UNIT 3

Cross Validation

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code calculates the loss using holdout validation.

load data

cvpt = cvpartition(groupData.group,"HoldOut",0.3);

trainData = groupData(training(cvpt),:);

testData = groupData(test(cvpt),:);

holdoutMdl = fitcdiscr(trainData,"group");

holdoutLoss = loss(holdoutMdl,testData)

holdoutLoss = 0.0379

**Task 1**

cvpt=cvpartition(groupData.group,"KFold",5);

**Task 2**

mdl = fitcdiscr(groupData,"group","CVPartition",cvpt);

**Task 3**

kfLoss=kfoldLoss(mdl)

kfLoss = 0.0761

**Task 4**

mdl2 = fitcdiscr(groupData,"group","Leaveout","on");

kfLoss2 = kfoldLoss(mdl2)

kfLoss2 = 0.0870

**Further Practice**

# Heart Disease Analysis

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

rng(0)

heartData = readtable("heartdataNumMulti.txt");

The script holdoutAnalysis calculates the loss for different models using holdout validation.

holdoutAnalysis

## **Task 1**

Create models and calculate loss

% Create partition

part = cvpartition(heartData.HeartDisease,"KFold",7);

% k-NN

mdlKnn = fitcknn(heartData,"HeartDisease","NumNeighbors",5,"CVPartition",part);

lossKnn = kfoldLoss(mdlKnn);

% Discriminant analysis

mdlDa = fitcdiscr(heartData,"HeartDisease","CVPartition",part);

lossDa = kfoldLoss(mdlDa);

Display the results.

KFoldLoss = [lossKnn;lossDa];

results = table(KFoldLoss);

results.Properties.RowNames = ["kNN" "Discriminant Analysis"];

disp("Seven-fold cross-validated results")

disp(results)

## **Further Practice**

# Heart Disease Analysis

This code loads and partitions the data.

heartData = readtable("heartdataNumMulti.txt");

rng(1234)

pt = cvpartition(heartData.HeartDisease,"Holdout",0.2);

heartTrain = heartData(training(pt),:);

heartTest = heartData(~training(pt),:);

This code fits a k-NN model to the training data and calculates the loss.

m = fitcknn(heartTrain,"HeartDisease");

trainLoss = resubLoss(m)

trainLoss = 0

testLoss = loss(m,heartTest)

testLoss = 0.5833

## **Task 1**

Set optimization options.

cvpt = cvpartition(heartTrain.HeartDisease,"KFold",10);

opt = struct("CVPartition",cvpt,"MaxObjectiveEvaluations",20);

## **Optimize hyperparameters**

Calculate loss.

Principal Component Analysis

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the data.

rng(0)

load data

dataOrig

dataOrig = 154×6 table

|  | **P1** | **P2** | **P3** | **P4** | **P5** | **R** |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | -1.2539 | -1.1178 | -4.1511 | 1.2458 | 2.1275 | B |
| **2** | 1.1176 | -1.7874 | -0.4522 | -2.1508 | 0.5305 | C |
| **3** | -2.0149 | -0.0593 | -3.3889 | 2.7783 | 1.4625 | B |
| **4** | 2.0259 | 2.6283 | 2.2820 | 1.6122 | 3.0237 | A |
| **5** | -0.6001 | -1.3114 | -1.6471 | -0.8356 | 1.9933 | B |
| **6** | 0.9108 | 1.4592 | 2.0335 | 3.8287 | -1.2584 | A |
| **7** | -0.6914 | -1.2603 | -4.4747 | 1.9510 | 1.2883 | B |
| **8** | 0.8586 | -1.4930 | -1.5366 | 1.9835 | 2.6220 | B |
| **9** | -0.5116 | 3.8282 | 3.0187 | -4.7132 | -0.3218 | A |
| **10** | 0.8554 | 1.3721 | 0.8799 | -1.7261 | 1.9298 | A |
| **11** | 2.0108 | 1.0084 | 0.6663 | 0.4765 | 3.2294 | A |
| **12** | 0.9903 | 2.0877 | 1.8697 | -10.5791 | -1.1924 | A |
| **13** | -2.2038 | -0.3924 | -1.7183 | 1.2358 | 2.6251 | B |
| **14** | -1.2551 | -1.2900 | -1.8300 | 0.8597 | 1.6396 | B |
| **15** | -0.2214 | 1.4063 | 0.5509 | -3.3436 | 1.0012 | A |
| **16** | -0.8494 | 0.9453 | 1.7221 | 1.0876 | 1.0414 | A |
| **17** | -0.0885 | 0.6134 | 1.5238 | 2.9290 | 1.0799 | A |
| **18** | 0.1385 | -2.0203 | -1.5750 | -0.3429 | 2.9986 | B |
| **19** | -1.1319 | -0.6499 | -1.4379 | -0.8543 | 1.1252 | B |
| **20** | 1.9496 | 2.2062 | 0.2498 | -0.0290 | 1.4227 | A |
| **21** | 1.5252 | 1.8602 | 2.8620 | -4.3732 | 0.0515 | A |
| **22** | -0.7076 | 2.1399 | 1.7143 | -0.5725 | -0.6702 | A |
| **23** | -0.1507 | -0.2149 | 0.0984 | -2.7173 | -0.2976 | C |
| **24** | 0.1454 | -1.2099 | 0.5581 | 0.9308 | 2.0802 | B |
| **25** | -0.5587 | -0.8903 | -0.8467 | 2.3312 | 1.9662 | B |
| **26** | 0.3868 | 1.4751 | 2.7015 | -1.2146 | 0.2658 | A |
| **27** | -1.6291 | -2.6989 | -3.5272 | 0.4927 | 2.2817 | B |
| **28** | 2.9927 | 1.8614 | 2.0229 | -4.2564 | 0.4100 | A |
| **29** | -2.3981 | 0.1287 | -0.8702 | 0.5811 | 3.2030 | B |
| **30** | -0.8356 | 0.2616 | -1.1921 | 1.8998 | 1.9980 | B |
| **31** | 2.5461 | 1.8627 | 0.6680 | -4.9550 | 2.0205 | A |
| **32** | -0.8078 | -0.5982 | -1.5888 | -0.6045 | 1.3356 | B |
| **33** | -1.0942 | -1.3268 | -1.3778 | 2.4580 | 0.6591 | B |
| **34** | -1.9047 | -0.8907 | -1.6113 | 0.4777 | 2.3009 | B |
| **35** | -0.2523 | -0.5246 | -1.4162 | 0.6999 | 2.6504 | B |
| **36** | 0.4790 | 0.8045 | 0.4464 | 0.5860 | 2.8660 | B |
| **37** | -1.7841 | -1.3086 | -0.5196 | 0.5619 | 2.5717 | B |
| **38** | 0.0078 | 0.0360 | 0.1443 | -0.0933 | 1.7642 | B |
| **39** | -1.2730 | 0.1741 | -3.3958 | 2.0294 | 2.1059 | B |
| **40** | 1.4514 | 1.6150 | 3.6555 | -2.4060 | 0.3096 | A |
| **41** | -1.4204 | 2.5942 | 1.1521 | 2.9044 | 2.2503 | A |
| **42** | -2.8054 | -2.0966 | -2.2800 | 0.5913 | 2.1506 | B |
| **43** | -0.6725 | -2.4814 | -2.8541 | 1.6293 | 2.2062 | B |
| **44** | 0.9575 | 2.3465 | 1.1345 | -1.8259 | -0.6118 | A |
| **45** | 1.4156 | 1.1756 | 2.5201 | 4.3193 | 2.0061 | A |
| **46** | -0.7170 | 1.2419 | 2.3914 | -2.0137 | -1.4863 | A |
| **47** | 0.0655 | -0.8897 | 1.8448 | -2.9461 | -0.0850 | C |
| **48** | -0.3353 | -0.8445 | -1.7498 | 0.7870 | 1.4501 | B |
| **49** | -1.8608 | -1.7231 | -2.8606 | 0.5617 | 1.9692 | B |
| **50** | -3.1237 | 1.4245 | -2.0461 | 0.5664 | 2.5046 | B |
| **51** | -0.1743 | 0.8779 | -4.8595 | 1.7512 | 0.8413 | B |
| **52** | 1.2661 | -2.2659 | 2.1032 | -2.2301 | 0.3724 | C |
| **53** | 2.5802 | 4.0563 | 2.4550 | 0.1078 | 1.6770 | A |
| **54** | 1.1094 | 2.9578 | 2.3335 | -2.6059 | 0.9292 | A |
| **55** | -0.2201 | -0.1198 | 2.5378 | -2.5157 | 1.0648 | C |
| **56** | 0.9984 | 2.5479 | 2.2117 | -1.2662 | -1.8381 | C |
| **57** | -0.8864 | -0.3887 | -2.8724 | 1.9726 | 2.4537 | B |
| **58** | -1.9300 | -1.5700 | -3.4043 | 1.0668 | 1.5122 | B |
| **59** | -1.4809 | -1.6568 | -2.9294 | 2.0128 | 2.7312 | B |
| **60** | 0.5763 | -0.8731 | -0.9258 | 0.6549 | 2.9934 | B |
| **61** | 0.8395 | -1.5789 | 1.2877 | -0.6251 | -1.8041 | C |
| **62** | 3.3055 | 1.4312 | 1.0208 | -0.1734 | -0.2128 | A |
| **63** | 1.5565 | -0.0656 | 5.3056 | -2.8508 | 0.6939 | C |
| **64** | 0.2996 | -2.0051 | -0.0288 | -1.7536 | -1.9908 | C |
| **65** | -0.0484 | -0.9251 | 1.7185 | -2.8546 | -2.9590 | C |
| **66** | -0.1448 | 1.5284 | 1.8668 | -1.6079 | 1.9248 | A |
| **67** | 2.9379 | 0.7310 | 1.7062 | 4.0706 | 1.3502 | A |
| **68** | -3.1638 | 2.1325 | 1.2855 | -1.8667 | 1.0000 | A |
| **69** | -1.8223 | 0.4702 | -1.8403 | 2.0289 | 1.9735 | B |
| **70** | 1.4424 | 2.2146 | 2.3075 | -1.4931 | 0.3484 | A |
| **71** | -0.3513 | -2.0360 | -2.0888 | 1.3893 | 2.1018 | B |
| **72** | -0.2153 | -0.4735 | -3.0424 | 3.0034 | 2.4118 | B |
| **73** | -1.0867 | -2.3270 | -0.7294 | 1.7059 | 2.2458 | B |
| **74** | -2.0570 | -0.4061 | -0.7738 | 1.6489 | 2.4245 | B |
| **75** | 3.1675 | 3.0159 | 2.1837 | -2.4452 | -1.3193 | A |
| **76** | 0.7326 | 0.6361 | 1.6070 | -2.1223 | 0.2535 | C |
| **77** | -2.4286 | -0.3008 | -3.0234 | 1.0700 | 2.3895 | B |
| **78** | 2.2291 | -2.0717 | 0.0597 | -2.7686 | -0.4333 | C |
| **79** | 0.6958 | -1.7976 | -0.6547 | -3.0086 | -0.8654 | C |
| **80** | -0.8630 | -1.8759 | -2.0776 | -0.2833 | 2.3675 | B |
| **81** | 1.7814 | 1.2518 | 2.5528 | -5.9491 | 1.4494 | A |
| **82** | -0.4638 | -1.9047 | -1.2285 | 3.7873 | 3.2348 | B |
| **83** | -1.2919 | -0.6801 | -3.5821 | -1.3290 | 1.9900 | B |
| **84** | 1.0658 | 0.5040 | 3.8135 | -2.5982 | -1.6644 | C |
| **85** | 0.6394 | -0.5512 | 2.7611 | -2.3376 | -2.6876 | C |
| **86** | 1.8877 | 3.2105 | 2.7914 | 0.2393 | -0.9488 | A |
| **87** | 6.0606 | 3.6620 | 1.6462 | -1.6881 | 1.2323 | A |
| **88** | -2.2706 | -1.3158 | -2.6011 | 0.7815 | 2.2925 | B |
| **89** | 1.2038 | -0.0845 | 1.5679 | -1.7689 | 1.8992 | C |
| **90** | -0.5747 | 2.6795 | 2.4517 | -0.9606 | 1.5812 | A |
| **91** | 1.1778 | -2.4831 | -0.6676 | 1.3165 | 2.3365 | B |
| **92** | 1.4845 | 1.2493 | -0.0518 | -1.6173 | 0.9692 | A |
| **93** | -0.2231 | 1.5755 | 0.7429 | -0.1270 | 2.1921 | A |
| **94** | -0.6768 | -1.5408 | -1.1580 | 0.7299 | 2.1729 | B |
| **95** | -0.4368 | -1.0638 | -1.3777 | 0.4684 | 2.1887 | B |
| **96** | 0.9690 | -2.8221 | 3.1952 | -2.2101 | -3.7673 | C |
| **97** | 2.0280 | 1.8277 | -0.0026 | -1.4619 | 1.3271 | A |
| **98** | 1.0779 | 0.4829 | 0.4539 | -2.6339 | -2.3999 | C |
| **99** | 1.4599 | 0.6331 | 1.7752 | -1.7875 | 1.9111 | A |
| **100** | -0.5402 | 2.5835 | 1.1513 | -1.0917 | 1.8577 | A |
| **⋮** |  |  |  |  |  |  |

This code fits a 10-fold cross-validated Naive Bayes model to the original data and calculates the loss.

mdlOrig = fitcnb(dataOrig,"R","KFold",10);

kfoldLoss(mdlOrig)

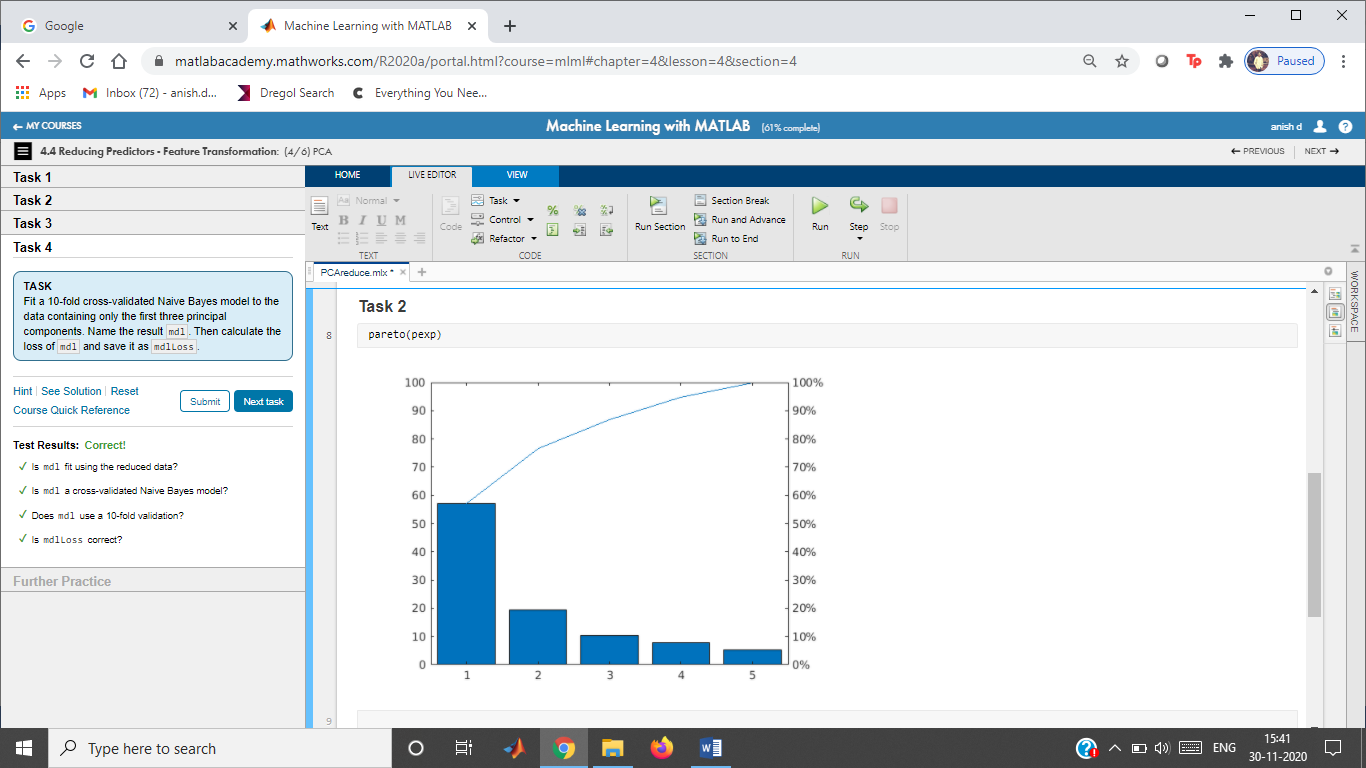
ans = 0.0195

**Task 1**

[~,scrs,~,~,pexp] = pca(dataOrig{:,1:end-1});

**Task 2**

pareto(pexp)



**Task 3**

dataRed = scrs(:,1:3);

**Task 4**

mdl = fitcnb(dataRed,dataOrig.R,"KFold",10);

mdlLoss = kfoldLoss(mdl)

mdlLoss = 0.0455

**Further Practice**

Heart Disease Analysis

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data and splits the data into predictors (numData) and response (resp).

rng(0)

heartData = readtable("heartDataNum.txt");

vars = heartData.Properties.VariableNames(1:end-1)

vars = 1×11 cell

'Age' 'Cholesterol' 'ExerciseDuration' 'METs' 'RestingHeartRate' 'RestingSystolic' 'RestingDiastolic' 'MaxHeartRate' 'PeakExSystolic' 'PeakExDiastolic' 'InducedSTDep'

numData = heartData{:,1:end-1};

resp = categorical(heartData.HeartDisease);

This code fits a 10-fold cross-validated Naive Bayes model using the kernel distribution to the original data and calculates the loss.

mdlOrig = fitcnb(numData,resp,"DistributionNames","kernel","KFold",10);

