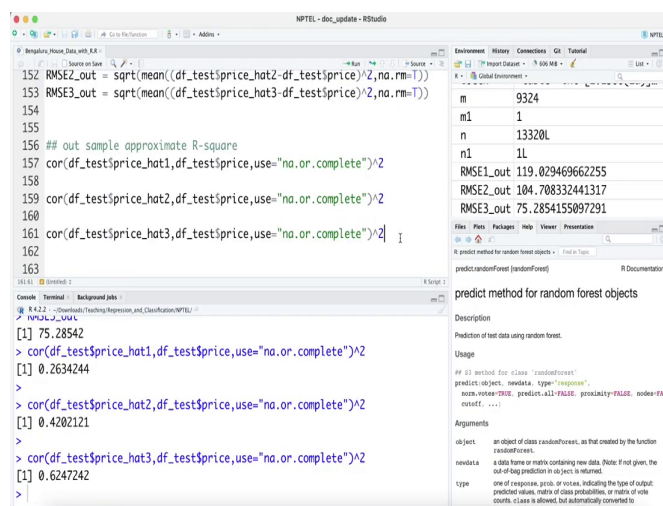


Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Chennai Mathematical Institute

Lecture - 56
Hands on with R : More Prediction of Bangalore House Price

Hello all. In this video, we are going to continue on the Bangalore House Price Data Analysis using R. So, in this particular video, we will whatever we have done in along with that, we will do few more models, and trying to understand some of the way we split the dataset is actually called a static random stratified sampling strategy and we will try to understand what is it doing.

(Refer Slide Time: 00:56)



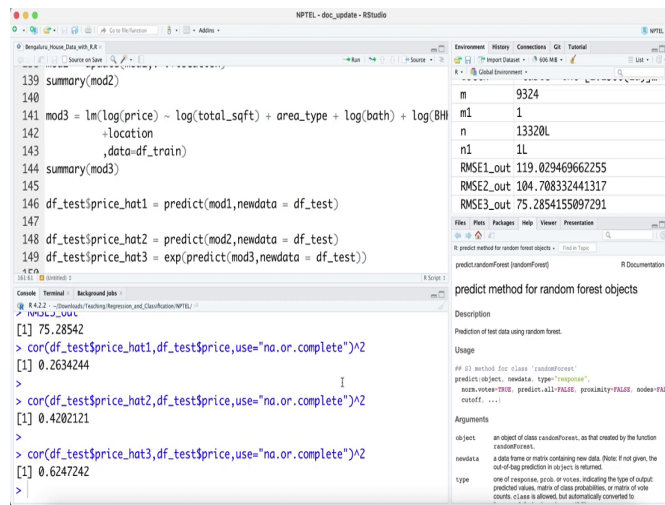
```
152 RMSE2_out = sqrt(mean((df_test$price_hat2-df_test$price)^2,na.rm=T))
153 RMSE3_out = sqrt(mean((df_test$price_hat3-df_test$price)^2,na.rm=T))
154
155
156 ## out sample approximate R-square
157 cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
158
159 cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
160
161 cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
[1] 75.28542
> cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
>
> cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
>
> cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
>
```

m	9324
m1	1
n	13320L
n1	1L
RMSE1_out	119.029469662255
RMSE2_out	104.708332441317
RMSE3_out	75.2854155097291



(Refer Slide Time: 00:58)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains R code for building and evaluating a random forest model. The console shows the output of the code, including the correlation of predicted values with actual values. The environment pane on the right shows the objects in the global environment, including the fitted model and test data.

```
139 summary(mod2)
140
141 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BH
142 +location
143 ,data=df_train)
144 summary(mod3)
145
146 df_test$price_hat1 = predict(mod1,newdata = df_test)
147
148 df_test$price_hat2 = predict(mod2,newdata = df_test)
149 df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
```

Console output:

```
[1] 75.28542
> cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
>
> cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
>
> cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
>
```

Environment pane:

Object	Value
m	9324
m1	1
n	13320L
n1	1L
RMSE1_out	119.029469662255
RMSE2_out	104.708332441317
RMSE3_out	75.2854155097291

predict method for random forest objects

Description

Prediction of test data using random forest.

Usage

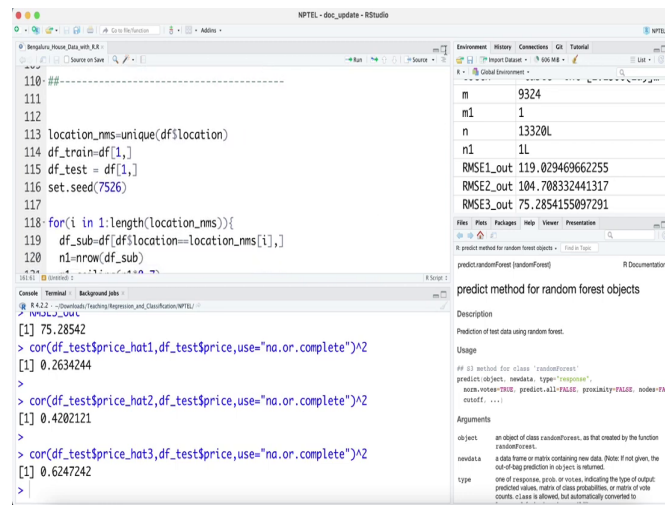
```
# R 61 method for class 'randomForest'
predict(object, newdata, type="response",
        node votes="TRUE", predict.all=FALSE, proximity=FALSE,
        outof1, ...)
```

Arguments

- object: an object of class "randomForest", as that created by the function randomForest.
- newdata: a data frame or matrix containing new data. (Note: If not given, the out-of-bag prediction in object is returned.)
- type: one of response, prob, or cvtree, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. ci=TRUE is allowed, but automatically converted to



(Refer Slide Time: 01:05)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The editor pane contains R code for splitting data by location and calculating correlations. The console shows the output of these commands. The Environment pane on the right displays the objects created, including 'm', 'm1', 'n', 'n1', 'RMSE1_out', 'RMSE2_out', and 'RMSE3_out'. The Help pane on the right shows the documentation for the 'predict' method for random forest objects.

```
110 ##-----
111
112 location_rms=unique(df$location)
113 df_train=df[1,]
114 df_test = df[1,]
115 set.seed(7526)
116
117 for(i in 1:length(location_rms)){
118   df_sub=df[df$location==location_rms[i],]
119   n1=nrow(df_sub)
120
121   [1] 75.28542
122   > cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
123   [1] 0.2634244
124   >
125   > cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
126   [1] 0.4202121
127   >
128   > cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
129   [1] 0.6247242
130   >
```

Environment: Global Environment
m 9324
m1 1
n 13320L
n1 1L
RMSE1_out 119.02946962255
RMSE2_out 104.708332441317
RMSE3_out 75.2854155097291

predict method for random forest objects
predict.randomForest (randomForest)

Description
Prediction of test data using random forest.

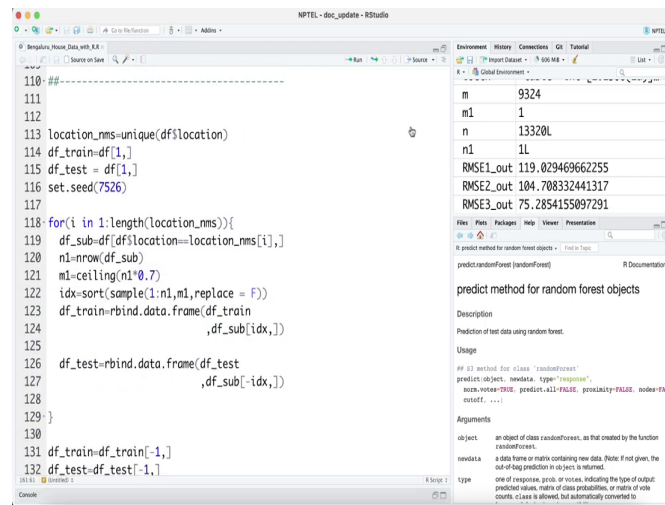
Usage
if all method for class 'randomForest'
predict(object, newdata, type="response",
 node.predict.all=FALSE, probability=FALSE, nodeid=FALSE,
 output = ...)

Arguments
object an object of class 'randomForest', as that created by the function randomForest.
newdata a data frame or matrix containing new data. (Note: If not given, the out-of-bag prediction in object is returned.)
type one of response, probs, or cvtree, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. (Values is allowed, but automatically converted to)



So, here in the we can let me first talk about the strategy of splitting the data by location.

(Refer Slide Time: 01:06)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor on the left contains R code for data partitioning and model evaluation. The Environment pane on the right shows the variables created. The console at the bottom shows the execution of the code.

```
110 #####
111
112 location_rms=unique(df$location)
113 df_train=df[1,]
114 df_test = df[1,]
115 set.seed(7526)
116
117 for(i in 1:length(location_rms)){
118   df_sub=df[df$location==location_rms[i],]
119   n1=nrow(df_sub)
120   m1=ceiling(n1*0.7)
121   idx=sort(sample(1:n1,m1,replace = F))
122   df_train=rbind.data.frame(df_train
123                             ,df_sub[idx,])
124
125   df_test=rbind.data.frame(df_test
126                             ,df_sub[-idx,])
127 }
128
129 }
130
131 df_train=df_train[-1,]
132 df_test=df_test[-1,]
```

The Environment pane shows the following variables:

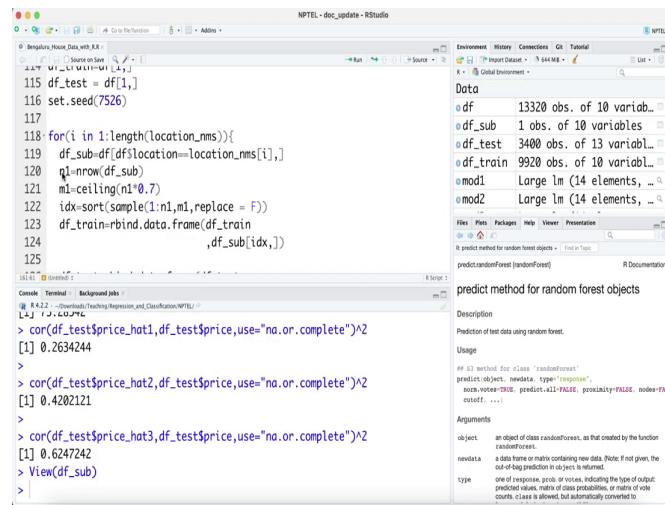
Variable	Value
m	9324
m1	1
n	13320L
n1	1L
RMSE1_out	119.02946962255
RMSE2_out	104.708332441317
RMSE3_out	75.2854155097291

The console shows the execution of the code, with the final output being the RMSE values for the three models.



So, here you remember that we put the we call the locations, unique locations and put them in location names. Then, we just created a space holder for train and test, I set the seed and then for each location, we created a sub dataset here.

(Refer Slide Time: 01:31)



```
115 df_test = df[,]
116 set.seed(7526)
117
118 for(i in 1:length(location_nms)){
119   df_sub=df[df$location==location_nms[i],]
120   n1=nrow(df_sub)
121   m1=ceiling(n1*0.7)
122   idx=sort(sample(1:n1,m1,replace = F))
123   df_train=rbind.data.frame(df_train
124                             ,df_sub[idx,])
125 }

> cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
>
> cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
>
> cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
> View(df_sub)
```

The RStudio interface shows the following data objects in the Environment pane:

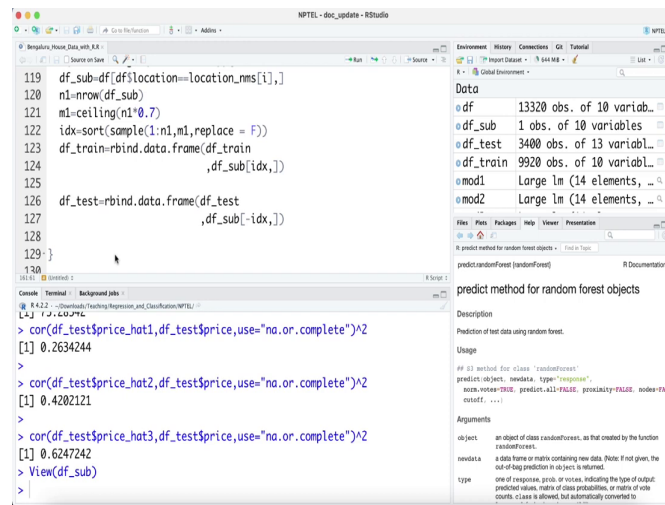
Object	Class	Size
df	data.frame	13320 obs. of 10 variables
df_sub	data.frame	1 obs. of 10 variables
df_test	data.frame	3400 obs. of 13 variables
df_train	data.frame	9920 obs. of 10 variables
mod1	lm	Large lm (14 elements, ...)
mod2	lm	Large lm (14 elements, ...)

The console output shows the correlation analysis results for three different models (price_hat1, price_hat2, price_hat3) against the actual price. The correlation coefficients are 0.2634244, 0.4202121, and 0.6247242 respectively.



And then we took the n1 and we put it in the ceiling, and then we draw the sample and in that sample, we just put it in the idx and then test and then we just append that in the test. So, therefore, each location we created a sub dataset and then for each location, whatever the sample we have, we just split it into 2 and into 30, 70 way into train and test dataset.

(Refer Slide Time: 01:57)



```
119 df_sub=df[df$location==location_nms[i],]
120 n1=nrow(df_sub)
121 m1=ceiling(n1*0.7)
122 idx=sort(sample(1:n1,m1,replace = F))
123 df_train=rbind.data.frame(df_train
124                           ,df_sub[idx,])
125
126 df_test=rbind.data.frame(df_test
127                          ,df_sub[-idx,])
128 }
129 }
130 }

> cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
>
> cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
>
> cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
> View(df_sub)
```

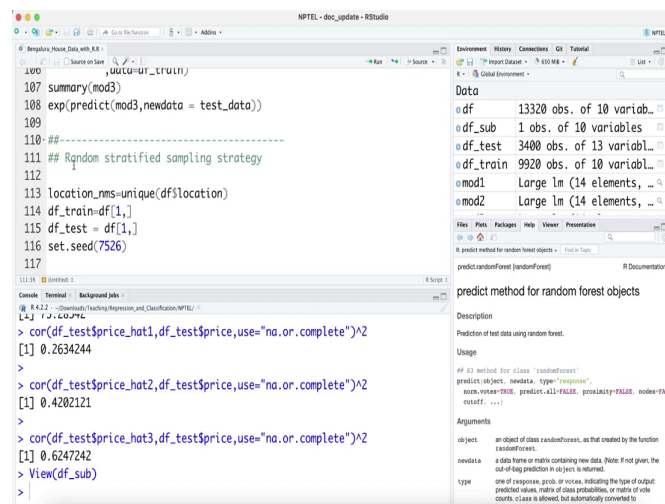
The RStudio interface displays the following data summary on the right:

Object	Description
df	13320 obs. of 10 variables
df_sub	1 obs. of 10 variables
df_test	3400 obs. of 13 variables
df_train	9920 obs. of 10 variables
mod1	Large lm (14 elements, ...)
mod2	Large lm (14 elements, ...)

The console output shows the correlation results for three different models (price_hat1, price_hat2, price_hat3) against the actual price, with values 0.2634244, 0.4202121, and 0.6247242 respectively. The 'View(df_sub)' command is also executed.



