is related to particular month only. You can observe that, most of the times they are linearly correlated, I mean closely correlated, high correlation.

Instead of generating this individual diagrams like this, here there is a command where you can generate all the things like this. It takes some time, just wait. This, in terms of values correlation, values correlation coefficient, correlation coefficient, basically, it indicates how exactly two variables are related to each other.

Now, coming back to importing this particular linear regression model. I am just writing down here, this is the main important thing, linear regression has been imported using this function. From a scalar library dot linear models, import linear regression. That linear related regression object, I am using right now, where you can see like this.

Once this is done, once the object is taken, lm.fit, this is the function we will have, x_train gamma, y_train, and I will obtain the R^2 value for this one. x_train, y_train, x_test gamma, y_test. They are all mostly closely related to each other. We can say that one from these two values.

Here, I am just doing the prediction errors, what is the y_prediction and y_test. This is just basically using several things, I have just extended this task, like I have mentioned earlier, mean square error can be calculated as a metric, even mean absolute error can be calculated as a metric and R^2 scores can also be used as a metric.

Here, this is the, main thing is, here, you have to focus, once the object is taken, just fit the model and obtain the score. Here, we are just obtaining the score for original trained data and test, trained data. Similarly, I am focusing now on x_test and y_test. If you observe here, both value, training value is more, even R^2, sorry, R^2 score of the test data is also more.

That means more or less, they are equal. So, we can say that model is good only. But here in this particular problem, you have used only 17 points. When we discuss next problem, you will get to know very well. These are just, you need not bother about all the remaining things. I am just extending the task like, I am just predicting the values here.

Here, I have taken directly R^2 score value, here, I am just taking the enter, of, if I give you x_test as an input, this is how the values are. y trained predict, also I have taken here. This is our test data, this is after fitting the model, I am just predicting the x trained values also. These are nothing but y predict value, y_test value, you can see here, the original value is 628 comma, my out is testing, test data, this is how the value is.

16939 means 1854, here. Like, 10, so last one if you take, 1611, here is 1489. This will, at least more or less close to the original values. This is good enough. This is like, mean square error, mean absolute error and R^2 score, I am just directly importing these functions instead of using this particular formula like lm.score, which I have used here. This is another way of doing this thing. This is just an extension, no need to bother.

Till now we have done, what I have done is just I have taken the data and I have splitted the data into x_train and y_train. Similarly, x_test and y_test. After doing that, and I have fitted the model, after fitting the model, I have found out the R^2 score value, after finding out this R^2 score value.

And even I have experimented with the mean square well, mean square error value and mean absolute error value also, but does not make any sense, not necessary to use so many things. You can take a single metric and you can comment on. Here, mean absolute error and mean square error should always be small, we will expect, whereas R^2 square value will remain in 0 to 1, but we expect the value to be 1, near to 1. I have just extended that one.

In most of the cases, in general, the data set, having several variables, suppose in this case we have x-rays, bed days and length. If you observe the data, like this here, see, all these data points, lengthwise, these are very small, actually, compared to the remaining variables. To let the model, not to get confused too much, we will bring all the columns to here following a particular distribution, like normal distribution.

So, we will bring the, all the columns of the features to follow a particular normal distribution. Normal distribution is nothing but standard deviation is having plus or minus 1, where this particular mean value should be 0. To even, to do that one, we have directly somehow, some libraries, from the library we can import that one even, see here. Standardize, like $\frac{x-\bar{x}}{\sigma}$. This is actually, this is the formula we will use, generally, for standardization of data.

Like suppose x, I have $\frac{Xi-\bar{X}i}{\sigma}$. In other words, we can say, $Xi = \frac{Xi-\bar{X}i}{\sigma}$, or $Xi = \frac{Xi-\mu}{\sigma}$. This is how we will have. Even to do this one, we have something like a standard scalar function is there. Similarly, min max scalar is there. Standard, scalar will have minus 1 to plus 1. We will bring the data to this value, whereas here, min max scalar, we will we will convert each and every column variable to values between 0 to 1. This is how we will do.

For that one, here I have just done the manual implementation, like $\frac{x-\bar{x}}{\sigma}$. See here, all the values will range between minus 1 to plus 1 only, even this one. No, sorry, σ value is plus 1, standard deviation, like standard normal distribution in the sense, like this one.

Oh sorry, I am sorry, suppose if this is how the data, this is 0, you will have up to plus sigma, minus sigma plus 2 sigma, minus 2 sigma, like this data will distribute. I am sorry,

I just told minus 1 to plus 1, that is not the case. We can say that mean is 0 and sigma deviation is 1. That is the case. That is good.

Just manual implementation, they have done. Now coming back to, let us check. Standardization has been done. What we are doing here is without standardization, we have done, similarly, with standardization, we are doing here. Since the, here, the data points which we are using is very less data points like 17 rows and only 4 columns but if you, the data set is more, the data point is more, you can see that particular difference.

Always this, normalization will always help to converge the faster, converge the model first, and at the same time, very well. Just, we have done the earlier things only, just, I am doing r2 square is not different, because I just did not run this particular program, I think. From r2, where it is, see, here, I did not run this one. That is why problem occurred. Standardization has been done for the training test data, training data.

Always remember, while doing this normalization, while doing this normalization, whatever, suppose I am doing the normalization for x_data, like trained data, we will compute mean and standard deviation for each and every variable, and I will explain as mu by sigma. I will do like this. Now, comes to test data $\frac{Xi-\mu}{\sigma}$.

Use the μ and σ of what we got, now when it comes to standardization, when comes to standardization, standardization, suppose I am doing the standardization for x_train data, we will use like $\frac{Xi-\mu}{\sigma}$. This μ and σ are computed from the training data only, data.

Assume that I am dealing, I am talking about the standardization for a particular variable like Xi. This Xi is a variable where number of data points I have is i is equal to 1 to m, and among these number of data points, like 1 to m data points, I am just focusing on particular variable of particular feature only.

Now, I have to standardize this data, like this formula, I will use, like $\frac{Xi-\mu}{\sigma}$. μ is nothing but $\bar{x}_i$ where σ is nothing but the standard deviation, how we compute normally. This is how we will do the standardization for the particular variable in the training, while standardizing the training data.

Now, it comes to test data standardization, where we will have only the same variable, I am talking about, like $\frac{Xi-\mu}{\sigma}$, use the same mean and standard deviation that I have used, are calculated for the training data. That is the basic difference. Please keep this point in mind.

Whenever we are doing standardization, while doing the standardization, when we are dealing with the training data to compute the standardization, we will use this formula like $\frac{Xi-\bar{X}i}{\sigma}$. Similarly, for the test standardization, we will use the same μ and σ. That means, we are not going to compute the new mean and sigma for this test data. Please keep that point in mind.

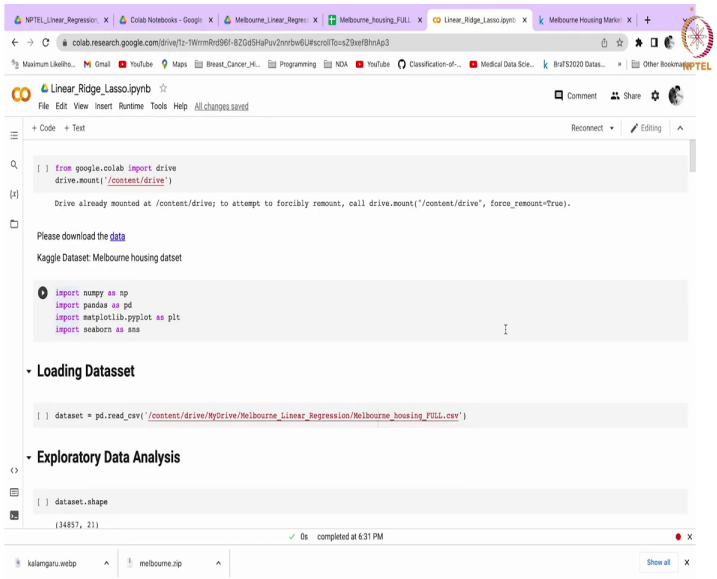
Even here, there are some functions like standard scalar or min max scalar, these functions are available directly even while doing these things, you just have to be careful how to use them.

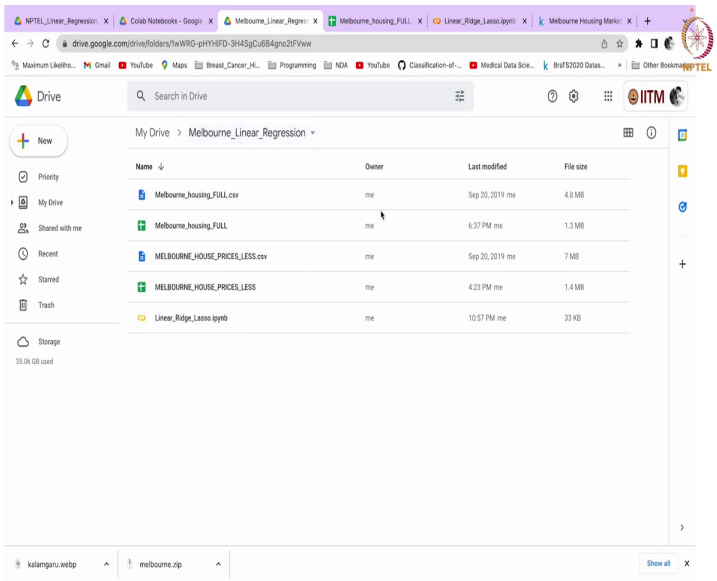
After doing normalization, again I have run the code. This is how the values are. If you observe clearly, there is no much difference, because in general the difference you can see, but here I have used the basic, very less number of data points and very less number of, that is where this problem is happening. So, in total, just I have just predicted the y actual data and predicting that like very y_test and y_predicted, just computed here, something like this, y_test and y test-train scatter plot for these, the plots I have done, scatter plot, you can observe. That is not that difficult.

Here, I have used another min max standardization. I have just, after doing this entire thing, you can even see the, those coefficient values also, you can see, like this, using this function, like lm.intercept, something like this. Just go with this code, you will easily understand what exactly is going on.

In total, as a summary, I can say that you should have to be familiar with NumPy library, pandas library, matplotlib library, Scikit-learn library for machine learning and the deep learning for PyTorch and TensorFlow. That is it.

(Refer Slide Time: 46:54)



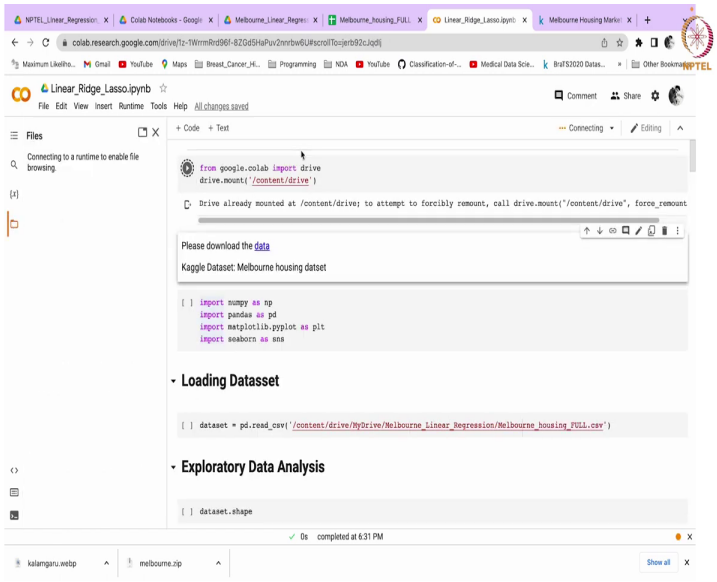


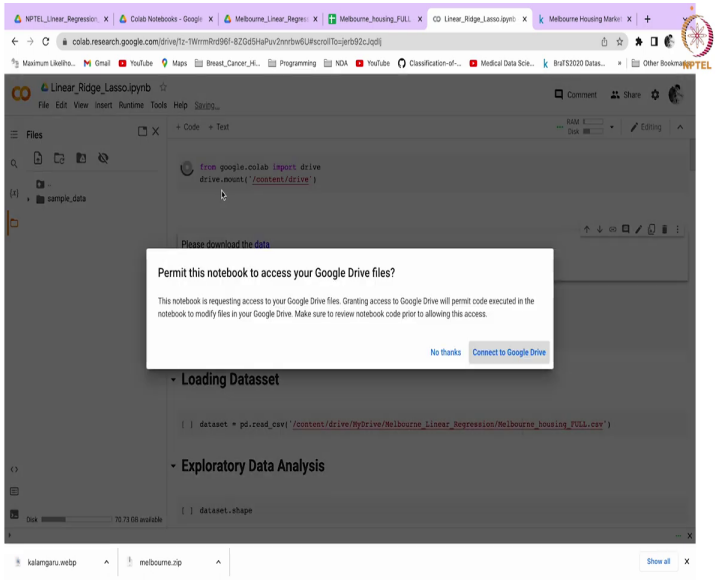
This is the new example which I told you earlier, that which it involves both overfitting problem and even here we will use this cross validation for the hyper parameter tuning. In this, for the hyper parameter tuning, mostly here, we are using, apart from just take that linear regression function, we are going to use something, two more functions like ridge regression and lasso regression.

