Regression Week 4: Ridge Regression Assignment 1

In this assignment, we will run ridge regression multiple times with different L2 penalties to see which one produces the best fit. We will revisit the example of polynomial regression as a means to see the effect of L2 regularization. In particular, we will:

- Use a pre-built implementation of regression to run polynomial regression

- Use matplotlib to visualize polynomial regressions

- Use a pre-built implementation of regression to run polynomial regression, this time with L2 penalty

- Use matplotlib to visualize polynomial regressions under L2 regularization

- Choose best L2 penalty using cross-validation.

- Assess the final fit using test data.

We will continue to use the House data from previous assignments. (In the next programming assignment for this module, you will implement your own ridge regression learning algorithm using gradient descent.)

IMPORTANT: Choice of tools

For the purpose of this assessment, you may choose between GraphLab Create and scikit-learn (with Pandas). You are free to experiment with other tools (e.g. R or Matlab), but they may not produce correct numbers for the quiz questions.

- If you are using GraphLab Create, download the IPython notebook and follow the instructions contained in the notebook.

- If you are using Pandas+scikit-learn combination, follow through the instructions in this reading.

What you need to download

If you are using GraphLab Create:

- Download the King County House Sales data In SFrame format: kc_house_data.gl.zip (https://eventing.coursera.org/api/redirectStrict/QaV9r4ZHTmI671q2M_wFUkMgglWM0u _Ah8sB-Gl0kXg0k37VxRhM29H-mr5FfPzIu2z100iiKIzAITDxZKeSyA.rVqYdpP3aEpqItm88ul- dg.lW8I1XcGjvv5pDAwi- IKOan3HQgz8wj924_ZILVS_aE6m1wHPyL4eYQw1rsBhzL1Radwtwdsy9adFqVlHi5nPQn82-

jGuQt0zX4VNuwlya2S_mQNI6Kw3XI-
2QMxT1Wsh63YDU6y7WfFg6FdB_8R4s8Snzhk5PeuLXpfw6J7DQva2PCWouh-
cJt2kSkp7paPDBWVv_gpMWaUyDqChUE2JgZVLGrXbxH1vKw3KOpF74Q9eJd2FvgH4qnrPe
M-Zmgp--
KR1FKIuv2g0tu5YeE0D539aEmnUu1fcoZ2C1k39IDFdCqqJrQCoyRlnhmevk5pfYgpZChCxR0
vl9HcufmdZH22YpGhki3CiU3P5maKVQxCSed1Ai2fr58dtz6pJV3F_DHqqvl5NafdR7dkTQGit
h3HaKW9THcV_txxzCAN5CeuYTrqKma-gI6gb0ozXL6W)

- Download the companion IPython Notebook: week-4-ridge-regression-assignment-1-blank.ipynb (https://eventing.coursera.org/api/redirectStrict/k5tQb82hK0zyHFua-
bQH89_yUUIj2xywGTYTjDcjmxc7x66SPyhDliGBc-
znTst9J2gEyxnBArfUT6yiPlibEQ.Cp9hMBawVEssHnMC688PYQ.IaosUItqMiQMoMrhXDluG
4CIgjAWO3N-wp2WMtON6IjylWVreiAMtVy2aHhNOOTAv48zVWAnNDzxRJQ2FFDK-
bgqkcmcyqutbmueCH7SdsbrHnOuwRRUWZH-
E7z2YyOdnlicnA0ZByNe8QAOFwwbsEhiZed2GjOPIJYO25d6k_7R5peAlgy7rAqc_qk3ShyuBj
RmrikW4PLMm0RKsu7yI7I9KmGq4mbIz3MWcw8I1EQQxVwEWq4i7DOc6AhoDdB-
oDmOH8a2il0aSPwBE7PMZivnQU_GpyTucs7x3cWy2sW9MOpOwqLeTt0eLO4KTXWc98KR
9gGWu02kngEi_zcD1yVE6bxgghTnZljxvoVBp4JIVHUVxVhJjrvbdR3J8mL8_Jf0gp5NBVMkFM
wn5hy_e8GnIoHOUd4FrMO6E-o_W3VoszsNpabPspLhe-
B8BaNWn1c5oOV91rHRI5K5euMH7fvY1khHc-FLzZhGkFcToSwpplo-8vekCIjMwPRXq7C3)

- Save both of these files in the same directory (where you are calling IPython notebook from) and unzip the data file.

