EE 511 HW #1

P1. MLE

① $p(\underline{E}, \underline{H}; \lambda, p) = \prod_{i=1}^{n} \left(\frac{\lambda^{E_i}}{E_i!} e^{-\lambda} \cdot p^{H_i} (1-p)^{E_i - H_i} \right)$

$$\ln p(\underline{E}, \underline{H}; \lambda, p) = \sum_{i=1}^{n} \ln \left[\frac{\lambda^{E_i}}{(E_i)!} e^{-\lambda} \cdot p^{H_i} (1-p)^{E_i - H_i} \right]$$

$$= \sum_{i=1}^{n} \left[E_i \ln\lambda + (-\lambda) - \ln \frac{1}{(E_i)!} + H_i \ln p + (E_i - H_i) \ln(1-p) \right]$$

② $$\frac{\partial \ln p(\underline{E}, \underline{H}; \lambda, p)}{\partial \lambda} = \sum_{i=1}^{n} \left[\frac{E_i}{\lambda} - 1 \right] = 0 \Rightarrow \frac{\sum E_i}{\lambda} = n \Rightarrow \hat{\lambda} = \frac{\sum E_i}{n}$$

$$\frac{\partial \ln p(\underline{E}, \underline{H}; \lambda, p)}{\partial p} = \sum_{i=1}^{n} \left[H_i \cdot \frac{1}{p} - (E_i - H_i) \frac{1}{1-p} \right] = 0 \Rightarrow \hat{p} = \frac{\sum_{i=1}^{n} H_i}{\sum_{i=1}^{n} E_i}$$

③ $\sum E_i = 8 + 9 + 6 + 4 + 1 + 5 + 2 + 12 + 9 + 7 = 63$

$\sum H_i = 5 + 6 + 4 + 3 + 0 + 5 + 2 + 9 + 8 + 6 = 48$

$\therefore \hat{\lambda}_{MLE} = \frac{63}{10} = 6.3 \qquad \hat{p}_{MLE} = \frac{48}{63} = \frac{16}{21}$

P2: $\sum_{i=1}^{n}(y_i - X\hat{\omega})^2 + \lambda\|w\|_1 \approx \sum_{i=1}^{n}(y_i - X\hat{\omega})^2$

2.1 (a) Small.

With λ approaching to 0, we ignore the regularization part.
The error on the training set should be small because of optimizing MLE

(b) Large.

If optimizing training set only by ordinary least square method, the error on the testing set should be large

(c) Large.

With a small λ, it's nonregularized.

(d) Large.

λ is too small to force relatively small weights to real zero.

Overestimate of λ

(a) Large

With such a large λ, too many weights become zero. The error on the training set became large.

(b) Large

Overfitting means that the model can't generalize to test set.

(c) Small.

λ is so large that most weights are forced to be zero.

(d) Small

Too many weights are forced to be real zero for overestimated λ.

2.2

1. $\lambda \|w_1\| = \lambda \sum_{i=1}^{d} |w_i|$

$$\frac{\partial E_L}{\partial w_i} = \begin{cases} \lambda & w_i > 0 \\ -\lambda & w_i < 0 \end{cases}$$

2. $\lambda \|w\|_2^2 = \lambda \sum_{i=1}^{d} (w_i)^2$

$$\frac{\partial E_R}{\partial w_i} = 2\lambda w_i$$

3. If $w_i \neq \pm \frac{1}{2}$, their behaviors differ.

When λ is very large:

For Lasso, such a large λ and linear penalty pushes more weights to zero. It allows for a type of feature selection.

For Ridge, relatively small weights may trade off such a large λ.

Part II. Programming

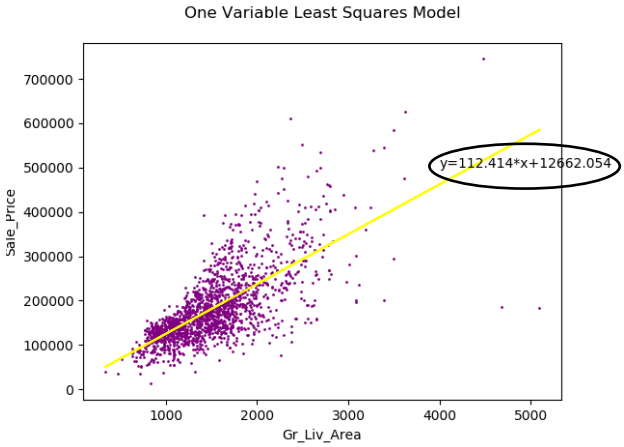
The first part is data preprocessing:

After replacing missing values, **line 63** starts to separate data into training, validation and test set based on requirements.

The second part is one variable least squares model:

Line 87 starts to build model based on equation one variable OLS model, which is the function named as best fit slope and intercept. Then function names as rmse is applied to calculate the Root Mean Squared Error(RMSE). The equation for the line is:

y = 112.414*x+12662.054



The third part is all features least squares model:

Line 119 starts to build all features least squares model based on imported package, which is from sklearn.linear_model to import LinearRegression.

Before training the model, we are required to transform the categorical features to one-hot encoding so that they can be used in the model. **Line 123**, we employ panda get_dummies to transform categorical features. We should pay attention to this case that we only need k-1 dummy variables to indicate a feature with k variables. Hence, drop_first=True is applied. (finally, there are 330 feature columns)

Then we train all features least squares model with the training set, validate it with the validation set.