Theoretical part is, according to your convenience you can browse for it, it is not that necessary, but coming, let me show you the implementation. I have already given you

this particular entire data set or this particular folder. I have given you. Just upload this particular folder to your drive, and just to open this linear ridge, lasso regression iPython notebook.

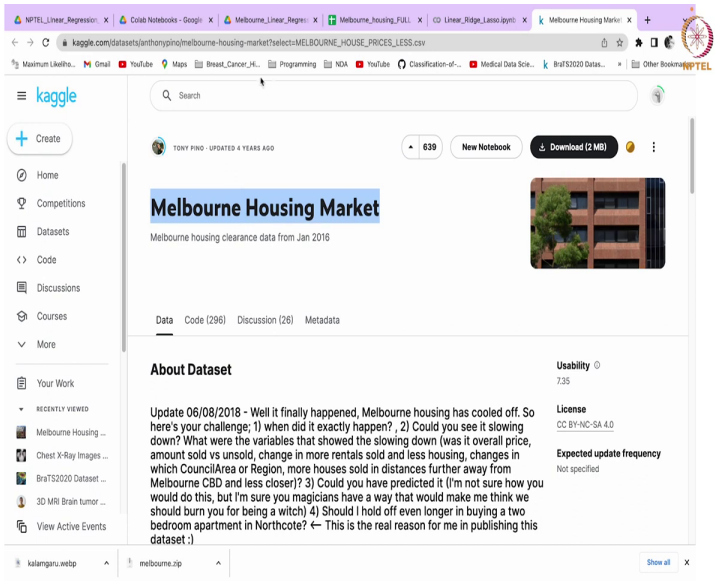
(Refer Slide Time: 47:46)





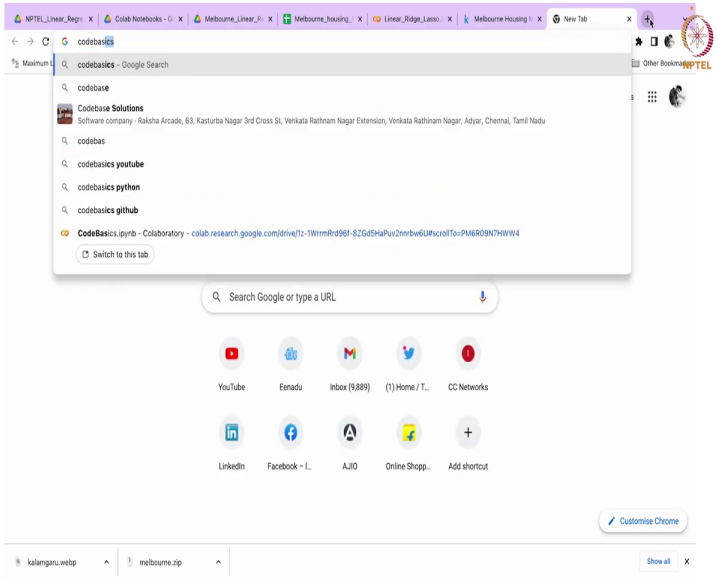
Once you double click over this one, you will get here. If you observe, here, our first job is to just mount your drive. Let me check once this drive is being mounted.

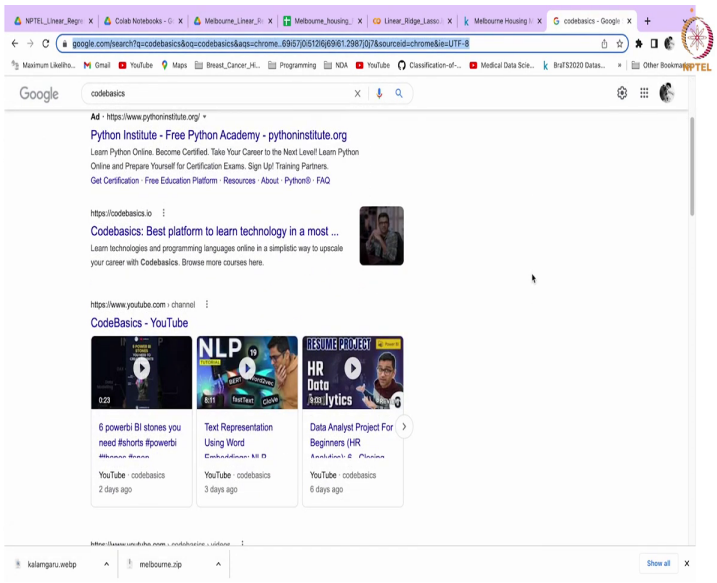
(Refer Slide Time: 48:17)



The data set which we are using here is, just let me click on this, the data set which we are using here is from Kaggle dataset, where you can call this one as Melbourne Housing Market. And for doing this tutorial, I have taken help from Codebasics channels, that is, in YouTube.

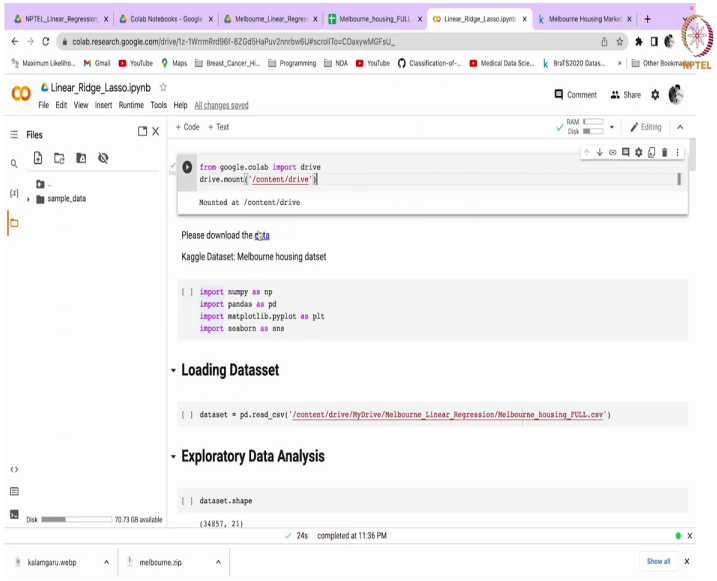
(Refer Slide Time: 48:38)

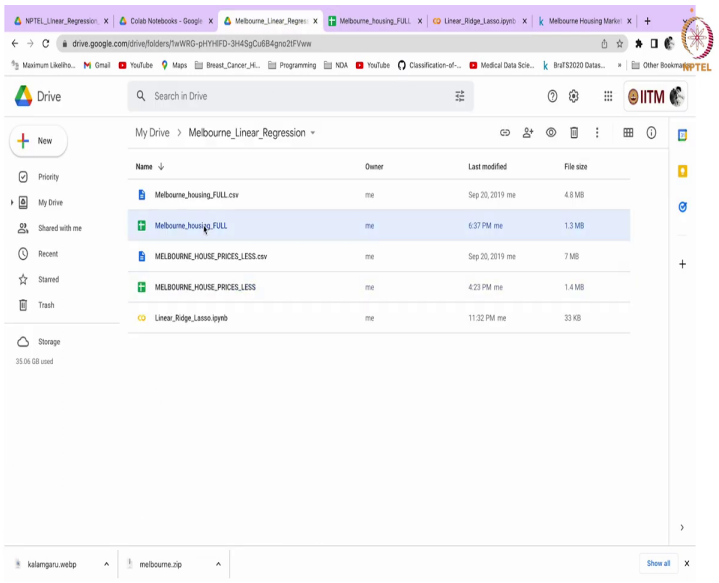


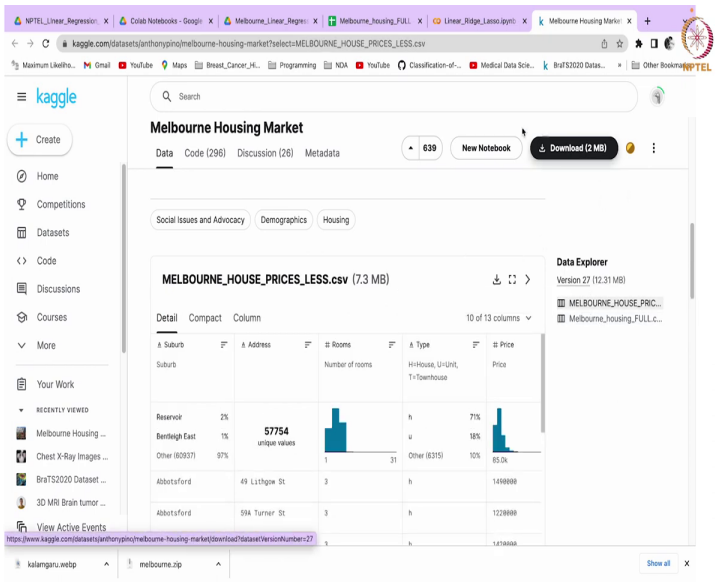


c-o-d-e s-i-c-s, YouTube channel is there. I am referring his videos actually. This is just for implementation part. If you have time and if you want to explore more algorithms, you can go and check that. I have taken reference from there.

(Refer Slide Time: 48:59)

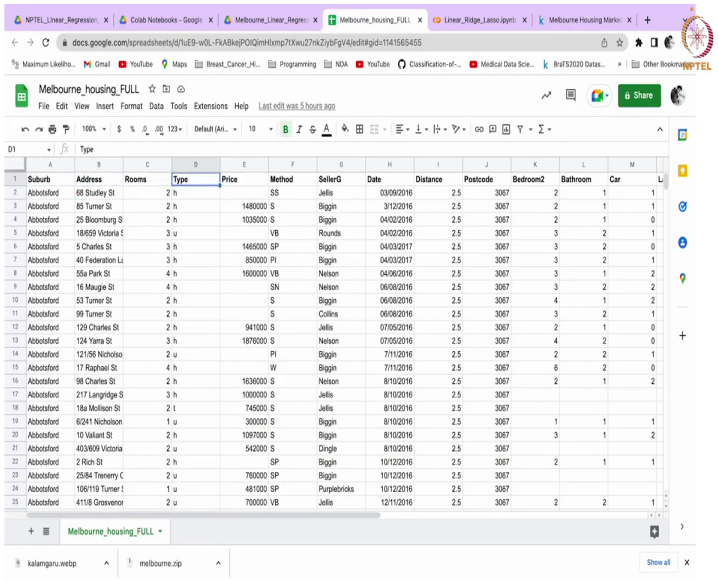






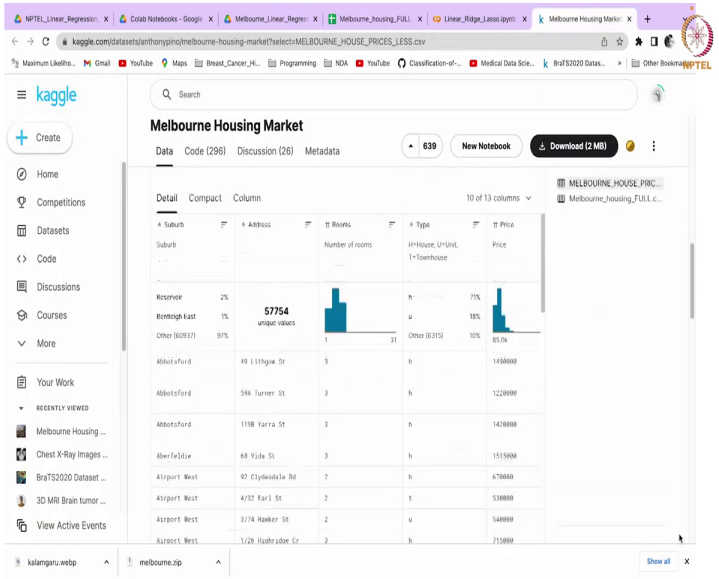
Now let me explain, this is the data set, just click here, this particular data. This is for namesake, I am just explaining but in the folder, I have already downloaded data from both the points, from Kaggle. Click here data, if you go down, you can see both these files. Just click this download button, enter download button here, you can download all the files, data