If you are not using GraphLab Create:

- Download the King County House Sales data csv file: kc_house_data.csv (https://eventing.coursera.org/api/redirectStrict/bd9jdEmVv2NYeLk3YPIj31XtEzKVbAfIwi7
At-
T37cHeyN89gCVvYRGnZhrUjjK3rH24jttw_56zYfGPVN103w.CxNfTtGAqDD6tX0FQ3s4VQ.71
ZhgmpD0way8tcDdC8dQ3LcVccGT-Gd9P1JhdvrdG2-
Y1OszLOtVmGltb1GwdygN3uoBFA93wH5KuLXWGG1RWIxd8VjZMTTgzwoP3bZ1f09hUMQ
YHopNQdLSNpXZW-v13_NLmtENqQM-
QlwQ_cFuF7X7OKEU5_QZPNO72k_aqx69Belt3hNMJ5soIkPFRlpql5Q-
xzxHZAUNxHbgQrW2PqgU5cjFcoLmsxEaDdbncdOygGbDxJvUnpLjfLGnSeDzd5nhSlw2VU
X94lYAK8N6g8w9AsecogJCwyrg6LI9_-Ekj7lg9C4pzfiAdq-eRs_JiNgZ_Qy-
O6T83TVTy8o6UzBYnN-TtNIn4IYADEewtkB5wtgY-
IR3QQBbkpGaGmEmhNns0ylT7UrZB4Tav_DU52ZC5iU-
aQymVWVDE3yQ8rYqCju3ZiGmTVilV8E9ZuD)

- Download the King County House Sales training data csv file: wk3_kc_house_train_data.csv (https://eventing.coursera.org/api/redirectStrict/7p7tZHgOYeGAPkIrhpzqD67J6_Em4RV-
N3qwUWVdNXDysZWd7GcDgvbi-mijvgSgdS6in15x86OUOWmpFYMzsg.nk1-YVpSYJnI-
okbCuzcfQ.FWIs11_C_8JAUpw-lAyP0OH3BB-bmmKOGUfyTPrfLxS5-
8Hk2kWulzvBV9KPsRTCTG98UUcAo-
FiSpT6dl7HsLmTLQQK7ViS7pLTns361pJdaCckZOFfwneoNUCcAQzQsCCTRMWdO47diizho

zQ1QiGCslubHnfd-
_pyKVS_sAZFzM0A9YAVSI1QoswoufaGl8JTYTaT4M6OpdBOIl0_bSz0q21T6hk_7boWedWs1
Fhi9w6BCXUXM___t31ZsnYqbCXY_DLzXr-8XLifAlh2eUovmqP7y5Ex-
jVuI0QrtyIUdjWu3aVa8H1tfAjxTQA7efWCx5LRV3yRIGszmFBQJktzKKNscGtexpyMBRFxyJYT
5IqzQliEjToNBt4rIsdN6TPJ6uQJFROVQSaT_d0ywOaIRAKbb3vi5lPBs8BwIVqSYqfL5lAOpqCn
POUK536QwiLrUvfpdw7LzUarKFQp8g)