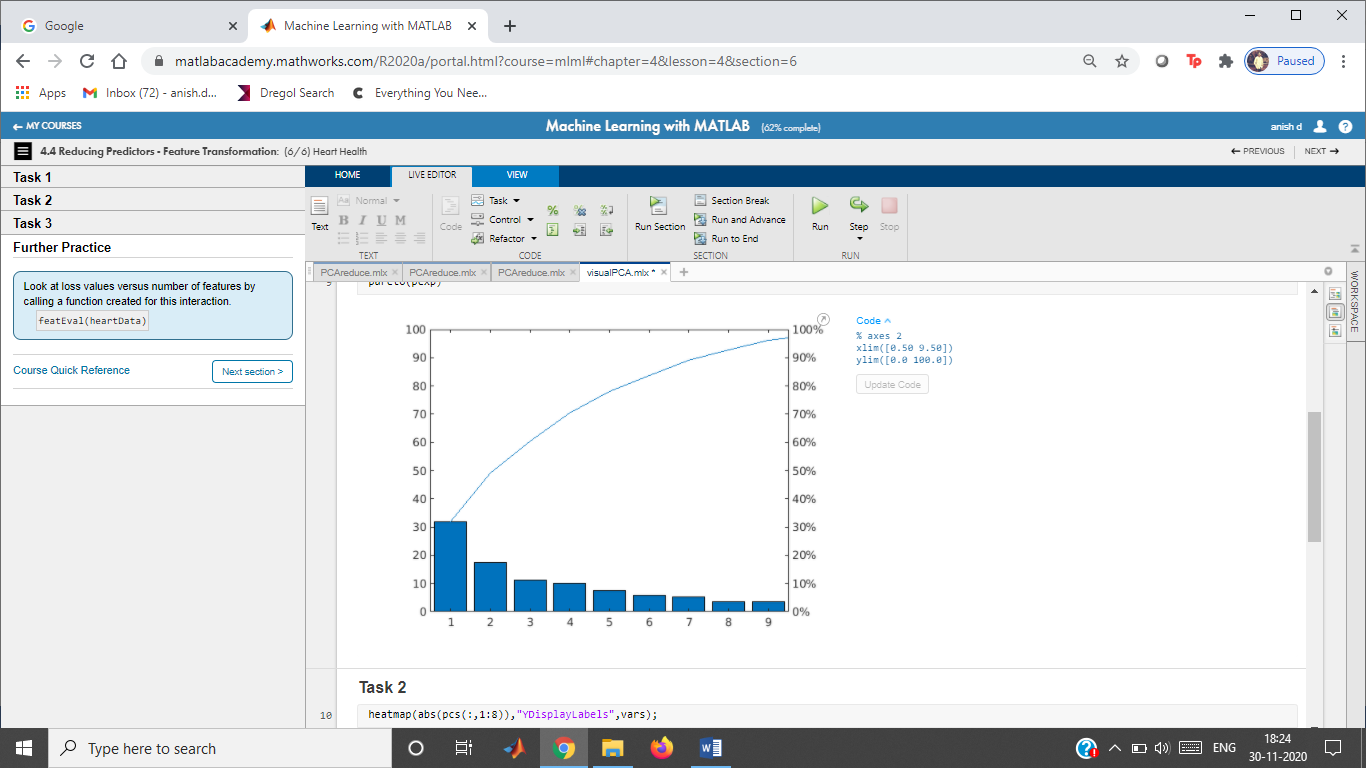
lossOrig = kfoldLoss(mdlOrig)

lossOrig = 0.2740

**Task 1**

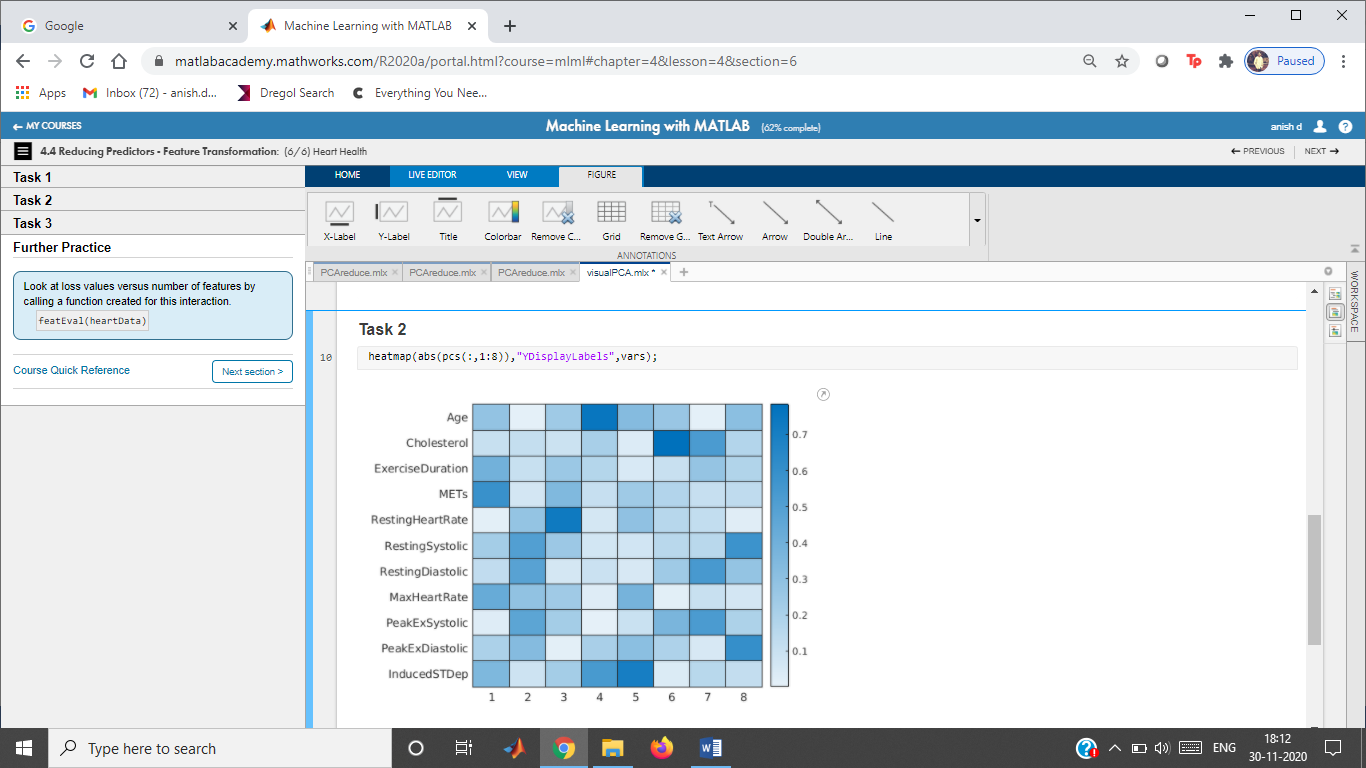
[pcs,scrs,~,~,pexp] = pca(numData);

pareto(pexp)



**Task 2**

heatmap(abs(pcs(:,1:8)),"YDisplayLabels",vars);



**Task 3**

mdl = fitcnb(scrs(:,1:8),resp,"DistributionNames","kernel","KFold",10);

pcaLoss = kfoldLoss(mdl)

pcaLoss = 0.2834

Feature Selection with Categorical Data

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the data.

rng(0)

load data.mat

data

data = 154×6 table

|  | **size** | **score** | **center** | **comparison** | **grit** | **bin** |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | -0.7429 | -2.3981 | -0.4037 | 1.7989 | ultra | B |
| **2** | 0.5697 | 0.3728 | 2.9002 | -1.5993 | super | C |
| **3** | -1.5900 | -3.1321 | -3.2852 | 0.3593 | extra | B |
| **4** | 0.7101 | 0.9942 | 3.1275 | -1.6632 | extra | A |
| **5** | -2.2128 | -2.4286 | -1.0112 | 0.2764 | ultra | B |
| **6** | 2.2193 | 3.0919 | 1.7275 | -4.5343 | super | A |
| **7** | 0.8244 | 2.3452 | 2.3502 | 0.9770 | very | A |
| **8** | -0.8994 | -1.0942 | -2.4622 | 1.4400 | super | B |
| **9** | -3.4863 | 0.4790 | 0.5519 | 2.7298 | extra | B |
| **10** | 3.0342 | 1.9417 | 3.3514 | 3.8385 | extra | A |
| **11** | 1.4424 | 2.2146 | 2.3075 | -1.4931 | very | A |
| **12** | 3.1068 | 1.8746 | 1.7009 | -0.4206 | extra | A |
| **13** | 1.5252 | 1.8602 | 2.8620 | -4.3732 | super | A |
| **14** | -1.6509 | -0.5587 | -1.8449 | 1.8155 | super | B |
| **15** | 0.2503 | -0.9148 | -0.8424 | 0.6925 | extra | B |
| **16** | -0.5747 | 2.6795 | 2.4517 | -0.9606 | super | A |
| **17** | 6.0606 | 3.6620 | 1.6462 | -1.6881 | very | A |
| **18** | 2.5802 | 4.0563 | 2.4550 | 0.1078 | super | A |
| **19** | -1.0549 | 0.5763 | -1.8205 | 1.7596 | extra | B |
| **20** | -0.5736 | -0.1743 | 0.6557 | -1.0220 | super | B |
| **21** | 3.3055 | 1.4312 | 1.0208 | -0.1734 | very | A |
| **22** | -0.5402 | 2.5835 | 1.1513 | -1.0917 | super | A |
| **23** | 1.6914 | 2.4926 | 0.8436 | -3.9772 | very | A |
| **24** | 1.0495 | -0.7503 | 0.5818 | -2.5499 | super | C |
| **25** | 0.0826 | 0.1385 | -3.4429 | 1.3005 | extra | B |
| **26** | -0.4054 | -0.6725 | -4.0950 | 0.3961 | super | B |
| **27** | 4.9166 | 1.5647 | 1.1764 | -1.9585 | very | A |
| **28** | 0.0061 | -3.4969 | -2.6321 | 0.6269 | super | B |
| **29** | -0.7076 | 2.1399 | 1.7143 | -0.5725 | very | A |
| **30** | -0.6498 | -0.3353 | -1.7801 | 1.1769 | extra | B |
| **31** | -0.0702 | -0.1190 | -2.4138 | 0.8682 | ultra | B |
| **32** | 1.1094 | 2.9578 | 2.3335 | -2.6059 | very | A |
| **33** | 0.2269 | -1.5924 | 3.1240 | -1.3890 | extra | C |
| **34** | 0.8809 | 0.2725 | 3.2226 | -1.7183 | super | C |
| **35** | -1.9485 | -2.0570 | -1.1601 | 1.8671 | super | B |
| **36** | -0.7602 | -0.6768 | -2.7648 | 1.5954 | ultra | B |
| **37** | -0.1423 | -2.4694 | -4.0379 | 1.1837 | extra | B |
| **38** | -0.9988 | -1.4677 | 0.5654 | 1.6011 | extra | B |
| **39** | -1.2365 | -1.4711 | -0.8865 | 1.6125 | extra | B |
| **40** | -1.6904 | -1.7841 | -2.4365 | 2.0468 | super | B |
| **41** | 0.9575 | 2.3465 | 1.1345 | -1.8259 | very | A |
| **42** | 1.4514 | 1.6150 | 3.6555 | -2.4060 | extra | A |
| **43** | -1.0708 | -1.1249 | -1.5587 | 1.0923 | extra | B |
| **44** | -2.6118 | -1.6045 | -2.2556 | 0.7617 | ultra | B |
| **45** | 2.5461 | 1.8627 | 0.6680 | -4.9550 | very | A |
| **46** | -0.6222 | 2.5107 | 1.2018 | -3.2516 | extra | A |
| **47** | 3.1675 | 3.0159 | 2.1837 | -2.4452 | very | A |
| **48** | -1.0245 | -0.8966 | -1.9352 | 1.2296 | ultra | B |
| **49** | -1.6912 | -0.8078 | -1.4317 | 1.2908 | extra | B |
| **50** | 1.0237 | -0.8630 | -3.2387 | 0.9451 | ultra | B |
| **51** | -2.6642 | -1.1768 | -3.4506 | 1.9750 | extra | B |
| **52** | 0.1355 | -1.5854 | 1.9755 | -2.9777 | ultra | C |
| **53** | 1.7814 | 1.2518 | 2.5528 | -5.9491 | very | A |
| **54** | 0.9903 | 2.0877 | 1.8697 | -10.5791 | super | A |
| **55** | 2.2574 | -0.1100 | 2.0822 | -1.1723 | ultra | C |
| **56** | -0.2980 | -1.6196 | 1.4225 | -0.7970 | extra | C |
| **57** | -0.9201 | -1.2339 | -1.1513 | 1.9105 | extra | B |
| **58** | -0.9338 | -1.0209 | -1.6187 | 0.4067 | super | B |
| **59** | 0.1921 | 0.8586 | -2.6972 | 1.3277 | extra | B |
| **60** | -1.6516 | -2.8054 | -3.5508 | 0.8020 | ultra | B |
| **61** | 1.2238 | -1.6198 | -0.0014 | -1.8271 | extra | C |
| **62** | 2.0280 | 1.8277 | -0.0026 | -1.4619 | very | A |
| **63** | 1.6378 | -1.3819 | -1.7992 | -2.0338 | ultra | C |
| **64** | -0.8494 | 0.9453 | 1.7221 | 1.0876 | extra | A |
| **65** | 1.2753 | -1.4880 | 3.7820 | -1.6105 | very | C |
| **66** | 2.5748 | -1.4210 | 2.7164 | -1.3496 | very | C |
| **67** | 1.4599 | 0.6331 | 1.7752 | -1.7875 | super | A |
| **68** | 0.6585 | 2.0708 | 4.5260 | -5.7629 | extra | A |
| **69** | -0.0676 | 1.6896 | 1.4110 | -4.3653 | super | A |
| **70** | 0.6934 | -1.1984 | 1.3763 | -1.9198 | very | C |
| **71** | 0.7668 | 2.5228 | 1.8235 | 0.4022 | very | A |
| **72** | 0.0001 | -1.9300 | -2.8061 | 0.0070 | super | B |
| **73** | -0.5506 | -1.8223 | 0.0792 | 1.1129 | extra | B |
| **74** | -1.1319 | -0.6499 | -1.4379 | -0.8543 | extra | B |
| **75** | -0.4638 | -1.9047 | -1.2285 | 3.7873 | ultra | B |
| **76** | 0.6810 | 1.1459 | 0.6383 | 1.9527 | super | A |
| **77** | 1.0251 | 0.3909 | 2.5498 | -2.0020 | super | C |
| **78** | -1.9444 | -1.2551 | -2.4101 | 1.1202 | ultra | B |
| **79** | 1.1667 | 0.3754 | -0.8383 | -1.2688 | super | C |
| **80** | 1.1738 | -1.7386 | 0.6835 | -1.7938 | super | C |
| **81** | -0.5773 | -1.6291 | -4.4026 | -0.0799 | extra | B |
| **82** | -0.9017 | -3.1237 | 1.4287 | 0.9674 | extra | B |
| **83** | -1.0308 | -1.3826 | -1.3938 | 1.5894 | extra | B |
| **84** | -2.6702 | -2.2038 | -1.1407 | 1.1992 | ultra | B |
| **85** | 1.0472 | -1.5780 | 3.2359 | -2.3604 | extra | C |
| **86** | -0.2214 | 1.4063 | 0.5509 | -3.3436 | extra | A |
| **87** | 5.2920 | 1.8639 | 2.5080 | -0.8378 | extra | A |
| **88** | 1.2294 | -0.6982 | -2.7895 | 0.3736 | ultra | B |
| **89** | 1.8648 | -1.3012 | 5.0246 | -1.6635 | ultra | C |
| **90** | -0.0885 | 0.6134 | 1.5238 | 2.9290 | super | A |
| **91** | -1.0000 | -1.2730 | -0.3395 | 0.0130 | super | B |
| **92** | -0.9090 | 2.5290 | 0.4229 | -0.9271 | super | A |
| **93** | 1.8877 | 3.2105 | 2.7914 | 0.2393 | very | A |
| **94** | -0.1383 | -1.9047 | -1.8454 | 1.2748 | super | B |
| **95** | -3.3193 | -0.6914 | -2.3681 | -0.7499 | extra | B |
| **96** | 0.9511 | 0.6174 | 1.0005 | -1.8941 | very | C |
| **97** | -0.1134 | 1.1883 | 1.9800 | -3.7693 | extra | A |
| **98** | -0.6468 | -2.4752 | -0.1428 | -1.9840 | very | C |
| **99** | -1.4204 | 2.5942 | 1.1521 | 2.9044 | very | A |
| **100** | 2.5565 | 1.6686 | 3.0391 | -1.2673 | extra | A |
| **⋮** |  |  |  |  |  |  |

**Task 1**

dvGrit = dummyvar(data.grit)

dvGrit = 154×4

0 0 0 1  
 0 0 1 0  
 0 1 0 0  
 0 1 0 0  
 0 0 0 1  
 0 0 1 0  
 1 0 0 0  
 0 0 1 0  
 0 1 0 0  
 0 1 0 0

**Task 2**

predictors = [data{:,1:4} dvGrit];

ferror = @(XTrain,yTrain,XTest,yTest) nnz(yTest ~= predict(fitcknn(XTrain,yTrain),XTest));

toKeep = sequentialfs(ferror,predictors,data.bin)

toKeep = 1×8 logical array

0 1 0 1 1 0 0 1

**Task 3**

mdlPart = fitcknn(predictors(:,toKeep),data.bin,"KFold",7);

partLoss = kfoldLoss(mdlPart)

partLoss = 0.0584

Ensemble Learning - Random Forest

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the data.

rng(1234)

load data

data;

This code fits a classification tree and calculates the loss.

cvpt = cvpartition(data.R,"KFold",3);

mdlTree = fitctree(data,"R","CVPartition",cvpt);

lossTree = kfoldLoss(mdlTree)

lossTree = 0.0714

**Task 1**

mdlEns = fitcensemble(data,"R","Method","Bag");

lossEns = resubLoss(mdlEns)

lossEns = 0

**Task 2**

mdlEns2 = fitcensemble(data,"R","Method","Bag",...

"NumLearningCycles",30,"CVPartition",cvpt);

lossEns2 = kfoldLoss(mdlEns2)

lossEns2 = 0.0455

Heart Disease Analysis

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the data.

heartData = readtable("heartDataAll.txt");

heartData = convertvars(heartData,12:22,"categorical");

heartDataNum = heartData(:,[1:11 end]);

This code partitions the data and fits a tree model.

rng(0)

cvpt = cvpartition(heartDataNum.HeartDisease,"KFold",7);

mdlTree = fitctree(heartDataNum,"HeartDisease","CVPartition",cvpt);

lossTree = kfoldLoss(mdlTree)

lossTree = 0.3396

**Tasks 1 & 2**

Fit tree ensemble and calculate loss.

tmodel = templateTree("Prune","on");

mdlEns = fitcensemble(heartDataNum,"HeartDisease","CVPartition",cvpt,...