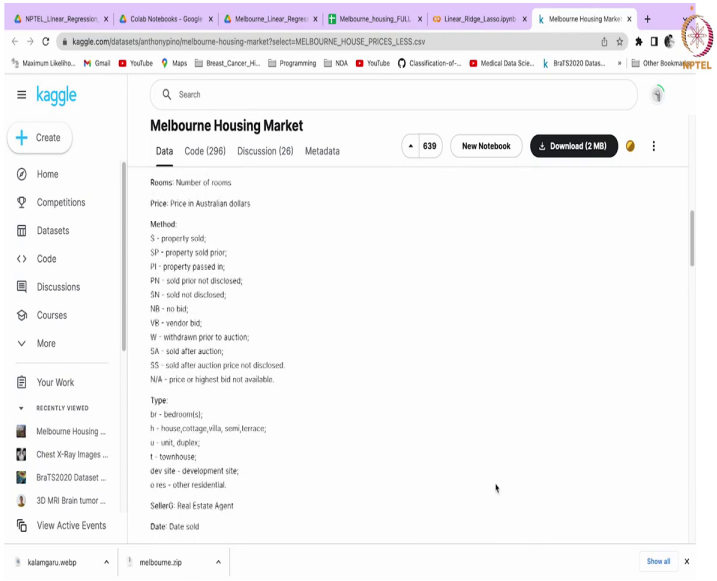
.(Refer Slide Time: 49:27)

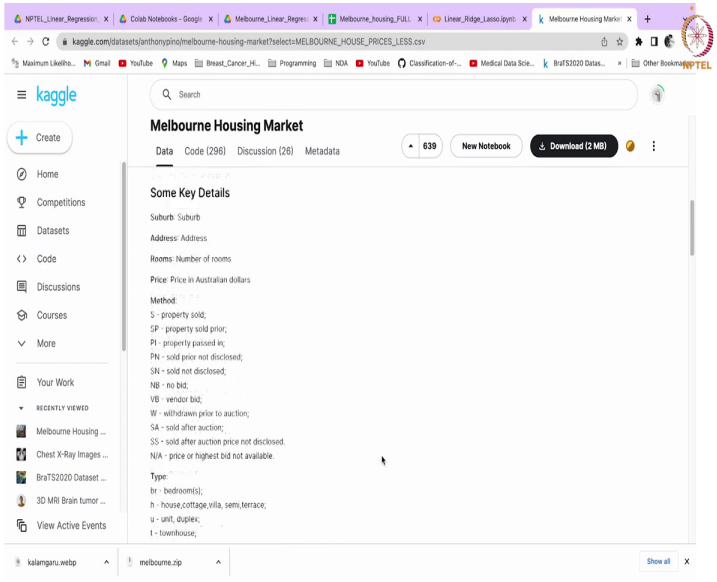


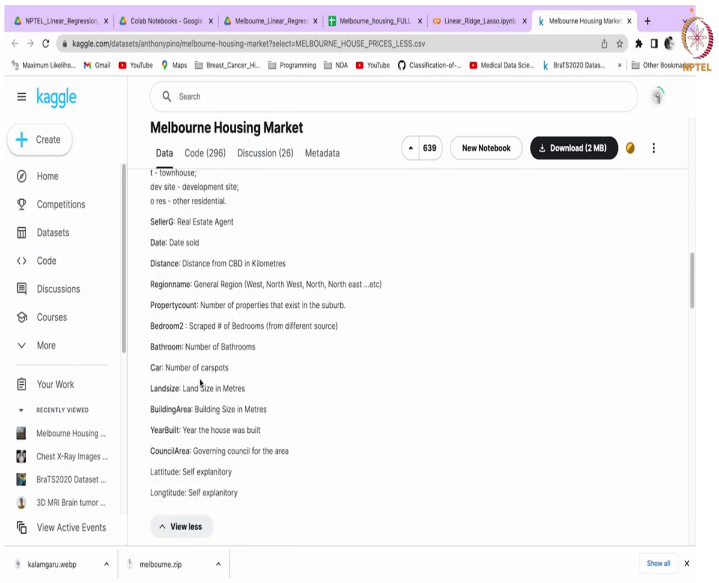
This is how the data has been given, like Suburb and Address, Rooms, Type, Price, Method, SellerG.

(Refer Slide Time: 49:37)



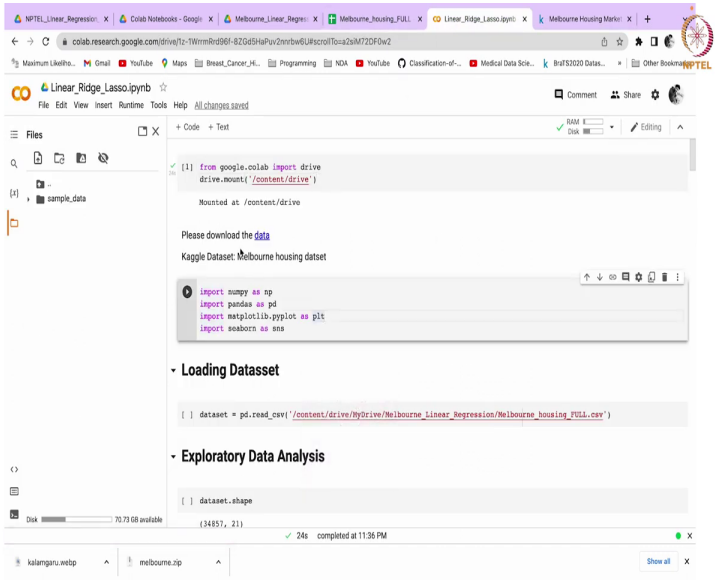

kaggle.com/datasets/anthonypino/melbourne-housing-market?select=MELBOURNE_HOUSE_PRICES_LESS.csv
kaggle
Search
Create
Home
Competitions
Datasets
Code
Discussions
Courses
More
Your Work
RECENTLY VIEWED
Melbourne Housing ...
Chest X-Ray Images ...
BraTS2020 Dataset ...
3D MRI Brain tumor ...
View Active Events
Melbourne Housing Market
Data
Code (296)
Discussion (26)
Metadata
639
New Notebook
Download (2 MB)
Rooms: Number of rooms
Price: Price in Australian dollars
Method:
S - property sold;
SP - property sold prior;
PI - property passed in;
PN - sold prior not disclosed;
SN - sold not disclosed;
NB - no bid;
VB - vendor bid;
W - withdrawn prior to auction;
SA - sold after auction;
SS - sold after auction price not disclosed.
N/A - price or highest bid not available.
Type:
br - bedroom(s);
h - house,cottage,villa, semi,terrace;
u - unit, duplex;
t - townhouse;
dev site - development site;
o res - other residential.
SellerG: Real Estate Agent
Date: Date sold
kalamgaru.webp
melbourne.zip
Show all

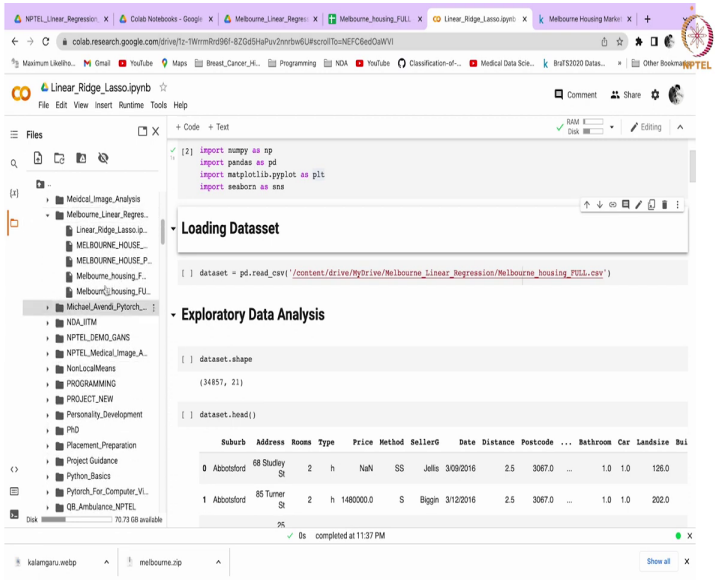

kaggle.com/datasets/anthonypino/melbourne-housing-market?select=MELBOURNE_HOUSE_PRICES_LESS.csv
kaggle
Search
Create
Home
Competitions
Datasets
Code
Discussions
Courses
More
Your Work
RECENTLY VIEWED
Melbourne Housing ...
Chest X-Ray Images ...
BraTS2020 Dataset ...
3D MRI Brain tumor ...
View Active Events
Melbourne Housing Market
Data
Code (296)
Discussion (26)
Metadata
639
New Notebook
Download (2 MB)
Some Key Details
Suburb: Suburb
Address: Address
Rooms: Number of rooms
Price: Price in Australian dollars
Method:
S - property sold;
SP - property sold prior;
PI - property passed in;
PN - sold prior not disclosed;
SN - sold not disclosed;
NB - no bid;
VB - vendor bid;
W - withdrawn prior to auction;
SA - sold after auction;
SS - sold after auction price not disclosed.
N/A - price or highest bid not available.
Type:
br - bedroom(s);
h - house,cottage,villa, semi,terrace;
u - unit, duplex;
t - townhouse;
kalamgaru.webp
melbourne.zip
Show all

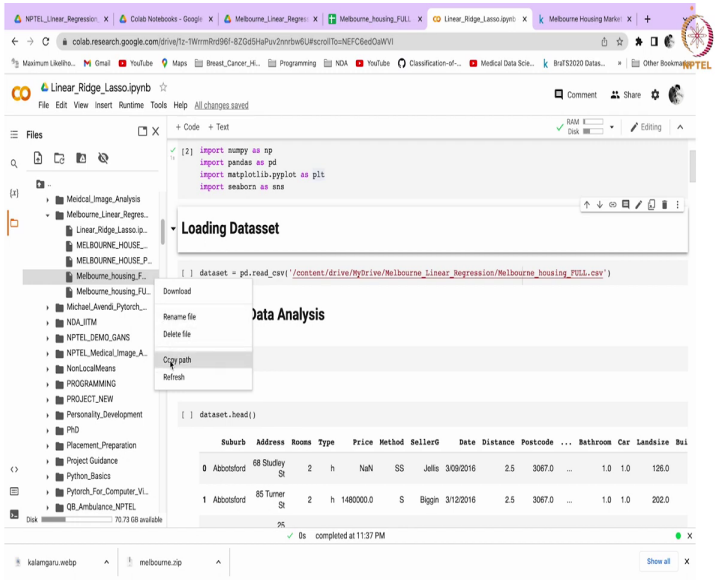


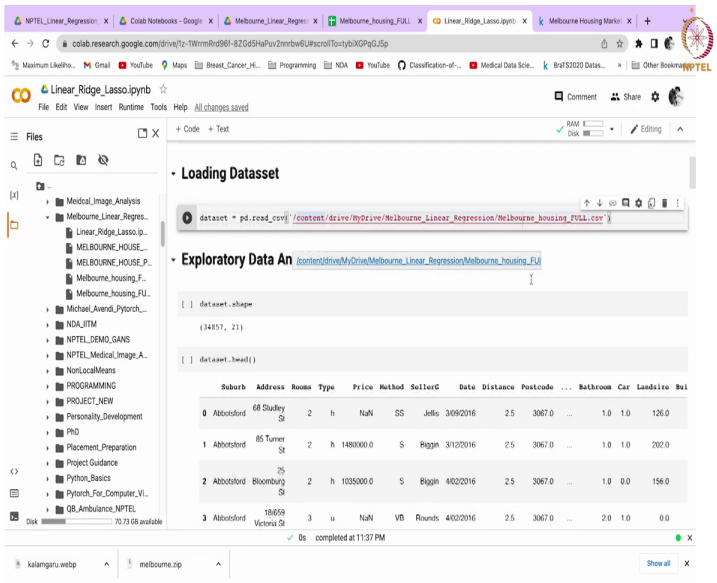
If you want you can further explore here. Each and every detail has been given, what exactly each data point, each column will describe. Method, Type, SellerG, Date sold, all the, each and every variable, particular description has been given.