(Refer Slide Time: 02:09)



```
107 summary(mod3)
108 exp(predict(mod3,newdata = test_data))
109
110 ##-----
111 ## Random stratified sampling strategy
112
113 location_nms=unique(df$location)
114 df_train=df[1,]
115 df_test = df[1,]
116 set.seed(7526)
117
118 > cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
119
120 > cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
121
122 > cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
123 > View(df_sub)
```

The RStudio interface also shows the Environment pane with the following data objects:

Object	Class	Size
df	data.frame	13320 obs. of 10 variables
df_sub	data.frame	1 obs. of 10 variables
df_test	data.frame	3400 obs. of 13 variables
df_train	data.frame	9920 obs. of 10 variables
mod1	lm	Large lm (14 elements, ...)
mod2	lm	Large lm (14 elements, ...)

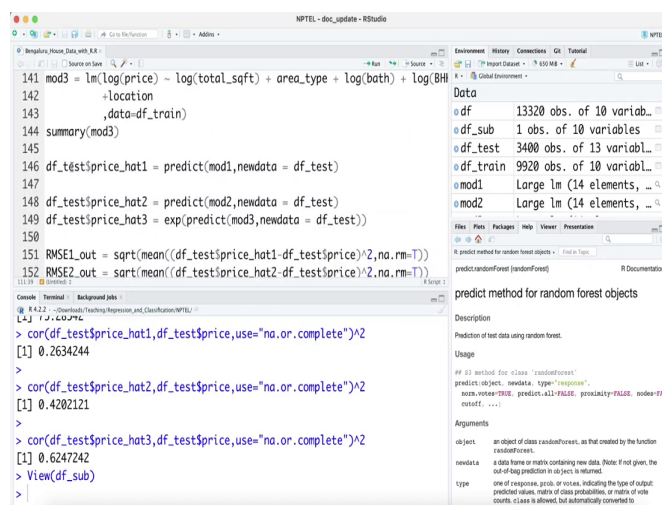


Now, this strategy is known as random stratified sampling strategy. The idea is imagine you want to draw sample all over the India. You can either draw simple random sample with replacement. In that case, what might happen that you made quite a few samples from big states like Maharashtra, Tamil Nadu, or you know Uttar Pradesh, but there are some small state like Sikkim, Goa, ok. In those states, you may not find even a single sample.

In this, this is not a very good strategy because you want you want representative of from each sample. Similarly, here each location, for each location, I want a representation. So, that is exactly what we are doing. This strategy is called random stratified sampling strategy and because of that, we avoid the issue of inconsistent modeling. Now, our model is consistent across all locations.

So, you can apply this strategy when you will handle state level data or any location or any data where which will have a some kind of categorical variable or too many categorical or categorical variable with quite a few locations. It is good to stratify the samples using or random draw the samples using stratified sampling strategy.

(Refer Slide Time: 04:18)

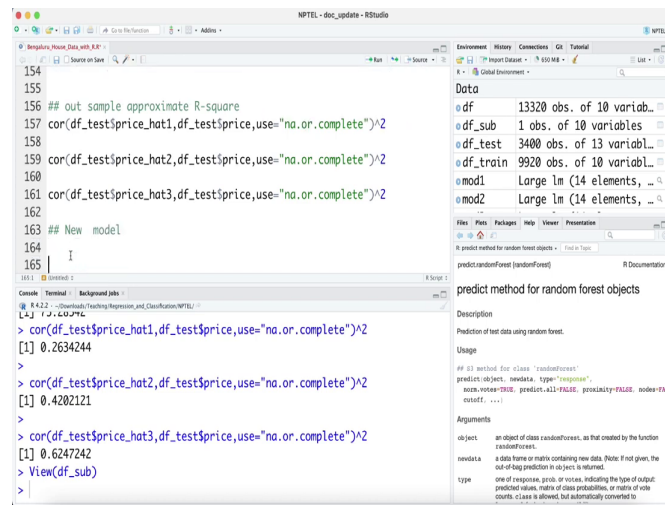


```
141 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BH)
142 +location
143 ,data=df_train)
144 summary(mod3)
145
146 df_test$price_hat1 = predict(mod1,newdata = df_test)
147
148 df_test$price_hat2 = predict(mod2,newdata = df_test)
149 df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
150
151 RMSE1_out = sqrt(mean((df_test$price_hat1-df_test$price)^2,na.rm=T))
152 RMSE2_out = sqrt(mean((df_test$price_hat2-df_test$price)^2,na.rm=T))
153
154 > cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
155
156 > cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
157
158 > cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
159 > View(df_sub)
```

The screenshot shows the RStudio interface. The console displays the execution of R code for training a linear model (mod3) and evaluating its performance against two other models (mod1 and mod2) using RMSE and correlation. The environment pane on the right shows the data objects: df (13320 obs. of 10 variables), df_sub (1 obs. of 10 variables), df_test (3400 obs. of 13 variables), df_train (9920 obs. of 10 variables), mod1 (Large lm (14 elements, ...)), and mod2 (Large lm (14 elements, ...)). The help pane on the right shows the documentation for the predict method for random forest objects.



(Refer Slide Time: 04:19)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains R code for calculating the correlation coefficient between predicted and actual values for three different models. The console shows the output of these calculations. The sidebar on the right displays the "Data" pane with variables `df`, `df_sub`, `df_test`, and `df_train`. Below the data pane, the "Environment" pane shows the objects `mod1` and `mod2`. The "Files" pane shows the "predict" method for random forest objects. The "Documentation" pane shows the documentation for the `predict` method for random forest objects.

```
154
155
156 ## out sample approximate R-square
157 cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
158
159 cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
160
161 cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
162
163 ## New model
164
165 |
```

```
> cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
>
> cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
>
> cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
> View(df_sub)
```

Environment: History: Connections: Global Environment

Data

- `df` 13320 obs. of 10 variab...
- `df_sub` 1 obs. of 10 variables
- `df_test` 3400 obs. of 13 variabl...
- `df_train` 9920 obs. of 10 variabl...
- `mod1` Large lm (14 elements, ...)
- `mod2` Large lm (14 elements, ...)

Files: Packages: Help: Viewer: Presentation

predict method for random forest objects

predict.randomForest (randomForest)

R Documentation

predict method for random forest objects

Description

Prediction of test data using random forest.

Usage

```
# S3 method for class 'randomForest'
predict(object, newdata, type="response",
        node.predict.all=FALSE, proximity=FALSE, nodesize=100,
        cutoff, ...)
```

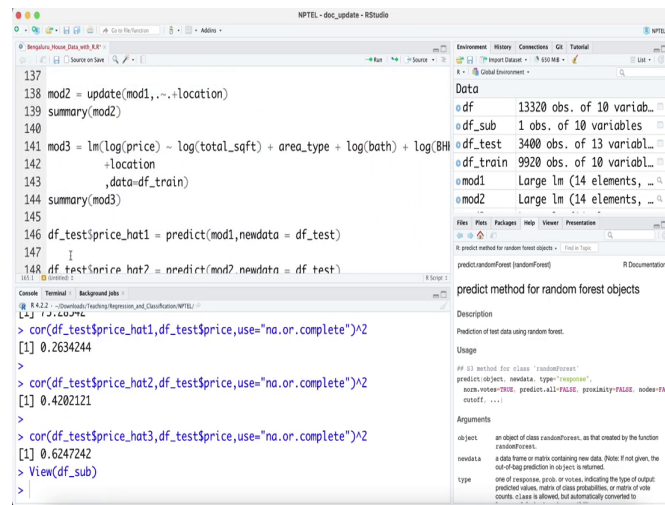
Arguments

- `object` an object of class `randomForest`, as that created by the function `randomForest`.
- `newdata` a data frame or matrix containing new data. (Note: If not given, the out-of-bag prediction in `object` is returned.
- `type` one of response, prob, or cvtree, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. `cutoff` is allowed, but automatically converted to



So, that is why we avoided all those problems and we able to move on to new models. So, let us try to fit a new model. So, new model.

(Refer Slide Time: 04:39)



```
137
138 mod2 = update(mod1, ~. + location)
139 summary(mod2)
140
141 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BH
142         + location
143         , data=df_train)
144 summary(mod3)
145
146 df_test$price_hat1 = predict(mod1, newdata = df_test)
147 I
148 df_test$price_hat2 = predict(mod2, newdata = df_test)
149
150 > cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
[1] 0.2634244
>
> cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
[1] 0.4202121
>
> cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
[1] 0.6247242
> View(df_sub)
```

The RStudio interface shows the following data objects in the Environment pane:

- df: 13320 obs. of 10 variables
- df_sub: 1 obs. of 10 variables
- df_test: 3400 obs. of 13 variables
- df_train: 9920 obs. of 10 variables
- mod1: Large lm (14 elements)
- mod2: Large lm (14 elements)

The Documentation pane shows the details for the `predict` method for random forest objects:

predict method for random forest objects

Description
Prediction of test data using random forest.

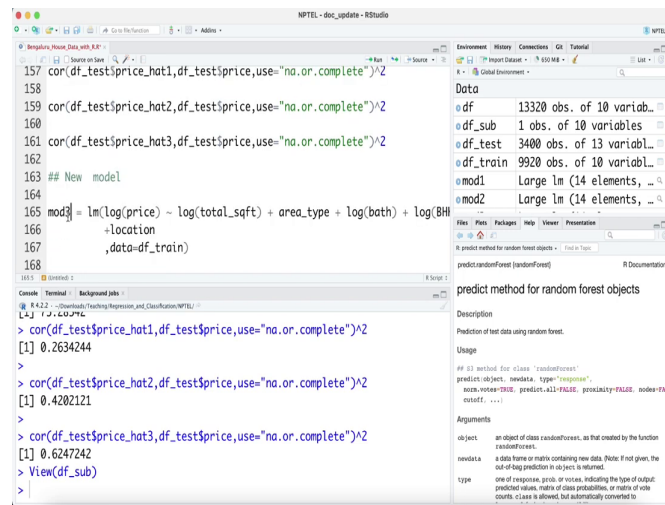
Usage
S3 method for class 'randomForest'
`predict(object, newdata, type="response",
 node.predict.all=FALSE, proximity=FALSE, nodesize=10,
 cutoff, ...)`

Arguments
object: an object of class 'randomForest', as that created by the function `randomForest`.
newdata: a data frame or matrix containing new data. (Note: If not given, the out-of-bag prediction in object is returned.)
type: one of response, probs, or cvtree, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. `cutoff` is allowed, but automatically converted to



So, this was our best model. So, for the third model was the best model. And can we add few more thing in the best model?

(Refer Slide Time: 04:50)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor contains the following R code:

```
157 cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
158
159 cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
160
161 cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
162
163 ## New model
164
165 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BH
166         + location
167         , data=df_train)
168
```

The console shows the output of the correlation calculations:

```
> cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
[1] 0.2634244
>
> cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
[1] 0.4202121
>
> cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
[1] 0.6247242
> View(df_sub)
```

The Environment pane on the right shows the following objects:

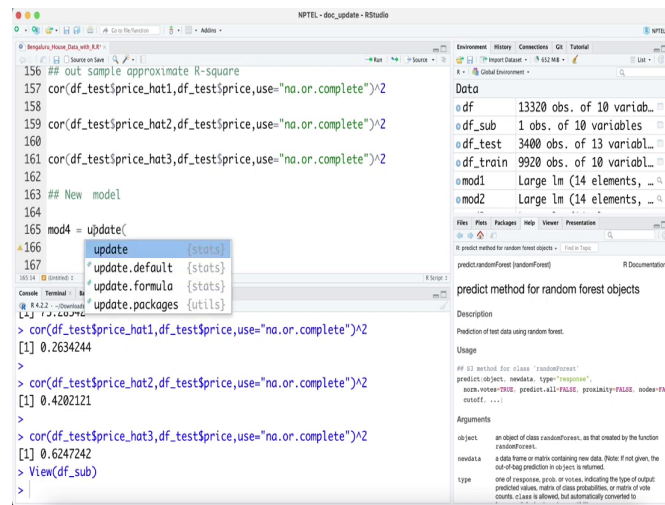
- df: 13320 obs. of 10 variab...
- df_sub: 1 obs. of 10 variables
- df_test: 3400 obs. of 13 variabl...
- df_train: 9920 obs. of 10 variabl...
- mod1: Large lm (14 elements, ...)
- mod2: Large lm (14 elements, ...)

The Help pane on the right shows the documentation for the `predict.randomForest` function.



So, mod 4; oops mod 4 and maybe what we will do, we will just update the first, we do not want to again run the model 3.

(Refer Slide Time: 05:02)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The script editor contains R code for calculating out-of-sample approximate R-squared values and updating a model. The console shows the results of these calculations. The Environment pane on the right lists objects: df (13320 obs. of 10 variables), df_sub (1 obs. of 10 variables), df_test (3400 obs. of 13 variables), df_train (9920 obs. of 10 variables), mod1 (Large lm (14 elements, ...)), and mod2 (Large lm (14 elements, ...)). The Files pane shows the "predict" method for random forest objects. The Help pane displays the documentation for the predict method for random forest objects, including a description, usage, and arguments.

```
156 ## out sample approximate R-square
157 cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
158
159 cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
160
161 cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
162
163 ## New model
164
165 mod4 = update(
166   update {stats}
167   *update.default {stats}
168   *update.formula {stats}
169   *update.packages {utils}
170 )
171
172 > cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
173 [1] 0.2634244
174
175 > cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
176 [1] 0.4202121
177
178 > cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
179 [1] 0.6247242
180 > View(df_sub)
```

Environment: 6 objects, 612 MB
Global Environment

Data

- df: 13320 obs. of 10 variables
- df_sub: 1 obs. of 10 variables
- df_test: 3400 obs. of 13 variables
- df_train: 9920 obs. of 10 variables
- mod1: Large lm (14 elements, ...)
- mod2: Large lm (14 elements, ...)

Files: Packages: Help: Viewer: Presentation

predict method for random forest objects

predict.randomForest (randomForest)

R Documentation

predict method for random forest objects

Description

Prediction of test data using random forest.