"Learners",tmodel,"NumLearningCycles",50);

lossEns = kfoldLoss(mdlEns)

lossEns = 0.2693

# Wine Quality - Reducing Predictors

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the wine data and a trained 7-fold cross-validated quadratic DA model of redData.

rng(0)

load wineDataRed

redData;

mdlFull

mdlFull =

ClassificationPartitionedModel  
 CrossValidatedModel: 'Discriminant'  
 PredictorNames: {'FixedAcidity' 'VolatileAcidity' 'CitricAcid' 'ResidualSugar' 'Chlorides' 'FreeSulfurDioxide' 'TotalSulfurDioxide' 'Density' 'pH' 'Sulphates' 'Alcohol'}  
 ResponseName: 'QCLabel'  
 NumObservations: 1593  
 KFold: 7  
 Partition: [1×1 cvpartition]  
 ClassNames: [A B C D E]  
 ScoreTransform: 'none'  
  
  
 Properties, Methods

fullLoss = kfoldLoss(mdlFull)

fullLoss = 0.4488

## **Reduce Predictors**

Fit model with fewer predictors. Target loss value less than 0.45.

#### Method: Feature Selection (Tree)

tModel = fitctree(redData,"QCLabel");

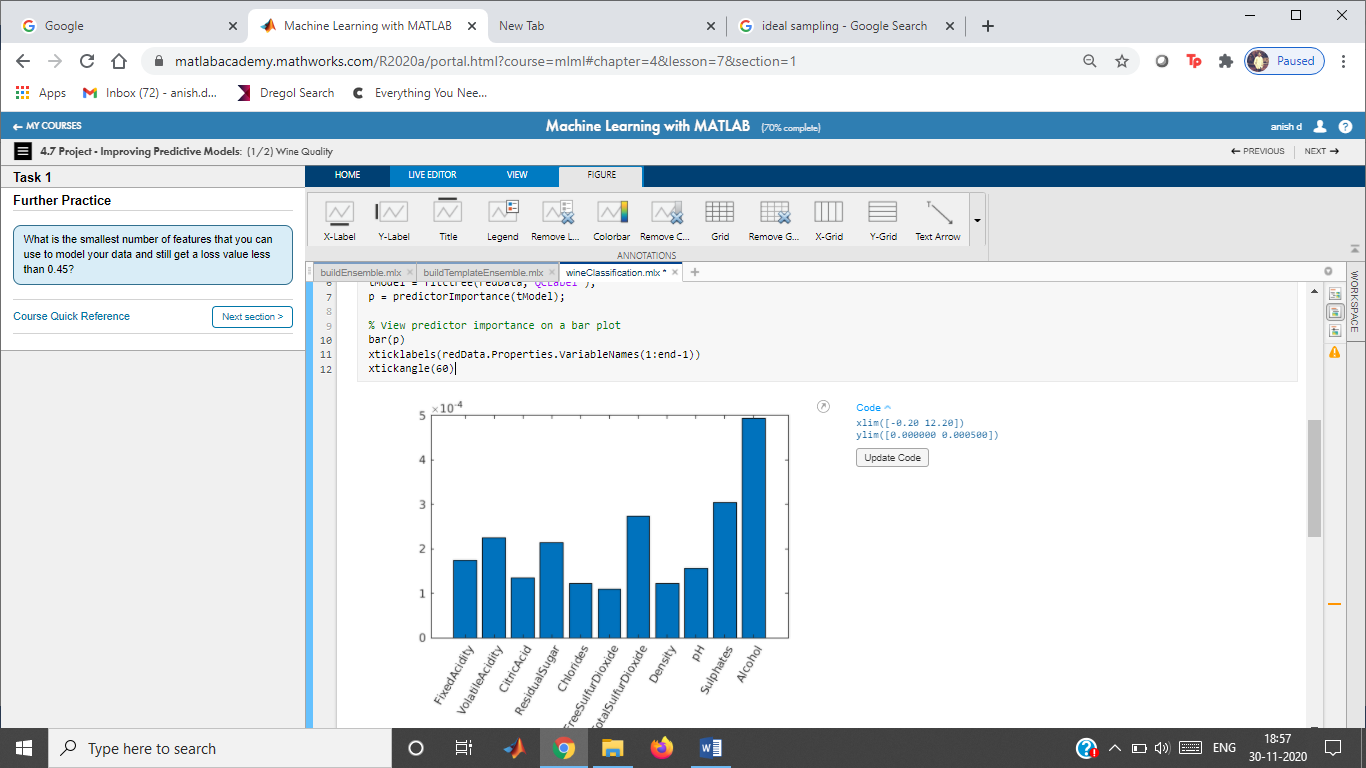
p = predictorImportance(tModel);

% View predictor importance on a bar plot

bar(p)

xticklabels(redData.Properties.VariableNames(1:end-1))

xtickangle(60)



% Sort out the top predictors

[~,iSorted] = sort(p);

selected = [iSorted(1:6) width(redData)];

% Create tree model to reduced data

mdl = fitctree(redData(:,selected),"QCLabel","KFold",7);

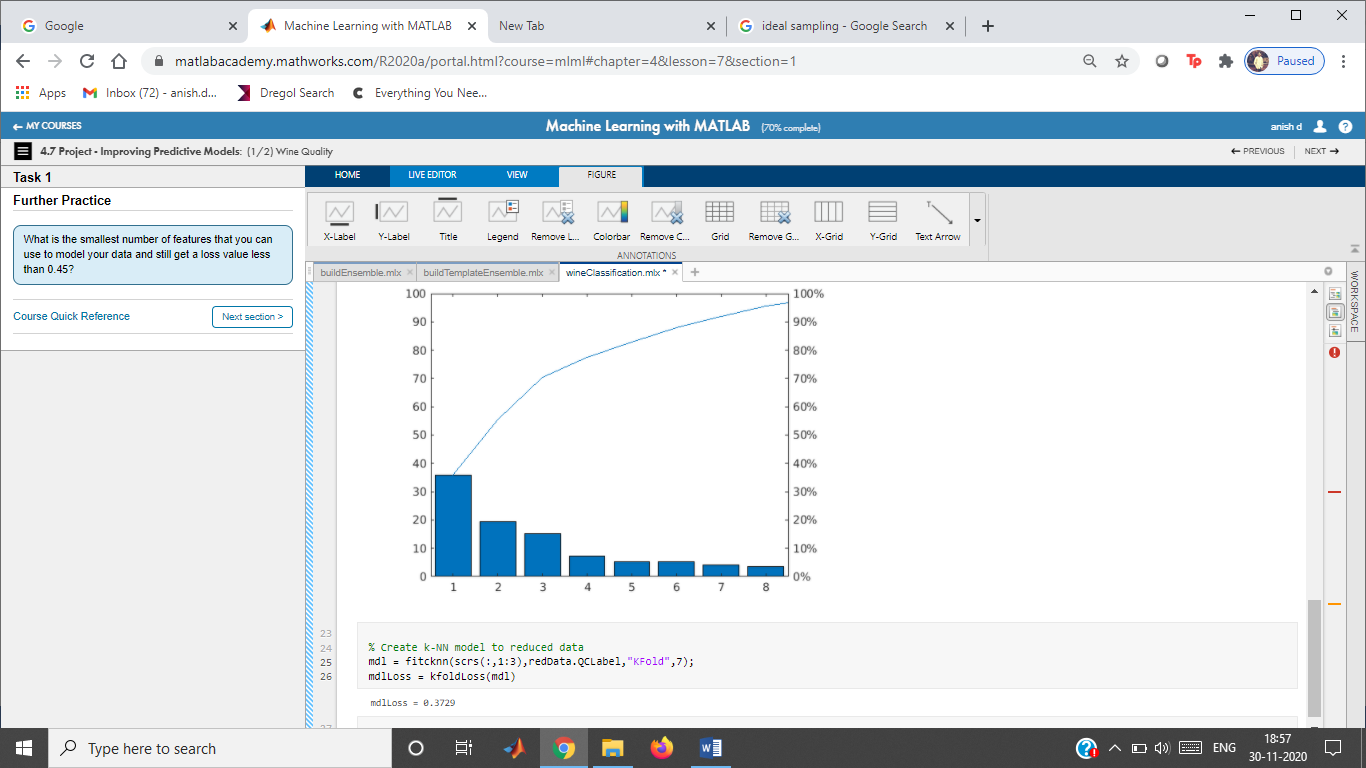
mdlLoss = kfoldLoss(mdl)

mdlLoss = 0.4476

#### Method: Feature Transformation (PCA)

[pcs,scrs,~,~,pexp] = pca(redData{:,1:end-1});

pareto(pexp)



% Create k-NN model to reduced data

mdl = fitcknn(scrs(:,1:3),redData.QCLabel,"KFold",7);

mdlLoss = kfoldLoss(mdl)

mdlLoss = 0.3729

# Classify Credit Ratings

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the credit ratings data and a 7-fold cross-validated tree model of creditRatings.

rng(0)

load creditData

creditRatings;

mdlFull

mdlFull =

ClassificationPartitionedModel  
 CrossValidatedModel: 'Tree'  
 PredictorNames: {'WC\_TA' 'RE\_TA' 'EBIT\_TA' 'MVE\_BVTD' 'S\_TA'}  
 ResponseName: 'Rating'  
 NumObservations: 250  
 KFold: 7  
 Partition: [1×1 cvpartition]  
 ClassNames: [A AA AAA B BB BBB CCC]  
 ScoreTransform: 'none'  
  
  
 Properties, Methods

fullLoss = kfoldLoss(mdlFull)

fullLoss = 0.3440

## **Task 1**

Fit model with 3 or fewer predictors.

#### Method: Feature Selection (Tree)

tModel = fitctree(creditRatings,"Rating");

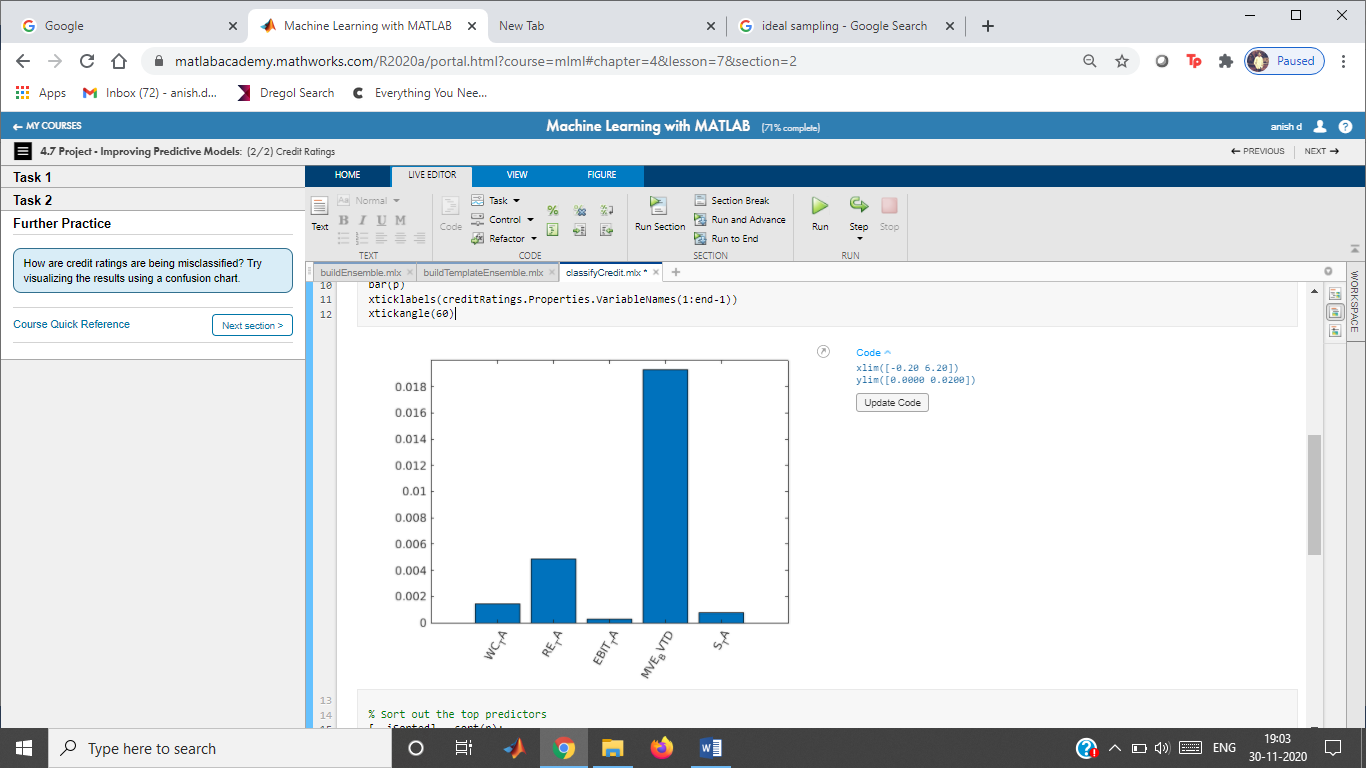
p = predictorImportance(tModel);

% View predictor importance on a bar plot

bar(p)

xticklabels(creditRatings.Properties.VariableNames(1:end-1))

xtickangle(60)



% Sort out the top predictors

[~,iSorted] = sort(p);

selected = [iSorted(1:2) width(creditRatings)];

% Create tree model to reduced data

mdl = fitctree(creditRatings(:,selected),"Rating","KFold",7);

mdlLoss = kfoldLoss(mdl)

mdlLoss = 0.4760

#### Method: Feature Transformation (PCA)

[pcs,scrs,~,~,pexp] = pca(creditRatings{:,1:end-1});

% Create tree model to reduced data

mdl = fitctree(scrs(:,1:3),creditRatings.Rating,"KFold",7);

mdlLoss = kfoldLoss(mdl)

## **Task 2**

Fit ensemble with 3 or fewer predictors.

#### Method: Feature Selection (Tree)

mdlEns = fitcensemble(creditRatings(:,selected),"Rating","Method","Bag",...

"NumLearningCycles",50,"Learners","tree","KFold",7);

lossEns = kfoldLoss(mdlEns)

lossEns = 0.5400

#### Method: Feature Transformation (PCA)

mdlEns = fitcensemble(scrs(:,1),creditRatings.Rating,"Method","Bag",...

"NumLearningCycles",50,"Learners","tree","KFold",7);

lossEns = kfoldLoss(mdlEns)

Sequential Feature Selection

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the data.

rng(0)

load data.mat

data

data = 154×6 table

Full = fitcknn(data,"R","KFold",7);

fullLoss = kfoldLoss(mdlFull)

fullLoss = 0.0519

**Task 1**

ferror = @(XTrain,yTrain,XTest,yTest) nnz(yTest ~= predict(fitcknn(XTrain,yTrain),XTest))

ferror = *function\_handle with value:*

@(XTrain,yTrain,XTest,yTest)nnz(yTest~=predict(fitcknn(XTrain,yTrain),XTest))

**Task 2**

toKeep = sequentialfs(ferror,data{:,1:end-1},data.R)

toKeep = 1×5 logical array

0 1 1 1 1

dataPart = data(:,[toKeep true])

dataPart = 154×5 table

**Task 3**

mdlPart = fitcknn(dataPart,"R","KFold",7);

partLoss = kfoldLoss(mdlPart)

partLoss = 0.0649

# Heart Disease Analysis

Load the data.

heartData = readtable("heartDataAll.txt");

heartData = convertvars(heartData,12:22,"categorical");

Extract the response variable and make a partition for evaluation.

HD = heartData.HeartDisease;

rng(1234)

cvpt = cvpartition(HD,"KFold",10);

### Convert categorical predictors to numeric dummy variables

[X,XNames] = cattable2mat(heartData(:,1:end-1))

### Fit a Naive Bayes model to the full data

dists = [repmat("kernel",1,11),repmat("mvmn",1,10)];

mFull = fitcnb(heartData,"HeartDisease","DistributionNames",dists,"CVPartition",cvpt);

### Perform sequential feature selection

rng(1234)

fmodel = @(X,y) fitcnb(X,y,"DistributionNames","kernel");

ferror = @(Xtrain,ytrain,Xtest,ytest) nnz(predict(fmodel(Xtrain,ytrain),Xtest) ~= ytest);

toKeep = sequentialfs(ferror,X,HD,"cv",cvpt,"options",statset("Display","iter"));

% Which variables are in the final model?