- Download the King County House Sales validation data csv file:
wk3_kc_house_valid_data.csv
(https://eventing.coursera.org/api/redirectStrict/WXQ7ieNyf7LRQs-
cIt71RqtFksespWGnoQG6rAKfiDrv2f7cqA7-
KAuN8bOCmo7dFJe99_yab0IqtiNqdPtuFA.bXMbK763EnmnYMo-
ahZGxA.Ap75iJqZdiU0drT3rie6xvOgOfZPqX_rcqiZJr1iuGE18HXyrUMWjvYvledtIoiMZmhXVe
oVVVu7dpDGwQGfx1PkZkVQ_mpeUlhAdfeFg2gFnWqQifGp0yGqS_IBrte-otUzWgIkMo7-
kzHkMp38Ns0IrPXKtlJEKo_pWMcq0aZjXkQ8GMLkUhUplZtXVJkPS3zJpSDGWr5EZBlV5vZ7T
YHvgx2TVNbTwPmL-jC2jOiHlwCy917O_sEB86Hzmj2z0MKnoVcHk4wjqWp1hPQPeqE1O-
yYVUuvleut5Oy5gDPTx89M7yFcClO3rox-C6n_O-
xFWjJF0HLJLnffV3DWvzhkHUbHzo9tFaKNTOGWEdcocFyU5BYN-
xSAreuJd7oR8lGnxUMcAkCF7mO5B4E0wUl0_jSOH7oldW8B3oQrarVECriwDuxwoVUqnGn
weqMGNyG1maSIhxR8vJFTLetuYQ)

- Download the King County House Sales testing data csv file: wk3_kc_house_test_data.csv
(https://eventing.coursera.org/api/redirectStrict/RIUMWW9Tnca4GqcTzurTBtC-
ynkcXOHP6j51gSKLZr07M9DG79S0SnTqmJzN0aXLlfL4UC9Ml7fcJ5Rv394k9g.ipYoD4XcgEo
bWSmtNzDnnA.SIA_rs_ahkwycDcXjhAyfBtdf8Uj9r48IfzU8RSBVk1gBupw7jJTxsiTPI5soQsO
qWefOjdjLDl0KZfxXLH3Svr28Cvut71DEuIZvM2V-ouS9-
YCsp9bsnlBxwFkOtfXkpQ99fsmUVqIgoId4YHFoWt4Dc_S1c02tVZtPK_e2uhqybEjVHnWY0O
y0SlW4aZ9k--
GiAwEPO1UEdcbbnHGazF6hGbrxbUdBkOg624aFS23FSaNSNXSrsKTYzzfVaSNDrnY2OQj9y
kls2zgJt5qJASZehaGVOKal_7cy-lvUY8P4ybW8DNoKmcRs8Nif7R_ulpbUwVNqAVt4KzL8g-
rElFB863ZibW7FAZy3CBrhCfAHjOpifYRoJ6UdveWjwPBpW_QG-ycJKAcDP0XuIQ7f0Q-
93172x_8IFqkQlkIVPuu0AHaZX5Zty97sBe8PdYvwV0C8BY6xxnPohuervg3CQ)

- Download csv file containing randomly shuffled rows of training and validation data
combined: wk3_kc_house_train_valid_shuffled.csv
(https://eventing.coursera.org/api/redirectStrict/CPIyPMxDgyjPIgiQmCTIKUzEHRo0lvq7KL
i0Hx_5lX_L37lM0vozl12yXyizclKy3Sa3_WcvvAj-9ceFVPol-Q.Sp8BNsMsElJ-
YoKsSzfs3w.2QaS2pdnQweWe9JJ8gflNzqNZ3GMFSYmh1d7m8F_WHwDRwlp7hbWr5Btn3
omDsAN6SK39CRTAIo3RKx34npXeu9Okph4ScTjoFS7iKmefZV-
eJSZhpuv0Eo0orZWpJgEH2e43Vy-
I9Wn3HNDEZxcedPEVchsDV2XhX290eMULnVjcQtBpSc6VlRRnhQa2JzLuRcQn7TmnkgNRj5
Y7uKP1aNtPvtiui3CU6K2Za4Zlk0ye_g9QDlmZwmx1HF8nR0MDVEl_lGBU9l31wchOaLMBP
9rYp2MgMi--xEtj-iav3AetLEityHtPg3-
2EDPc7vhua4GfY3rgRzw_07B7llJV_dNNtlFPkC3qpmlDPdzCPzR28M2hsOMw2kaS3i2Jmx_
mQc8ANj4EQPbfrWdljZphcvUf671hvNDYjRfKPISzMMPusdwrdCP4YPKToCh9kO9UQbWQr
qlFzs9dF95q2GmqmDNBsV3QvBmcLq-E10icJo)