Usage

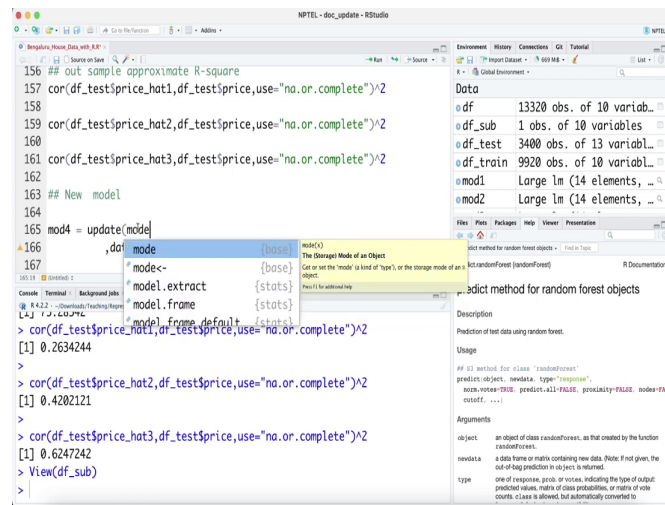
```
# S3 method for class 'randomForest'
predict(object, newdata, type="response",
        node.predict="TRUE", predict.all=FALSE, proximity=FALSE,
        outofbag=FALSE, ...)
```

Arguments

- object: an object of class "randomForest", as that created by the function randomForest.
- newdata: a data frame or matrix containing new data. (Note: If not given, the out-of-bag prediction in object is returned.)
- type: one of response, probs, or cvtree, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. ci=TRUE is allowed, but automatically converted to



(Refer Slide Time: 05:07)



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for calculating out-of-sample R-squared values and fitting a new model.
- Environment:** Lists objects in the workspace: df (13320 obs. of 10 variables), df_sub (1 obs. of 10 variables), df_test (3400 obs. of 13 variables), df_train (9920 obs. of 10 variables), mod1 (Large lm (14 elements, ...)), and mod2 (Large lm (14 elements, ...)).
- Console:** Shows the execution of the R code, resulting in three R-squared values: 0.2634244, 0.4202121, and 0.6247242.
- Pop-up:** A documentation window for the `rfc` method, which is a prediction method for random forest objects.

```
156 ## out sample approximate R-square
157 cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
158
159 cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
160
161 cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
162
163 ## New model
164
165 mod4 = update(mod1,
166               data = df_sub,
167               model = model.extract(mod1),
168               model.frame = model.frame(mod1, data = df_sub))
169
170 > cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
171 [1] 0.2634244
172
173 > cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
174 [1] 0.4202121
175
176 > cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
177 [1] 0.6247242
178 > View(df_sub)
```

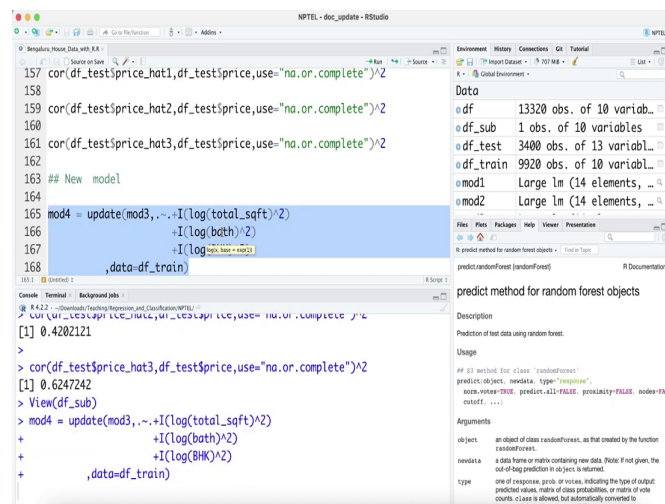
rfc Method Documentation:

- Description:** Prediction method for random forest objects.
- Usage:** `rfc(object, newdata, type="response", nodesize="TRUE", predict.all=FALSE, proximity=FALSE, nodesize="TRUE", costf, ...)`
- Arguments:**
 - `object`: an object of class `randomForest`, as that created by the function `randomForest`.
 - `newdata`: a data frame or matrix containing new data. (Note: If `newdata` is a matrix, the out-of-bag prediction in `object` is returned.)
 - `type`: one of `"response"`, `"prob"`, or `"matrix"`, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. `costf` is allowed, but automatically converted to `nodesize`.



So, we will just mod 3.

(Refer Slide Time: 05:12)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor contains the following R code:

```
157 cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
158
159 cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
160
161 cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
162
163 ## New model
164
165 mod4 = update(mod3, ~. + I(log(total_sqft)^2)
166               + I(log(bath)^2)
167               + I(log(bhk, na.rm = TRUE)^2)
168               , data=df_train)
```

The console shows the output of the correlation calculations:

```
[1] 0.4202121
>
> cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
[1] 0.6247242
> View(df_sub)
```

The Environment pane on the right shows the following objects:

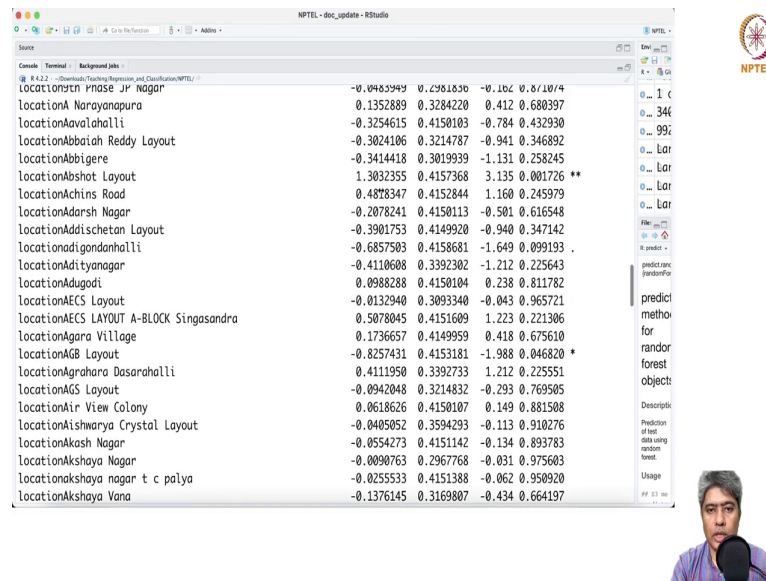
- df: 13320 obs. of 10 variables
- df_sub: 1 obs. of 10 variables
- df_test: 3400 obs. of 13 variables
- df_train: 9920 obs. of 10 variables
- mod1: Large lm (14 elements, ...)
- mod2: Large lm (14 elements, ...)

The Files pane shows the 'predict' method for random forest objects.



We will just tick that and whatever we have along with this, maybe we put a squared term of this, a square term of that plus I log of bath square plus I log of BHK square. So, I just want this. It takes a little time. Yeah, it is done. So, I just put a summary of mod4, model 4th model, ok. It is similar to 83 percent.

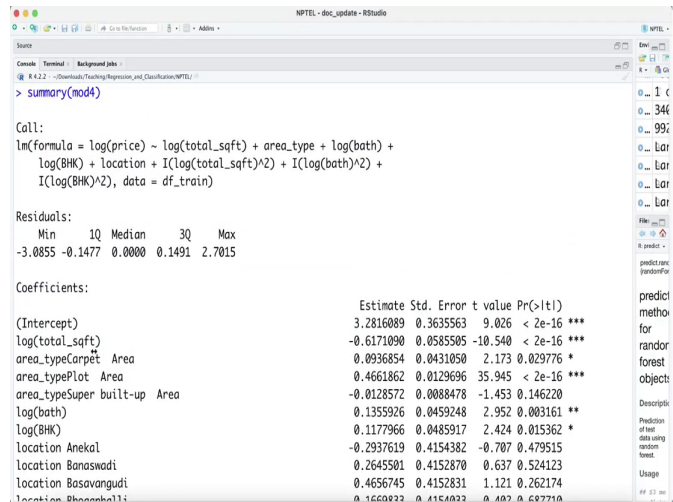
(Refer Slide Time: 06:42)



Location	Coordinate 1	Coordinate 2	Coordinate 3	Coordinate 4
locationnrt Phase JV Nagar	-0.10483949	0.2981830	-0.106	0.871074
locationA Narayanapura	0.1352889	0.3284220	0.412	0.680397
locationAvalahalli	-0.3254615	0.4150103	-0.784	0.432930
locationAbbaiah Reddy Layout	-0.3024106	0.3214787	-0.941	0.346892
locationAbbigere	-0.3414418	0.3019939	-1.131	0.258245
locationAbshot Layout	1.3032355	0.4157368	3.135	0.001726 **
locationAchins Road	0.4878347	0.4152844	1.160	0.245979
locationAdarsh Nagar	-0.2078241	0.4150113	-0.501	0.616548
locationAddisichetan Layout	-0.3901753	0.4149920	-0.940	0.347142
locationadigondanahalli	-0.6857503	0.4158681	-1.649	0.099193
locationAdityanagar	-0.4110608	0.3392302	-1.212	0.225643
locationAdugodi	0.0988288	0.4150104	0.238	0.811782
locationAECs Layout	-0.0132940	0.3093340	-0.043	0.965721
locationAECs LAYOUT A-BLOCK Singasandra	0.5078045	0.4151609	1.223	0.221306
locationAgara Village	0.1736657	0.4149959	0.418	0.675610
locationAGB Layout	-0.8257431	0.4153181	-1.988	0.046820 *
locationAgrahara Dasarahalli	0.4111950	0.3392733	1.212	0.225551
locationAGS Layout	-0.0942048	0.3214832	-0.293	0.769505
locationAir View Colony	0.0618626	0.4150107	0.149	0.881508
locationAishwarya Crystal Layout	-0.0405052	0.3594293	-0.113	0.910276
locationAakash Nagar	-0.0554273	0.4151142	-0.134	0.893783
locationAkhaya Nagar	-0.0090763	0.2967768	-0.031	0.975603
locationAkhaya nagar t c poly	-0.0255533	0.4151388	-0.062	0.950920
locationAkhaya Vana	-0.1376145	0.3169807	-0.434	0.664197

Not much. Let me just little bit increase the place. If we can just emit more space. Yeah. And see a shot layout Ambedkar Colony, AGB Layout, ok. These are like a shot layout, ok. 8th Block Jayanagar as in. Anyway, we will see what we have here. (Refer Time: 07:31). So, then these third stage. So, what we are seeing here, ok.

(Refer Slide Time: 07:47)



NPTEL - doc_update - RStudio

```
> summary(mod4)
```

Call:
lm(formula = log(price) ~ log(total_sqft) + area_type + log(bath) +
log(BHK) + location + I(log(total_sqft)^2) + I(log(bath)^2), data = df_train)

Residuals:

Min	1Q	Median	3Q	Max
-3.0855	-0.1477	0.0000	0.1491	2.7015

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	3.2816089	0.3635563	9.026	< 2e-16 ***
log(total_sqft)	-0.6171090	0.0585505	-10.540	< 2e-16 ***
area_typeCarpet Area	0.0936854	0.0431050	2.173	0.029776 *
area_typePlot Area	0.4661862	0.0129696	35.945	< 2e-16 ***
area_typeSuper built-up Area	-0.0128572	0.0088478	-1.453	0.146220
log(bath)	0.1355926	0.0459248	2.952	0.003161 **
log(BHK)	0.1177966	0.0485917	2.424	0.015362 *
location Anekal	-0.2937619	0.4154382	-0.707	0.479515
location Banaswadi	0.2645501	0.4152870	0.637	0.524123
location Basavangudi	0.4656745	0.4152831	1.121	0.262174
location Bhogalalli	0.1600932	0.4154032	0.387	0.697710


NPTEL

predict new
randomFor

predict metho
for
random
forest
object

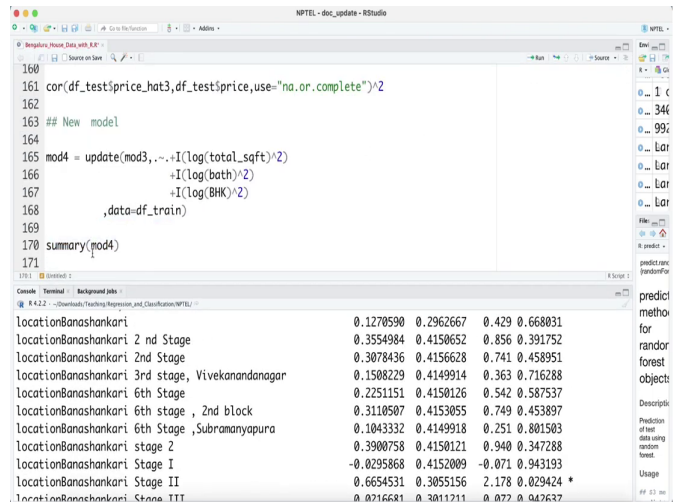
Description:
Prediction of new
data using
random
forest.

Usage
[3] ...



There is some issue. It has square terms, but we are not seeing the square terms, ok. Maybe they have added the square terms at the end. So, but they are not ending it here.

(Refer Slide Time: 08:04)





The screenshot shows an RStudio session. The script editor contains the following code:

```
160
161 cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
162
163 ## New model
164
165 mod4 = update(mod3, ~., +I(log(total_sqft)^2)
166               +I(log(bath)^2)
167               +I(log(BHK)^2)
168               , data=df_train)
169
170 summary(mod4)
171
```

The console output shows the summary of the model, including coefficients and their standard errors:

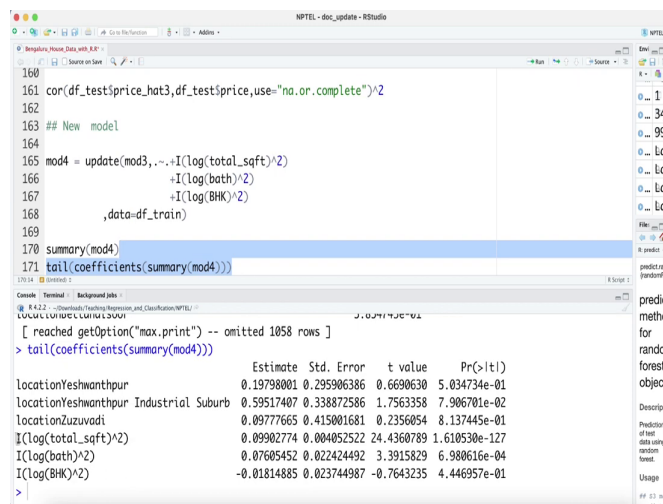
locationBanashankari	0.1270590	0.2962667	0.429	0.668031
locationBanashankari 2 nd Stage	0.3554984	0.4150652	0.856	0.391752
locationBanashankari 2nd Stage	0.3078436	0.4156628	0.741	0.458951
locationBanashankari 3rd stage, Vivekanandanagar	0.1508229	0.4149914	0.363	0.716288
locationBanashankari 6th Stage	0.2251151	0.4150126	0.542	0.587537
locationBanashankari 6th stage, 2nd block	0.3110507	0.4153055	0.749	0.453897
locationBanashankari 6th Stage, Subramanyapura	0.1043332	0.4149918	0.251	0.801503
locationBanashankari stage 2	0.3900758	0.4150121	0.940	0.347288
locationBanashankari Stage I	-0.0295868	0.4152009	-0.071	0.943193
locationBanashankari Stage II	0.6654531	0.3055156	2.178	0.029424 *
locationBanashankari Stage TTT	0.0216681	0.3011211	0.077	0.047637

The right sidebar shows the 'Environment' pane with a list of objects, including 'mod4'. The 'Files' pane shows the project structure. The 'Console' pane shows the output of the 'summary(mod4)' command.



So, what we will do instead of that, we will just say coefficients mod 4, ok; it is just, ok. It is just coeff with this and this (Refer Time: 08:29), ok. And maybe I will just say tail of this tail or yeah. So, you can see at the end, the it is added these parameters.

(Refer Slide Time: 08:37)



The screenshot shows an RStudio window with the following code in the editor:

```
160
161 cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
162
163 ## New model
164
165 mod4 = update(mod3, ~. + I(log(total_sqft)^2)
166               + I(log(bath)^2)
167               + I(log(BHK)^2)
168               , data=df_train)
169
170 summary(mod4)
171 tail(coefficients(summary(mod4)))
```

The console output shows the tail of the coefficients for the model:

```
[ reached getOption("max.print") -- omitted 1058 rows ]
> tail(coefficients(summary(mod4)))
```

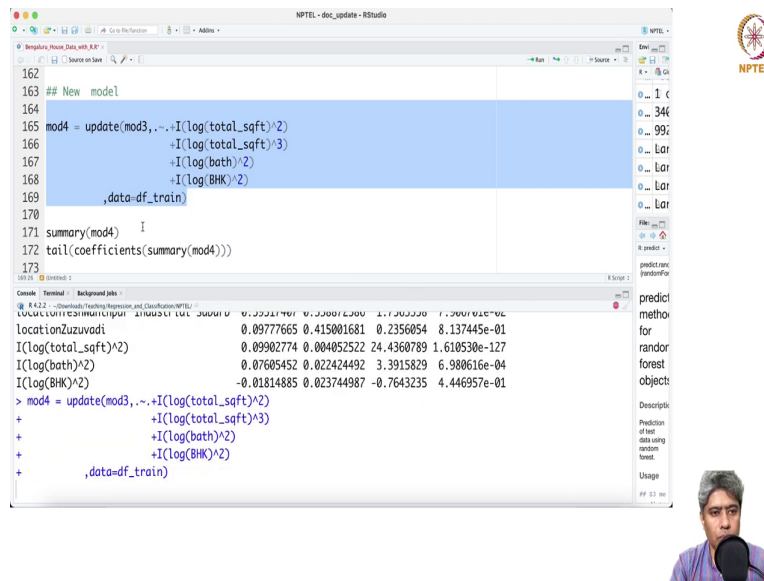
	Estimate	Std. Error	t value	Pr(> t)
locationYeshwanthpur	0.19798001	0.295906386	0.6690630	5.034734e-01
locationYeshwanthpur Industrial Suburb	0.59517407	0.338872586	1.7563358	7.906701e-02
locationZuzuvadi	0.09777665	0.415001681	0.2356054	8.137445e-01
I(log(total_sqft)^2)	0.09902774	0.004052522	24.4360789	1.610530e-127
I(log(bath)^2)	0.07605452	0.022424492	3.3915829	6.980616e-04
I(log(BHK)^2)	-0.01814885	0.023744987	-0.7643235	4.446957e-01

A small video inset in the bottom right corner shows a man with grey hair and a beard, wearing a blue shirt, speaking into a microphone.