XNames(toKeep)

% Fit a model with just the given variables

mPart = fitcnb(X(:,toKeep),HD,'Distribution','kernel','CVPartition',cvpt);

% Display loss values

lossFull = kfoldLoss(mFull)

lossPart = kfoldLoss(mPart)

Local function: cattable2mat

function [dummymat,dummyvarnames] = cattable2mat(data)

% Makes a matrix from a table, with categorical variables

% replaced by (numeric) dummy variables

vars = string(data.Properties.VariableNames);

idxCat = varfun(@iscategorical,data,"OutputFormat","uniform");

for k = find(idxCat)

% get list of categories

c = categories(data.(vars(k)));

% replace variable with matrix of dummy variables

data = convertvars(data,vars(k),@dummyvar);

% split dummy variable and make new variable names (by appending

% the category value to the categorical variable name)

varnames = vars(k) + "\_" + replace(c," ","\_");

data = splitvars(data,vars(k),"NewVariableNames",varnames);

end

% return the numeric values

dummymat = data.Variables;

dummyvarnames = string(data.Properties.VariableNames);

end

Decision Trees: Predictor Importance

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the data.

rng(0)

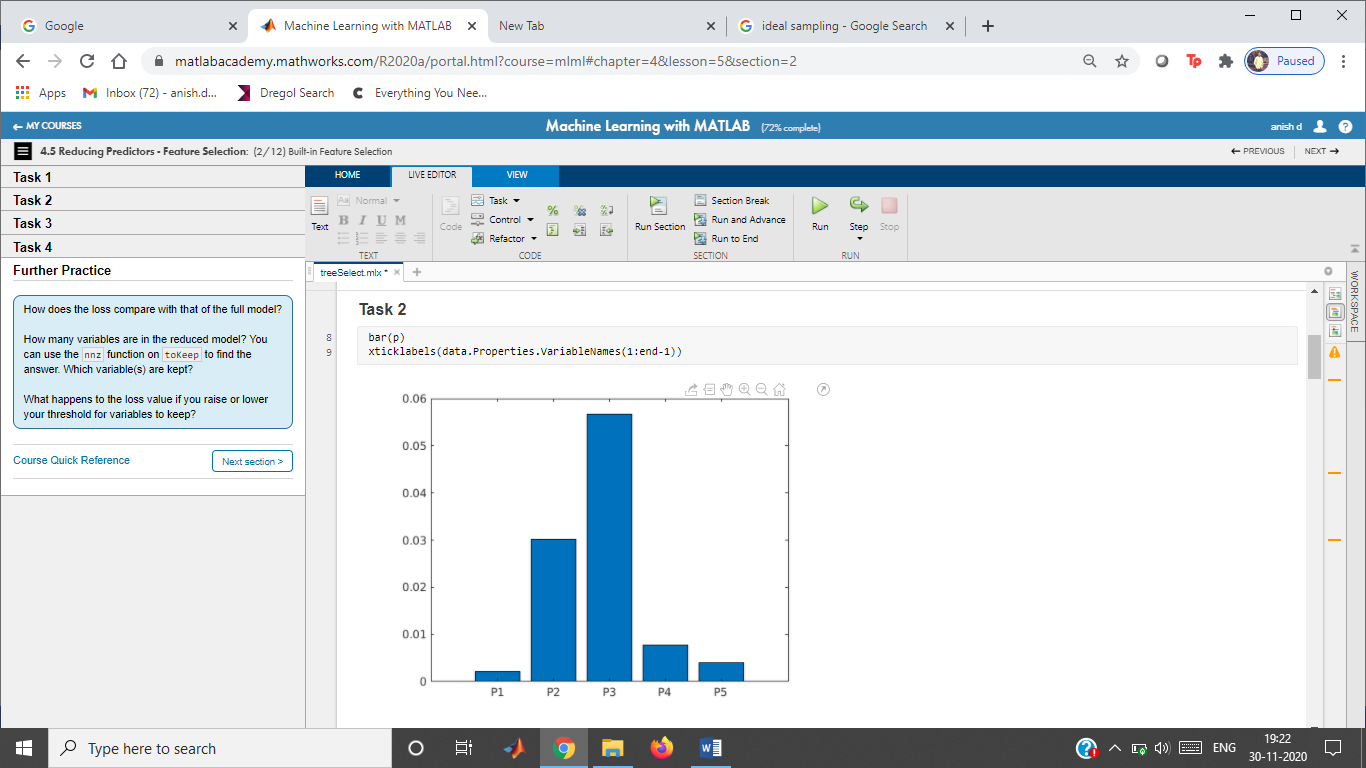
load data.mat

data

This code fits a 7-fold cross-validated classification tree model to the original data and calculates the loss.

mdlFull = fitctree(data,"R","KFold",7);

fullLoss = kfoldLoss(mdlFull)



**Task 1**

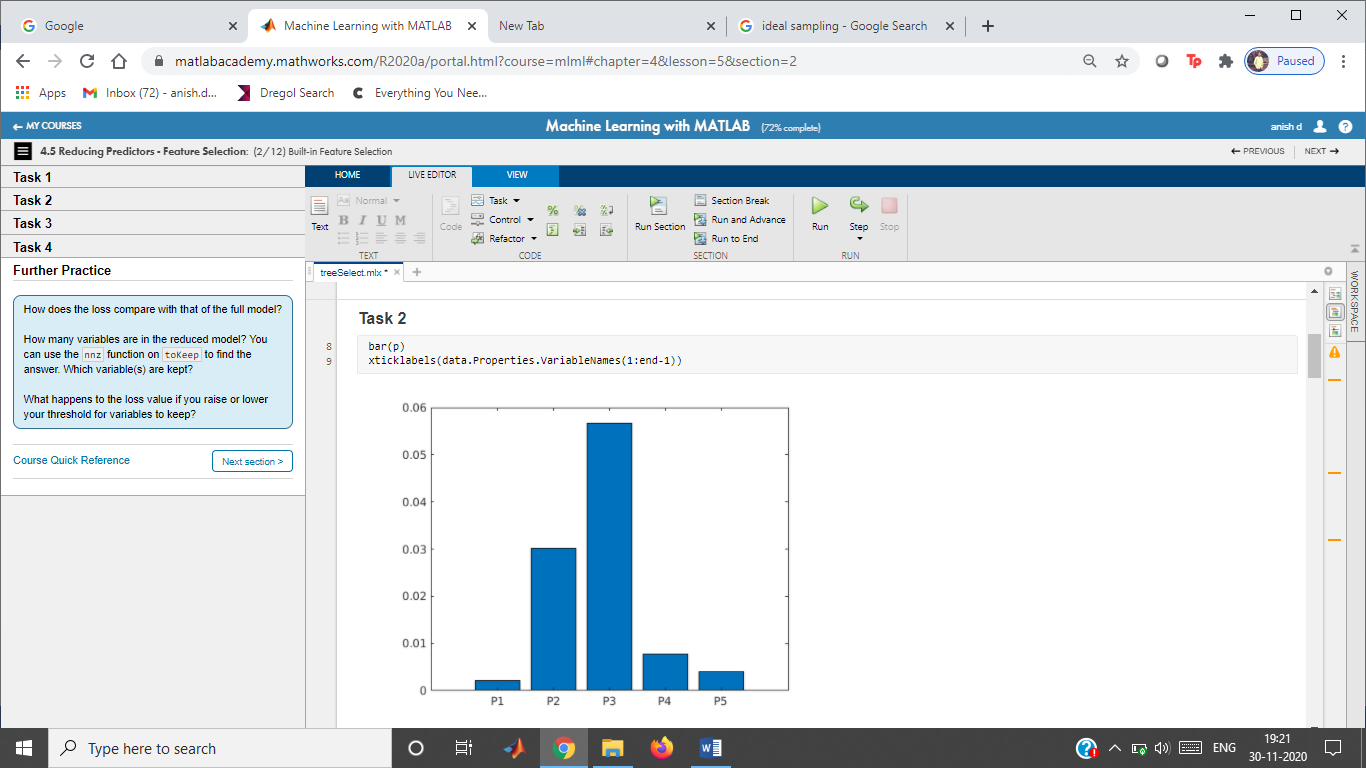
mdl = fitctree(data,"R");

p = predictorImportance(mdl)

**Task 2**

bar(p)

xticklabels(data.Properties.VariableNames(1:end-1))

**Task 3**

toKeep = p > 0.005

dataPart = data(:,[toKeep true])

**Task 4**

mdlPart = fitctree(dataPart,"R","KFold",7);

partLoss = kfoldLoss(mdlPart)

**Further Practice**

## **Unit 5**

Fitting a Line

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and plots the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

plot(data.x,data.y,".")

**Task 1**

Fitting a Line

Instructions are in the task pane to the left. Complete and submit each task one at a time.

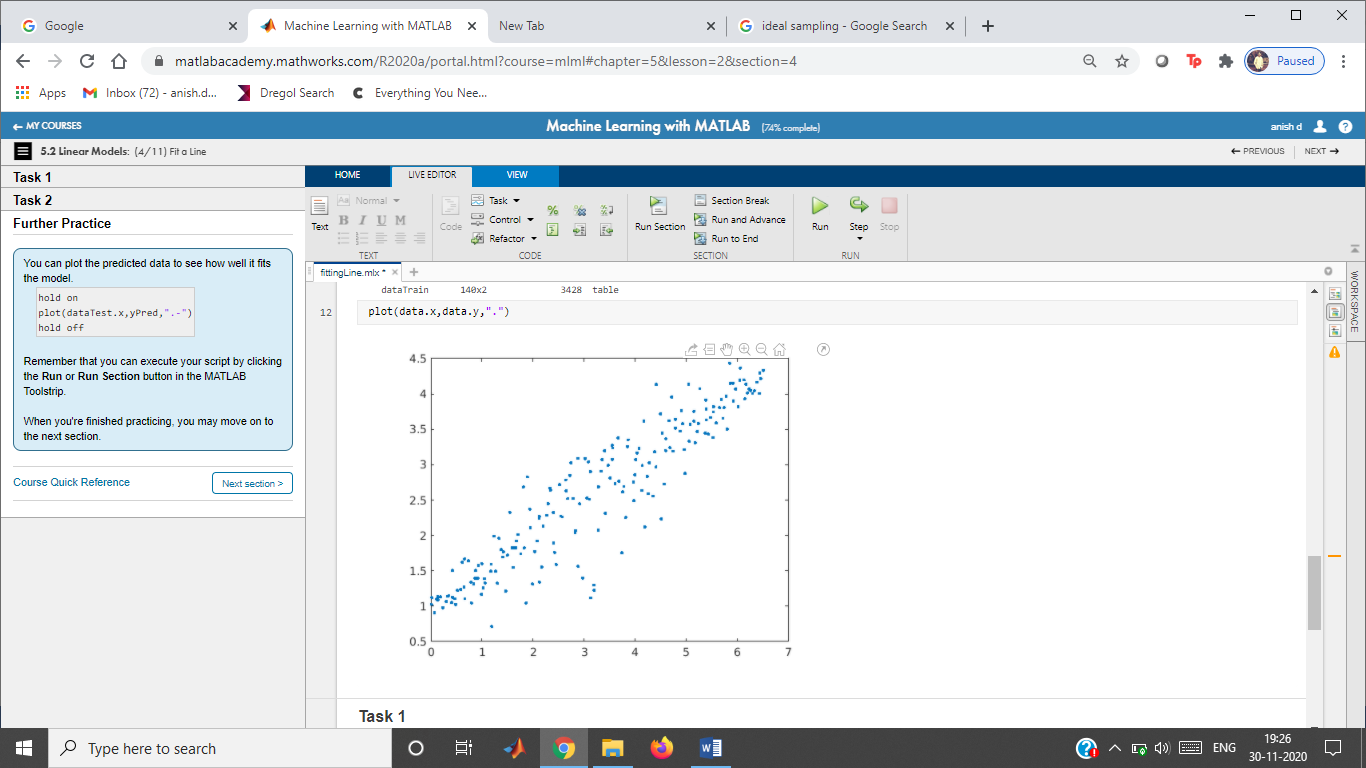
This code loads and plots the data.

load data

whos data dataTrain dataTest

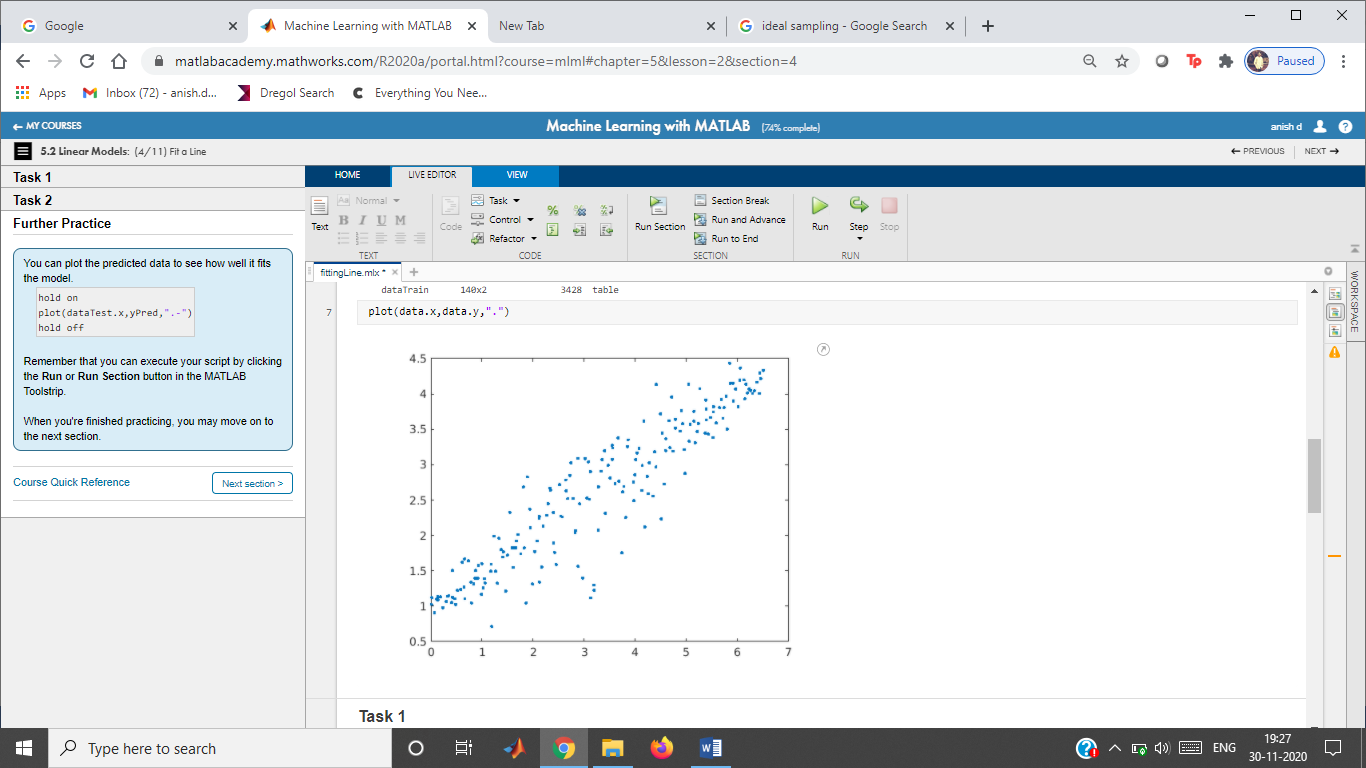
Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

plot(data.x,data.y,".")



**Task 1**

mdl = fitlm(dataTrain);



**Task 2**

Fitting a Line

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and plots the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

plot(data.x,data.y,".")

**Task 1**

mdl = fitlm(dataTrain);

**Task 2**

yPred = predict(mdl,dataTest);

Fitting a Polynomial

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

**Tasks 1, 2, 3**

mdl = fitlm(dataTrain,"quadratic","RobustOpts","on")

mdl =

Linear regression model (robust fit):  
 y ~ 1 + x + x^2  
  
Estimated Coefficients:  
 **Estimate** **SE** **tStat** **pValue**   
 **\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_**  
  
 **(Intercept)** 5.0238 0.30936 16.239 9.8548e-34  
 **x**  -3.8354 0.23458 -16.35 5.3122e-34  
 **x^2**  1.0068 0.037078 27.153 5.3905e-57  
  
  
Number of observations: 140, Error degrees of freedom: 137  
Root Mean Squared Error: 1.35  
R-squared: 0.946, Adjusted R-Squared: 0.945  
F-statistic vs. constant model: 1.2e+03, p-value = 1.37e-87

yPred = predict(mdl,dataTest);

Plot the results.

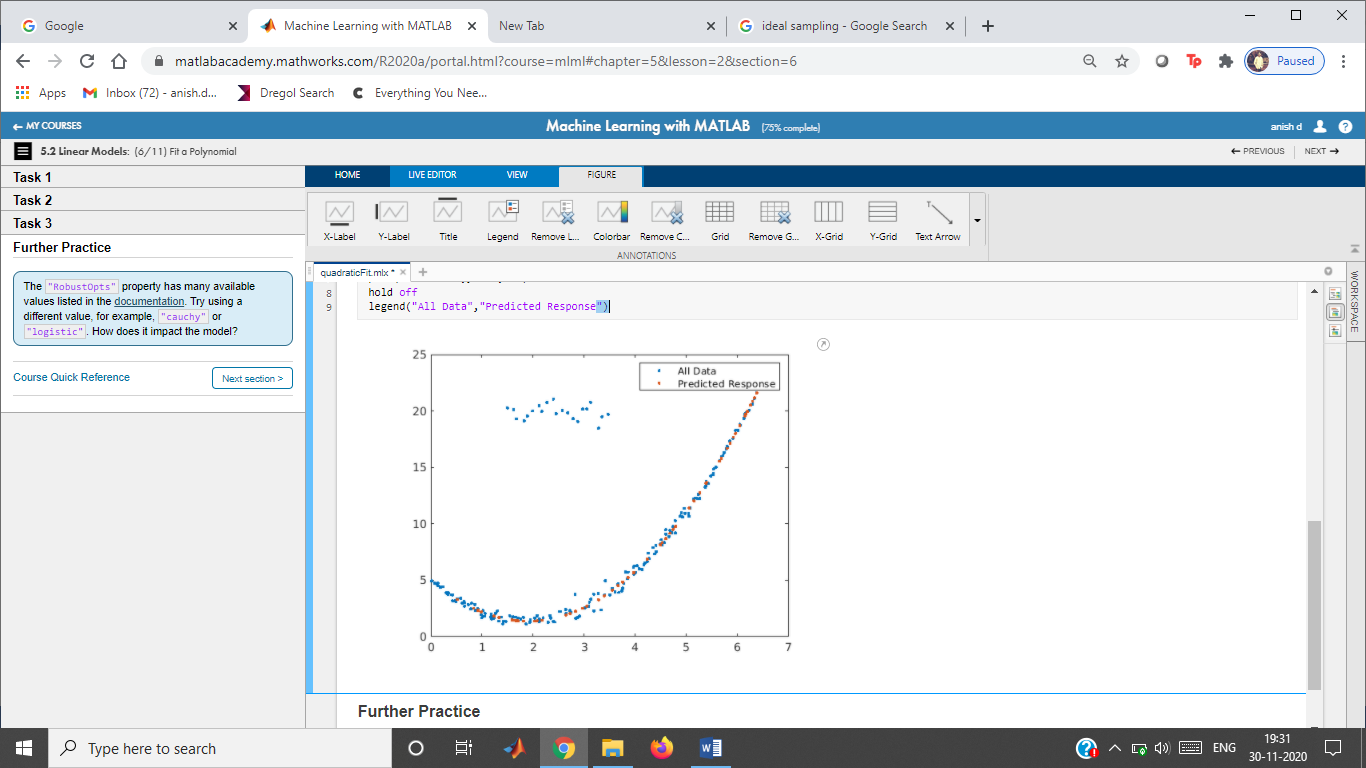
plot(data.x,data.y,'.')

hold on

plot(dataTest.x,yPred,'.')

hold off

legend("All Data","Predicted Response")



Multivariable Linear Regression

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data and displays the table variable names.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 100x4 4814 table   
 dataTest 30x4 2574 table   
 dataTrain 70x4 3854 table

data.Properties.VariableNames

ans = 1×4 cell

'X1' 'X2' 'X3' 'Y'

**Tasks 1 & 2**

mdl = fitlm(dataTrain)

mdl =

Linear regression model:  
 Y ~ 1 + X1 + X2 + X3  
  
Estimated Coefficients:  
 **Estimate** **SE** **tStat** **pValue**   
 **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_**  
  
 **(Intercept)** -8.7253 7.1477 -1.2207 0.22682  
 **X1**  -0.0067133 0.00052754 -12.726 6.0508e-19  
 **X2**  -0.14815 0.13473 -1.0996 0.27575  
 **X3**  0.71534 0.085609 8.3558 9.5691e-12  
  
  
Number of observations: 66, Error degrees of freedom: 62  
Root Mean Squared Error: 2.91  
R-squared: 0.867, Adjusted R-Squared: 0.86  
F-statistic vs. constant model: 134, p-value = 4.47e-27

yPred = predict(mdl,dataTest);

Compare actual and predicted response.

plot(dataTest.Y,yPred,".")

hold on

Multivariable Linear Regression

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data and displays the table variable names.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 100x4 4814 table   
 dataTest 30x4 2574 table   
 dataTrain 70x4 3854 table

data.Properties.VariableNames

ans = 1×4 cell

'X1' 'X2' 'X3' 'Y'

**Tasks 1 & 2**

mdl = fitlm(dataTrain,"Y ~ X3 + X1^2")

mdl =

Linear regression model:  
 Y ~ 1 + X1 + X3 + X1^2  
  
Estimated Coefficients:  
 **Estimate** **SE** **tStat** **pValue**   
 **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_**  
  
 **(Intercept)** -0.38043 8.3718 -0.045442 0.9639  
 **X1**  -0.014043 0.0037879 -3.7073 0.00044911  
 **X3**  0.71314 0.081914 8.7059 2.3795e-12  
 **X1^2**  1.1953e-06 5.9868e-07 1.9966 0.050265  
  
  
Number of observations: 66, Error degrees of freedom: 62  
Root Mean Squared Error: 2.85  
R-squared: 0.872, Adjusted R-Squared: 0.866  
F-statistic vs. constant model: 141, p-value = 1.18e-27

yPred = predict(mdl,dataTest);

Compare actual and predicted response.

plot(dataTest.Y,yPred,".")