- Download the King County House Sales subset 1 data csv file:

wk3_kc_house_set_1_data.csv
(https://eventing.coursera.org/api/redirectStrict/P1EaaeoohUtlqSGWeFfzL_bEABFkLIMZtz
TlFBbjBozChcy0F5142R64-HiKbDVj10xVliUWAogeZiVJ9G9WFg.ZbXO-
a_87f5BLJ2fuEPeXg.qg4vsqjghODW2HwnIODosfcZdFqq1-
3p7XEAsjFu0hfeLm_ocd_ddDkvrNka42HqiXgAFPRDqP5OrqRSaYNjb8ZM0fpwprBb-
4bpmF57AlUNj8oYpjYOLXs-
F4ciFgl3CoGHyPHMO92zFctB_wyYF_pnwUMOX9m0pUbtkWDd4fnmL8wZ1FpDoj_Dxz2XUj
jM6-4JVrHpv-lOBpSwnTKiex1tgvXDw5VKvOxE9ibvkRxp8iJE4-
Gd13uCTqAAynMG5ze8d1PhlXiG0Rf6evvh_LX-
0HNmJqRllECfGJ6C20T2vNSH3qRP7LN5iHygG_2patydm3an2xIXszyQYckQLbg_BAb1-
GbRonx3E6PnVWqsmJFAQ4is4OTHW3n75YfOA3TLgAAP44ajlTh2QKehz83-
FlqzIH5WVjcLSOXF_K_XkyJR9a2mlJHPVwVGIXKmL3AAjGQ6Pgy0X6F1GWTCNQ)

- Download the King County House Sales subset 2 data csv file:
  wk3_kc_house_set_2_data.csv
  (https://eventing.coursera.org/api/redirectStrict/FHAf6kgVD2A3hc3MWa8EKZ2kqXgOinLt
  3v1ei5TVBlVE9AaV9O6hZcEt3NKsYmf8HvyaNEEO0fNS5UbRjkWhVQ.1FkaJuyaf8CPodDCn
  plCCQ.r07v0MoXkeMpyEv4OLC-DV2IAeow5ncZoD2BrnLfBy6dV4DPaeoDnnBodFJGINj-
  B0MLMYHVhrINoWJfbEf0pZ9YBhkvw01D7ONb-
  rJxDUfkGUtRC11S7VgRtehtAcYckzkWPLZOm0fEayp1yN0gG7Ai1e6_J8awd40kx1U_znFphp
  215tAm3yKK5PlrtNgS8uSb6fL1Sn3qQckUAbjd1UcjdMOroFroI4XVtyl3jE9TYkR5d-
  B2tsS8QSq4PKpU1HwocTXUqaQBvyE4uy5DUpOOlTiGuHmWhAIrGIGgfpDDDDyUM3oW3i
  nQbdM7vPNX1tMJY0pPNCsqxrcUYl8qo8-wcQd4OSEqIxxdH-
  LrTN_TO06gtdrXvwfXopKDLnJ9LauOnSjKOHYdzeGe_5T83fnnpyYmEGpiqVm72i7RhjHiy6h
  nJhKVJj4HcigdQJOr-bYkewV908TPOmqY9-S_qQ)