So, you can see that the t value for log total square feet square is very high. Log bath square is also very high. And log BHK square does not have much effect. t value is very small. So, square quadratic effect for bath in log scale and total square feet in log scale does have a quite significant effect. Looks like.

(Refer Time: 09:16); what we can do is you can keep adding more and more samples, more and more things, so maybe you can try one more. Total square feet cube. I just, ok.

(Refer Slide Time: 09:43)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The script editor contains the following R code:

```
162  
163 ## New model  
164  
165 mod4 = update(mod3, ~. + I(log(total_sqft)^2)  
166               + I(log(total_sqft)^3)  
167               + I(log(bath)^2)  
168               + I(log(BHK)^2)  
169               , data=df_train)  
170  
171 summary(mod4)  
172 tail(coefficients(summary(mod4)))  
173
```

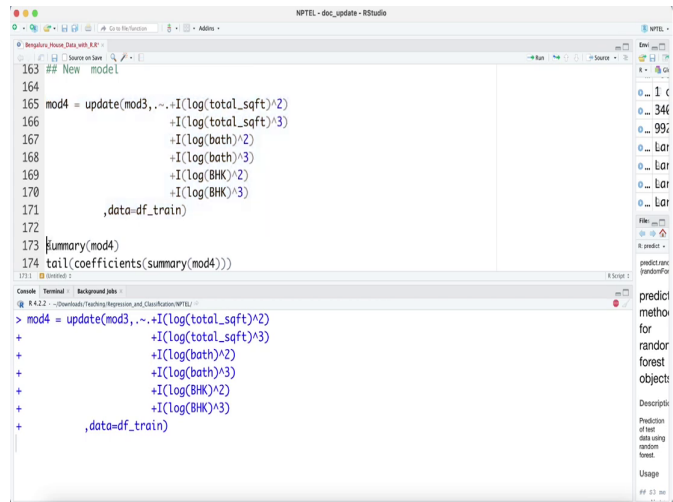
The console output shows the following coefficients:

```
locationZuZuvadi 0.09777665 0.415001681 0.2356054 8.137445e-01  
I(log(total_sqft)^2) 0.09902774 0.004052522 24.4360789 1.610530e-127  
I(log(bath)^2) 0.07605452 0.022424492 3.3915829 6.980616e-04  
I(log(BHK)^2) -0.01814885 0.023744987 -0.7643235 4.446957e-01
```

Below the code, a small video inset shows a man speaking into a microphone. To the right of the RStudio window, there is a small NPTEL logo and a vertical list of items: "predict", "method", "for", "rand", "forest", "object", "Description", "Prediction", "of new", "data using", "random", "forest", "Usage", "## 13 min".

4 and then we just see this. Yeah, both square and cube does have effect looks like. Bath square does have a effect. So, What about bath cube, and BHK cube?

(Refer Slide Time: 10:25)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor contains the following R code:

```
163 ## New model
164
165 mod4 = update(mod3, ~., +I(log(total_sqft)^2)
166               +I(log(total_sqft)^3)
167               +I(log(bath)^2)
168               +I(log(bath)^3)
169               +I(log(BHK)^2)
170               +I(log(BHK)^3)
171               ,data=df_train)
172
173 summary(mod4)
174 tail(coefficients(summary(mod4)))
```

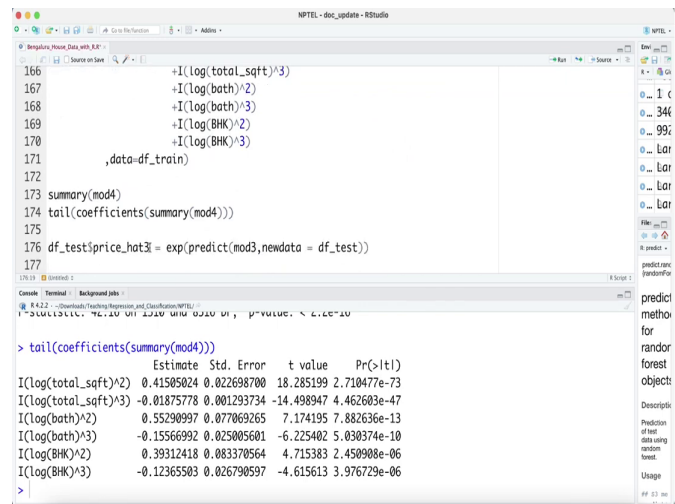
The console shows the execution of the code:

```
> mod4 = update(mod3, ~., +I(log(total_sqft)^2)
+               +I(log(total_sqft)^3)
+               +I(log(bath)^2)
+               +I(log(bath)^3)
+               +I(log(BHK)^2)
+               +I(log(BHK)^3)
+               ,data=df_train)
```

On the right side of the RStudio window, there is a sidebar with a list of files and a description of the model. The description reads: 'Prediction of test data using random forest.' Below this, there is a small video feed of a person speaking into a microphone.

Let us run this, ok. So, if I just add the end. So, BHK square and BHK cube both are now effective. Bath square and bath cube is also effective. And square fit square and square fit cube are also effective. So, we can keep doing it. But let us see if we by doing this, by adding this predictors, engineer predictors if we really adding any value or not.

(Refer Slide Time: 11:41)





The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The script editor contains the following code:

```
166 +I(log(total_sqft)^3)
167 +I(log(bath)^2)
168 +I(log(bath)^3)
169 +I(log(BHK)^2)
170 +I(log(BHK)^3)
171 ,data=df_train)
172
173 summary(mod4)
174 tail(coefficients(summary(mod4)))
175
176 df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
177
```

The console output shows the summary of the model coefficients:

```
> tail(coefficients(summary(mod4)))
              Estimate Std. Error t value Pr(>|t|)
I(log(total_sqft)^2) 0.41505024 0.022698700 18.285199 2.710477e-73
I(log(total_sqft)^3) -0.01875778 0.001293734 -14.498947 4.462603e-47
I(log(bath)^2)        0.55290997 0.077069265  7.174195 7.882636e-13
I(log(bath)^3)        -0.15566992 0.025005601 -6.225402 5.030374e-10
I(log(BHK)^2)         0.39312418 0.083370564  4.715383 2.450908e-06
I(log(BHK)^3)        -0.12365503 0.026790597 -4.615613 3.976729e-06
```



So, let us try model this test, ok, alright.

(Refer Slide Time: 11:58)

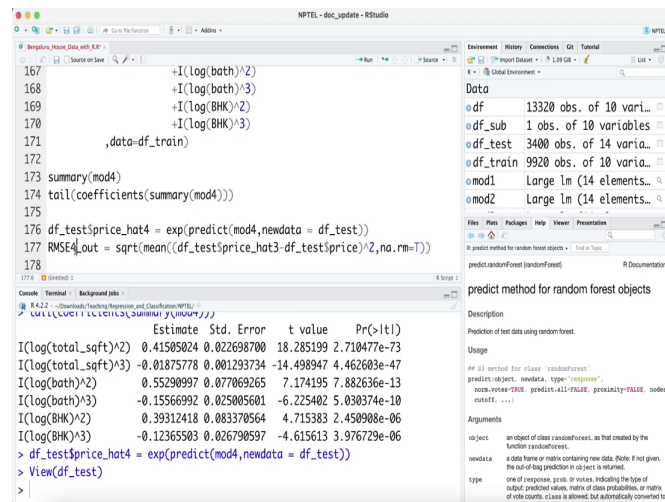
The screenshot shows the RStudio interface with the following components:

- Environment Pane:** Displays a data frame 'df' with 13320 observations and 10 variables. It also shows 'df_sub' (1 obs, 10 variables), 'df_test' (3400 obs, 14 variables), 'df_train' (9920 obs, 10 variables), 'mod1' (Large lm (14 elements...)), and 'mod2' (Large lm (14 elements...)).
- Console:** Shows the execution of a linear model and a prediction. The model summary for 'I(log(total_sqft)^2)' is displayed, followed by the prediction command: `> df_test$price_hat4 = exp(predict(mod4, newdata = df_test))`.
- Help Window:** Shows the documentation for the `predict.ranger` function, which is used for predicting values from a random forest model.



And then this is df test, this is price hat. And looks like it is doing fine, yeah. So, we will see and then let us write.

(Refer Slide Time: 12:37)



```

167      +I(log(bath)^2)
168      +I(log(bath)^3)
169      +I(log(BHK)^2)
170      +I(log(BHK)^3)
171      ,data=df_train)
172
173 summary(mod4)
174 tail(coefficients(summary(mod4)))
175
176 df_test$price_hat4 = exp(predict(mod4,newdata = df_test))
177 RMSE4_out = sqrt(mean((df_test$price_hat3-df_test$price)^2,na.rm=T))
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Console Output:

	Estimate	Std. Error	t value	Pr(> t)
I(log(total_sqft)^2)	0.41585024	0.022698700	18.285199	2.710477e-73
I(log(total_sqft)^3)	-0.01875778	0.001293734	-14.498947	4.462603e-47
I(log(bath)^2)	0.55290997	0.077069265	7.174195	7.882636e-13
I(log(bath)^3)	-0.15566992	0.025005601	-6.225402	5.030374e-10
I(log(BHK)^2)	0.39312418	0.083370564	4.715383	2.450908e-06
I(log(BHK)^3)	-0.12365503	0.026790597	-4.615613	3.976729e-06

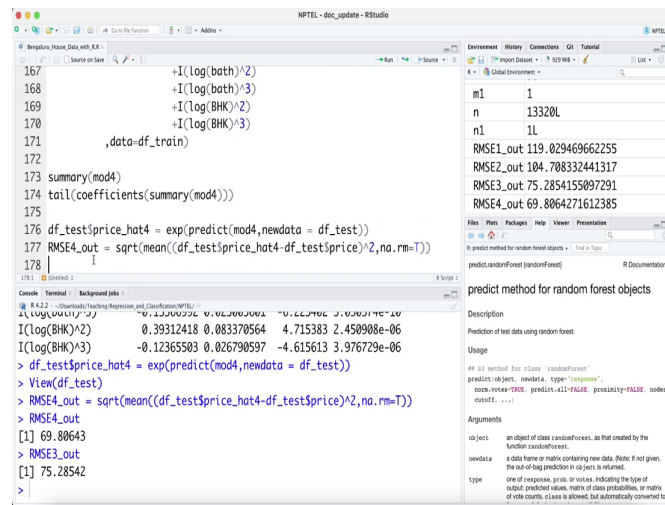
RMSE4_out = 4.69

df_test\$price_hat4 = exp(predict(mod4,newdata = df_test))

View(df_test)

RMSE 4 out, this is 4, hat 4 and 69 whereas, RMSE 3 output 75. So, adding these adding these numbers adding these features actually helped a bit. So, you can see feature engineering is helping. So, these are out of sample remember that. These are out of the sample RMSE, root mean square error.

(Refer Slide Time: 13:20)



```

167      +I(log(bath)^2)
168      +I(log(bath)^3)
169      +I(log(BHK)^2)
170      +I(log(BHK)^3)
171      ,data=df_train)
172
173 summary(mod4)
174 tail(coefficients(summary(mod4)))
175
176 df_test$price_hat4 = exp(predict(mod4,newdata = df_test))
177 RMSE4_out = sqrt(mean((df_test$price_hat4-df_test$price)^2,na.rm=T))
178

```

Environment

m1	1
n	13320L
n1	1L
RMSE1_out	119.029469662255
RMSE2_out	104.708332441317
RMSE3_out	75.2854155097291
RMSE4_out	69.8064271612385

predict method for random forest objects

Description

Prediction of test data using random forest.

Usage

```

## S3 method for class "randomForest"
predict(object, newdata, type="response",
  norm.roots="TRUE", predict.all=FALSE, proximity=FALSE, nodes
  output, ...)

```

Arguments

object: an object of class randomForest, as that created by the function randomForest.

newdata: a data frame or matrix containing new data. Note: if not given, the out-of-bag prediction in object is returned.

type: one of response, prob, or ovs, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. Classes is allowed, but automatically converted to

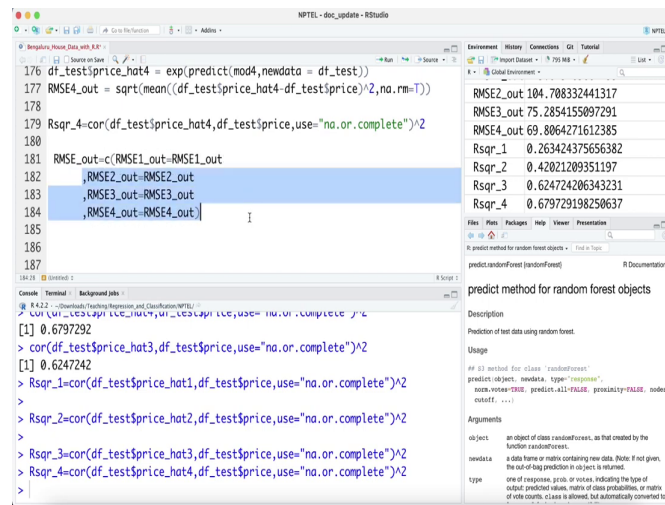


So, probably this is a good thing to have and 4 (Refer Time: 13:26). So, let me take my 4 and 0.679 whereas, my this is 62; this is even pushed further 70 percent. So, model 4 is so far best, right. So, I can just put it there R, to R square 1.



These are obviously out sample R square 2, 3 and this is 4, right. And so, we can see (Refer Time: 14:44) 1, 1, 2, 2.

(Refer Slide Time: 15:10)



The screenshot shows an RStudio session with the following code in the script editor:

```
176 df_test$price_hat4 = exp(predict(mod4, newdata = df_test))
177 RMSE4_out = sqrt(mean((df_test$price_hat4 - df_test$price)^2, na.rm=T))
178
179 Rsqr_4 = cor(df_test$price_hat4, df_test$price, use="na.or.complete")^2
180
181 RMSE_out = c(RMSE1_out, RMSE2_out,
182             , RMSE3_out, RMSE4_out)
183
184 Rsqr_out = c(Rsqr1_out, Rsqr2_out,
185             , Rsqr3_out, Rsqr4_out)
186
187
```

The console shows the output of the calculations:

```
[1] 0.6797292
> cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
[1] 0.6247242
> Rsqr_1 = cor(df_test$price_hat1, df_test$price, use="na.or.complete")^2
>
> Rsqr_2 = cor(df_test$price_hat2, df_test$price, use="na.or.complete")^2
>
> Rsqr_3 = cor(df_test$price_hat3, df_test$price, use="na.or.complete")^2
> Rsqr_4 = cor(df_test$price_hat4, df_test$price, use="na.or.complete")^2
>
```

The Environment pane on the right shows the following objects:

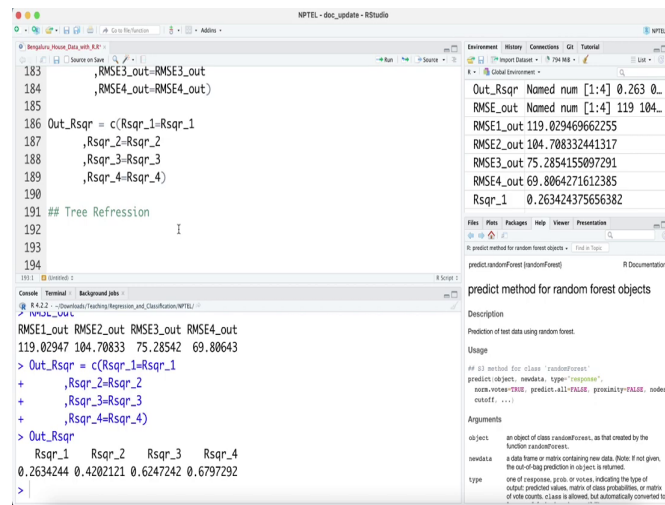
Object	Value
RMSE2_out	104.708332441317
RMSE3_out	75.2854155097291
RMSE4_out	69.8064271612385
Rsqr_1	0.263424375656382
Rsqr_2	0.42021209351197
Rsqr_3	0.624724206343231
Rsqr_4	0.679729198250637

The R Documentation pane on the right shows the documentation for the `predict.randomForest` function.



So, clearly RMSE has dropped over on, 4th model is much better. And out of the sample 4th model is this, and then R square 4 equal to R square 4, R square 3 equal to R square 3, R square 2 equal to R square 2, R square 1 equal to R square, right.

(Refer Slide Time: 16:04)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor on the left contains R code for calculating RMSE and R-squared values. The environment pane on the right shows the values of the variables created. The console at the bottom shows the output of the code.