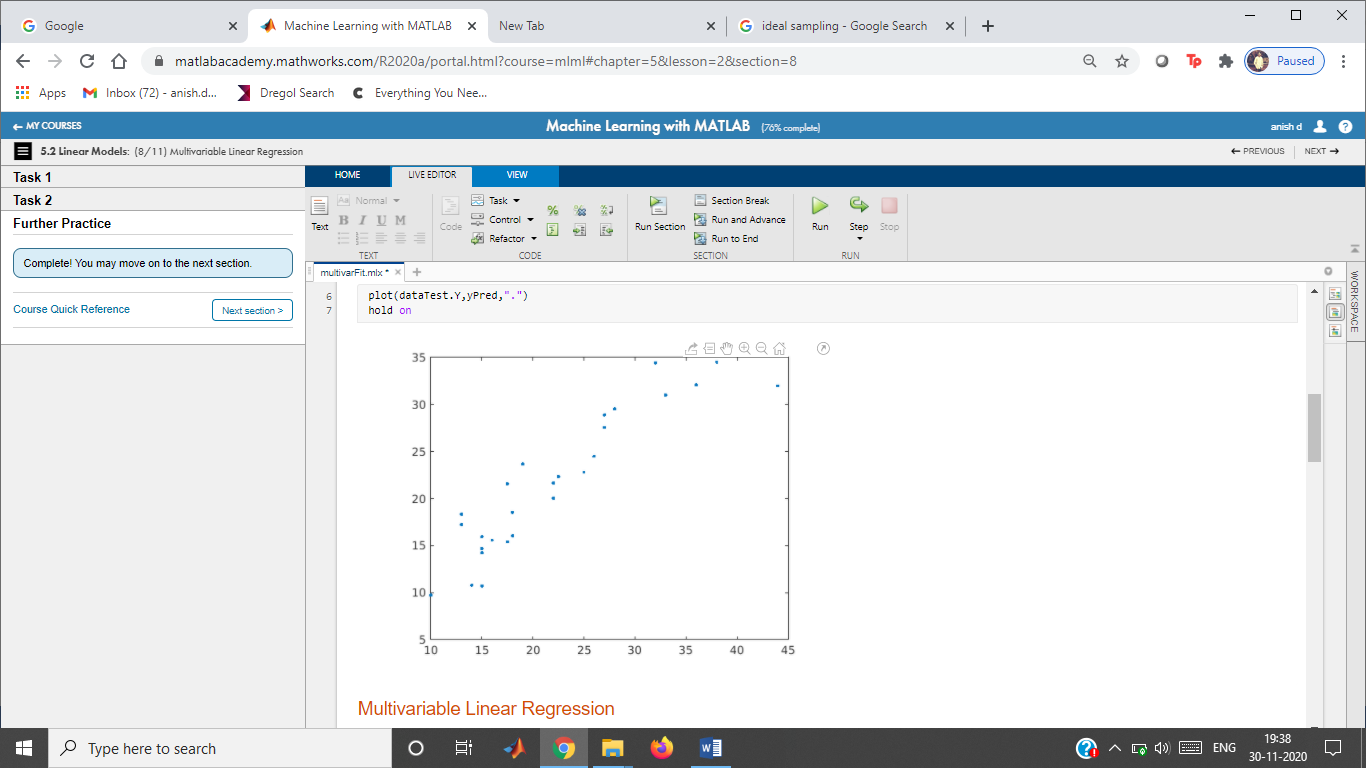
hold on

plot(dataTest.Y,dataTest.Y)

hold off

xlabel("Actual Response")

ylabel("Predicted Response")

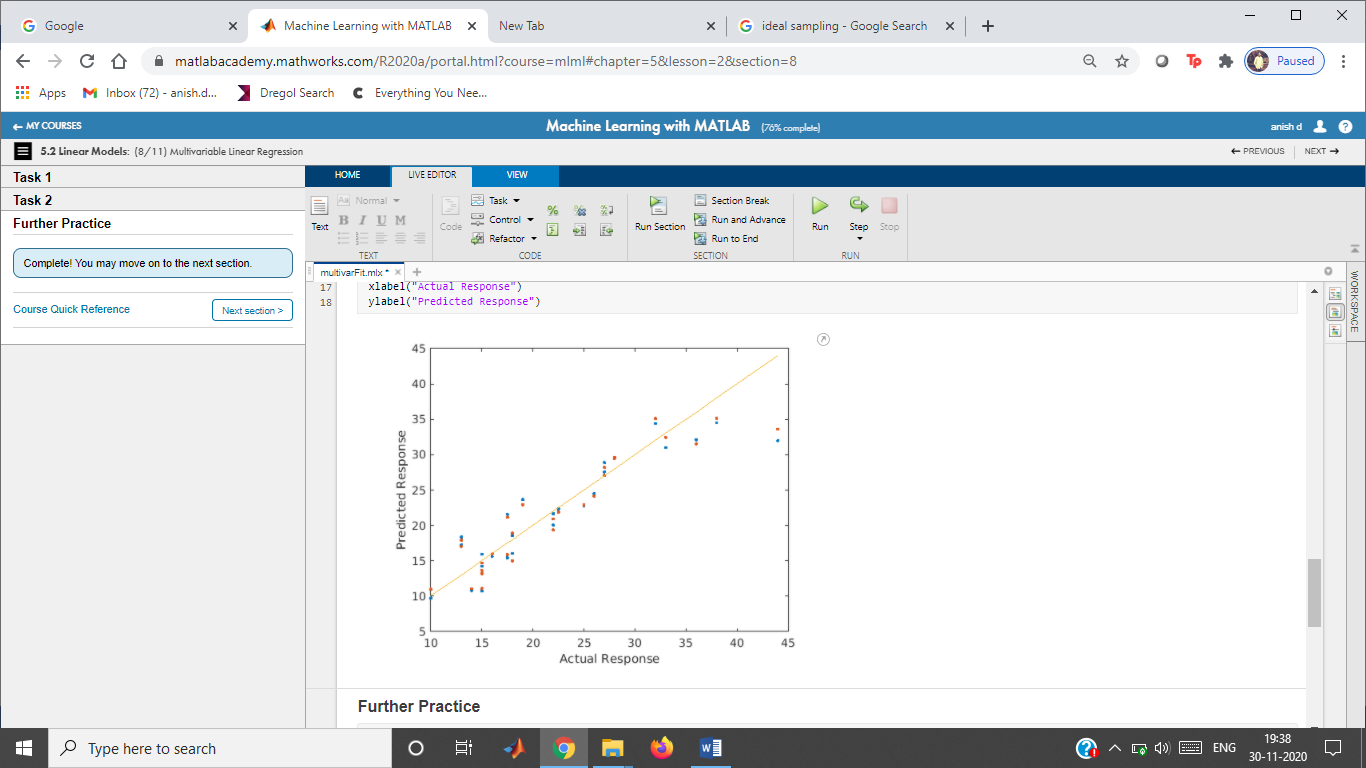
**Further Practice**

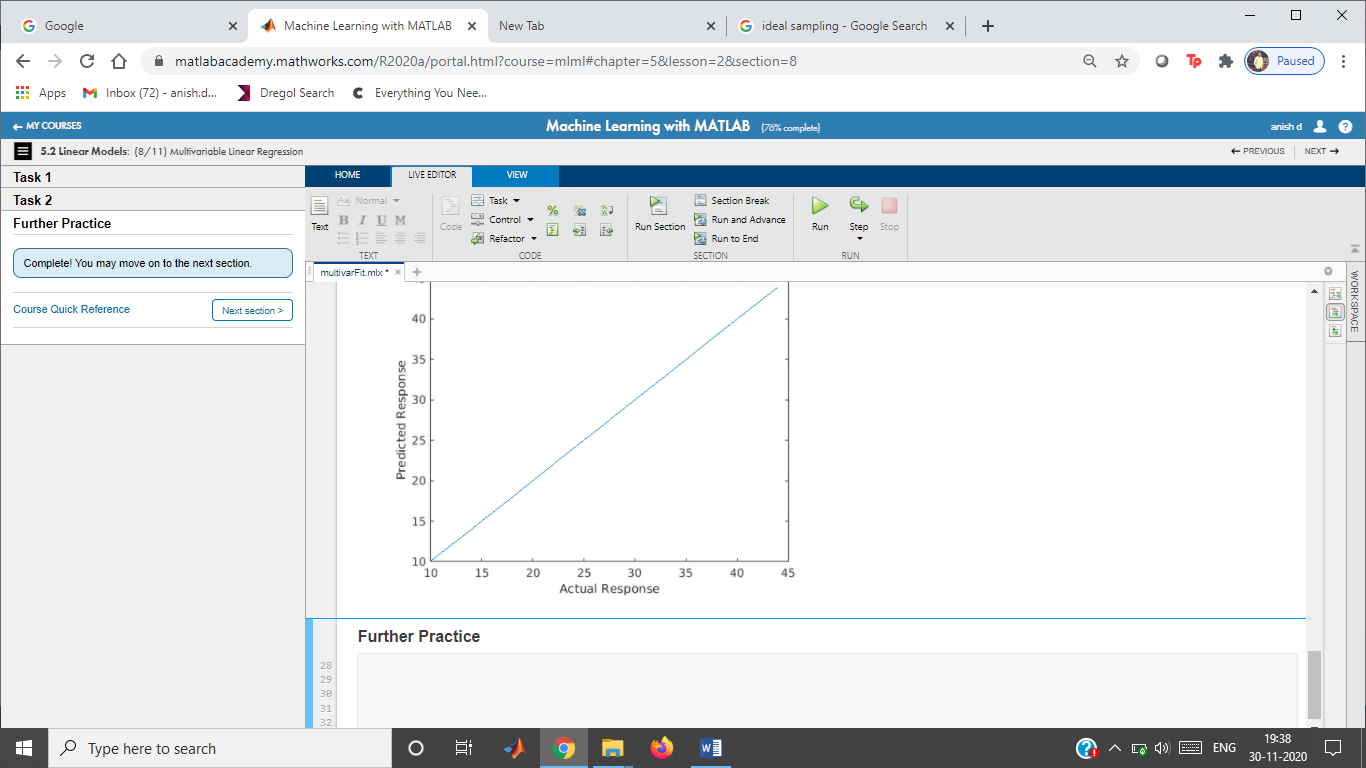
plot(dataTest.Y,dataTest.Y)

hold off

xlabel("Actual Response")

ylabel("Predicted Response")





Fuel Economy Analysis

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load carEcon

whos carData carTrain carTest

Name Size Bytes Class Attributes  
  
 carData 600x15 48238 table   
 carTest 192x15 20086 table   
 carTrain 408x15 34990 table

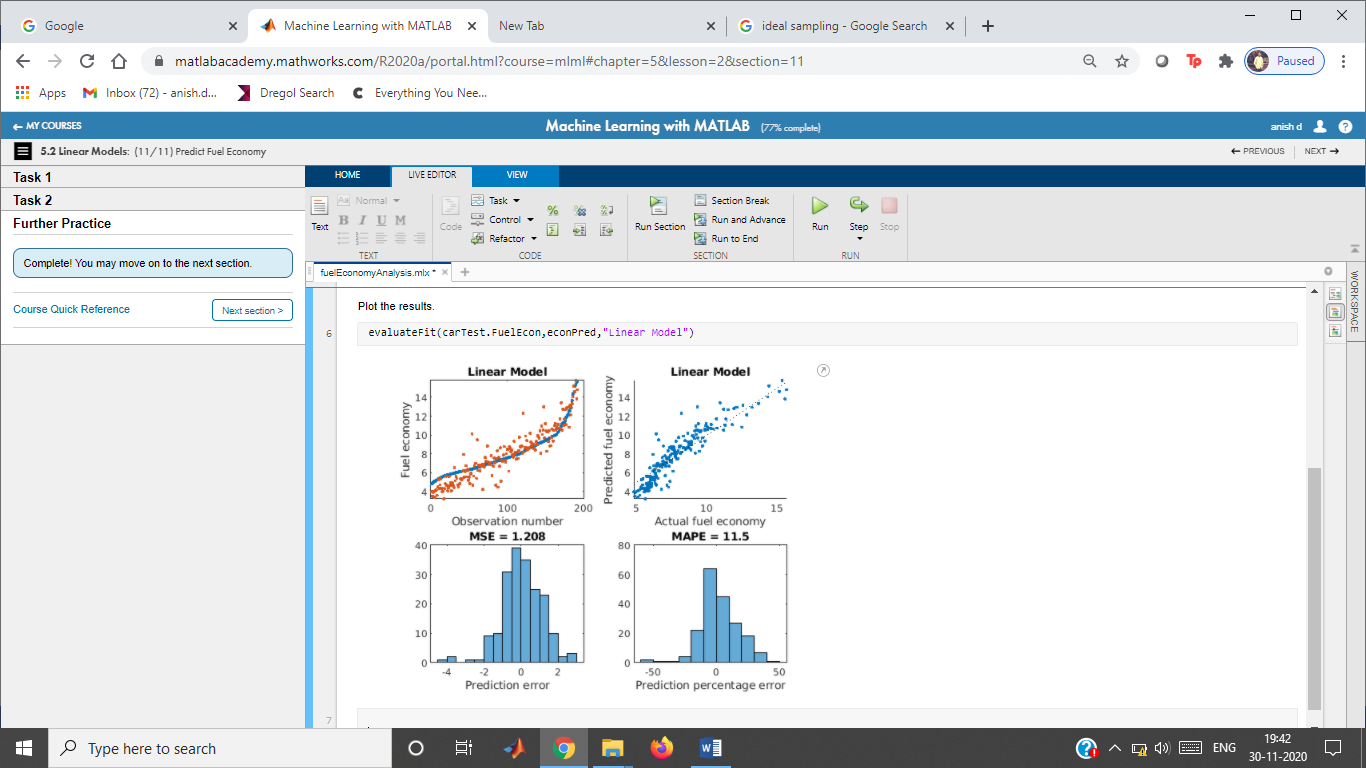
**Tasks 1 and 2**

mdl = fitlm(carTrain,"RobustOpts","cauchy");

econPred = predict(mdl,carTest);

Plot the results.

evaluateFit(carTest.FuelEcon,econPred,"Linear Model")



Fitting a Stepwise Linear Model

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 100x4 4814 table   
 dataTest 30x4 2574 table   
 dataTrain 70x4 3854 table

**Tasks 1 & 3**

mdl = stepwiselm(dataTrain,"purequadratic");

1. Removing X2^2, FStat = 0.84426, pValue = 0.36192  
2. Removing X2, FStat = 0.59788, pValue = 0.44243

yPred = predict(mdl,dataTest);

**Task 2**

formula = mdl.Formula

formula = Y ~ 1 + X1 + X3 + X1^2 + X3^2

Plot predicted vs. actual response.

plot(dataTest.Y,yPred,".")