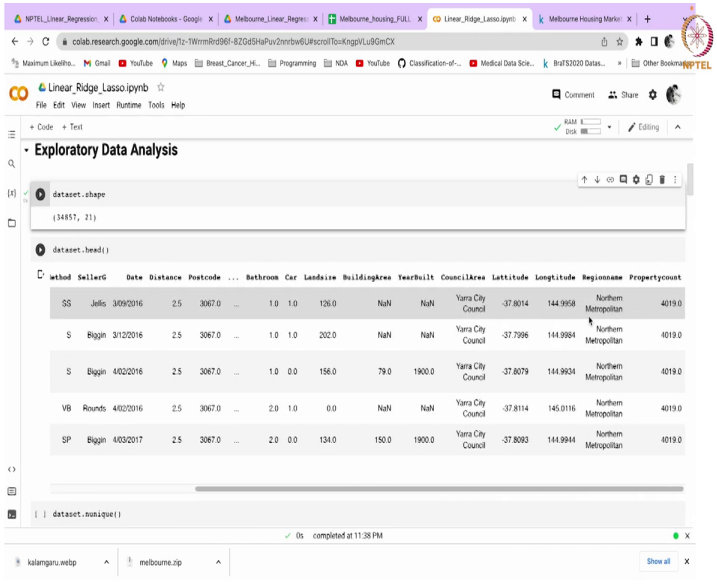
(Refer Slide Time: 49:56)

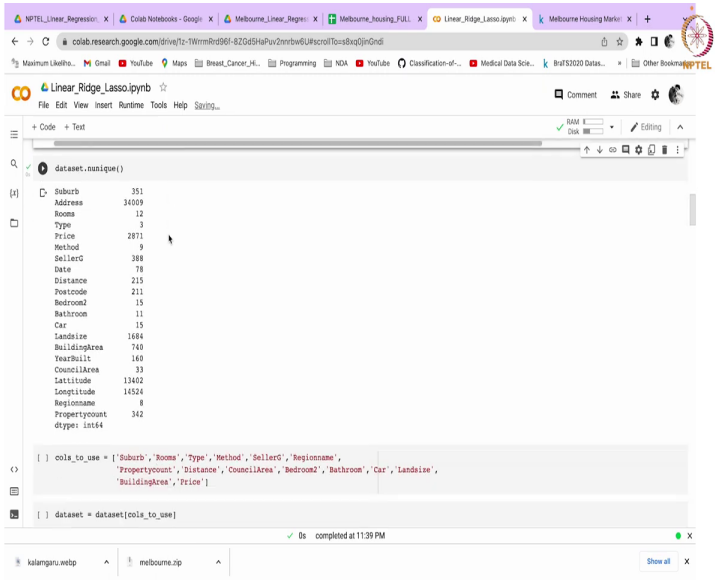


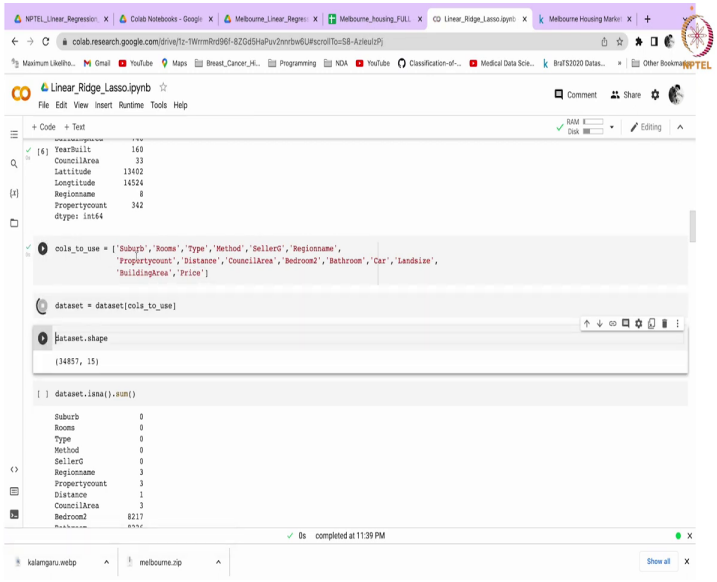

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Editing
Files
+ Code + Text
[2] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
Loading Datasset
[ ] dataset = pd.read_csv('/content/drive/MyDrive/Melbourne_Linear_Regression/Melbourne_housing_FULL.csv')
Exploratory Data Analysis
[ ] dataset.shape
(34857, 21)
[ ] dataset.head()
Suburb Address Rooms Type Price Method SellerG Date Distance Postcode ... Bathroom Car Landsize Bui
0 Abbotsford 68 Studley St 2 h NaN SS Jellis 3/09/2016 2.5 3067.0 ... 1.0 1.0 126.0
1 Abbotsford 85 Turner St 2 h 1480000.0 S Biggin 3/12/2016 2.5 3067.0 ... 1.0 1.0 202.0
0s completed at 11:37 PM
kalamgaru.webp
melbourne.zip
Show all

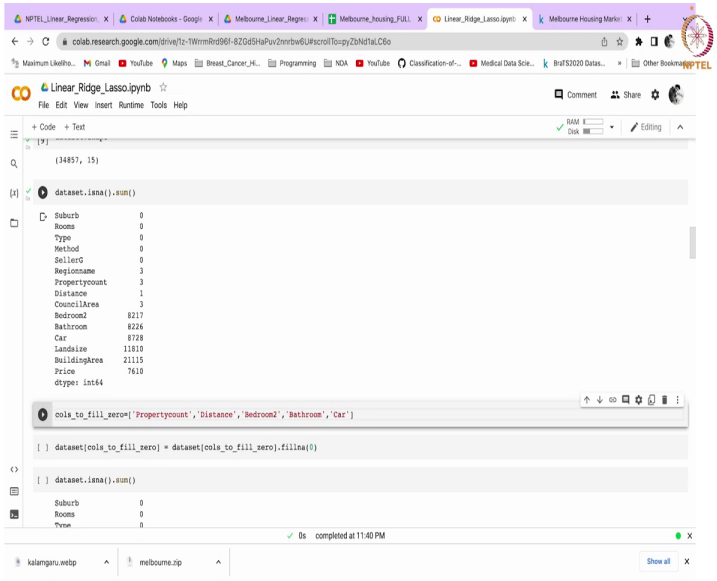

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Editing
Files
+ Code + Text
[2] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
Loading Datasset
[ ] dataset = pd.read_csv('/content/drive/MyDrive/Melbourne_Linear_Regression/Melbourne_housing_FULL.csv')
Download
Rename file
Delete file
Copy path
Refresh
[ ] dataset.head()
Suburb Address Rooms Type Price Method SellerG Date Distance Postcode ... Bathroom Car Landsize Bui
0 Abbotsford 68 Studley St 2 h NaN SS Jellis 3/09/2016 2.5 3067.0 ... 1.0 1.0 126.0
1 Abbotsford 85 Turner St 2 h 1480000.0 S Biggin 3/12/2016 2.5 3067.0 ... 1.0 1.0 202.0
0s completed at 11:37 PM
kalamgaru.webp
melbourne.zip
Show all

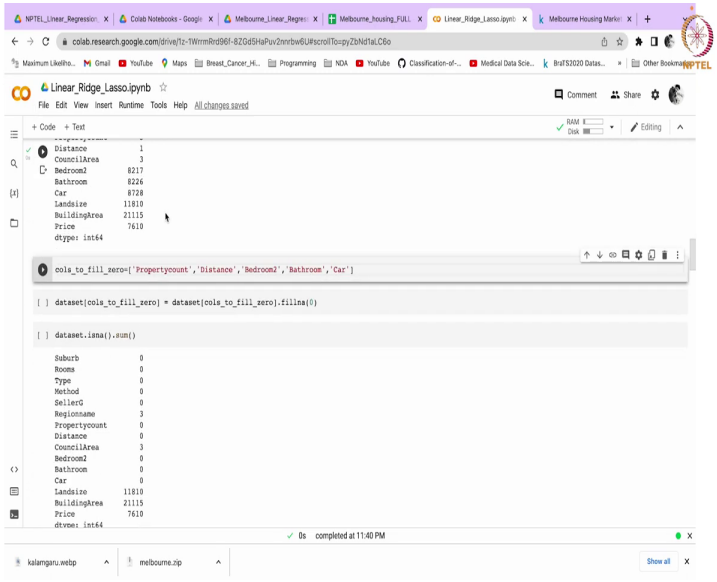

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Editing
Files
Loading Datasset
dataset = pd.read_csv('/content/drive/MyDrive/Melbourne_Linear_Regression/Melbourne_housing_FULL.csv')
Exploratory Data An
/content/drive/MyDrive/Melbourne_Linear_Regression/Melbourne_housing_FUl
[ ] dataset.shape
(34857, 21)
[ ] dataset.head()
Suburb Address Rooms Type Price Method SellerG Date Distance Postcode ... Bathroom Car Landsize Bui
0 Abbotsford 68 Studley St 2 h NaN SS Jellis 3/09/2016 2.5 3067.0 ... 1.0 1.0 126.0
1 Abbotsford 85 Turner St 2 h 1480000.0 S Biggin 3/12/2016 2.5 3067.0 ... 1.0 1.0 202.0
2 Abbotsford 25 Bloomburg St 2 h 1035000.0 S Biggin 4/02/2016 2.5 3067.0 ... 1.0 0.0 156.0
3 Abbotsford 18/659 Victoria St 3 u NaN VB Rounds 4/02/2016 2.5 3067.0 ... 2.0 1.0 0.0
0s completed at 11:37 PM
kalamgaru.webp
melbourne.zip
Show all

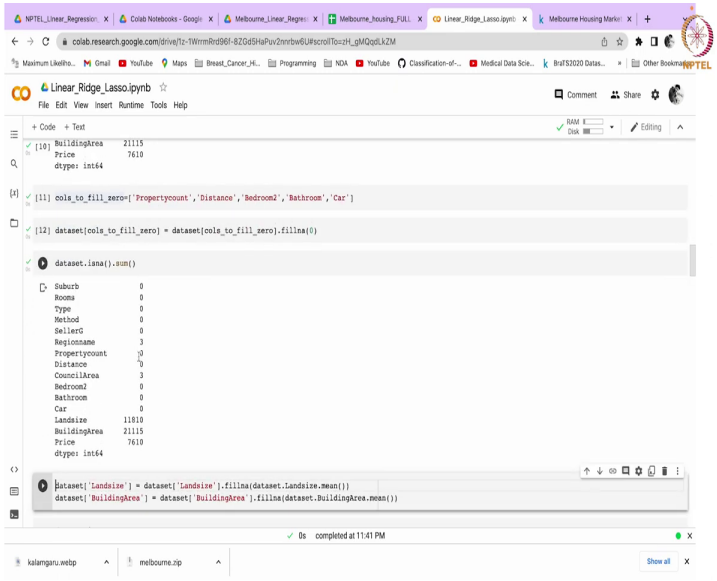

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
Editing
Exploratory Data Analysis
dataset.shape
(34857, 21)
dataset.head()
Method SellerG Date Distance Postcode ... Bathroom Car Landsize BuildingArea YearBuilt CouncilArea Lattitude Longtitude Regionname Propertycount
SS Jellis 3/09/2016 2.5 3067.0 ... 1.0 1.0 126.0 NaN NaN Yarra City Council -37.8014 144.9958 Northern Metropolitan 4019.0
S Biggin 3/12/2016 2.5 3067.0 ... 1.0 1.0 202.0 NaN NaN Yarra City Council -37.7996 144.9984 Northern Metropolitan 4019.0
S Biggin 4/02/2016 2.5 3067.0 ... 1.0 0.0 156.0 79.0 1900.0 Yarra City Council -37.8079 144.9934 Northern Metropolitan 4019.0
VB Rounds 4/02/2016 2.5 3067.0 ... 2.0 1.0 0.0 NaN NaN Yarra City Council -37.8114 145.0116 Northern Metropolitan 4019.0
SP Biggin 4/03/2017 2.5 3067.0 ... 2.0 0.0 134.0 150.0 1900.0 Yarra City Council -37.8093 144.9944 Northern Metropolitan 4019.0
[ ] dataset.nunique()
0s completed at 11:38 PM
kalamgaru.webp
melbourne.zip
Show all