```
183 RMSE3_out - RMSE3_out
184 RMSE4_out - RMSE4_out)
185
186 Out_Rsqr = c(Rsqr_1=Rsqr_1
187 ,Rsqr_2=Rsqr_2
188 ,Rsqr_3=Rsqr_3
189 ,Rsqr_4=Rsqr_4)
190
191 ## Tree Refression
192 I
193
194
```

Environment:

Variable	Value
Out_Rsqr	Named num [1:4] 0.263 0.263 0.263 0.263
RMSE_out	Named num [1:4] 119 104 104 104
RMSE1_out	119.029469662255
RMSE2_out	104.708332441317
RMSE3_out	75.2854155097291
RMSE4_out	69.8064271612385
Rsqr_1	0.2634244375656382

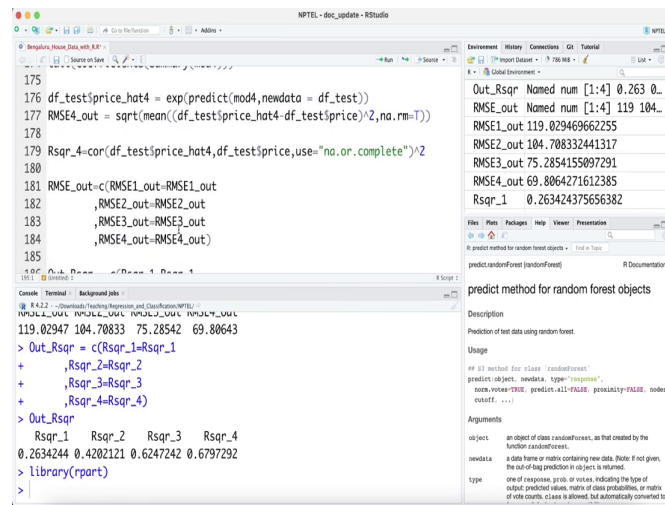
Console:

```
R 4.2.2 -- commands: Teaching-Regression_and_Classification-NPTEL --
RMSE1_out RMSE2_out RMSE3_out RMSE4_out
119.02947 104.70833 75.28542 69.80643
> Out_Rsqr = c(Rsqr_1=Rsqr_1
+ ,Rsqr_2=Rsqr_2
+ ,Rsqr_3=Rsqr_3
+ ,Rsqr_4=Rsqr_4)
> Out_Rsqr
Rsqr_1 Rsqr_2 Rsqr_3 Rsqr_4
0.2634244 0.4202121 0.6247242 0.6797292
```



And sample R square, R square is this. So, it has gone up to 70 percent 67, 68 percent 67.9 is the maximum out of the sample R square that we can found. So, what we now we can try? We can also try implement tree regression, tree regression, ok. So, tree regression is way we can do that library R part, ok, R part, ok.

(Refer Slide Time: 17:30)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor contains R code for model evaluation. The console shows the output of the code, including the calculation of RMSE values and the correlation coefficient. The environment pane on the right shows the objects created in the global environment.

```
175 df_test$price_hat4 = exp(predict(mod4,newdata = df_test))
176 RMSE4_out = sqrt(mean((df_test$price_hat4-df_test$price)^2,na.rm=T))
177
178
179 Rsqr_4=cor(df_test$price_hat4,df_test$price,use="na.or.complete")/2
180
181 RMSE_out=c(RMSE1_out-RMSE1_out
182           ,RMSE2_out-RMSE2_out
183           ,RMSE3_out-RMSE3_out
184           ,RMSE4_out-RMSE4_out)
185
```

Console Output:

```
195.1 Out_Rmse = c(Rmse_1,Rmse_2,Rmse_3,Rmse_4)
196.1 RMSE1_out RMSE2_out RMSE3_out RMSE4_out
119.02947 104.70833 75.28542 69.80643
> Out_Rsqr = c(Rsqr_1=Rsqr_1
+             ,Rsqr_2=Rsqr_2
+             ,Rsqr_3=Rsqr_3
+             ,Rsqr_4=Rsqr_4)
> Out_Rsqr
Rsqr_1 Rsqr_2 Rsqr_3 Rsqr_4
0.2634244 0.4202121 0.6247242 0.6797292
> library(rpart)
```

Environment Pane:

Object	Class	Attributes
Out_Rsqr	Named num	[1:4] 0.263 0.420 0.625 0.680
RMSE_out	Named num	[1:4] 119 104 75 69.8
RMSE1_out	num	119.029469662255
RMSE2_out	num	104.708332441317
RMSE3_out	num	75.2854155097291
RMSE4_out	num	69.8064271612385
Rsqr_1	num	0.263424375656382

Documentation Pane:

predict.randomForest (randomForest)

Description

Prediction of test data using random forest.

Usage

```
# S3 method for class "randomForest"
predict(object, newdata, type="response",
        norm.predict=FALSE, predict.all=FALSE, proximity=FALSE,
        outofbag=FALSE, ...)
```

Arguments

object: an object of class randomForest, as that created by the function randomForest.

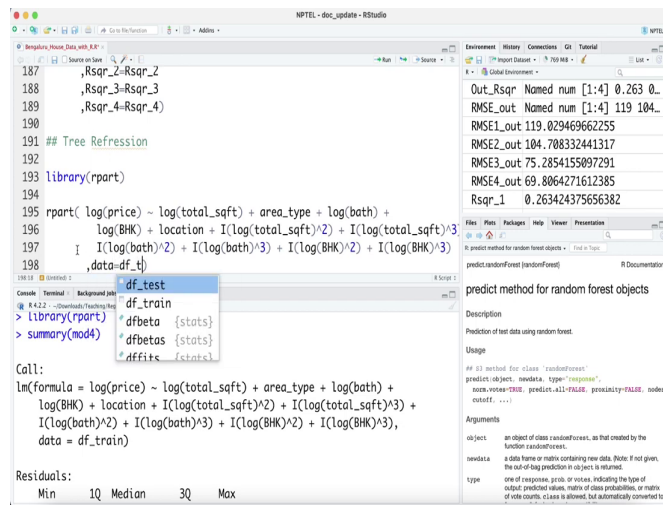
newdata: a data frame or matrix containing new data. Note: if not given, the out-of-bag prediction in object is returned.

type: one of response, prob, or matrix, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts. If case is allowed, but automatically converted to



And then, then model4 dot mod4. So, what we can do is essentially I can just fit R part and then I just run it. So, this is the formula, right, ok. So, y naught data equal to df train.

(Refer Slide Time: 18:50)



The screenshot shows an RStudio session with the following components:

- Script Editor:** Contains R code for a linear model and a random forest model.

```
187 ,Rsqr_2=Rsqr_2
188 ,Rsqr_3=Rsqr_3
189 ,Rsqr_4=Rsqr_4
190
191 ## Tree Regression
192
193 library(rpart)
194
195 rpart( log(price) ~ log(total_sqft) + area_type + log(bath) +
196        log(BHK) + location + I(log(total_sqft)^2) + I(log(total_sqft)^3)
197        I(log(bath)^2) + I(log(bath)^3) + I(log(BHK)^2) + I(log(BHK)^3)
198        ,data=df_t)
```

A dropdown menu is open over the `df_test` variable, showing options: `df_train`, `dfbeta {stats}`, `dfbetas {stats}`, and `dfefits {stats}`.
- Environment:** Displays the following objects:
 - `Out_Rsqr` Named num [1:4] 0.263 0...
 - `RMSE_out` Named num [1:4] 119 104...
 - `RMSE1_out` 119.029469662255
 - `RMSE2_out` 104.788332441317
 - `RMSE3_out` 75.2854155897291
 - `RMSE4_out` 69.8064271612385
 - `Rsqr_1` 0.263424375656382
- Console:** Shows the execution of `library(rpart)` and `summary(mod4)`. Below this, the `Call` for the linear model is displayed:

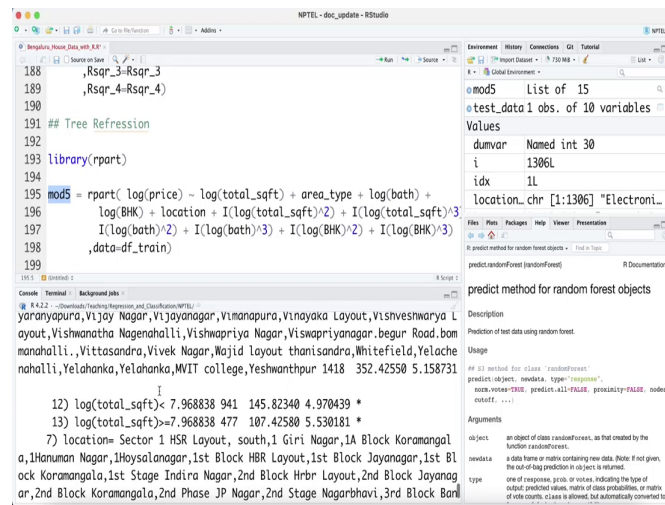
```
lm(formula = log(price) ~ log(total_sqft) + area_type + log(bath) +
    log(BHK) + location + I(log(total_sqft)^2) + I(log(total_sqft)^3) +
    I(log(bath)^2) + I(log(bath)^3) + I(log(BHK)^2) + I(log(BHK)^3),
    data = df_train)
```

Below the call, the `Residuals:` are listed: Min, 1Q, Median, 3Q, Max.
- Help:** Shows the documentation for the `predict` method for random forest objects.



Well, this is mod3, no mod5. I hope this will be ok. This has given me something.

(Refer Slide Time: 19:11)



The screenshot shows the RStudio interface with the following content:

```
188 ,Rsqr_3=Rsqr_3
189 ,Rsqr_4=Rsqr_4)
190
191 ## Tree Refression
192
193 library(rpart)
194
195 mod5 = rpart( log(price) ~ log(total_sqft) + area_type + log(bath) +
196               log(BHK) + location + I(log(total_sqft)^2) + I(log(total_sqft)^3)
197               I(log(bath)^2) + I(log(bath)^3) + I(log(BHK)^2) + I(log(BHK)^3)
198               ,data=df_train)
199
```

The Environment pane on the right shows:

- `mod5`: List of 15
- `test_data`: 1 obs. of 10 variables
- Values:
 - `dumvar`: Named int 30
 - `i`: 1306L
 - `idx`: 1L
 - `location_chr`: [1:1306] "Electroni_"

The console shows the output of the model:

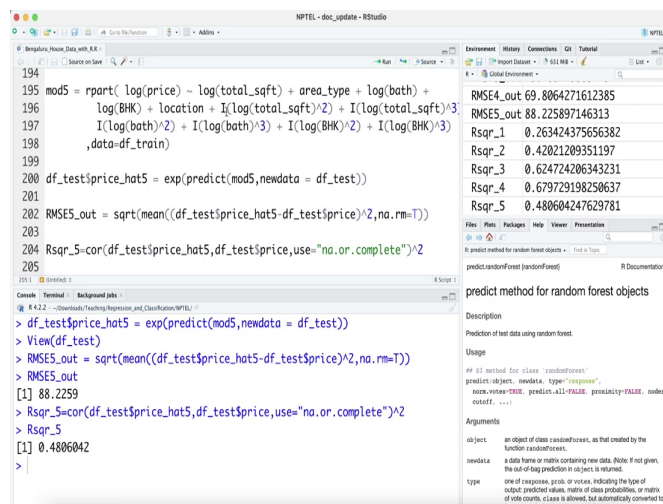
```
12) log(total_sqft)< 7.968838 941 145.82340 4.970439 *
13) log(total_sqft)>=7.968838 477 107.42580 5.530181 *
7) location= Sector 1 HSR Layout, south,1 Giri Nagar,1A Block Koramangal
a,1Hanuman Nagar,1Hoyasolanagar,1st Block HBR Layout,1st Block Jayanagar,1st Bl
ock Koramangala,1st Stage Indira Nagar,2nd Block Hrb Layout,2nd Block Jayanag
ar,2nd Block Koramangala,2nd Phase JP Nagar,2nd Stage Nagarbhavi,3rd Block Barl
```

The right pane shows the documentation for the `predict.randomForest` function.



And then alright, quite complicated.

(Refer Slide Time: 19:44)



The image shows an RStudio window with the following content:

```
194
195 mod5 = rpart( log(price) ~ log(total_sqft) + area_type + log(bath) +
196               log(BHK) + location + I(log(total_sqft)^2) + I(log(total_sqft)^3)
197               I(log(bath)^2) + I(log(bath)^3) + I(log(BHK)^2) + I(log(BHK)^3)
198               ,data=df_train)
199
200 df_test$price_hat5 = exp(predict(mod5,newdata = df_test))
201
202 RMSE5_out = sqrt(mean((df_test$price_hat5-df_test$price)^2,na.rm=T))
203
204 Rsqr_5=cor(df_test$price_hat5,df_test$price,use="na.or.complete")^2
205
```

Console output:

```
> df_test$price_hat5 = exp(predict(mod5,newdata = df_test))
> View(df_test)
> RMSE5_out = sqrt(mean((df_test$price_hat5-df_test$price)^2,na.rm=T))
> RMSE5_out
[1] 88.2259
> Rsqr_5=cor(df_test$price_hat5,df_test$price,use="na.or.complete")^2
> Rsqr_5
[1] 0.4806042
>
```

Environment pane shows:

Variable	Value
RMSE4_out	69.8064271612385
RMSE5_out	88.225897146313
Rsqr_1	0.263424375656382
Rsqr_2	0.42021209351197
Rsqr_3	0.624724206343231
Rsqr_4	0.679729198250637
Rsqr_5	0.480604247629781

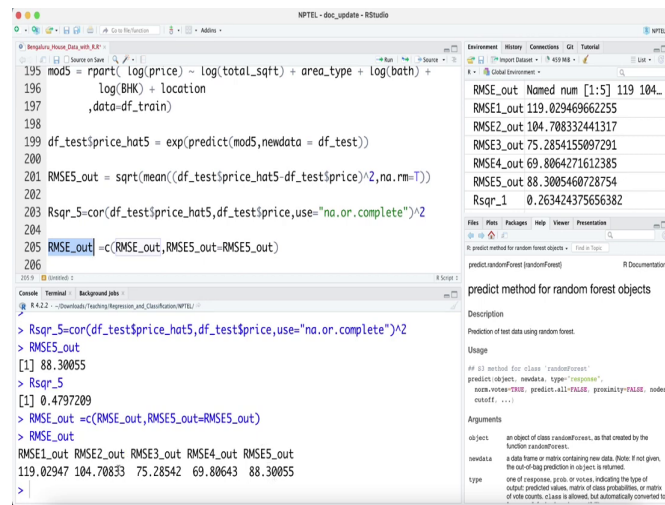
Help pane shows the documentation for `predict.randomForest`.



Let us now try to; let us use it as a black box. First is, we have to predict the model 5 as a black box let us see how it is doing, ok, looks like it is doing ok. And then RMSE 5 and RMSE square, RMSE 5 out this and this 88; RMSE 5 as this, alright. So, this is some what not doing so, when the regression tree or what you can do you know what, you just drop this, if you just drop this engineered feature and let us see what it does in the. So, RMSE (Refer Time: 20:50) similar not much.

So, whether you put the engineer feature or not tree regression, tree is doing slightly worse than I would say the RMSE out is, RMSE out in out comma is equal to out, RMSE out. So, now, so we can see that it is some doing somewhere between third model and the second model.

(Refer Slide Time: 21:24)



The image shows the RStudio interface with the following content:

```
195 mod5 = rpart( log(price) ~ log(total_sqft) + area_type + log(bath) +
196               log(BHK) + location
197               ,data=df_train)
198
199 df_testprice_hat5 = exp(predict(mod5,newdata = df_test))
200
201 RMSE5_out = sqrt(mean((df_testprice_hat5-df_testprice)^2,na.rm=T))
202
203 Rsqr_5=cor(df_testprice_hat5,df_testprice,use="na.or.complete")^2
204
205 RMSE_out =c(RMSE_out,RMSE5_out)
206
```

Environment pane:

RMSE_out	Named num [1:5]	119	104_
RMSE1_out	119.029469662255		
RMSE2_out	104.708332441317		
RMSE3_out	75.2854155097291		
RMSE4_out	69.8064271612385		
RMSE5_out	88.3005460728754		
Rsqr_1	0.263424375656382		

Console:

```
> Rsqr_5=cor(df_testprice_hat5,df_testprice,use="na.or.complete")^2
> RMSE5_out
[1] 88.30055
> Rsqr_5
[1] 0.4797209
> RMSE_out =c(RMSE_out,RMSE5_out)
> RMSE_out
RMSE1_out RMSE2_out RMSE3_out RMSE4_out RMSE5_out
119.02947 104.70833 75.28542 69.80643 88.30055
```

Documentation pane (predict.randomForest):

predict method for random forest objects

Description

Prediction of test data using random forest.

Usage

```
# S3 method for class "randomForest"
predict(object, newdata, type="response",
       norm.res=TRUE, predict.all=FALSE, proximity=FALSE, nodes
       output, ...)
```

Arguments

object: an object of class randomForest, as that created by the function randomForest.

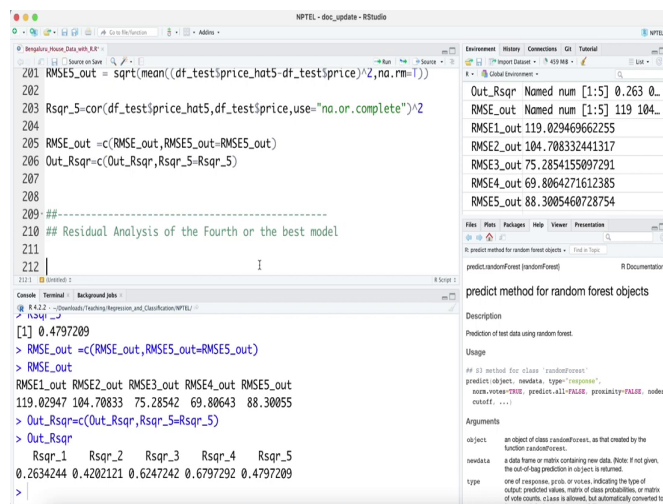
newdata: a data frame or matrix containing new data. Note: If not given, the out-of-bag prediction in object is returned.

type: one of response, prob, or nodes, indicating the type of output: predicted values, matrix of class probabilities, or matrix of node counts. If case is allowed, but automatically converted to



And R square 5 is been out (Refer Time: 21:40) R square 5 equal to Rsqr 5.

(Refer Slide Time: 21:49)



The screenshot shows the RStudio interface with the following content:

```
201 RMSE5_out = sqrt(mean((df_test$price_hat5-df_test$price)^2, na.rm=T))
202
203 Rsqr_5=cor(df_test$price_hat5,df_test$price,use="na.or.complete")^2
204
205 RMSE_out =c(RMSE_out,RMSE5_out)
206 Out_Rsqr=c(Out_Rsqr, Rsqr_5)
207
208
209 ##-----
210 ## Residual Analysis of the Fourth or the best model
211
212
```

Console output:

```
[1] 0.4797209
> RMSE_out =c(RMSE_out,RMSE5_out)
> RMSE_out
RMSE1_out RMSE2_out RMSE3_out RMSE4_out RMSE5_out
119.02947 104.70833 75.28542 69.80643 88.30055
> Out_Rsqr=c(Out_Rsqr, Rsqr_5)
> Out_Rsqr
Rsqr_1 Rsqr_2 Rsqr_3 Rsqr_4 Rsqr_5
0.2634244 0.4202121 0.6247242 0.6797292 0.4797209
```

Environment pane shows:

Object	Class	Attributes
Out_Rsqr	Named num	[1:5] 0.263 0.420 0.625 0.680 0.480
RMSE_out	Named num	[1:5] 119 104 75 69 88
RMSE1_out	num	119.029469662255
RMSE2_out	num	104.708332441317
RMSE3_out	num	75.2854155097291
RMSE4_out	num	69.8064271612385
RMSE5_out	num	88.3005460728754

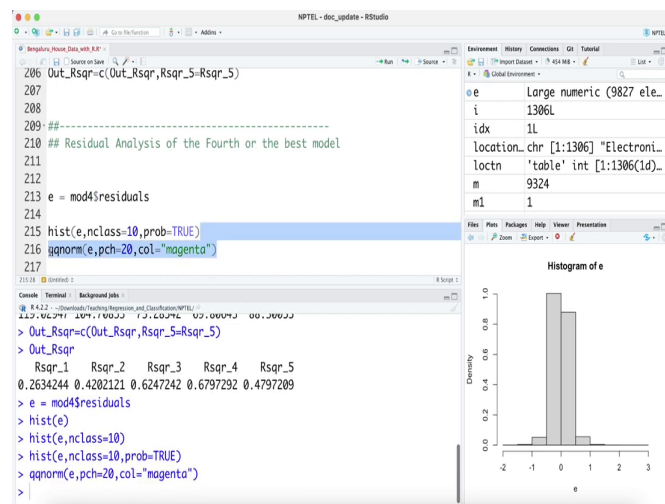
Documentation pane shows the `predict` method for random forest objects.



So, out of R square is also here also it is doing somewhere in between second model and third model. So far the third 4th model is the best model that we have find. Now, since 4th model is the best model what we will do we will try to do some residual analysis of the 4th model, ok, residual analysis of the 4th of the best model. So, you can if you want to take a break you can pause the video, you can take a break and come back and see it because now we are going to turn a bit, ok.

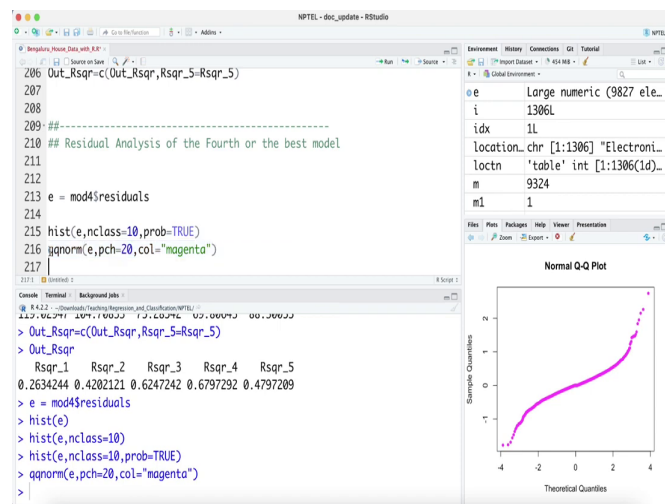
Alright, I hope you have taken your break and let us try to do the residual analysis of 4th model of the best model. And we hope that our 4th model has done reasonable job in terms of the fitting the; you know now here is the residuals.

(Refer Slide Time: 23:19)



So, let me just take it as e , ok. Now, first thing probably what we will do, I will just put `hist` equal to e . Let us see how the histogram looks like. Looks like its normal n class equal to 10 if I just do that, I do not know, alright. Probability equals to true. And then what we do is we can try to do `qqnorm` as e and then `pch` equal to 20 colour equal to say magento, ok.

(Refer Slide Time: 24:40)



That is not very good. So, that means, definitely the qq line lwd equal to 20 2. So that means, this error is heavily heavy tail distribution, error has definitely a heavy tail distribution. Though it is behaved like a bell shaped, it as doing reasonably bell shape, but it is definitely a as a heavy tail distribution on both sides, ok. On both sides there is some quite a bit of underestimation and overestimations are happening.

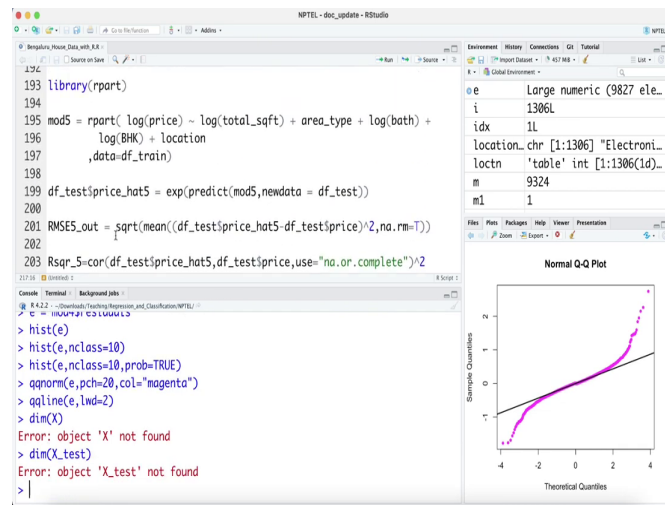
So, in this case I mean effectively if your prediction is doing well, probably you do not want to be your it should be fine. But at the same time if it is heavy tail; that means, you should be aware that there are cases where it is either doing heavy underestimation or heavy overestimation. So, that means, these are the points where this probably happened heavy over-estimations and these are the places where heavy underestimations have happened, right.

So, in this case, what can we do? I mean you have to be if you take it as with a pinch of salt then you say that, ok I am fine with this, right. And or you trying to put more features and you try to figure out if you can tame this underestimation or overestimations.

So, one way to put this underestimation and overestimation under control is put more and more new features which probably affect the price you brought those into the in your analysis. Just not location and bathrooms and how many BHKs, and you know all the only and total square feet area types. These are the only 5 variables effectively we have used. But other than that you should use other variables also, if you have available.

Now, for example, ready to move. You can try to add those variables as well. So, now if you want to still do the you know analysis, say whether if you are interested in doing inference statistical inference, so statistical inference will probably in this analysis will be valid. You can ask me why, because the sample size even in the training sample you have 9000 samples whereas, your dimension of your x is so maybe x test; wait a minute.

(Refer Slide Time: 28:02)



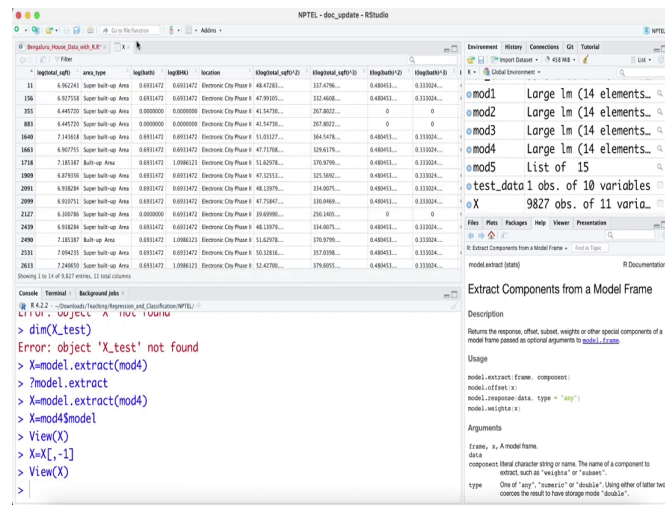
So, what you can do is sorry now what I have to do probably, I will tell you exactly what will be the model. So, model4, mod4, model dot extract, model dot extract, extract, model4, x and so, model dot extract and mod4. That is it, ok.

(Refer Slide Time: 29:07)



Or, one way of doing it is simply mod4 dollar model, if you do that. And so, first then is equal to x comma minus 1.

(Refer Slide Time: 29:48)



The screenshot shows an RStudio interface with the following components:

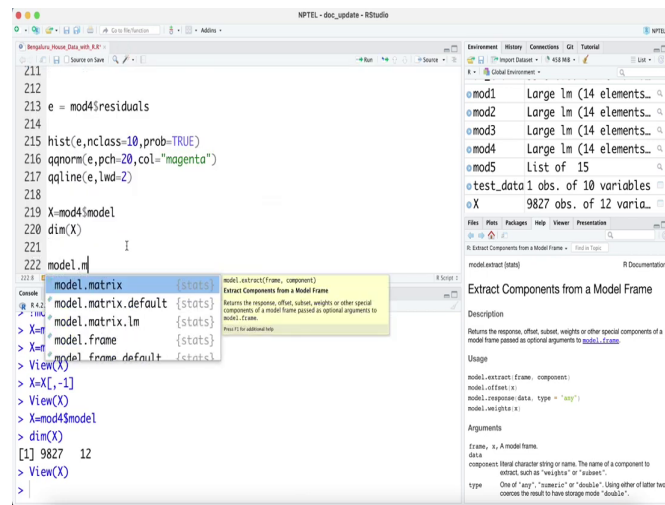
- Environment:** Lists several objects including 'mod1' through 'mod5', 'test_data', and 'X'. 'mod1' through 'mod5' are 'Large lm (14 elements...)' and 'test_data' is 'List of 15'. 'X' is '9827 obs. of 11 varia...'.
- Console:** Contains the following R code and an error message:


```
> dim(X_test)
Error: object 'X_test' not found
> X=model.extract(mod4)
> ?model.extract
> X=model.extract(mod4)
> X=model$X
> View(X)
> X=X[,1]
> View(X)
> 
```
- Help Window:** Displays the documentation for the `model.extract` function, including its description, usage, arguments, and type information.



So, the number of actually dimension of x, just run it, so 12. So, you have 9827 training samples that you are using that model 4 have used and there are only 12 parameters.

(Refer Slide Time: 30:28)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor on the left contains the following R code:

```
211
212
213 e = mod4$residuals
214
215 hist(e, nclass=10, prob=TRUE)
216 qqnorm(e, pch=20, col="magenta")
217 qqline(e, lwd=2)
218
219 X=mod4$model
220 dim(X)
221
222 model.matrix
```

A tooltip for `model.matrix` is visible, showing its usage: `model.matrix.default`, `model.matrix.lm`, `model.frame`, and `model.frame.default`. The console on the bottom left shows the following output:

```
> X=mod4$model
> dim(X)
[1] 9827 12
> View(X)
```

The environment pane on the right shows the following objects:

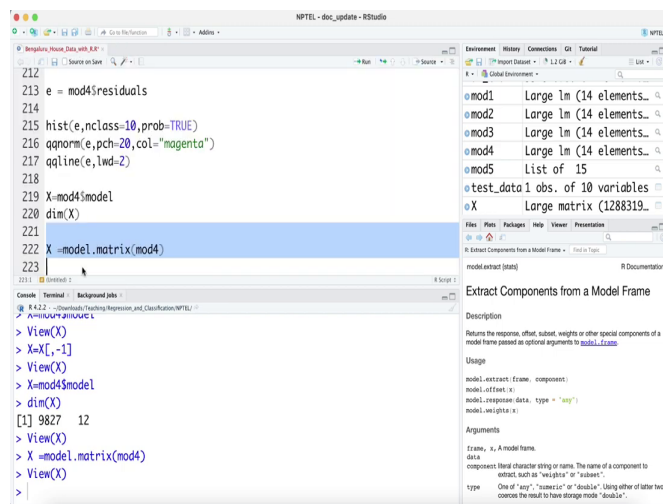
- mod1: Large lm (14 elements...)
- mod2: Large lm (14 elements...)
- mod3: Large lm (14 elements...)
- mod4: Large lm (14 elements...)
- mod5: List of 15
- test_data: 1 obs. of 10 variables
- X: 9827 obs. of 12 variables

The help pane on the right shows the documentation for `model.extract`, titled 'Extract Components from a Model Frame'.



But if you put all locations, ok; that is where you need the locations fine, alright. So, model dot matrix mod 4, I think if you just say x; and now if you are saying that we have intercept log square feet area type, and then you are getting all the locations dummy variables.

(Refer Slide Time: 30:57)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor contains the following R code:

```
212  
213 e = mod4$residuals  
214  
215 hist(e, nclass=10, prob=TRUE)  
216 qqnorm(e, pch=20, col="magenta")  
217 qqline(e, lwd=2)  
218  
219 X=mod4$model  
220 dim(X)  
221  
222 X=model.matrix(mod4)  
223
```

The environment pane on the right shows the following objects:

- mod1: Large lm (14 elements...)
- mod2: Large lm (14 elements...)
- mod3: Large lm (14 elements...)
- mod4: Large lm (14 elements...)
- mod5: List of 15
- test_data: 1 obs. of 10 variables
- X: Large matrix (1288319...)

The console shows the following output:

```
> View(X)  
> X=X[, -1]  
> View(X)  
> X=mod4$model  
> dim(X)  
[1] 9827 12  
> X=mod4$model  
> X=model.matrix(mod4)  
> View(X)  
>
```

The right pane shows the documentation for the `model.extract` function, which returns the response, offset, subset, weights or other special components of a model frame.

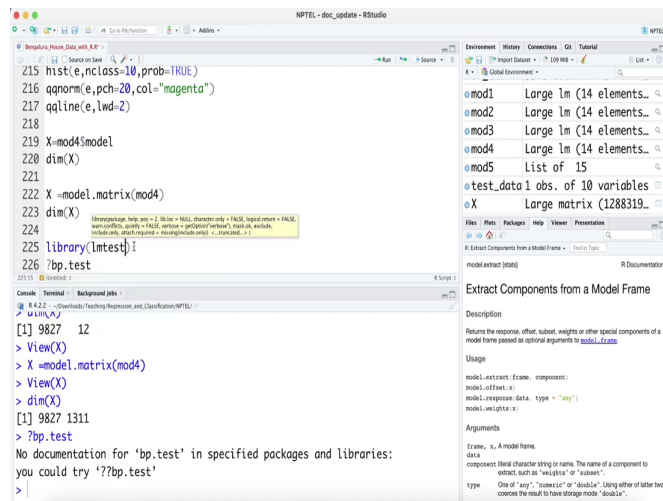


And then, if you put dimension of x, then what you will get is, ok 9827 samples and 1311 columns you are fitting it is a dimension. So, it is just 2th n x 10th a bit of a high dimension I would say, but not bad. So, you have about for each parameter you have about 10 samples. So, not bad, I would not say not bad.

So, probably, probably central limit theorem will kick in and you will be able to do the analysis reasonably, ok. I do not think you will be facing much of a trouble in this case. So, this the testing of hypothesis for the coefficients are not valid this argument probably will not be, I mean is ok, not be correct, but you can probably say reasonably that central limit theorem probably will kick in and your testing of hypothesis based on these coefficient of t value and should be fine.

But you have to be careful that there are certain cases where you are doing quite a bit of underestimation and overestimations. So, one thing one should do that you should do Breusch Pagan test bp test, right, bp test, ok.

(Refer Slide Time: 33:03)



```
215 hist(e,nclass=10,prob=TRUE)
216 qqnorm(e,pch=20,col="magenta")
217 qqline(e,lwd=2)
218
219 X=mod4$model
220 dim(X)
221
222 X=model.matrix(mod4)
223 dim(X)
224
225 library(lmtest)
226 ?bp.test
```

Console

```
[1] 9827 12
> View(X)
> X=model.matrix(mod4)
> View(X)
> dim(X)
[1] 9827 1311
> ?bp.test
No documentation for 'bp.test' in specified packages and libraries:
you could try '?bp.test'
```

Environment

Object	Class	Attributes
mod1	Large lm	(14 elements...)
mod2	Large lm	(14 elements...)
mod3	Large lm	(14 elements...)
mod4	Large lm	(14 elements...)
mod5	List of 15	
test_data	1 obs. of 10 variables	
X	Large matrix	(1288319...)

Files | Plots | Packages | Help | Viewer | Presentations

R Extract Components from a Model Frame

model.extract (stats)

R Documentation

Extract Components from a Model Frame

Description

Returns the response, offset, subset, weights or other special components of a model frame passed as optional arguments to [glm\(\)](#).

Usage

```
model.extract (Ename, component,
               model.offset, x,
               model.offset.data, type = "any")
model.weights(x)
```

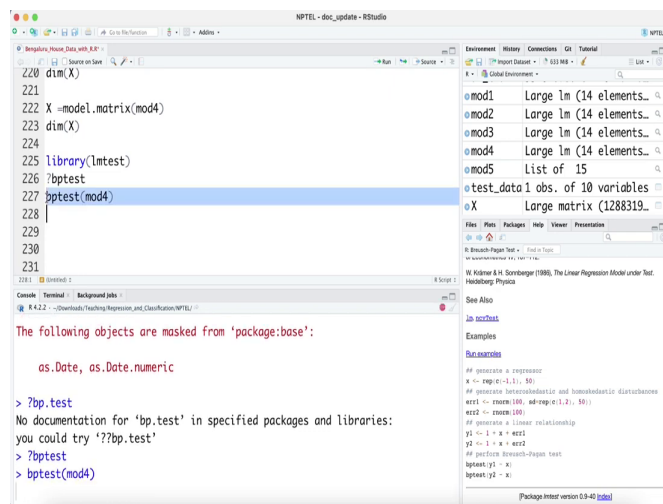
Arguments

Ename, x, A model frame
data
component literal character string or name. The name of a component to extract, such as "weights" or "offset".
type One of "any", "numeric" or "double". Using either of latter two coerces the result to have storage mode "double".



Now, I think this will come under library. I am not, if am not making, a mistake I think it is under library lm test I think bp test. Yes, Breusch Pagan test, lm test, correct, correct, ok.

(Refer Slide Time: 33:34)



The image shows a screenshot of the RStudio interface. The script editor on the left contains the following R code:

```
220 dim(X)
221
222 X = model.matrix(mod4)
223 dim(X)
224
225 library(lmtest)
226 ?bptest
227 bptest(mod4)
228
229
230
231
```

The console on the right shows the output of the `bptest(mod4)` command:

```
The following objects are masked from 'package:base':
  as.Date, as.Date.numeric

> ?bp.test
No documentation for 'bp.test' in specified packages and libraries:
you could try '?bp.test'
> ?bptest
> bptest(mod4)
```

The environment pane on the right shows the following objects:

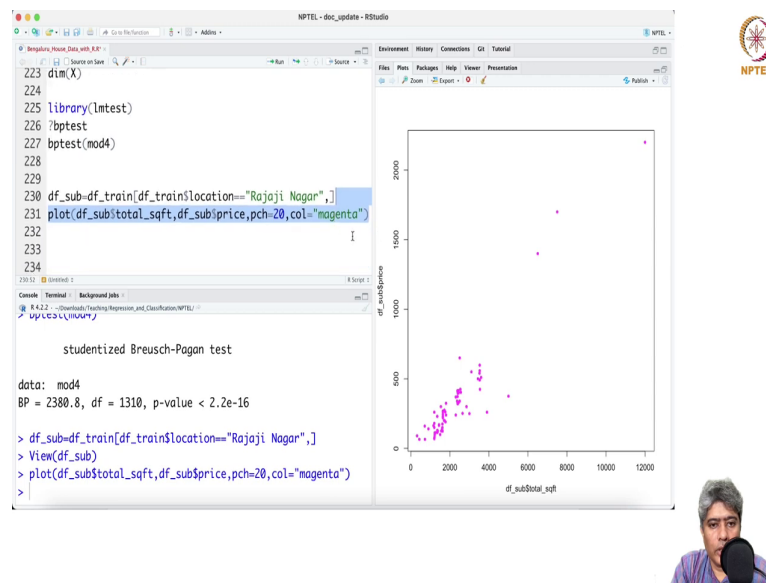
- mod1: Large lm (14 elements)
- mod2: Large lm (14 elements)
- mod3: Large lm (14 elements)
- mod4: Large lm (14 elements)
- mod5: List of 15
- test_data: 1 obs. of 10 variables
- X: Large matrix (1288319)



So, may be what I will just do that I will just make a mod4. So, it takes times in general, ok. So, even Breusch Pagan test is rejecting so that means, there is quite bit of heterogeneity is there, heterogeneity is there. So, let us try to see how the heterogeneities are there. Let me see. The location the coefficients. What we can do is very simple is, let us take few places like say Rajaji Nagar or something, right.

So, let us take this kind of places and we can try to take a df sub create a sub value and let us try to understand. So, this in these area where these are the values that we have.

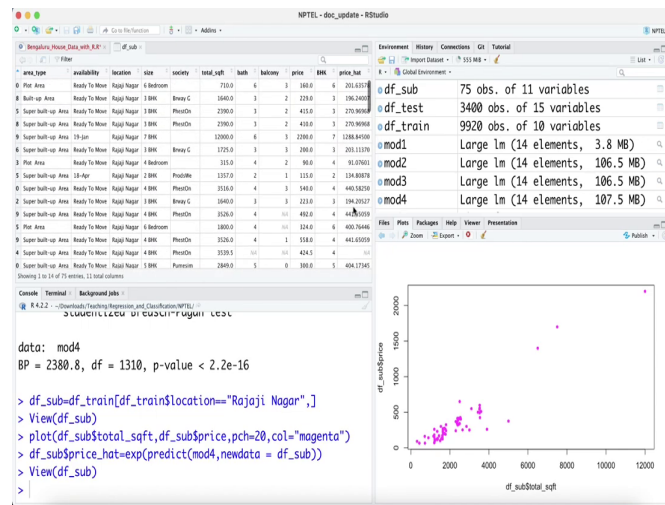
(Refer Slide Time: 35:23)



And for these values df_sub what we will have is, we can plot the df_sub dollar square feet comma df_sub dollar price df_sub equal 20 and color equal to magenta. So, this is pretty much data that we have. And then if we; so, this is the data that we have. So, this is the total square feet and this is the data. And now, if we do a df_sub in the df_sub dollar price hat, equal to exp.

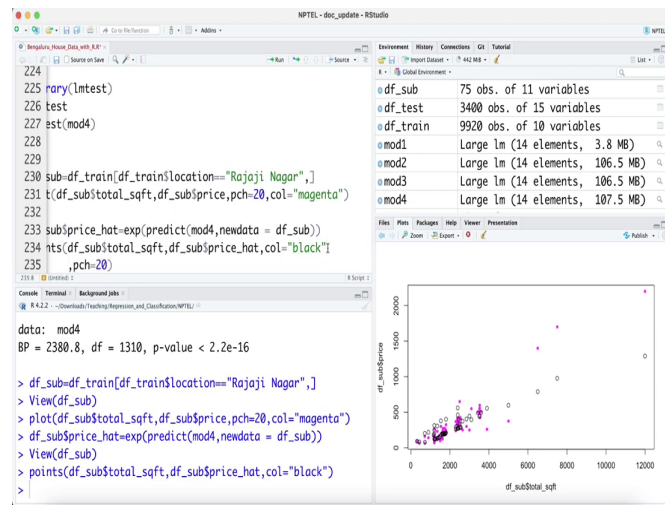
What we can do is mod 4, let us take mod 4 and new predicator which we have to first credit mod 4 comma new data equal to df_sub , just for Rajaji Nagar, we are doing the prediction for Rajaji Nagar in this case, ok. So, let us see df_sub . So, now, I have price, predicted price for these values.

(Refer Slide Time: 37:24)



So, most of the places I have values there are few cases where I do not have values like NA. So, I have NA is here, so which I cannot do much. And then what I can do is essentially points; I can plot is essentially this comma df price hat is color equal to black, alright. Let us try to get that pch equal to 20.

(Refer Slide Time: 38:28)

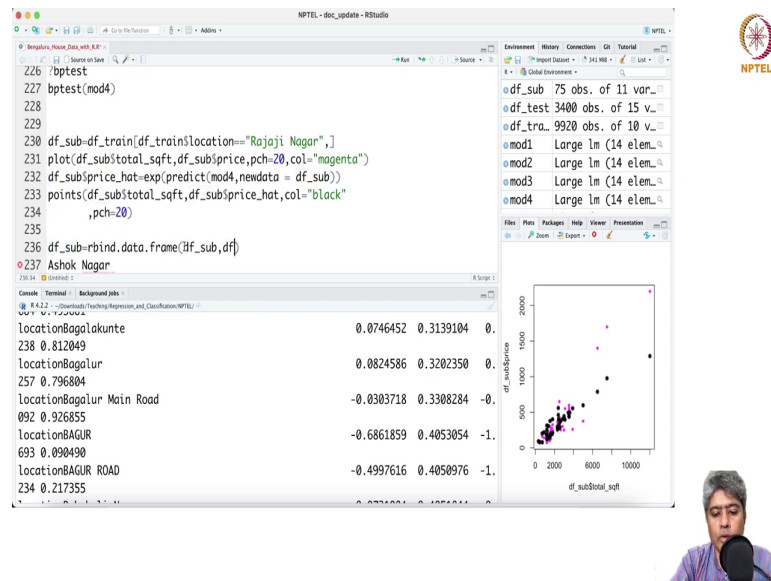


So, this is where we are doing and these are the points, this model missing these points, you can see that the large points this point is underestimating, clearly underestimating, ok. And that is why we are seeing for different here it is trying to do very good, because we have all the values and all these things, but here clearly it is bit of underestimating.

So, now similarly, if we just go and take another area. So, we did for say other than Rajaji Nagar. Here I think we have the Rajaji Nagar, alright.

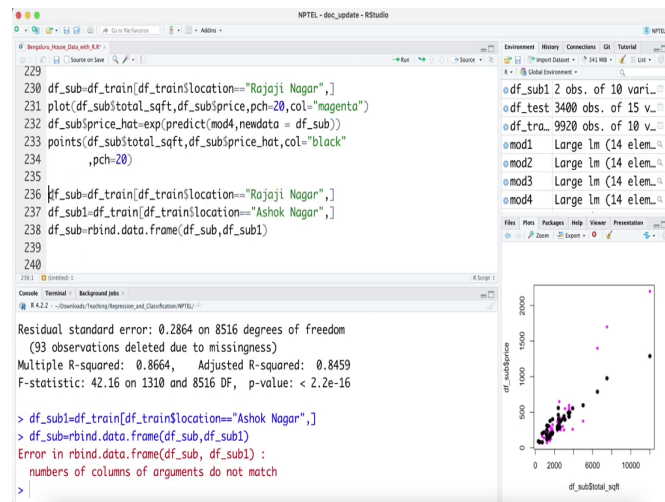


(Refer Slide Time: 40:26)



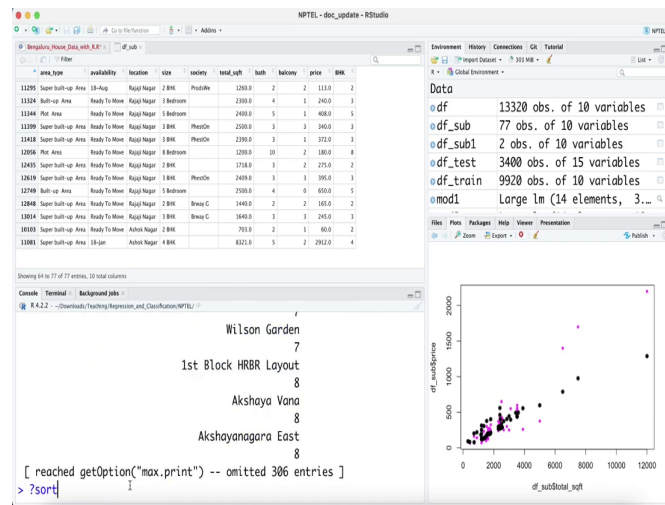
Now, you can tell me what is df sub 1. I have no I do not have any df sub 1, but I am going to make 1; that is df train and here and instead of Ashok Nagar. This is at Ashok Nagar. So, df sub 1 will have, right, ok, alright. So, ok I will do this, ok. Let me just do this guy.

(Refer Slide Time: 41:19)



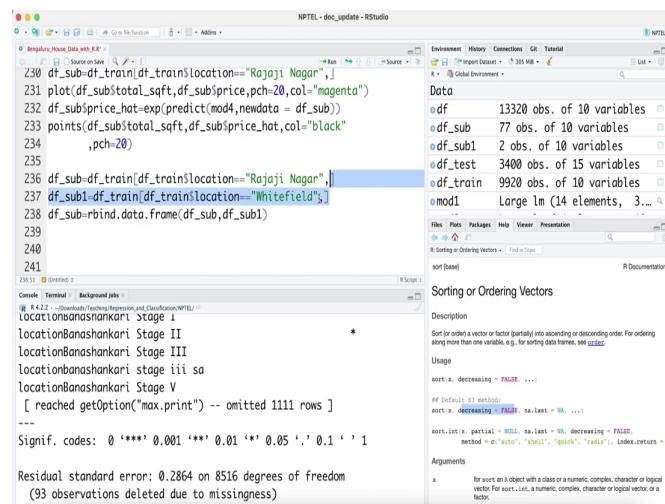
Run it once more and now it is fine. Now, if you look into the df sub, it has first Rajaji Nagar datas and then two Ashok Nagar, ok. That is where the problem probably, ok. So, what I will do is df dollar location, table, and sort; wait a minute sort descending is equal to true. Descending equal to true.

(Refer Slide Time: 42:11)



First, I do not know why it is not happening. So, I can, I want few more data set actually. I want a location with a little bit more data set.

(Refer Slide Time: 42:41)



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for data manipulation and plotting.