The forth part is Lasso regression model:

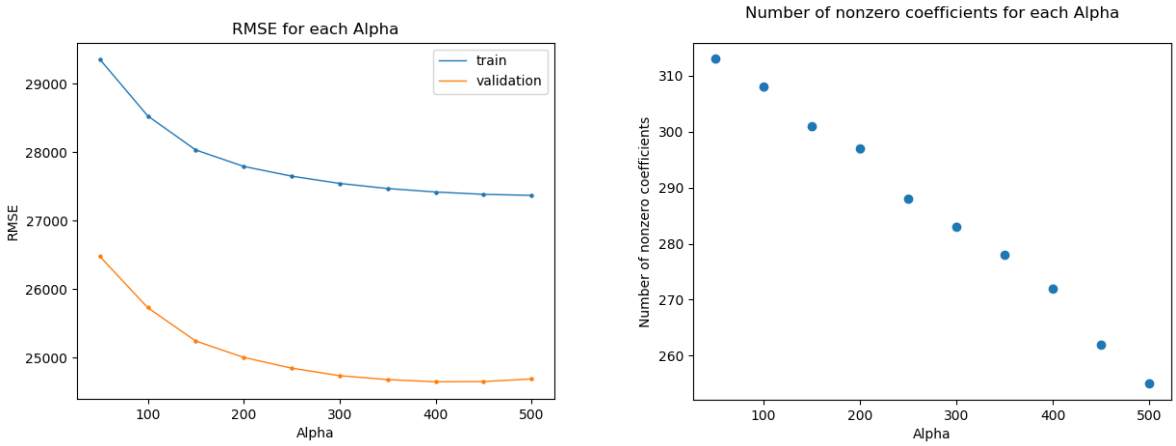
Line 151 starts to build Lasso regression model based on imported package, which is from sklearn.linear_model to import Lasso.

Before training the model, we are required to normalize training, validation and test set by substracting training set mean and dividing by training set std. **Line 153** starts to normalize those data sets.

Line 164 starts to train Lasso regression model. To employ cross validation, we concatenate training and validation set, then separate them into 4 folders. For each Alpha, we train the model by randomly picking three of folders and validate it by the rest one. Then, we average the RMSE calculated by these 4 folders.

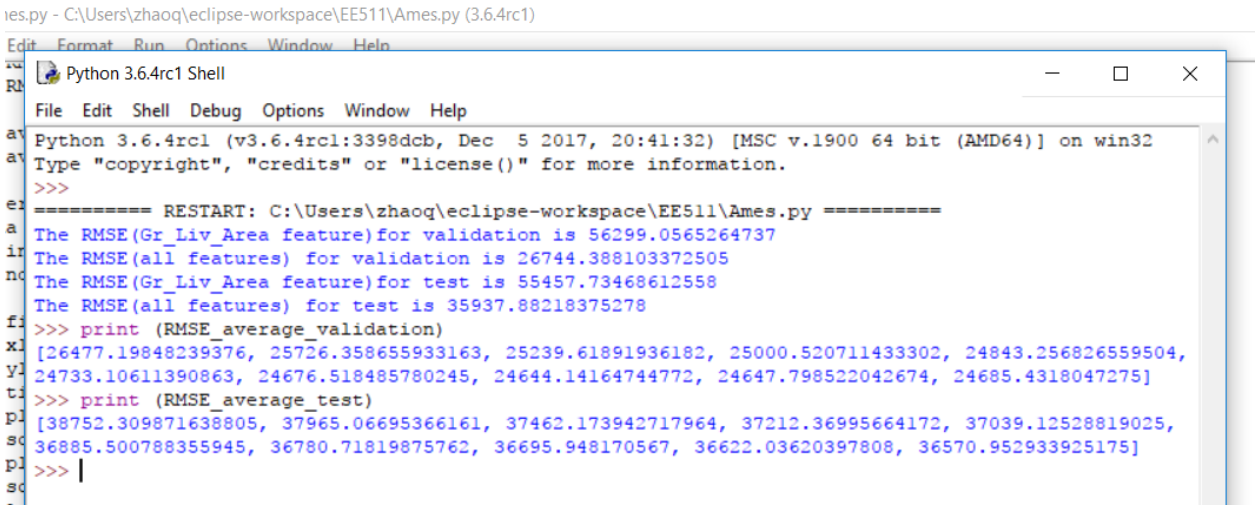
After training and validating, the RMSE for each alpha is listed. And we could find that the lowest error on the validation set appear at Alpha=400.

As graph shown, the RMSE for training data keeps decreasing, however, the RMSE for validation data simultaneously decrease before Alpha=400 and starts to increase after Alpha=400. This is the beginning of over-fitting. The phenomenon named as over-fitting is that the model exactly fits the training set but fails to generalize to validation and test sets.



The fifth part is using the test set:

Line 226 applies the single variable model, the all features least squares model and the regularized model to the test set. The RMSE for each condition for both the validation set and the test set are listed in the table.



Model\RMSE	Validation Set	Test Set
Single variable model	56299.056	55457.734
All features Least Squares Model	26744.388	35937.882
Lasso Regression Model (Alpha =50)	26477.198	38752.310
Lasso Regression Model (Alpha =100)	25726.359	37965.067
Lasso Regression Model (Alpha =150)	25239.619	37462.174
Lasso Regression Model (Alpha =200)	25000.521	37212.370
Lasso Regression Model (Alpha =250)	24843.257	37039.125
Lasso Regression Model (Alpha =300)	24733.106	36885.501
Lasso Regression Model (Alpha =350)	24676.518	36780.718
Lasso Regression Model (Alpha =400)	24644.142	36695.948
Lasso Regression Model (Alpha =450)	24647.799	36622.036
Lasso Regression Model (Alpha =500)	24685.432	36570.953

The RMSE for validation data decreases before Alpha=400 and starts to increase after Alpha=400. Alpha=400 is the beginning of over-fitting. The phenomenon named as over-fitting is that the model exactly fits the training set but fails to generalize to validation and test sets.