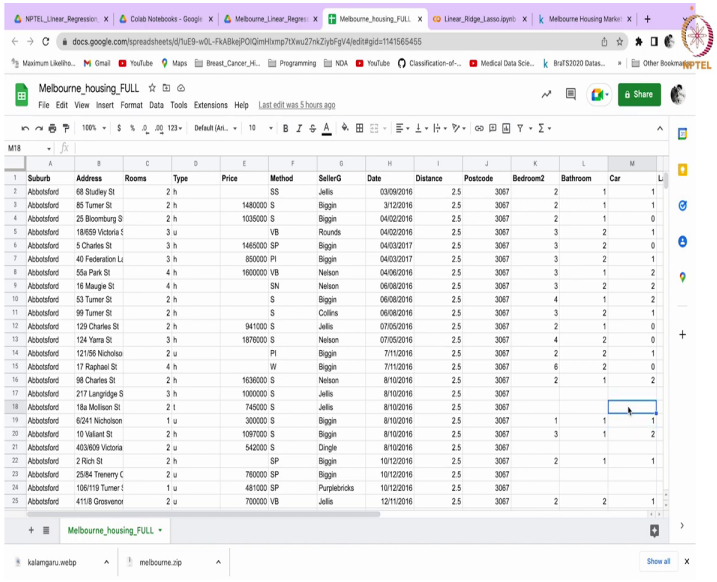

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
RAM
Disk
Editing
dataset.nunique()
Suburb 351
Address 34009
Rooms 12
Type 3
Price 2871
Method 9
SellerG 388
Date 78
Distance 215
Postcode 211
Bedroom2 15
Bathroom 11
Car 15
Landsize 1684
BuildingArea 740
YearBuilt 160
CouncilArea 33
Lattitude 13402
Longtitude 14524
Regionname 8
Propertycount 342
dtype: int64
[ ] cols_to_use = ['Suburb','Rooms','Type','Method','SellerG','Regionname',
'Propertycount','Distance','CouncilArea','Bedroom2','Bathroom','Car','Landsize',
'BuildingArea','Price']
[ ] dataset = dataset[cols_to_use]
0s completed at 11:39 PM
kalamganu.webp
melbourne.zip
Show all

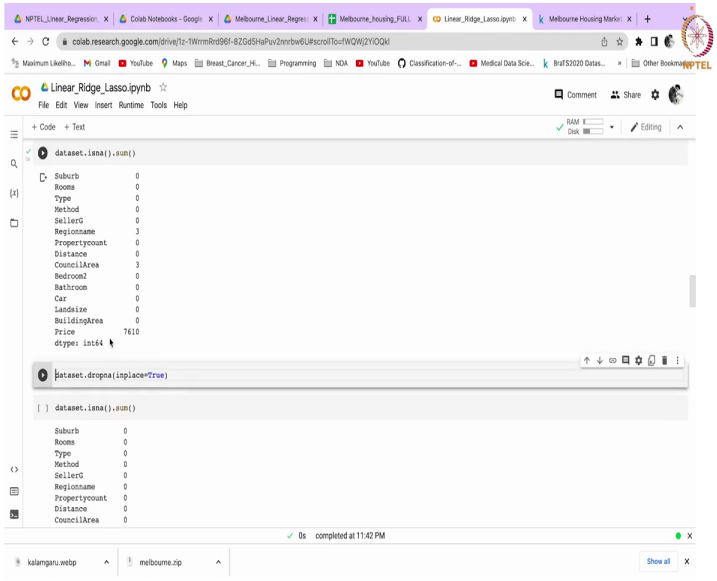

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
[6] YearBuilt 160
CouncilArea 33
Lattitude 13402
Longtitude 14524
Regionname 8
Propertycount 342
dtype: int64
cols_to_use = ['Suburb','Rooms','Type','Method','SellerG','Regionname',
'Propertycount','Distance','CouncilArea','Bedroom2','Bathroom','Car','Landsize',
'BuildingArea','Price']
dataset = dataset[cols_to_use]
dataset.shape
(34857, 15)
[ ] dataset.isna().sum()
Suburb 0
Rooms 0
Type 0
Method 0
SellerG 0
Regionname 3
Propertycount 3
Distance 1
CouncilArea 3
Bedroom2 8217
0s completed at 11:39 PM
kalamganu.webp
melbourne.zip
Show all

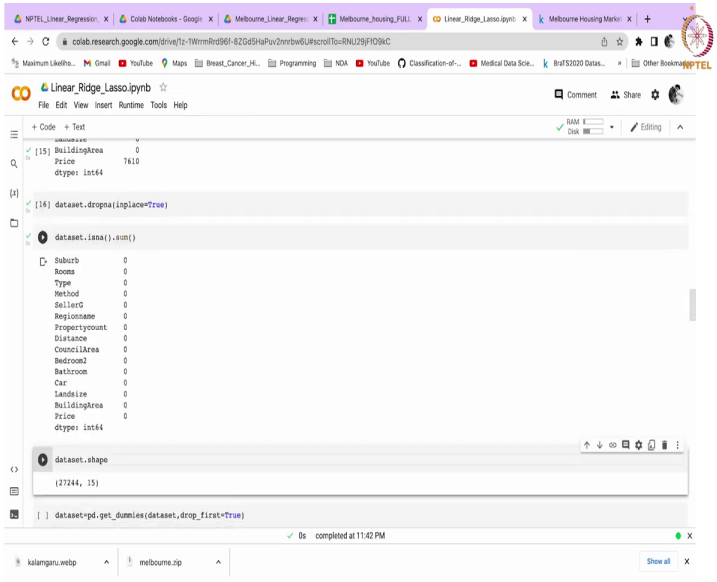

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
(34857, 15)
dataset.isna().sum()
Suburb 0
Rooms 0
Type 0
Method 0
SellerG 0
Regionname 3
Propertycount 3
Distance 1
CouncilArea 3
Bedroom2 8217
Bathroom 8226
Car 8728
Landsize 11810
BuildingArea 21115
Price 7610
dtype: int64
cols_to_fill_zero=['Propertycount','Distance','Bedroom2','Bathroom','Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)
dataset.isna().sum()
Suburb 0
Rooms 0
0s completed at 11:40 PM
kalamganu.webp
melbourne.zip
Show all

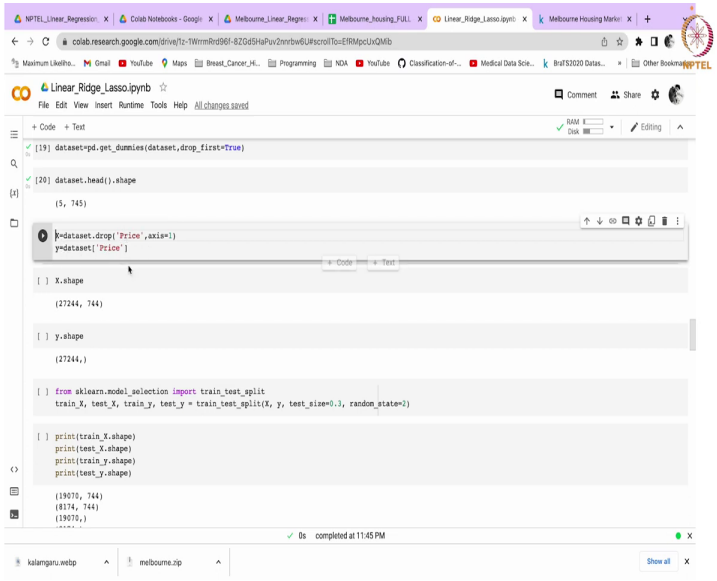

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
Distance 1
CouncilArea 3
Bedroom2 8217
Bathroom 8226
Car 8728
Landsize 11810
BuildingArea 21115
Price 7610
dtype: int64
cols_to_fill_zero=['Propertycount','Distance','Bedroom2','Bathroom','Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)
dataset.isna().sum()
Suburb 0
Rooms 0
Type 0
Method 0
SellerG 0
Regionname 3
Propertycount 0
Distance 0
CouncilArea 3
Bedroom2 0
Bathroom 0
Car 0
Landsize 11810
BuildingArea 21115
Price 7610
0s completed at 11:40 PM
kalamganu.webp
melbourne.zip
Show all

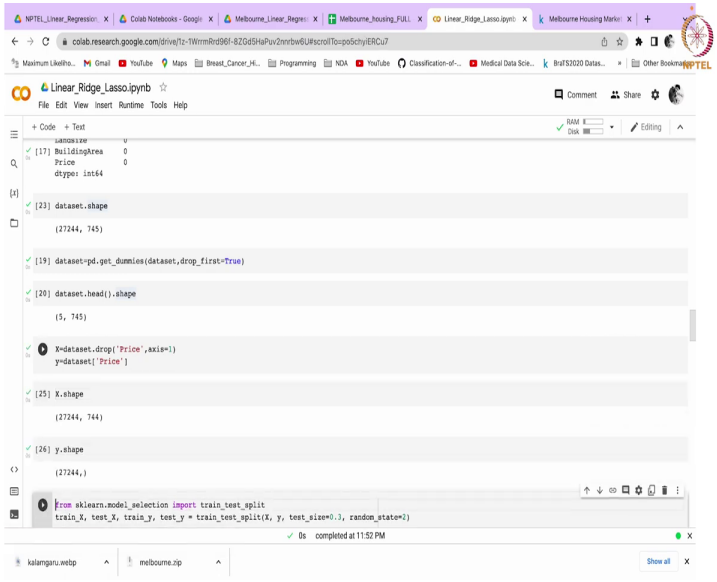

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
[10] BuildingArea 21115
Price 7610
dtype: int64
[11] cols_to_fill_zero=['Propertycount','Distance','Bedroom2','Bathroom','Car']
[12] dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)
dataset.isna().sum()
Suburb 0
Rooms 0
Type 0
Method 0
SellerG 0
Regionname 3
Propertycount 0
Distance 0
CouncilArea 3
Bedroom2 0
Bathroom 0
Car 0
Landsize 11810
BuildingArea 21115
Price 7610
dtype: int64
dataset['Landsize'] = dataset['Landsize'].fillna(dataset.Landsize.mean())
dataset['BuildingArea'] = dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())
0s completed at 11:41 PM
kalamganu.webp
melbourne.zip
Show all