- Download the King County House Sales subset 3 data csv file:
  wk3_kc_house_set_3_data.csv
  (https://eventing.coursera.org/api/redirectStrict/orl7ayVmcgnwRle8XyIkABGn-
  uJvRuHbcDZPePHrVMjby2f5aTbnMRGdjjZgTNGwHbxs73qUm50ieBg09ezkUA.IS_jRu_xH7
  mHPNiAduJKhg.QS5zztr5fGaS3jRXOkEt--
  o3GPQi4fNXE5ASSXtfjQVX8SfLoGPVFxwPazzFFdNcuB64I5dDT8rHiQc_9Mm4sJd3o25c6-
  VuY2DJVD5NCS7JdIvcnuSsnrbAeSzL57w0RSDqYboWarDqoU1FWwI9RgaUsJOHVcAwM2K-
  P6p-ZCmB6yfGvsto21tf1LUy49xIxBG0EQgqFnajm-
  XKD8NDLLktHVlUcTTPfeJTbSHg7ZI0qac-_zjMXnK16UACsvx-Etmf3AEyU2B-
  WCCizkOh1jmE_LpwcQKzOMcvtlfk8w40W4ETsqt36IJropqQkY0XnrnSy9G1lm__ePijqoCh-
  CohkUYDPFW1AKuiinj7m3HqfI95IohV9ziexs9V8kkbcEjoq4zGUYzYNThj1FCaoZURej7__NJs
  7TC5k_9jpkZJgJWY4jP1MYp8xqQU6t9XrGfYQ7dn7rSBCMegqXmO8A)

- Download the King County House Sales subset 4 data csv file:
  wk3_kc_house_set_4_data.csv
  (https://eventing.coursera.org/api/redirectStrict/kmfKSLGWLQX7tJJjZ0XDYWQZAWbcXnK
  nkCgdylWVWUSaX7pEZUhzmYrMtmv5oMV_iONqJCcUdAlfCms1PeCVfw.L5H5C9oiKx6QmI
  twhZaffw.HyHtfzeZuBb78Fodc5k4GszeTpLeZFvnwZrGzIM5qYig0GUcCekOkpyHNEMLCy2
  7BdPBXKBscPfX6o-U0Lk3W6aVVRxkUG-
  xkZI8yMfi4fjveFPCPDWc2ysHRQs9b7LGaWl0_1IzunUdZQJGNocm2AQb__eNG-
  66apNKSluQh7KnAplBbLx7nwSSKgY2JXc6cBHoO_paHZ0Y4m0klKRSkXjwWTOIouerDrdSv
  hfCQNCw144NtOZk-

gOrSi0I2ggs_aeDayUpQTZibtzDShYZnfWvr6zTTjzwxUIg6pP00FD6BZjVWc1S-
23QLh5r7yeaRw2x7uuq_2dwqy7CIspKsfirXnef5EOznztniU19X3yUkwhMIfjg4fXw9pLzVn_zI
yigW5wMfa9G_tF7xPSEvlf5o4sZ-
2DuRH3YAclY8k4MZf7hQlisTZrsnfX3Oax8OH58htjghOD8wWWiU38Gog)

Useful resources

You may need to install the software tools or use the free Amazon EC2 machine. Instructions for both options are provided in the reading for Module 1 (Simple Regression).

If you are following the IPython Notebook and/or are new to numpy, then you might find the following tutorial helpful: numpy-tutorial.ipynb
(https://eventing.coursera.org/api/redirectStrict/lr0esQhBmJnPf7By6GnxSOIXewOPpOrmd--
A7dTlhyy7aMd_r9yuA5DPSzbnhftn73rAa6EtgGAl837xxvHwBg.tXueixbYTt_Xp2faVwb4pQ.-
c8E9NG0udJJUl3HE15Sipl-wAmgY7_bEXtzHUnnUO-lUfwboOHEXu7N7JO1jS4cbZT-
A9G0e3p2LJsZw69nvs3AvhMFjMKRW2OxHwgqFdEqHRu8tCDPLvCGzPmF0yuZp2AbOowXkia
1t5hKbwNMCtl4yA5BMxkdlo-787CJGjcjuebKxNmMsMn-
SiKAX2xyDfATz_kpUvuk8hcuJHieSFhhGFnkMCuqMT3Cv2zqtFqWr36Zs0MoqFmnbMNpewXz
04xCii_LKUy42PohBSJopwzNK3zwCtx0nfApp4__wgRMl_yj0h5Ey4pAi7eIx-
WsKRkaHAYY7tRCSu7er1I6ftFqhLrjZk4DqD3k0AXrR4_d8iiQt614KCx3pA8mhlpcTKHvCCyQ4IS
fh9zNAqG8L_BSczjw2haO_hMWfVgi9pKf7N4gyJUljBVM8BD3CRK1)

If you are using GraphLab Create and the companion IPython Notebook

Open the companion IPython notebook and follow the instructions in the notebook.