```

230 df_sub=df_train[df_train$location=="Rajaji Nagar",]
231 plot(df_sub$total_sqft,df_sub$price,pch=20,col="magenta")
232 df_sub$price_hat=exp(predict(mod4,newdata = df_sub))
233 points(df_sub$total_sqft,df_sub$price_hat,col="black",
234         ,pch=20)
235
236 df_sub=df_train[df_train$location=="Rajaji Nagar",]
237 df_sub1=df_train[df_train$location=="Whitefield",]
238 df_sub=rbind.data.frame(df_sub,df_sub1)
239
240
241
      
```
- Environment:** Lists objects in the workspace:
 - df: 13320 obs. of 10 variables
 - df_sub: 77 obs. of 10 variables
 - df_sub1: 2 obs. of 10 variables
 - df_test: 3400 obs. of 15 variables
 - df_train: 9920 obs. of 10 variables
 - mod1: Large lm (14 elements, 3...)
- Console:** Shows the output of the code execution, including a warning about reaching the maximum print limit and the residual standard error.


```

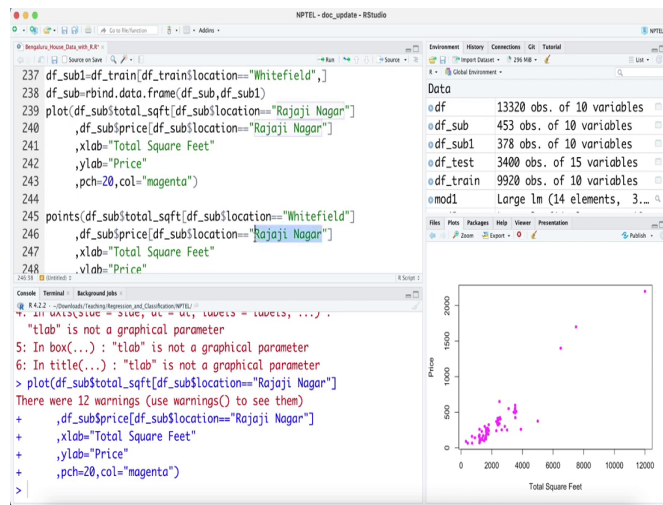
locationBanashankari Stage I
locationBanashankari Stage II
locationBanashankari Stage III
locationBanashankari stage iii sa
locationBanashankari Stage V
[ reached getOption("max.print") -- omitted 1111 rows ]
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2864 on 8516 degrees of freedom
(93 observations deleted due to missingness)
      
```
- Help:** Displays the documentation for the `sort` function, titled "Sorting or Ordering Vectors".



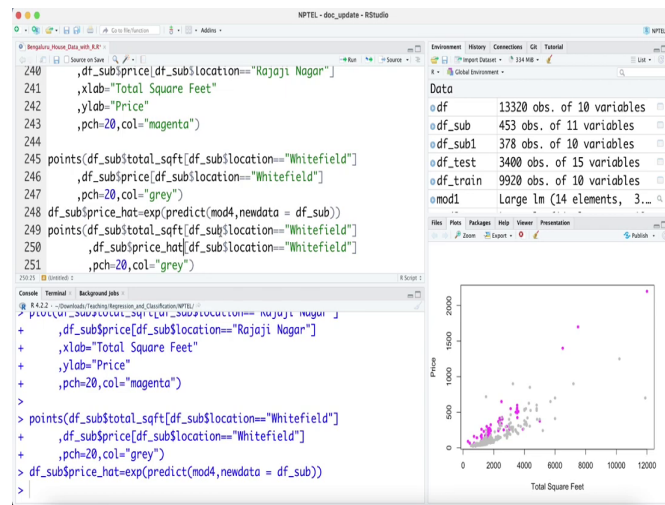
Summary equal to data, mod 4, maybe white field alright. So, maybe white field, ok. So, I have first Rajaji Nagar, then I have a bunch of white field, ok. Now, what I am going to do is first plot all the Rajaji Nagar cases; df sub (Refer Time: 43:51), ok. So, I have to put some x label to do this, total square feet square feet, and t lab equal to price, ok. Otherwise it is taking (Refer Time: 44:35), ok. So, I got that, alright. And then, I am going to put these points, points. And it is going to be white field, ok.

(Refer Slide Time: 45:00)



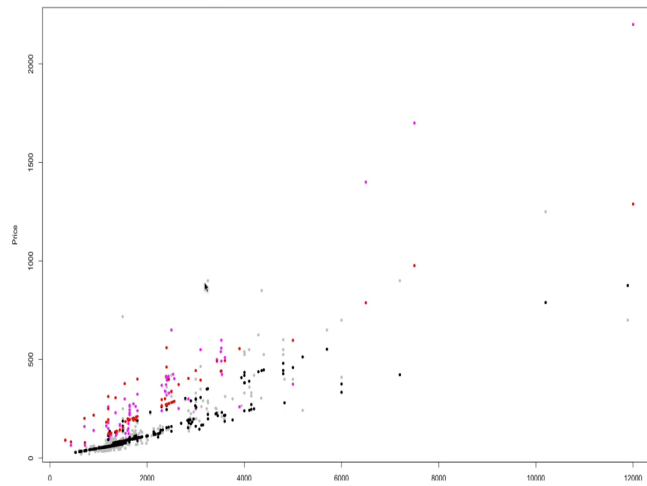
So, this is a bit of a different issue or grey. I think I can put a grey that will be probably better number, ok. Now, if you do the prediction, now if you do the prediction, and first, we are going to put the points, bunch of points for instead of price, 50 price hat.

(Refer Slide Time: 46:20)



But instead of grey using black, and ok. And here you have bunch of red and Instead of this, you just see Rajaji Nagar set. So, let me just zoom it. So, the magenta is the actual numbers.

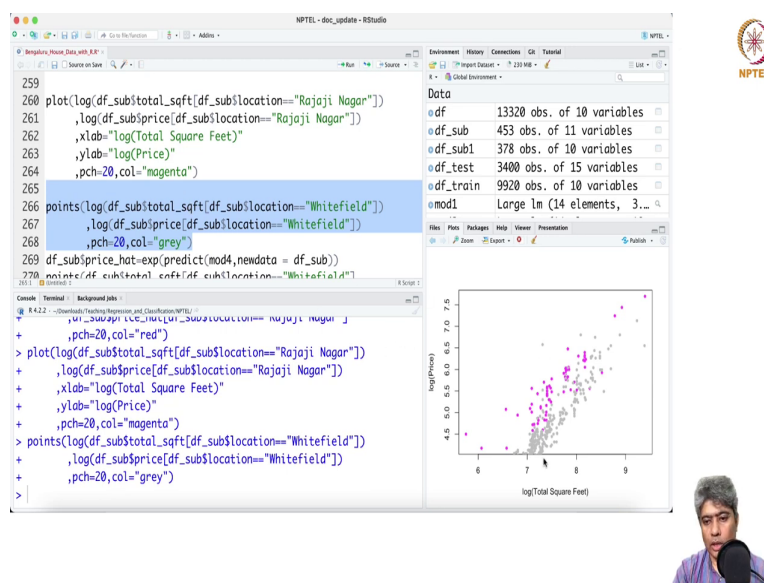
(Refer Slide Time: 47:12)



And you can see that rates are the actual the predictions. And then there is quite a bit of underestimations are happening. Whereas, the model here, it is doing pretty good in terms of you can see this, it is actually able to pick this, you know this kind of log linear behavior to an extent, right. It is quite great, actually in that sense.

And then we can try to put the same thing using maybe. Another plot I will just create out of it with log linear. Since, these are all long linear, I will try to put it in log linear scale. So, log of this and log of price in log scale total square feet log scale log of price. So, if you just plot this, so you can see that this is better and then if you just plot log of this and log of that; this is also, to an extent.

(Refer Slide Time: 49:07)

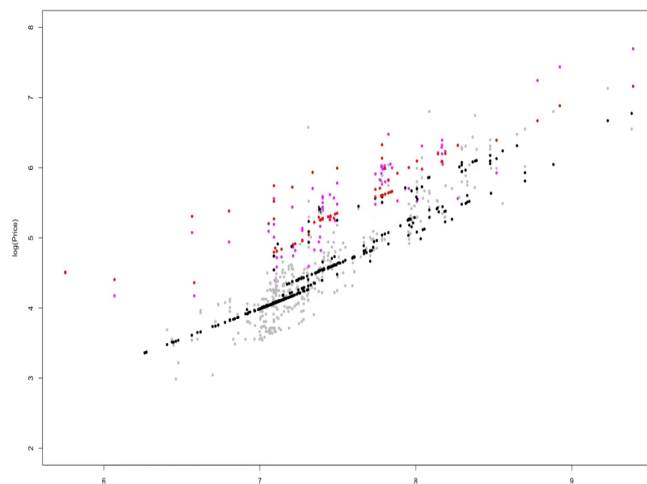


But so, but I need to, I think quite a few are actually down. So, maybe I will just increase the y limit, c from maybe from 2 to 8. I will just get 2 to 8. You can see systematically, the price on the white field is like, lower than the Rajaji Nagar. You can I mean, clearly this is the Rajaji Nagar. These points are in Rajaji Nagar. And whereas, these points are in the white field. Now, white field also when it, when the size of the house increases that time in terms of total square feet, the gap they are kind of overlapping.

At the, but at when the size of the flat in terms of total square feet decreases that time you can see there is a clear cut premium that you are paying for the Rajaji Nagar compared to Whitefield. For a large house, larger house this premium on gap is not that much. The slope is slightly different.

Whereas, now you do this thing and then we have to plot this in the clock still remember that. So, now, this is the; this is the predicted values of for the Whitefield. These are the predicted values of the Whitefield. So, Whitefield does reasonably, ok.

(Refer Slide Time: 51:33)



And if you just do this for the Rajaji Nagar with red now if you see that clearly the Rajaji Nagar's are the red points. And for the Rajaji Nagar the points are bit higher on the side, ok whereas, clearly they are on an average it is higher compared to the Whitefield. So, it is much more clear when you plot it in the log scale.

So, and that means, whenever you are doing modeling the housing price, in fact, any economic activity any economic activity which related to price, my strong and strong recommendation is always work with log price. And along the other variable with the log price, the feature of the economic activity also, like square feet in this case square feet

number of bathroom number of you know bedrooms, these are all part of the square feet and features, so take the log transformations.

The log transformations works in economics works amazingly while, beautiful because everything in economics sort of works; there is called economies of skill the economies of skill happens in the exponential scale. So, when it becomes; so, that is one of the reason you know if you can afford you should when if somebody is investing they say that you know you try to invest in larger apartment perhaps.

Because your appreciation will be better. People sometimes say that that because of the economies of skill. So, economies of skill gets captured in the log transformations. So, and we can see that visualization also clearly explain that. So, with this I will stop here. I will hope you enjoyed these particular weeks, complete week on the complete hands-on. One way to learn regression and classification using hands-on data analysis only.

I would recommend you try other models, try to apply more engineering, feature engineered features try to do some stepwise variable selections maybe Lasso, Ridge regression, try to check the multi-collinearity. I have not done any of those yet. So, if you do that on the; so far model fold looks like the best.

Try to apply all those advanced technique that we I have taught you maybe random forest on the top of this. I believe that will give you a very good results. You can improve the accuracy a much better. You can reduce the heaviness of the residual that is overestimations and under estimation to a great extent. And I hope you will enjoy this data analysis to a great extent.

So, I am going to share this code on NPTEL platform. Please download this code and try to run on your side, and check if you are able to reproduce these results. And then, try to improve these results using more feature engineer, engineered features more step-by selections very and then try to look into the variance inflation factor, maybe Ridge correction, Lasso correction you can try. All these will help you to improve the you know prediction accuracy of the model.

I hope you enjoyed it.

Thank you very much.

See you in the next week.