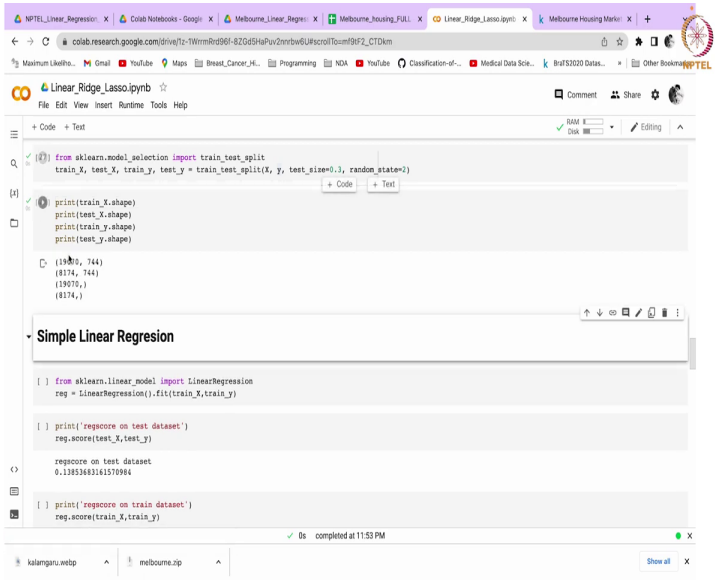

Melbourne_housing_FULL
File Edit View Insert Format Data Tools Extensions Help Last edit was 5 hours ago
Share
Suburb Address Rooms Type Price Method SellerG Date Distance Postcode Bedroom2 Bathroom Car
Abbotsford 68 Studley St 2 h SS Jellis 03/09/2016 2.5 3067 2 1 1
Abbotsford 85 Turner St 2 h 1480000 S Biggin 3/12/2016 2.5 3067 2 1 1
Abbotsford 25 Bloomburg St 2 h 1035000 S Biggin 04/02/2016 2.5 3067 2 1 0
Abbotsford 18/659 Victoria S 3 u VB Rounds 04/02/2016 2.5 3067 3 2 1
Abbotsford 5 Charles St 3 h 1465000 SP Biggin 04/03/2017 2.5 3067 3 2 0
Abbotsford 40 Federaton La 3 h 850000 PI Biggin 04/03/2017 2.5 3067 3 2 1
Abbotsford 55a Park St 4 h 1600000 VB Nelson 04/06/2016 2.5 3067 3 1 2
Abbotsford 16 Maugie St 4 h SN Nelson 06/08/2016 2.5 3067 3 2 2
Abbotsford 53 Turner St 2 h S Biggin 06/08/2016 2.5 3067 4 1 2
Abbotsford 99 Turner St 2 h S Collins 06/08/2016 2.5 3067 3 2 1
Abbotsford 129 Charles St 2 h 941000 S Jellis 07/05/2016 2.5 3067 2 1 0
Abbotsford 124 Yarra St 3 h 1876000 S Nelson 07/05/2016 2.5 3067 4 2 0
Abbotsford 121/56 Nicholso 2 u PI Biggin 7/11/2016 2.5 3067 2 2 1
Abbotsford 17 Raphael St 4 h W Biggin 7/11/2016 2.5 3067 6 2 0
Abbotsford 98 Charles St 2 h 1636000 S Nelson 8/10/2016 2.5 3067 2 1 2
Abbotsford 217 Langridge S 3 h 1000000 S Jellis 8/10/2016 2.5 3067
Abbotsford 18a Mollison St 2 t 745000 S Jellis 8/10/2016 2.5 3067
Abbotsford 6/241 Nicholson 1 u 300000 S Biggin 8/10/2016 2.5 3067 1 1 1
Abbotsford 10 Valiant St 2 h 1097000 S Biggin 8/10/2016 2.5 3067 3 1 2
Abbotsford 403/809 Victoria 2 u 542000 S Dingle 8/10/2016 2.5 3067
Abbotsford 2 Rich St 2 h SP Biggin 10/12/2016 2.5 3067 2 1 1
Abbotsford 25/84 Trenerry C 2 u 760000 SP Biggin 10/12/2016 2.5 3067
Abbotsford 106/119 Turner S 1 u 481000 SP Purplebricks 10/12/2016 2.5 3067
Abbotsford 411/8 Grosvenor 2 u 700000 VB Jellis 12/11/2016 2.5 3067 2 2 1
Melbourne_housing_FULL
kalamganu.webp
melbourne.zip
Show all

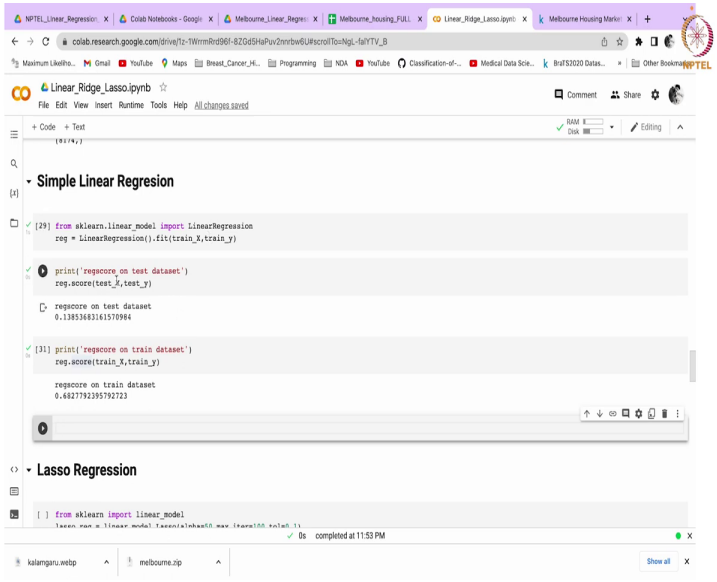

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
dataset.isna().sum()
Suburb 0
Rooms 0
Type 0
Method 0
SellerG 0
Regionname 3
Propertycount 0
Distance 0
CouncilArea 3
Bedroom2 0
Bathroom 0
Car 0
Landsize 0
BuildingArea 0
Price 7610
dtype: int64
dataset.dropna(inplace=True)
[ ] dataset.isna().sum()
Suburb 0
Rooms 0
Type 0
Method 0
SellerG 0
Regionname 0
Propertycount 0
Distance 0
CouncilArea 0
0s completed at 11:42 PM
kalamganu.webp
melbourne.zip
Show all

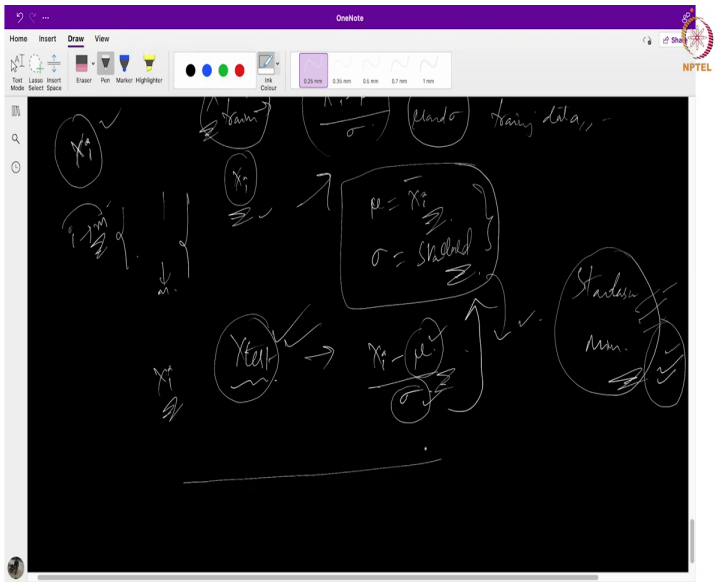

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
[15] BuildingArea 0
Price 7610
dtype: int64
[16] dataset.dropna(inplace=True)
dataset.isna().sum()
Suburb 0
Rooms 0
Type 0
Method 0
SellerG 0
Regionname 0
Propertycount 0
Distance 0
CouncilArea 0
Bedroom2 0
Bathroom 0
Car 0
Landsize 0
BuildingArea 0
Price 0
dtype: int64
dataset.shape
(27244, 15)
[ ] dataset=pd.get_dummies(dataset,drop_first=True)
0s completed at 11:42 PM
kalamganu.webp
melbourne.zip
Show all

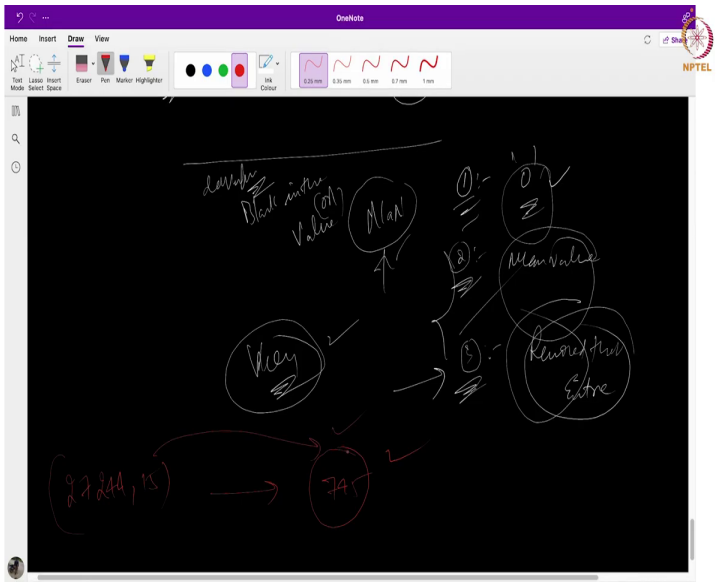

Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[19] dataset=pd.get_dummies(dataset,drop_first=True)
[20] dataset.head().shape
(5, 745)
X=dataset.drop('Price',axis=1)
y=dataset['Price']
[ ] X.shape
(27244, 744)
[ ] y.shape
(27244,)
[ ] from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=2)
[ ] print(train_X.shape)
print(test_X.shape)
print(train_y.shape)
print(test_y.shape)
(19070, 744)
(8174, 744)
(19070,)
0s completed at 11:45 PM
kalamganu.webp
melbourne.zip
Show all


Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
[17] BuildingArea 0
Price 0
dtype: int64
[23] dataset.shape
(27244, 745)
[19] dataset=pd.get_dummies(dataset,drop_first=True)
[20] dataset.head().shape
(5, 745)
X=dataset.drop('Price',axis=1)
y=dataset['Price']
[25] X.shape
(27244, 744)
[26] y.shape
(27244,)
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=2)
0s completed at 11:52 PM
kalamganu.webp
melbourne.zip
Show all


Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=2)
print(train_X.shape)
print(test_X.shape)
print(train_y.shape)
print(test_y.shape)
(19070, 744)
(8174, 744)
(19070,)
(8174,)
Simple Linear Regresion
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(train_X,train_y)
print('regscore on test dataset')
reg.score(test_X,test_y)
regscore on test dataset
0.13853683161570984
print('regscore on train dataset')
reg.score(train_X,train_y)
0s completed at 11:53 PM
kalamganu.webp
melbourne.zip
Show all


Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
Simple Linear Regresion
[29] from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(train_X,train_y)
print('regscore on test dataset')
reg.score(test_X,test_y)
regscore on test dataset
0.13853683161570984
[31] print('regscore on train dataset')
reg.score(train_X,train_y)
regscore on train dataset
0.6827792395792723
Lasso Regression
from sklearn import linear_model
0s completed at 11:53 PM
kalamganu.webp
melbourne.zip
Show all





After mounting the trained, I am just importing numPy, pandas, matplotlib and Seaburn libraries here. Just refresh here, after mounting your train. See, I am taking the path from Melbourne linear regression data. Melbourne, see here, from this folder, I am taking my data set, and the CSV file, which I am taking is from that particular full dot CSV, just to go to that particular file, wherever the particular file is there, Google Sheets, this one, not this one.

See here, just to go to this particular file and click here. You can see the copy path. After updating this particular copy path, just paste here. That is how you will give the data. Just

let me do path. So, my data set has been taken into your pandas data frame. Like pd.read_CSV.

Now, coming back to exploratory data analysis. I am just checking out the shape, like we, that means, I have 34,857 rows and 21 features, I have, not features, even along with the last variable, like price, we are talking about here, this one, this particular column is our target variable, remaining all the columns are features.

That means among those 21 variables, we just have to focus on only 20 variables, all the independent variables are features, and only one is target variable. Just, I am printing down what are the head files, like first five or four rows, I am checking out. This is the basic function people use.

And now, just check it. And this dataset.nunique is for checking out what each individual, suppose if you take the rooms, rooms you can see 12, that means, 12 unique numbers are there. Like, threes are, some, 3 is one number, unique number, 2 is one unique number. Likewise, we will have some 12 unique room numbers, rooms.

So, this 21 column, instead of taking all these 21 columns, I just have focused on these columns. So, let me check out from the data frame, I am just filtering out only these two, these number of columns, like 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, total 15 columns I am using, just let me check the shape here.