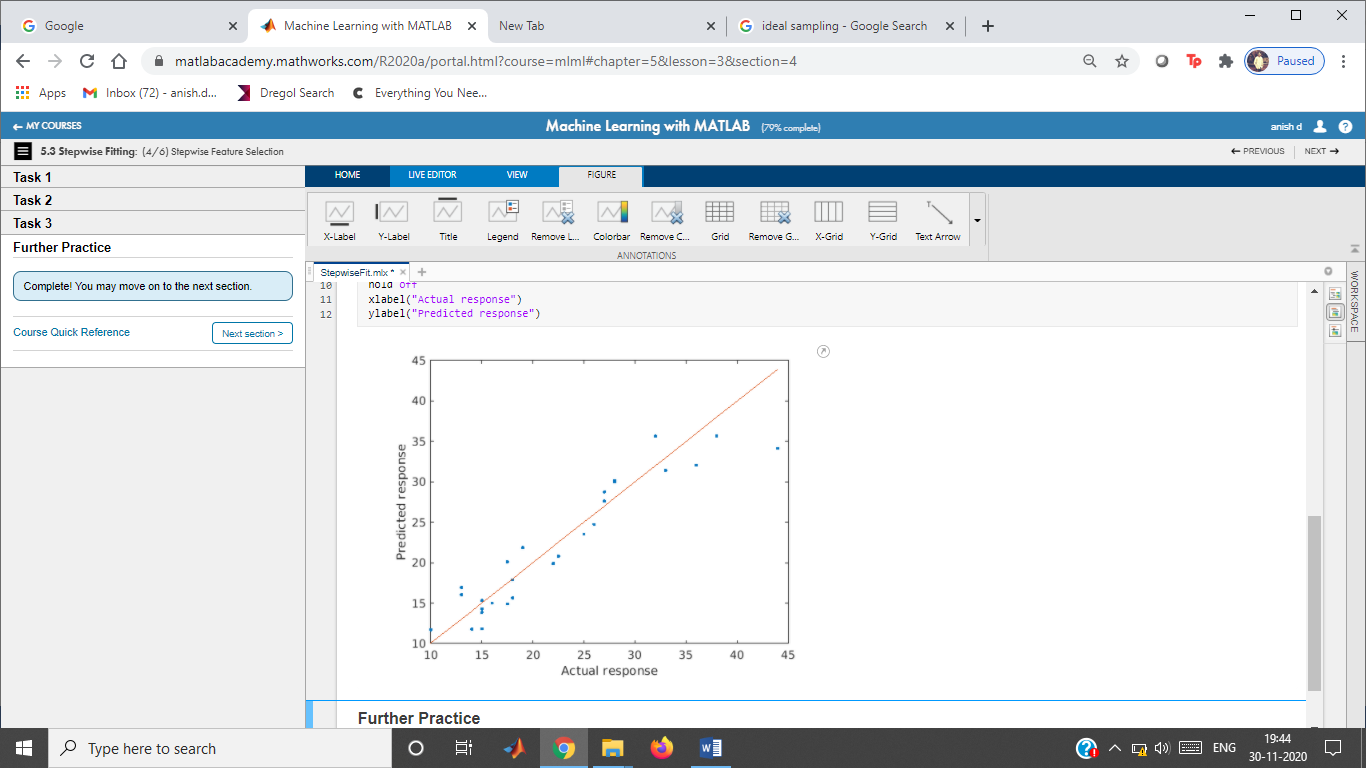
hold on

plot(dataTest.Y,dataTest.Y)

hold off

xlabel("Actual response")

ylabel("Predicted response")



Fuel Economy - Stepwise Linear Regression

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load carEcon

whos carData carTrain carTest

Name Size Bytes Class Attributes  
  
 carData 600x15 48718 table   
 carTest 192x15 20566 table   
 carTrain 408x15 35470 table

**Tasks 1 & 3**

mdl = stepwiselm(carTrain,"Upper","linear");

econPred = predict(mdl,carTest);

**Task 2**

RMSE = mdl.RMSE

Compare predicted and actual responses.

evaluateFit(carTest.FuelEcon,econPred,"Stepwise LM")

**Further Practice**

Fitting a Ridge Regression Model

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and plots the data.

load data

whos data XTrain XTest yTrain yTest

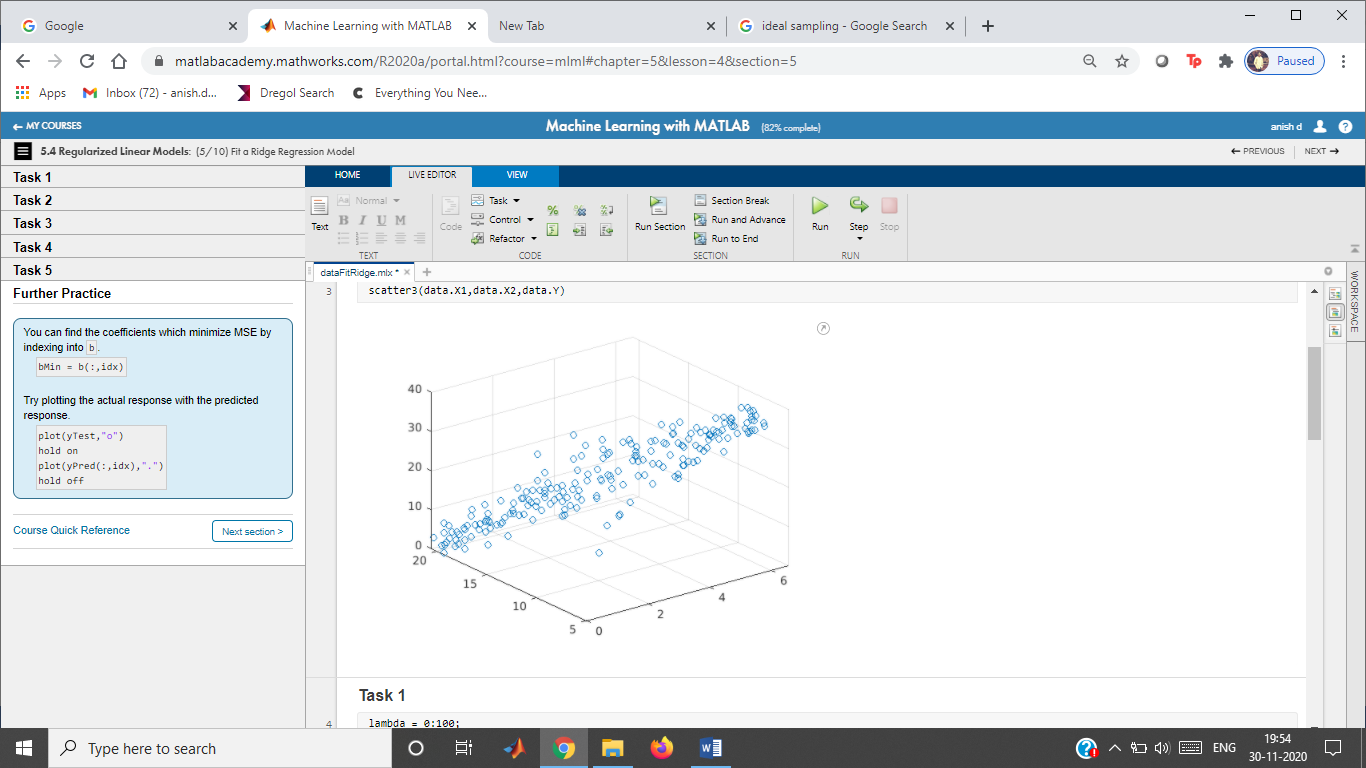
Name Size Bytes Class Attributes  
  
 XTest 60x2 960 double   
 XTrain 140x2 2240 double   
 data 200x3 6202 table   
 yTest 60x1 480 double   
 yTrain 140x1 1120 double

scatter3(data.X1,data.X2,data.Y)

**Task 1**

lambda = 0:100;

b = ridge(yTrain,XTrain,lambda,0);

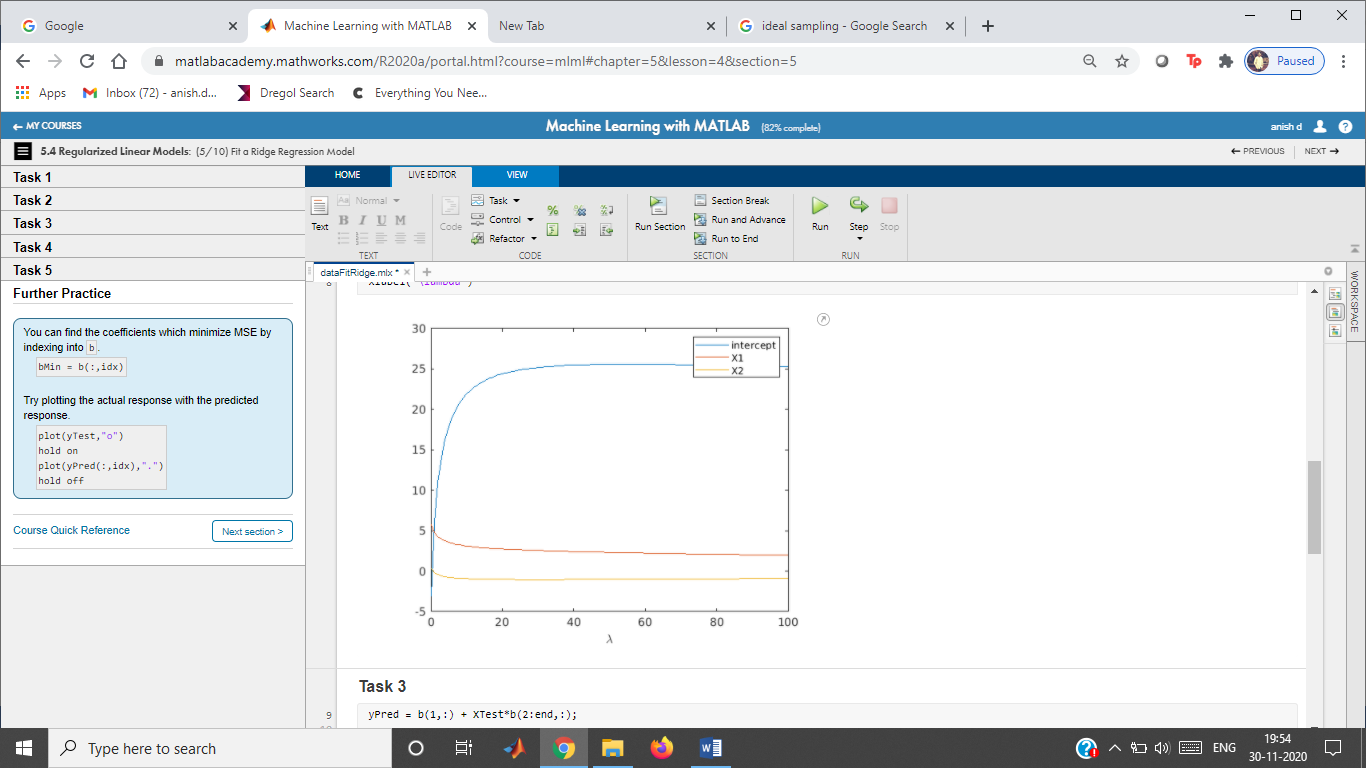


**Task 2**

plot(lambda,b)

legend("intercept","X1","X2")

xlabel("\lambda")



**Task 3**

yPred = b(1,:) + XTest\*b(2:end,:);

**Task 4**

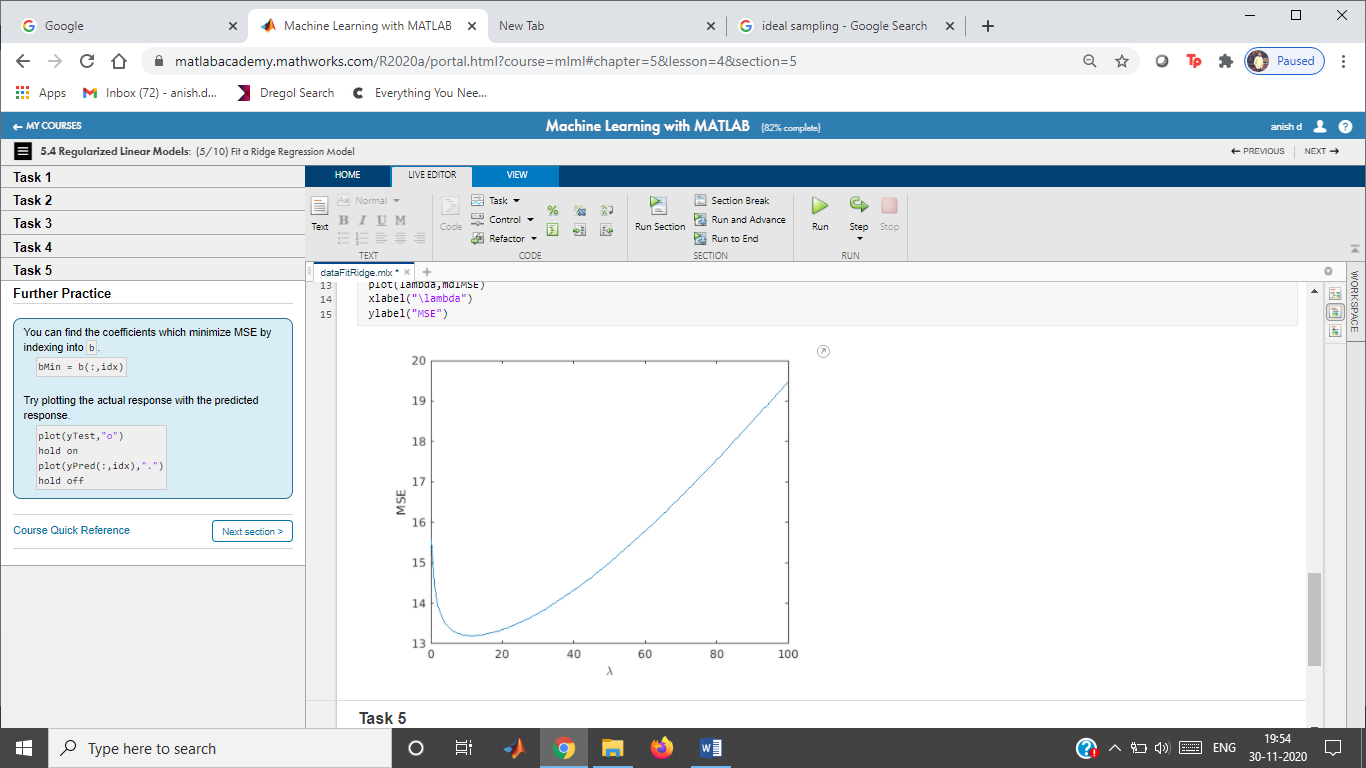
err = yPred - yTest;

mdlMSE = mean(err.^2);

plot(lambda,mdlMSE)

xlabel("\lambda")

ylabel("MSE")



**Task 5**

Find smallest MSE

[minMSE,idx] = min(mdlMSE)

minMSE = 13.1973

idx = 12

**Further Practice**

Display coefficients and plot predicted response

Fuel Economy - Regularized Linear Models

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load carEcon

whos XTrain XTest econTrain econTest

Name Size Bytes Class Attributes  
  
 XTest 192x29 44544 double   
 XTrain 408x29 94656 double   
 econTest 192x1 1536 double   
 econTrain 408x1 3264 double

**Task 1**

Fit ridge model

lambdaR = 0:300;

bR = ridge(econTrain,XTrain,lambdaR,0);

**Task 2**

Calculate and plot MSE. Find smallest MSE.

econPredR = bR(1,:) + XTest\*bR(2:end,:);

err = econPredR - econTest;

MSER = mean(err.^2);

[minMSER,idxR] = min(MSER)

minMSER = 1.0683

idxR = 109

plot(lambdaR,MSER)

xlabel("\lambda")

ylabel("MSE")

title("Ridge model")

**Task 3**

Fit lasso model

lambdaL = (0:300)/length(econTrain);

[bL,fitInfo] = lasso(XTrain,econTrain,"Lambda",lambdaL);

**Task 4**

Calculate and plot MSE. Find smallest MSE.

econPredL = fitInfo.Intercept + XTest\*bL;

err = econPredL - econTest;

MSEL = mean(err.^2);

[minMSEL,idxL] = min(MSEL)

minMSEL = 0.9954

idxL = 77

plot(lambdaL,MSEL)

xlabel("\lambda")

ylabel("MSE")

title("Lasso model")

Tree and SVM Models

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

**Task 1**

treeMdl = fitrtree(dataTrain,"y");

treeLoss = loss(treeMdl,dataTest)

treeLoss = 42.6345

yPred = predict(treeMdl,dataTest);

**Task 2**

svmMdl = fitrsvm(dataTrain,"y");

svmLoss = loss(svmMdl,dataTest)

svmLoss = 36.3405

yPred = predict(svmMdl,dataTest);

**Task 3**

svmMdl2 = fitrsvm(dataTrain,"y","KernelFunction","polynomial");

svmLoss2 = loss(svmMdl2,dataTest)

svmLoss2 = 28.9241

yPred = predict(svmMdl2,dataTest);

Plot the data and predicted response.

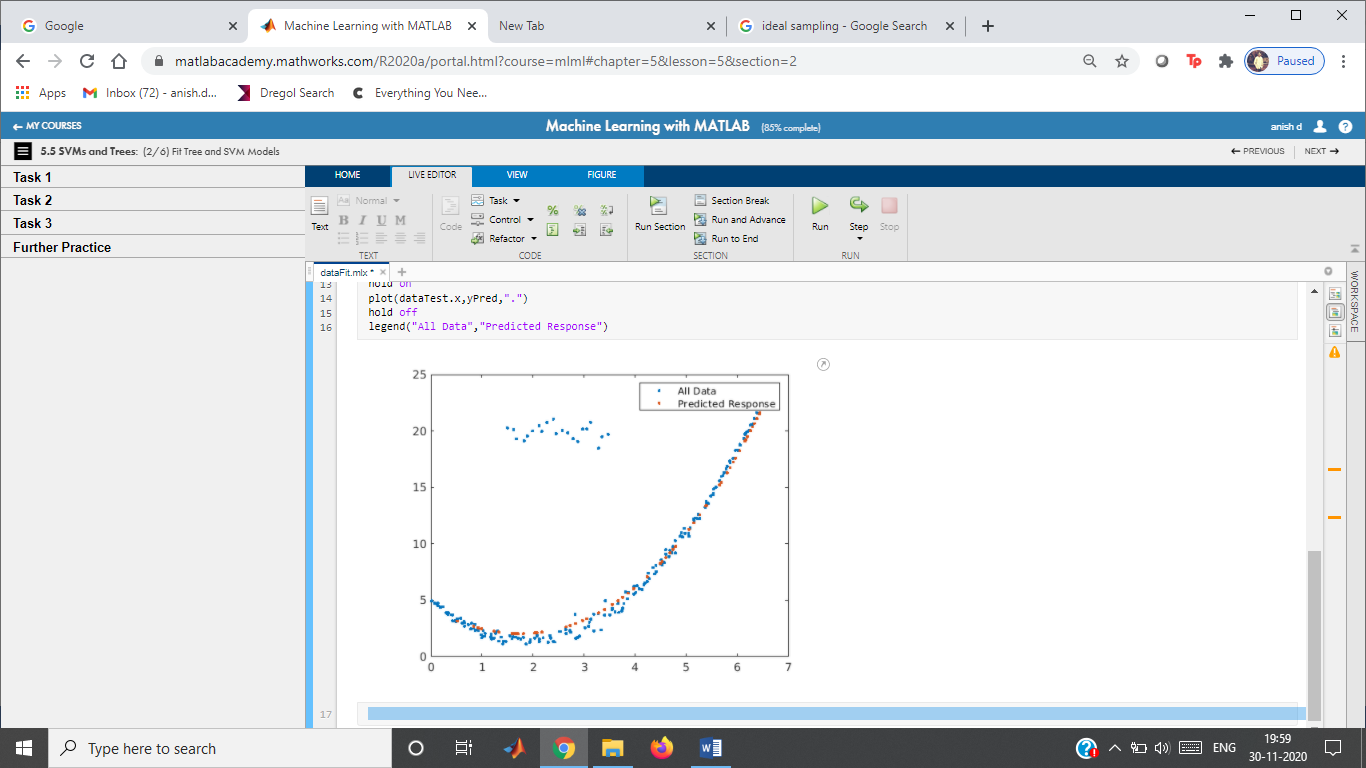
plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

hold off

legend("All Data","Predicted Response")



Choosing a Model

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and plots the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

plot(data.x,data.y,".")