If you are using scikit-learn with Pandas

1. Copy and paste an equivalent of 'polynomial_sframe' function from Module 3 (Polynomial Regression). This function accepts an array 'feature' (of type pandas.Series) and a maximal 'degree' and returns an data frame (of type pandas.DataFrame) with the first column equal to 'feature' and the remaining columns equal to 'feature' to increasing integer powers up to 'degree'.

2. For the remainder of the assignment we will be working with the house Sales data as in Module 3 (Polynomial Regression). Load in the data and also sort the sales data frame by 'sqft_living'. When we plot the fitted values we want to join them up in a line and this works best if the variable on the X-axis (which will be 'sqft_living') is sorted. For houses with identical square footage, we break the tie by their prices.

```
import pandas as pd

dtype_dict = {'bathrooms':float, 'waterfront':int, 'sqft_above':int, 'sqft
_living15':float, 'grade':int, 'yr_renovated':int, 'price':float, 'bedroom
s':float, 'zipcode':str, 'long':float, 'sqft_lot15':float, 'sqft_living':f
loat, 'floors':float, 'condition':int, 'lat':float, 'date':str, 'sqft_base
ment':int, 'yr_built':int, 'id':str, 'sqft_lot':int, 'view':int}

sales = pd.read_csv('kc_house_data.csv', dtype=dtype_dict)
sales = sales.sort(['sqft_living','price'])
```

3. Let us revisit the 15th-order polynomial model using the 'sqft_living' input. Generate polynomial features up to degree 15 using `polynomial_sframe()` and fit a model with these features. When fitting the model, use an L2 penalty of 1.5e-5:

```
l2_small_penalty = 1.5e-5
```

Note: When we have so many features and so few data points, the solution can become highly numerically unstable, which can sometimes lead to strange unpredictable results. Thus, rather than using no regularization, we will introduce a tiny amount of regularization (l2_penalty=1.5e-5) to make the solution numerically stable. (In lecture, we discussed the fact that regularization can also help with numerical stability, and here we are seeing a practical example.)

With the L2 penalty specified above, fit the model and print out the learned weights. Add "alpha=l2_small_penalty" and "normalize=True" to the parameter list of linear_model.Ridge:

```
from sklearn import linear_model
import numpy as np

poly15_data = polynomial_sframe(sales['sqft_living'], 15) # use equivalent
 of `polynomial_sframe`
model = linear_model.Ridge(alpha=l2_small_penalty, normalize=True)
model.fit(poly15_data, sales['price'])
```

4. *Quiz Question: What's the learned value for the coefficient of feature power_1?*

Observe Overfitting

5. Recall from Module 3 (Polynomial Regression) that the polynomial fit of degree 15 changed wildly whenever the data changed. In particular, when we split the sales data into four subsets and fit the model of degree 15, the result came out to be very different for each subset. The model had a high variance. We will see in a moment that ridge regression reduces such variance. But first, we must reproduce the experiment we did in Module 3.

6. For this section, please download the provided csv files for each subset and load them with the given list of types:

```
# dtype_dict same as above
set_1 = pd.read_csv('wk3_kc_house_set_1_data.csv', dtype=dtype_dict)
set_2 = pd.read_csv('wk3_kc_house_set_2_data.csv', dtype=dtype_dict)
set_3 = pd.read_csv('wk3_kc_house_set_3_data.csv', dtype=dtype_dict)
set_4 = pd.read_csv('wk3_kc_house_set_4_data.csv', dtype=dtype_dict)
```

7. Just as we did in Module 3 (Polynomial Regression), fit a 15th degree polynomial on each of the 4 sets, plot the results and view the weights for the four models. This time, set

```
l2_small_penalty=1e-9
```

and use this value for the L2 penalty. Make sure to add "alpha=l2_small_penalty" and "normalize=True" to the parameter list of linear_model.Ridge.