See here, I have only 34,857 rows and 15 columns. Earlier, the column number is 21, now the column number is 15. Now, check this function. What it indicates is, maybe in the given data set, you will be having some blanks, not filling any data, where people usually call them as nan values.

You have to deal in the data pre-processing stage or exploratory data analysis stage, you have to focus more on those details. This particular blank value can be replaced with zeros or even with the mean value of the remaining values whichever present to that particular variable or in another sense, you can just eliminate that particular row, even that is also accepted.

Because while feeding the deep, machine learning models, those models cannot accept the data that is having these nan values. So, please keep this point in your mind. So, dataset.isna.sum. Now, come to see here, for proper account, property count, distance, property count, we have three nan values, distance, one, and bedroom bathroom.

For all these four columns, I am just focusing to fill those columns with the zeros. I mean wherever those blank values are there, nan values are there, you just replace all those values with the zeros. See here. Suppose, if you go observe property count, you can see here, three blanks are there. Now here see, zero it became.

Similarly, take the distance, distance has only one. That even that 1 also becomes 0. Now, bathroom, car deck. See the car, 8728, car value now becomes 0. That means there is no blank values. What exactly blank values, I will show you. See here. Suppose, if you observe the price, there is no value here, there is no value here, there is no value here.

All these blanks see here, bedroom too, in this particular sheet, no value, no value, no value, no value, no value, no value. No value does not mean a zero value. Now, coming back to landsize and building area. See here, only these two are left, now landsize and building area. For those columns, I am just replacing with the mean values of the remaining values.

Take the land size, wherever those blanks are there, I am just filling those values with the mean of the remaining values, that are given in the particular land size variable. Just let me run this one. Now, let me run this one, some value. See here, only region name and council areas are left. Still, I have to record something for them. I am using just dropping out, the, those columns. So, those complete rows which are having same.

See here, now this function indicates, in the entire data set, whatever you have, like suppose, suppose you have a entire Excel sheet, you see, further now, whatever data I have prepared now, for all the, entire data wherever you see, you cannot find these kind of blanks anywhere in this particular data set, which I am going to fit to the model. So, keep that point in mind.

Here, I have all, used all the strategies like to fill the nan value, value, or a blank value or a blank in the data frame. I just, for a few columns, I have filled with zero values, for a few columns, I have filled with the mean value, for few rows, for few columns, wherever the nan are present, I just have removed that entire row, removed that entire row.

You can go for this one. Suppose those number of blanks are very less. Then only you can go for this one, because with very high efforts, we are collecting the data. So, the entire deep learning, machine learning algorithms, they are very much interested, they are very much trained well, if you have very good data. So, you do not try to lose even a single data point. So, always try to use every data point whichever you have. But for the explanation sake, this is how we can do several pre-processing tasks. I hope everybody got it.

Now coming back to, observe the shape here. 27,244 rows. Earlier, the row numbers are like 34,857. Because at the end, we have pricing area, pricing column, where it has 7,610 blanks. I have removed all those rows, whichever does not contain price. Now, coming back to, usually if you observe in the columns, like the suburb, you can see, usually deep learning models, or machine learning models, they can understand only, in numbers only.

But here, you can see there are strings or categorical variable. I have to convert all these categorical variables into one hot encoder, one hot encoded variables. To do that one hot encoding, just go and browse, what exactly one hot encoding means. To do that one hot encoding, you will have a function, something like dummies in pandas, that will help you to remove that categorical variable into convert this, converting, converting whatever variables into one hard encoded variables.

Just run this command so that everything will be easier. See here, I have already told you price is nothing but my point of interest, I mean this target variable. I am just separating out x variables and, sorry, x_data and y_data, I mean features and target, or input variables and output variable. Now coming to extract the shape. See here, 744, observe carefully.

Earlier, the shape is 27, earlier the shape is actually (27244,15). Observe here, after running this one here, and that is why this problem occurred, earlier we have (27244,15).

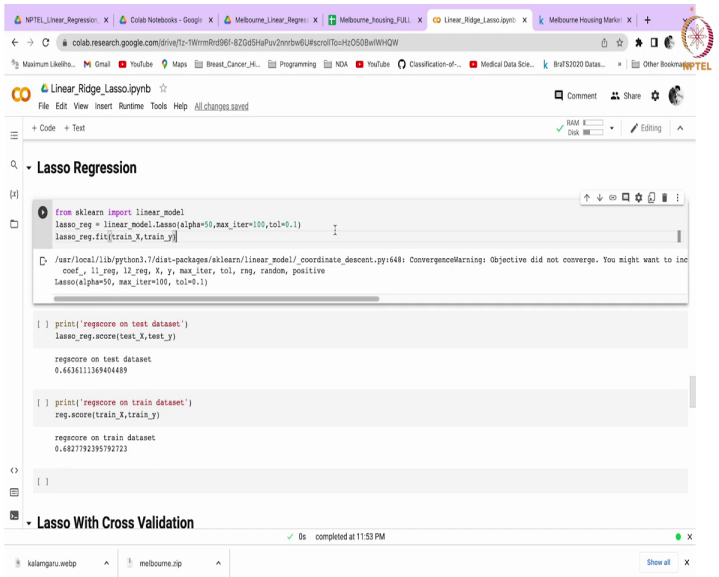
After using the dummy function, we got 745. So, keep that in mind. I mean those 15 features has been now converted into one hot encoded 745. That is what this value, dataset. shape.

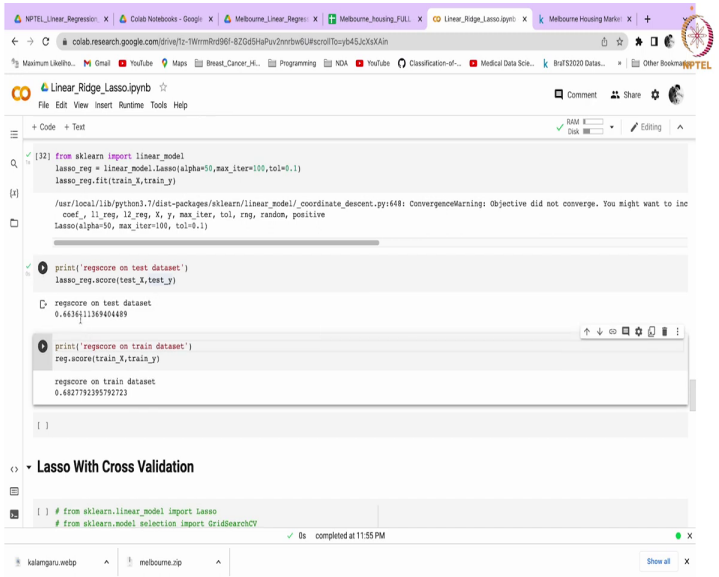
This one, 744, because we have separated out the price variable 2,744. As usually, now we have separated all the input variables and output variables. Now coming back to sklearn.model_selection import train test split, train_x,test_x, you can you usually know just x variable and y variable given, and see here, 0.3 is the, that means we are just converting the entire train data into, 70% of training data into trained data and 30% into test data.

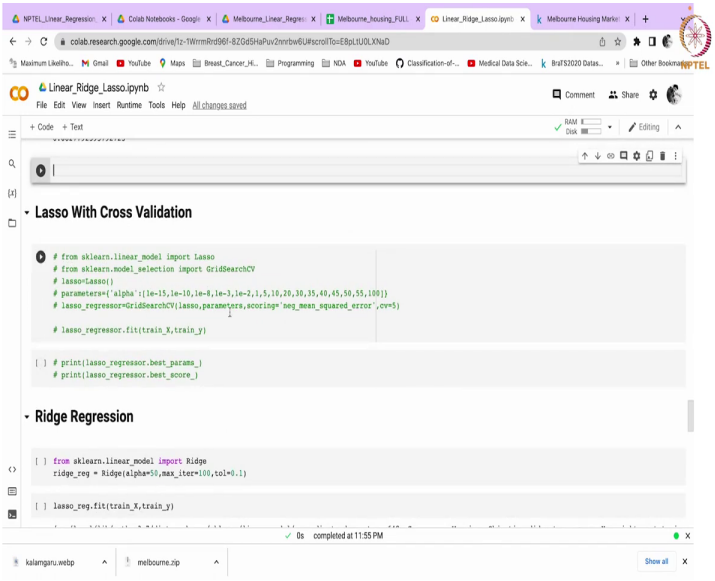
Observe the shapes, this is how. x_train, x_test, y_train y_test. Now, we just first implement the simple linear regression model to import that, just import this linear regression, linear regression filter, train_x , train_y. After this one, obtain the regression score or another r2 score. This is the turn test data coming back to, see here. This is, in this particular problem you can observe that after training this particular model, when I am checking out my test data values on this one, the score for the test that I got is 0.138 or 13 percentage.

Whereas the, in train for the train data, I have training error that particular score is 68 percent or 0.68. That means, you can see a huge difference between 68 and 13 percentage. That means training, here, r score indicates more or less accuracy. That means in terms of loss we can say that training error is less but the test error is more, that is why we call as overfitting problem.

(Refer Slide Time: 1:02:05)





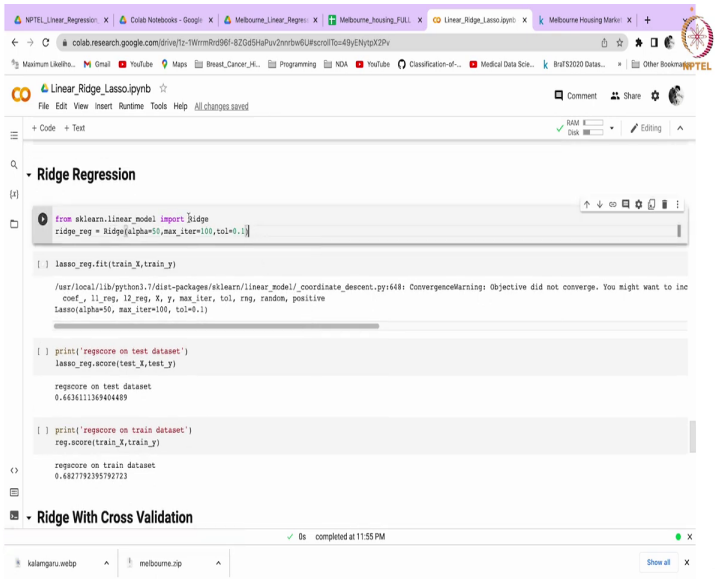


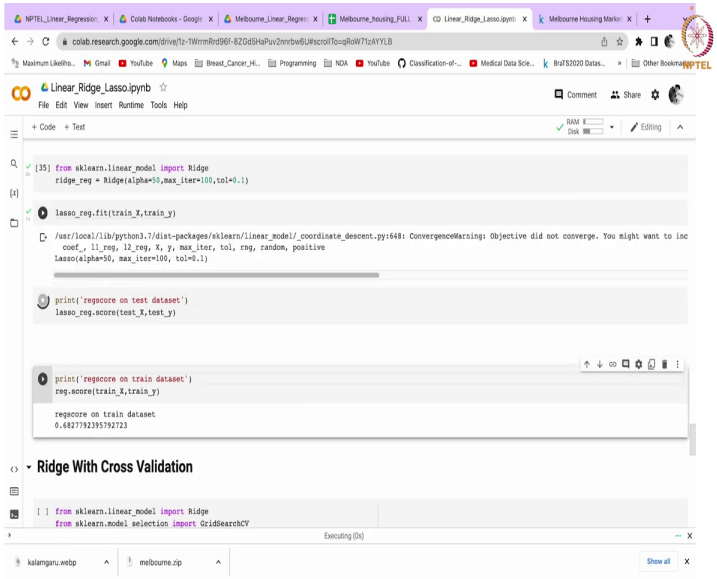
So, to deal this particular problem, we go for lasso regression or ridge regression. If you observe the lasso regression, see here, lasso, α is equal to 50, maximum iteration is like 100, tol is nothing but the tolerance level, 0.1. This is just functions only. You just have to learn how to pass this train x and train y into this one. α is equal to 50 indicates here that regularization parameter.