**Fit a model**

Fit a model

mdl = fitrsvm(dataTrain,"y","KernelFunction","gaussian");

Evaluate at test values.

mdlLoss = loss(mdl,dataTest)

mdlLoss = 0.1251

yPred = predict(mdl,dataTest);

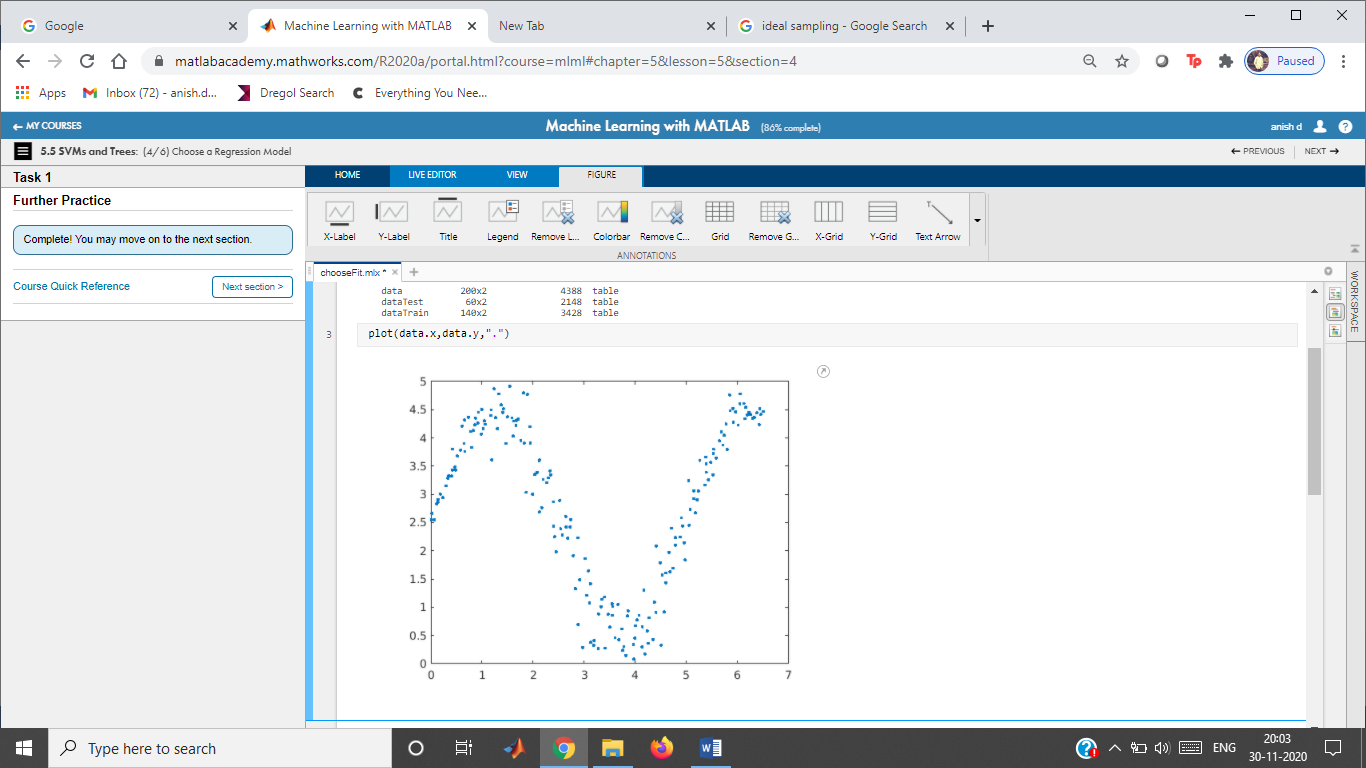
Plot the response.

plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

hold off



legend("All Data","Predicted Response")

# Fuel Economy Analysis - Tree

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load carEcon

whos carData carTrain carTest

Name Size Bytes Class Attributes  
  
 carData 600x15 48238 table   
 carTest 192x15 20086 table   
 carTrain 408x15 34990 table

**Tasks 1,2,3 - Fit the model**

Fit a model and evaluate at test values.

mdl = fitrtree(carTrain,"FuelEcon","MinLeafSize",5);

mdlLoss = loss(mdl,carTest)

mdlLoss = 0.9554

econPred = predict(mdl,carTest);

Plot the results.

evaluateFit(carTest.FuelEcon,econPred,"Tree")

## 

Fuel Economy Analysis - SVM

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load carEcon

whos carData carTrain carTest

**Tasks 1 and 2 - Fit the model**

Fit a model and evaluate at test values.

mdl = fitrsvm(carTrain,"FuelEcon","KernelFunction","polynomial",...

"Standardize",true);

mdlLoss = loss(mdl,carTest)

econPred = predict(mdl,carTest);

Plot the results.

evaluateFit(carTest.FuelEcon,econPred,"SVM")

# Fitting with Numeric Arrays

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos

Name Size Bytes Class Attributes  
  
 X 100x3 2400 double   
 XTest 30x3 720 double   
 XTrain 70x3 1680 double   
 y 100x1 800 double   
 yTest 30x1 240 double   
 yTrain 70x1 560 double

**Task 1**

mdl = fitlm(XTrain,yTrain)

mdl =

Linear regression model:  
 y ~ 1 + x1 + x2 + x3  
  
Estimated Coefficients:  
 **Estimate** **SE** **tStat** **pValue**   
 **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_**  
  
 **(Intercept)** -8.7253 7.1477 -1.2207 0.22682  
 **x1**  -0.0067133 0.00052754 -12.726 6.0508e-19  
 **x2**  -0.14815 0.13473 -1.0996 0.27575  
 **x3**  0.71534 0.085609 8.3558 9.5691e-12  
  
  
Number of observations: 66, Error degrees of freedom: 62  
Root Mean Squared Error: 2.91  
R-squared: 0.867, Adjusted R-Squared: 0.86  
F-statistic vs. constant model: 134, p-value = 4.47e-27

**Task 2**

XTrain13 = [XTrain(:,1),XTrain(:,3),XTrain(:,1).\*XTrain(:,3)];

mdl13 = fitlm(XTrain13,yTrain)

mdl13 =

Linear regression model:  
 y ~ 1 + x1 + x2 + x3  
  
Estimated Coefficients:  
 **Estimate** **SE** **tStat** **pValue**   
 **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_**  
  
 **(Intercept)** -75.128 24.394 -3.0798 0.0030857  
 **x1**  0.017304 0.008563 2.0208 0.047624  
 **x2**  1.5707 0.32465 4.8382 9.0428e-06  
 **x3**  -0.0003223 0.00011554 -2.7895 0.0070037  
  
  
Number of observations: 66, Error degrees of freedom: 62  
Root Mean Squared Error: 2.77  
R-squared: 0.879, Adjusted R-Squared: 0.873  
F-statistic vs. constant model: 150, p-value = 2.09e-28

**Further Practice**

Lasso and Elastic Net Regularization

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and plots the data.

load data

whos data XTrain XTest yTrain yTest

Name Size Bytes Class Attributes  
  
 XTest 60x2 960 double   
 XTrain 140x2 2240 double   
 data 200x3 6202 table   
 yTest 60x1 480 double   
 yTrain 140x1 1120 double

scatter3(data.X1,data.X2,data.Y)

# Fitting a GPR Model

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

Fitting a GPR Model

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

**Task 1**

Fit model and evaluate at test values.

mdl = fitrgp(dataTrain,"y");

yPred = predict(mdl,dataTest);

mdlMSE = loss(mdl,dataTest)

Plot the results.

plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

hold off

legend("All Data","Predictions","Location","Best")

**Further Practice**

**Task 1**

Fit model and evaluate at test values.

mdl = fitrgp(dataTrain,"y");

yPred = predict(mdl,dataTest);

mdlMSE = loss(mdl,dataTest)

mdlMSE = 0.1185

Plot the results.

plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

hold off

legend("All Data","Predictions","Location","Best")

**Further Practice**

**Tasks 1 & 3**

lambda = (0:100)/length(yTrain);

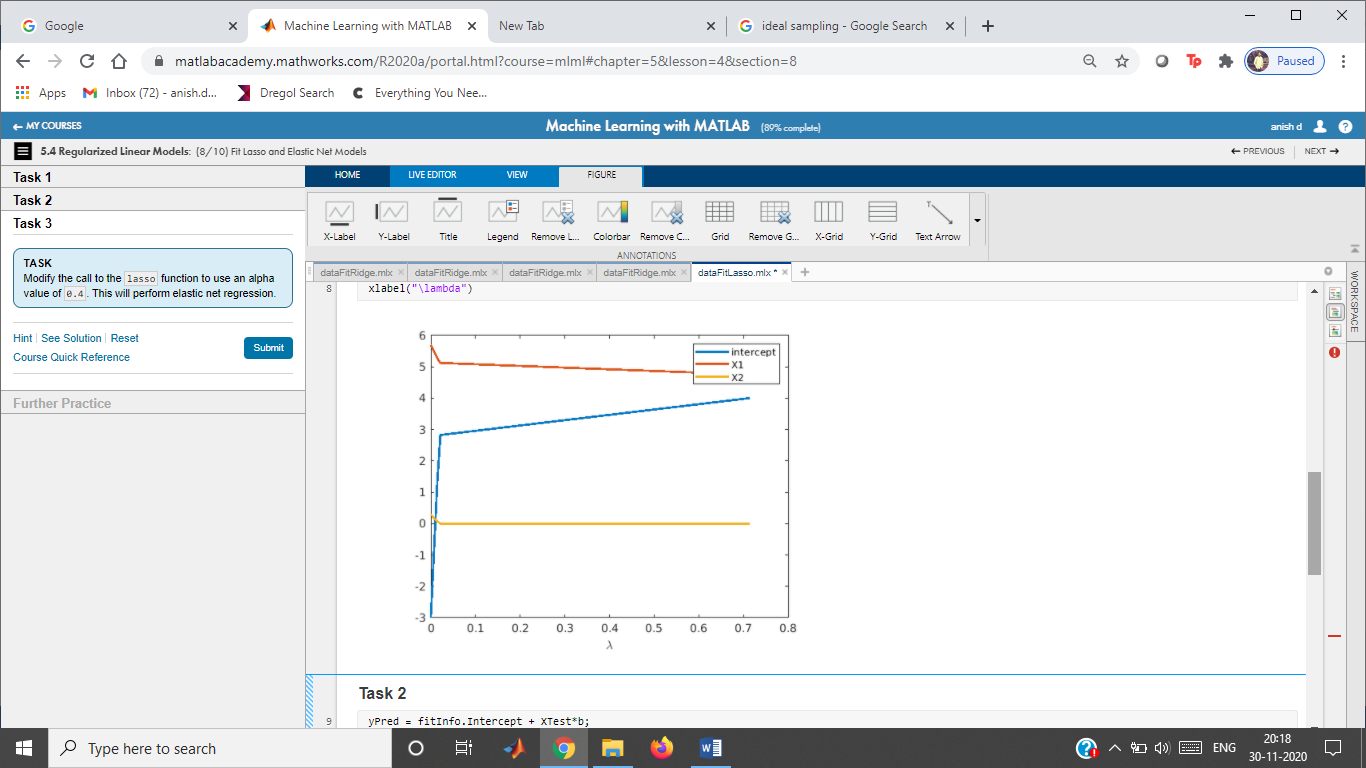
[b,fitInfo] = lasso(XTrain,yTrain,"Lambda",lambda);

Plot coefficients

plot(lambda,[fitInfo.Intercept;b],"LineWidth",2)

legend("intercept","X1","X2")

xlabel("\lambda")



**Task 2**

yPred = fitInfo.Intercept + XTest\*b;

mdlMSE = mean((yTest - yPred).^2);

[minMSE,idx] = min(mdlMSE)

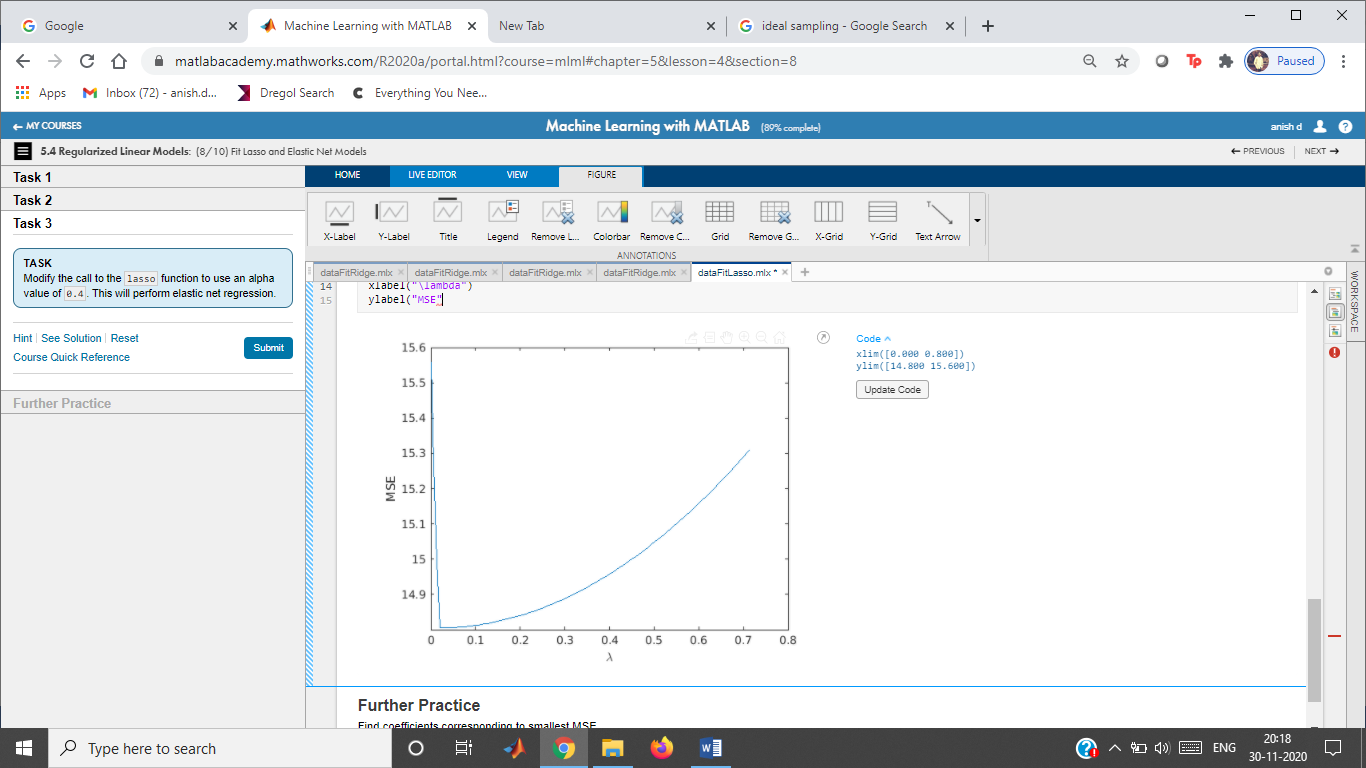
minMSE = 14.8052

idx = 4

plot(lambda,mdlMSE)

xlabel("\lambda")

ylabel("MSE")



**Further Practice**

Find coefficients corresponding to smallest MSE

Fitting a GPR Model

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4388 table   
 dataTest 60x2 2148 table   
 dataTrain 140x2 3428 table

Fitting a GPR Model

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

**Task 1**

Fit model and evaluate at test values.

mdl = fitrgp(dataTrain,"y");

yPred = predict(mdl,dataTest);

mdlMSE = loss(mdl,dataTest)

Plot the results.

plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

hold off

legend("All Data","Predictions","Location","Best")

**Further Practice**

**Task 1**

Fit model and evaluate at test values.

mdl = fitrgp(dataTrain,"y");

yPred = predict(mdl,dataTest);

mdlMSE = loss(mdl,dataTest)

mdlMSE = 0.1185



Plot the results.

plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

hold off

legend("All Data","Predictions","Location","Best")

Fitting Data with Outliers

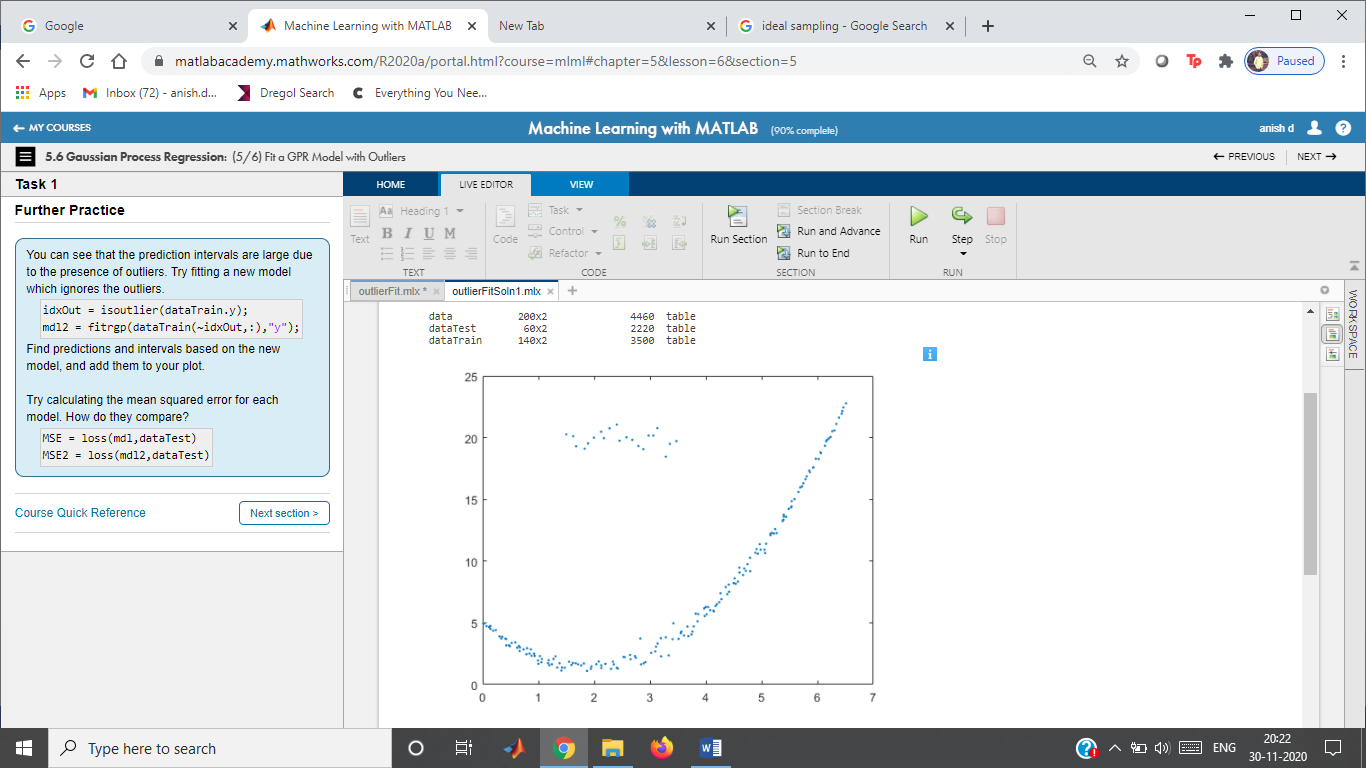
Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4460 table   
 dataTest 60x2 2220 table   
 dataTrain 140x2 3500 table