The four curves should differ from one another a lot, as should the coefficients you learned.

8. *Quiz Question: For the models learned in each of these training sets, what are the smallest and largest values you learned for the coefficient of feature power_1?* (For the purpose of answering this question, negative numbers are considered "smaller" than positive numbers. So -5 is smaller than -3, and -3 is smaller than 5 and so forth.)

Ridge regression comes to rescue

9. Generally, whenever we see weights change so much in response to change in data, we believe the variance of our estimate to be large. Ridge regression aims to address this issue by penalizing "large" weights. (The weights looked quite small, but they are not that small because 'sqft_living' input is in the order of thousands.)

10. Fit a 15th-order polynomial model on set_1, set_2, set_3, and set_4, this time with a large L2 penalty. Make sure to add "alpha=l2_large_penalty" and "normalize=True" to the parameter list, where the value of l2_large_penalty is given by

```
l2_large_penalty=1.23e2
```

These curves should vary a lot less, now that you introduced regularization.

11. *QUIZ QUESTION: For the models learned with regularization in each of these training sets, what are the smallest and largest values you learned for the coefficient of feature power_1?* (For the purpose of answering this question, negative numbers are considered "smaller" than positive numbers. So -5 is smaller than -3, and -3 is smaller than 5 and so forth.)

Selecting an L2 penalty via cross-validation

3/20/2016

12. Just like the polynomial degree, the L2 penalty is a "magic" parameter we need to select. We could use the validation set approach as we did in the last module, but that approach has a major disadvantage: it leaves fewer observations available for training. **Cross-validation** seeks to overcome this issue by using all of the training set in a smart way.

We will implement a kind of cross-validation called k-fold cross-validation. The method gets its name because it involves dividing the training set into k segments of roughtly equal size. Similar to the validation set method, we measure the validation error with one of the segments designated as the validation set. The major difference is that we repeat the process k times as follows:

- Set aside segment 0 as the validation set, and fit a model on rest of data, and evalutate it on this validation set

- Set aside segment 1 as the validation set, and fit a model on rest of data, and evalutate it on this validation set

- ...

- Set aside segment k-1 as the validation set, and fit a model on rest of data, and evalutate it on this validation set

After this process, we compute the average of the k validation errors, and use it as an estimate of the generalization error. Notice that all observations are used for both training and validation, as we iterate over segments of data.

13. To estimate the generalization error well, it is crucial to shuffle the training data before dividing them into segments. We reserve 10% of the data as the test set and randomly shuffle the remainder. Le'ts call the shuffled data 'train_valid_shuffled'.

For the purpose of this assignment, let us download the csv file containing pre-shuffled rows of training and validation sets combined: wk3_kc_house_train_valid_shuffled.csv (https://eventing.coursera.org/api/redirectStrict/CPIyPMxDgyjPIgiQmCTIKUzEHRo0lvq7KLi0 Hx_5lX_L37lM0vozl12yXyizclKy3Sa3_WcvvAj-9ceFVPol-Q.Sp8BNsMsElJ-YoKsSzfs3w.2QaS2pdnQweWe9JJ8gflNzqNZ3GMFSYmh1d7m8F_WHwDRwlp7hbWr5Btn3om DsAN6SK39CRTAIo3RKx34npXeu9Okph4ScTjoFS7iKmefZV-eJSZhpuv0Eo0orZWpJgEH2e43Vy-I9Wn3HNDEZxcedPEVchsDV2XhX290eMULnVjcQtBpSc6VlRRnhQa2JzLuRcQn7TmnkgNRj5Y7 uKP1aNtPvtiui3CU6K2Za4Zlk0ye_g9QDlmZwmx1HF8nR0MDVEI_lGBU9l31wchOaLMBP9rYp2 MgMi--xEtj-iav3AetLEityHtPg3-2EDPc7vhua4GfY3rgRzw_07B7llJV_dNNtlFPkC3qpmlDPdzCPzR28M2hsOMw2kaS3i2Jmx_mQ c8ANj4EQPbfrWdljZphcvUf671hvNDYjRfKPISzMMPusdwrdCP4YPKToCh9kO9UQbWQrqlFzs9 dF95q2GmqmDNBsV3QvBmcLq-E10icJo). In practice, you would shuffle the rows with a dynamically determined random seed.

```
train_valid_shuffled = pd.read_csv('wk3_kc_house_train_valid_shuffled.csv'
, dtype=dtype_dict)
test = pd.read_csv('wk3_kc_house_test_data.csv', dtype=dtype_dict)
```