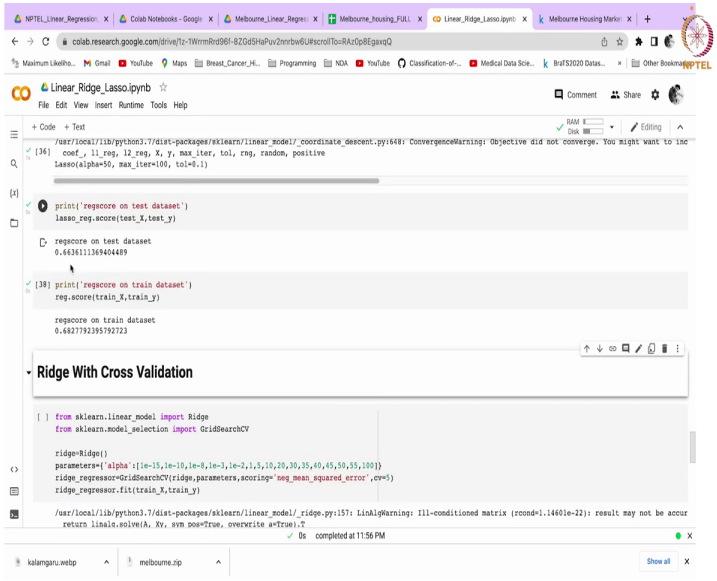
Suppose, let me train this model using lasso regression. See here, here also, I am not using any cross validation here. I am just using this α values, like regulation parameter only, I have used. I did not use any validation data here. Now, coming back to regression score, see here, on test data we have 0.6636, whereas on train data, we have 0.68. You can see, very minute difference is there. That means that particular overfitting problem has been dealt by the lasso regression. This is the basic difference, one should know.

To improve this model further, I have used with the cross validation, but it is taking more time to run the program. Once I explain this ridge regression and ridge cross validation, later you can observe this course. At the end, please try this one so that there is no complication.

(Refer Slide Time: 1:03:18)



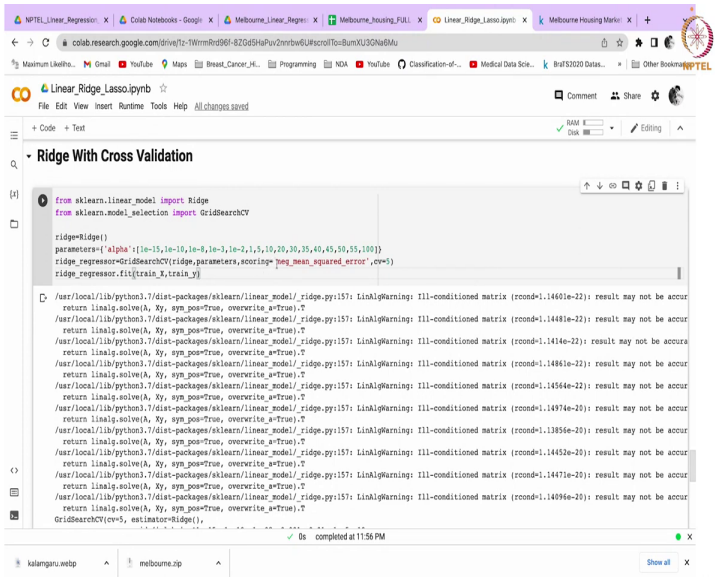


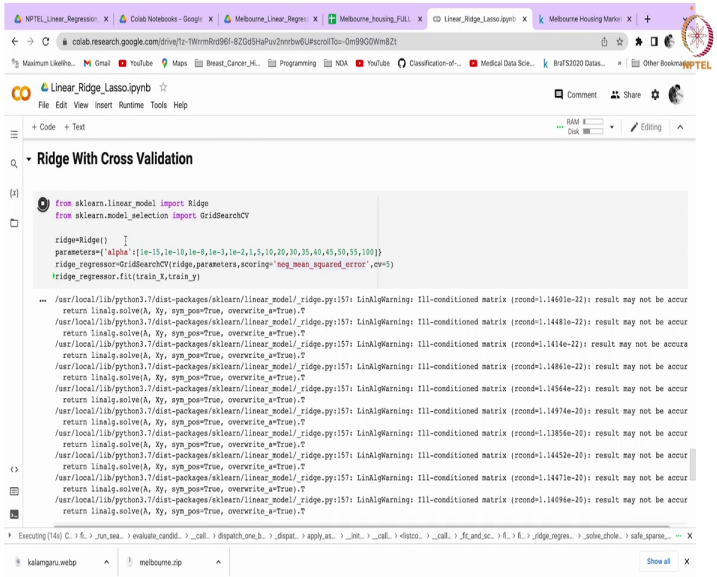


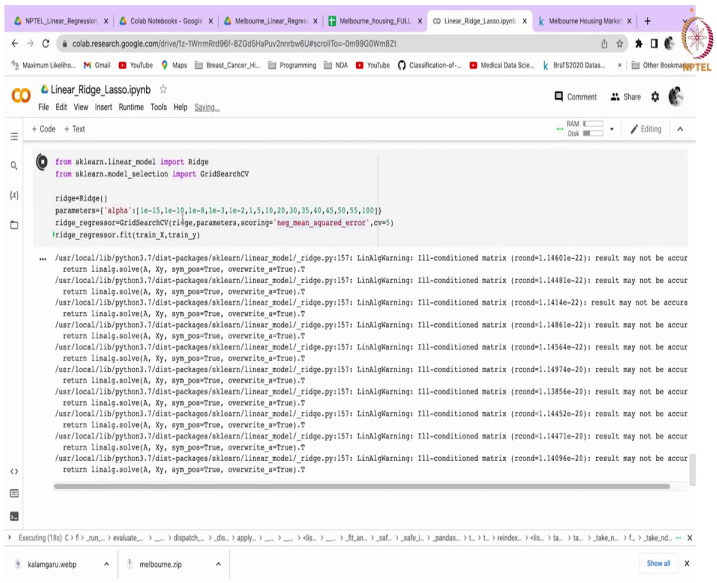
Now, coming back to ridge regression. Here also, this function, import this ridge regression function, α is equal to 50, max_iterations is equal to 100, tolerance level is equal to 0.1. That is it. Let me run this one.

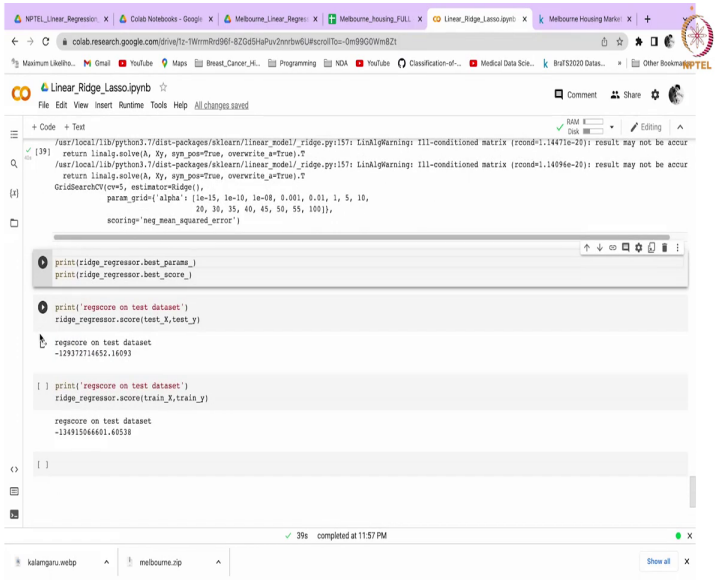
Next, lasso underscore regression dot fit of train_x , train_y. Let me check the regression score here. Here, this particular score, here that particular score indicates the r2 score only, or R^2 square value. See here, 0.66, 0.68, very minute difference is there, not much. Earlier, if you observe the simple linear regression, 13 percentage and 68 percentage, see how much difference is there. So, that is how you can dealt.

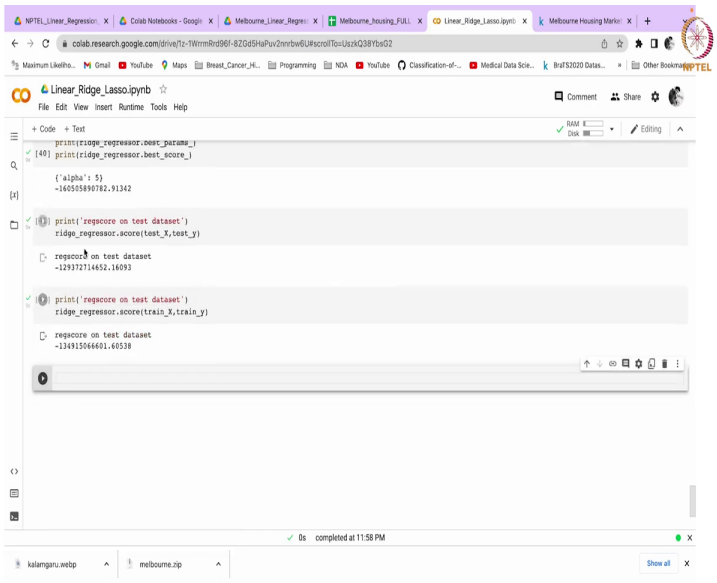
(Refer Slide Time: 1:04:21)






Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
RAM
Disk
Editing
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
ridge=Ridge()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
ridge_regressor.fit(train_X,train_y)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14601e-22): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14481e-22): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.1414e-22): result may not be accura
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14861e-22): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14564e-22): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14974e-20): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.13856e-20): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14452e-20): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14471e-20): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14096e-20): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
kalamgaru.webp
melbourne.zip
Show all


Linear_Ridge_Lasso.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14471e-20): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill-conditioned matrix (rcond=1.14096e-20): result may not be accur
return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
GridSearchCV(cv=5, estimator=Ridge(),
param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10,
20, 30, 35, 40, 45, 50, 55, 100]},
scoring='neg_mean_squared_error')
print(ridge_regressor.best_params_)
print(ridge_regressor.best_score_)
print('regscore on test dataset')
ridge_regressor.score(test_X,test_y)
regscore on test dataset
-129372714652.16093
print('regscore on test dataset')
ridge_regressor.score(train_X,train_y)
regscore on test dataset
-1349150666601.60538
39s completed at 11:57 PM
kalamgaru.webp
melbourne.zip
Show all



Till now, we did not use, to tune this hyper parameters, we did not use any cross validation, sorry, validation data till now here. Now, if you observe this ridge regression with the cross validation data, till now, I have dealt whatever I told you, simple linear regression and ridge regression and lasso regression, all are with, all are with only, without validation data.

Now ridge regression, we are dealing with the validation data. The score or the metric, I am observing here is negative mean squared error, observe here. This is not a r2 score value, please keep that point in mind. Now let me run this on ridge regression. Just learn how to use whichever algorithms, well, that is more than enough. See, it is taking some time. See, here, I am trying different α values, 1, 2, 3, 4, 5, 6, 7, 8, like this, from 10 to the power of minus 50 into 100.

Still, it is going. Our ultimate thing is to train error and test error or train whatever metric you are going to use, those metrics should be almost close to each other for the training and testing. Then only we can say that they are going to do as a, okay, here, I am observing the negative mean squared error, please keep that in mind, not R^2 value. Now, I am printing down the best parameters and matrix.

See here, for the CV is equal to 5, I have used that means I am dividing the entire thing into some cross validation of 5. Here is, that is the parameter. That means instead of using the entire train data, whatever I have given as a 70% data, among the data, few points will

be act as a validation data also here, please keep that in mind. That is where the cross validation part will come.

Now, use the regression score on this one, and regression score on this one. These are negative mean squared error only, not the, that one. See, observe both the values, you can see that both are almost close to each other, like 12 and 13. Maybe the value is large. That is how we will tackle the regression problems whenever there is any overfitting problem, we encounter with the either ridge regression or lasso regression.

You just observe, with, by removing these comments, whatever I have sent into that particular folder, data, just print this one, these two things also. GridSearchCV indicates that cross validation part only. Please go through this one. This is the end of the tutorial. Thank you so much.