**Task 1**

Fit a model and evaluate at test values.

mdl = fitrgp(dataTrain,"y");

[yPred,~,yInt] = predict(mdl,dataTest);

Plot the results.

plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

plot(dataTest.x,yInt,"k--")

hold off

legend("All Data","Predictions","95% Interval","Location","Best")

**Further Practice**

Fitting Data with Outliers

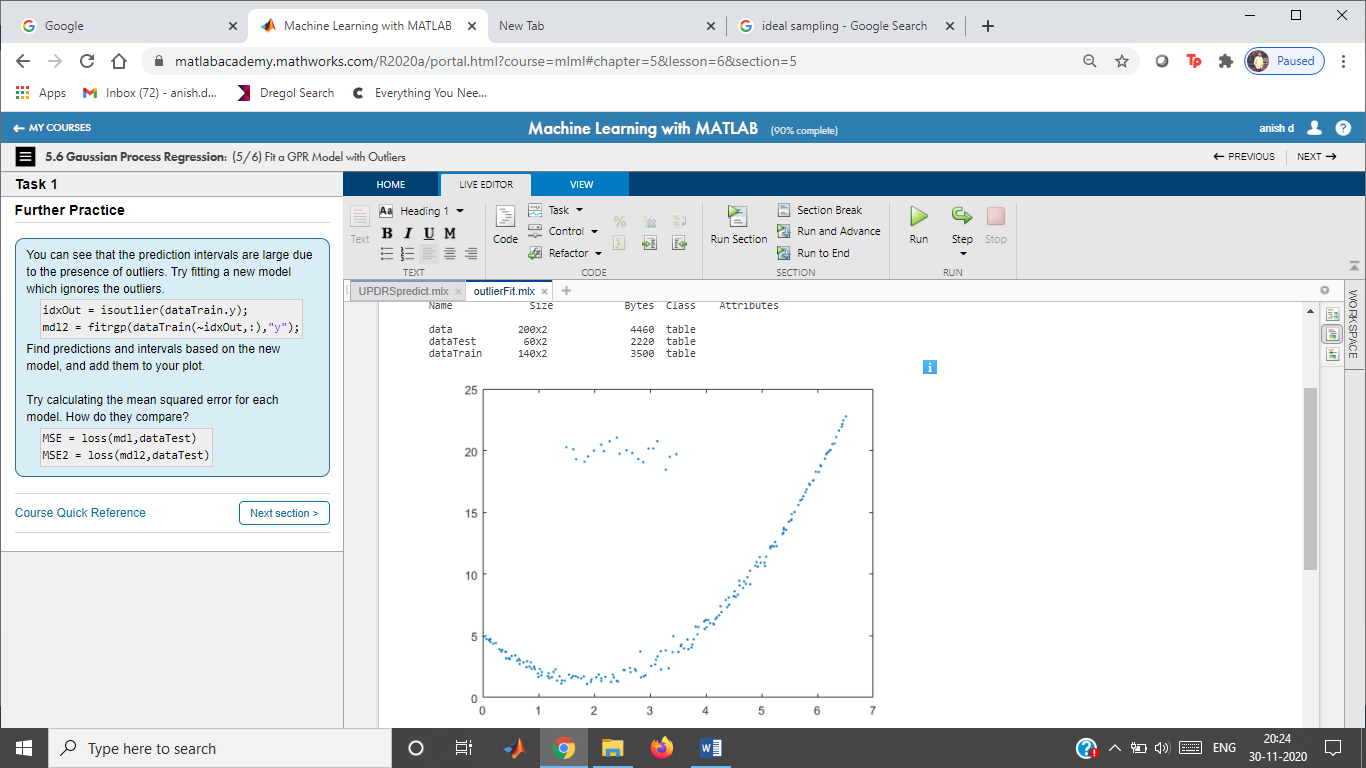
Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load data

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 200x2 4460 table   
 dataTest 60x2 2220 table   
 dataTrain 140x2 3500 table



**Task 1**

Fit a model and evaluate at test values.

mdl = fitrgp(dataTrain,"y");

[yPred,~,yInt] = predict(mdl,dataTest);

Plot the results.

plot(data.x,data.y,".")

hold on

plot(dataTest.x,yPred,".")

plot(dataTest.x,yInt,"k--")

hold off

legend("All Data","Predictions","95% Interval","Location","Best")

**Further Practice**

Telemonitoring Parkinson's Disease

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the data.

load parkinsons

whos data dataTrain dataTest

Name Size Bytes Class Attributes  
  
 data 5875x20 948606 table   
 dataTest 1762x20 290526 table   
 dataTrain 4113x20 666686 table

**Fit regression model**

mdl = fitrtree(dataTrain,"total\_UPDRS");

yPred = predict(mdl,dataTest);

Compare predicted and actual responses.

***MSE is displayed in the following plot:***

evaluateFit(dataTest.total\_UPDRS,yPred,"")

UNIT 6

Feed-Forward Classification Network (Interactive)

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads heart disease data.

load heartData

whos HDC numData

Name Size Bytes Class Attributes  
  
 HDC 427x1 655 categorical   
 numData 427x11 37576 double

**Task 1**

Create dummy variable

HD = dummyvar(HDC)

HD = 427×2

1 0  
 0 1  
 0 1  
 1 0  
 1 0  
 1 0  
 0 1  
 1 0  
 0 1  
 0 1

**Train the Network**

Launch the Neural Network Pattern Recognition app

See some predictions

Feed-Forward Classification Network (Interactive)

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads heart disease data.

load heartData

whos HDC numData

Name Size Bytes Class Attributes  
  
 HDC 427x1 655 categorical   
 numData 427x11 37576 double

**Task 1**

Create dummy variable

HD = dummyvar(HDC)

HD = 427×2

1 0  
 0 1  
 0 1  
 1 0  
 1 0  
 1 0  
 0 1  
 1 0  
 0 1  
 0 1

**Train the Network**

Launch the Neural Network Pattern Recognition app

See some predictions

Feed-Forward Classification Network (Commands)

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads and displays the heart disease data set.

load heartDisease

whos HDC heartData

Name Size Bytes Class Attributes  
  
 HDC 427x1 655 categorical   
 heartData 427x32 109312 double

**Task 1**

Initialize neural network

net = patternnet(15);

net.divideParam.trainRatio = 70/100;

net.divideParam.valRatio = 20/100;

net.divideParam.testRatio = 10/100;

**Task 2**

Train the network

heartData = heartData'; % transpose samples

HD = dummyvar(HDC)'; % convert categorical to dummy variable

[net,tr] = train(net,heartData,HD);

**Task 3**

Predict response

scoreTest = net(heartData(:,tr.testInd));

[~,yPred] = max(scoreTest);

**Task 4**

Evaluate classification with confusion matrix

HDtest = HDC(tr.testInd);

yTrue = double(HDtest);

confusionchart(yTrue,yPred');

**Task 5**

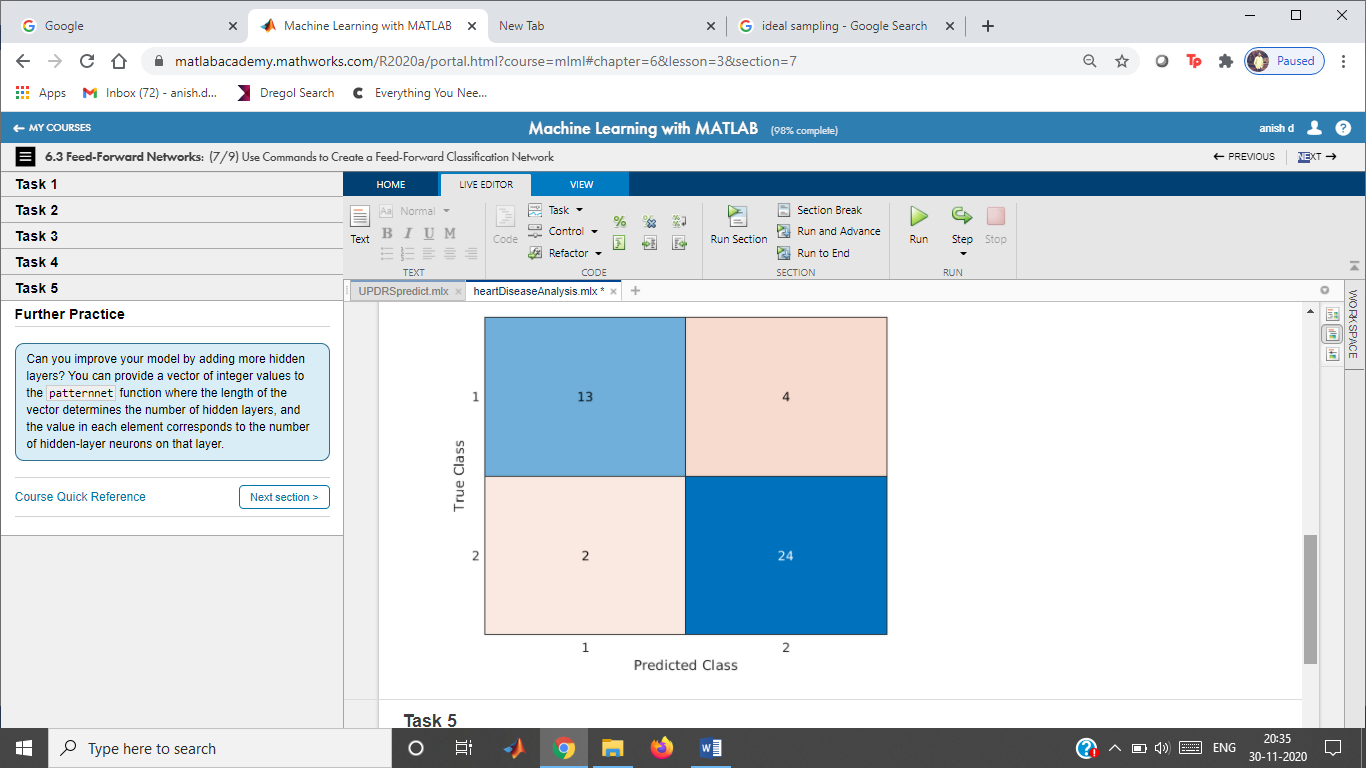
Determine validation error

HDtest = HDC(tr.testInd);

validErr = 100\*nnz(yPred' ~= double(HDtest))/length(HDtest)

validErr = 13.9535

**Further Practice**



Feed-Forward Regression Network (Commands)

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads fuel economy data.

load fuel

whos econ carData

Name Size Bytes Class Attributes  
  
 carData 600x33 158400 double   
 econ 600x1 4800 double

**Tasks 1, 3, & 4**

Initialize and train the neural network

net = fitnet([8 12]);

net.layers{1}.transferFcn = "logsig";

net.layers{2}.transferFcn = "radbas";

carData = carData';

econ = econ';

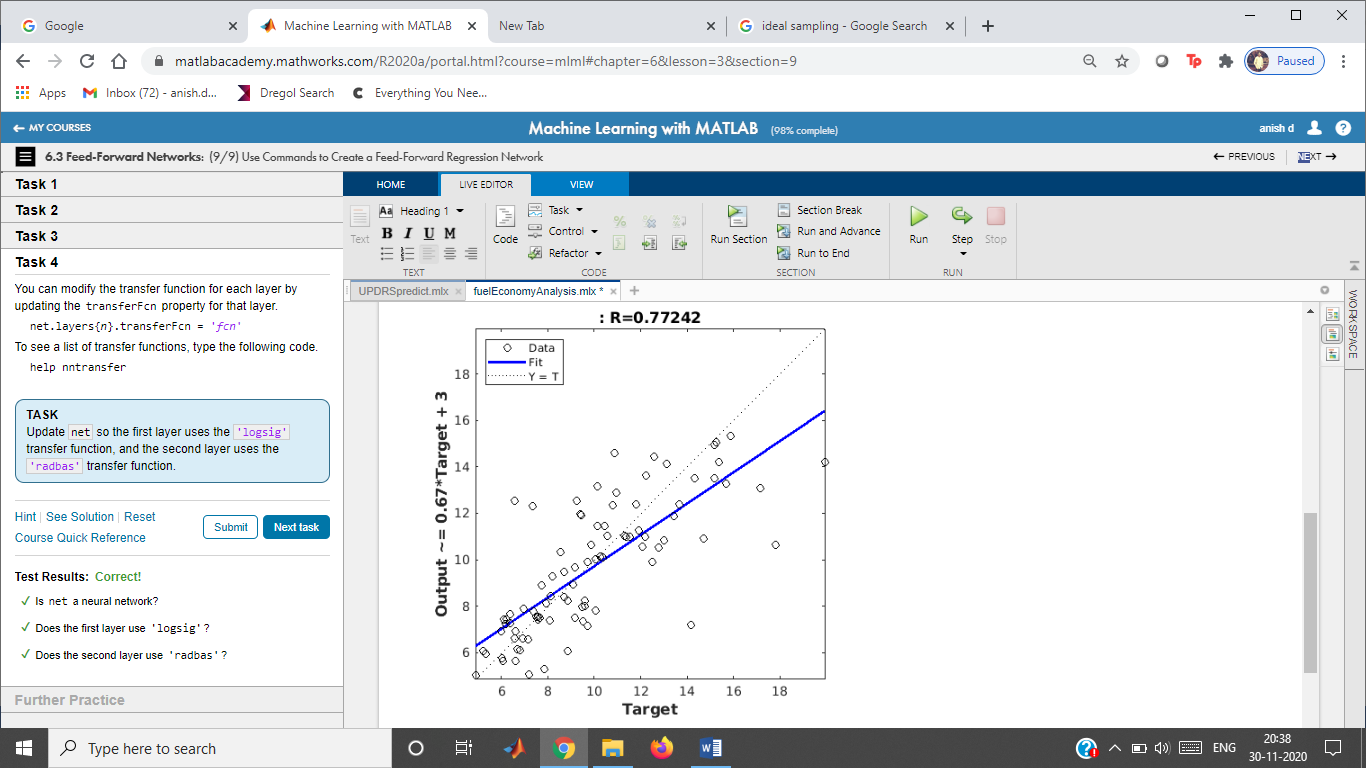
[net,tr] = train(net,carData,econ);

**Task 2**

Predict response and evaluate network performance

econPred = net(carData(:,tr.testInd));

plotregression(econ(tr.testInd),econPred)



Clustering Activity Data (Commands)

Instructions are in the task pane to the left. Complete and submit each task one at a time.

This code loads the matrix measurements.

load activityData

whos measurements

Name Size Bytes Class Attributes  
  
 measurements 2727x42 916272 double

**Task 1**

X = measurements';

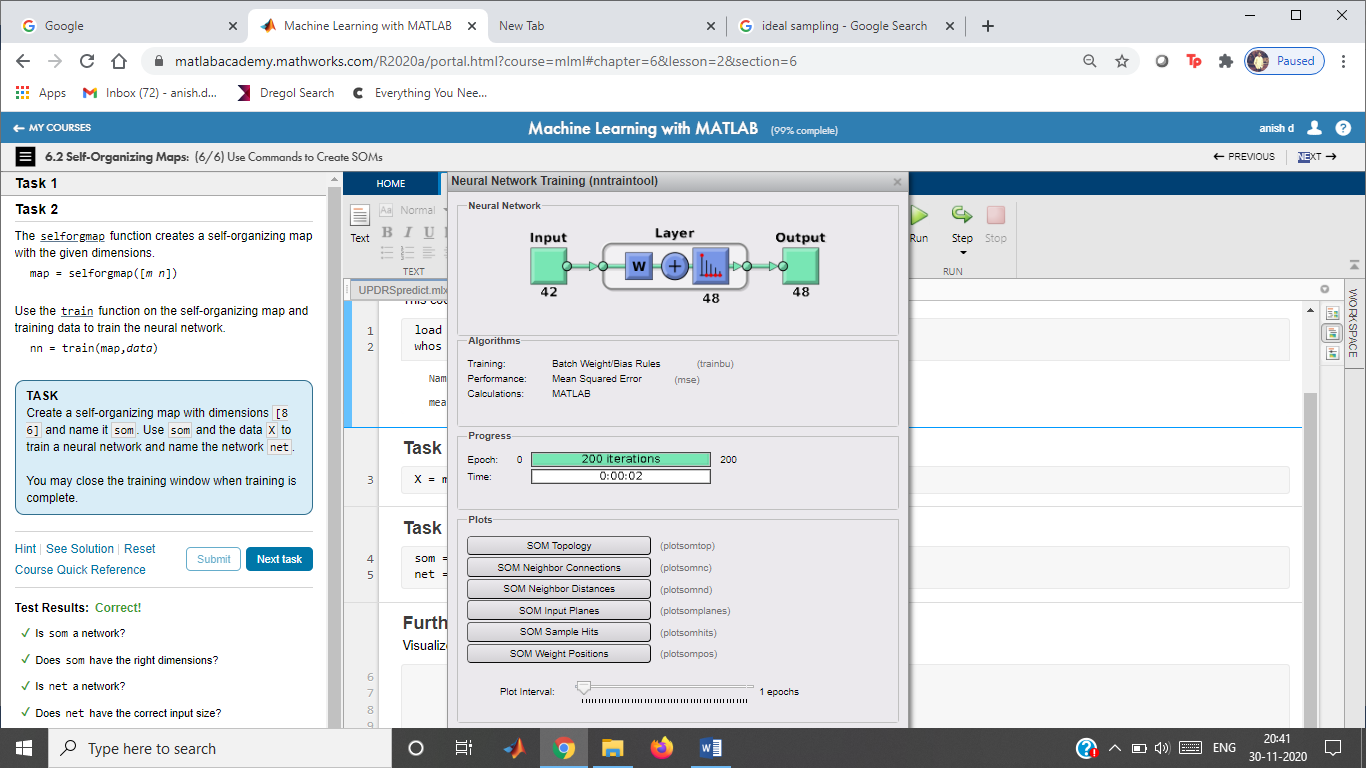
**Task 2**

som = selforgmap([8 6]);

net = train(som,X);

**Further Practice**

Visualize the SOM



## 