14. **D**ivide the combined training and validation set into equal segments. Each segment should receive n/k elements, where n is the number of observations in the training set and k is the number of segments. Since the segment 0 starts at index 0 and contains n/k elements, it ends at index (n/k)-1. The segment 1 starts where the segment 0 left off, at index (n/k). With n/k elements, the segment 1 ends at index (n*2/k)-1. Continuing in this fashion, we deduce that the segment i starts at index (n*i/k) and ends at (n*(i+1)/k)-1.

With this pattern in mind, we write a short loop that prints the starting and ending indices of each segment, just to make sure you are getting the splits right.

```
n = len(train_valid_shuffled)
k = 10 # 10-fold cross-validation

for i in xrange(k):
    start = (n*i)/k
    end = (n*(i+1))/k-1
    print i, (start, end)
```

Let us familiarize ourselves with array slicing with Pandas. To extract a continuous slice from a DataFrame, use colon in square brackets. For instance, the following cell extracts rows 0 to 9 of train_valid_shuffled. Notice that the first index (0) is included in the slice but the last index (10) is omitted.

```
train_valid_shuffled[0:10] # select rows 0 to 9
```

If the observations are grouped into 10 segments, the segment i is given by

```
start = (n*i)/10
end = (n*(i+1))/10
train_valid_shuffled[start:end+1]
```

Meanwhile, to choose the remainder of the data that's not part of the segment i, we select two slices (0:start) and (end+1:n) and paste them together.

```
train_valid_shuffled[0:start].append(train_valid_shuffled[end+1:n])
```

15. Now we are ready to implement k-fold cross-validation. Write a function that computes k validation errors by designating each of the k segments as the validation set. It accepts as parameters (i) k, (ii) l2_penalty, (iii) dataframe containing input features (e.g. poly15_data) and (iv) column of output values (e.g. price). The function returns the average validation error using k segments as validation sets. We shall assume that the input dataframe does not contain the output column.

For each i in [0, 1, ... k-1]:

• Compute starting and ending indices of segment i and call 'start' and 'end'

- Form validation set by taking a slice (start:end+1) from the data.

- Form training set by appending slice (end+1:n) to the end of slice (0:start).

- Train a linear model using training set just formed, with a given l2_penalty

- Compute validation error (RSS) using validation set just formed

e.g. in Python:

```
def k_fold_cross_validation(k, l2_penalty, data, output):
    ...
    return [average validation error]
```

16. Once we have a function to compute the average validation error for a model, we can write a loop to find the model that minimizes the average validation error. Write a loop that does the following:

- We will again be aiming to fit a 15th-order polynomial model using the sqft_living input

- For each l2_penalty in [10^3, 10^3.5, 10^4, 10^4.5, ..., 10^9] (to get this in Python, you can use this Numpy function: np.logspace(3, 9, num=13).): Run 10-fold cross-validation with l2_penalty.

- Report which L2 penalty produced the lowest average validation error.

Note: since the degree of the polynomial is now fixed to 15, to make things faster, you should generate polynomial features in advance and re-use them throughout the loop. Make sure to use train_valid_shuffled when generating polynomial features!

17. Quiz Question: What is the best value for the L2 penalty according to 10-fold validation?

18. Once you found the best value for the L2 penalty using cross-validation, it is important to retrain a final model on all of the training data using this value of l2_penalty. This way, your final model will be trained on the entire dataset.

19. Quiz Question: Using the best L2 penalty found above, train a model using all training data. What is the RSS on the TEST data of the model you learn with this L2 penalty?