

Homework 2

Haomiao Han (hh696), Hyein Baek (hb437)
CS5785 Applied Machine Learning

September 29, 2019

1 Programming Exercises

- Programming Exercises, Question 1

(a) We used `pandas` to load and split the dataset.

(b) The dataset has 1460 rows and 81 columns, meaning that there are 1460 samples and 80 features. Listed below are 43 categorical features (which we define as features whose value contains strings instead of numbers):

```
'MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',  
'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',  
'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',  
'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',  
'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',  
'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',  
'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish',  
'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence',  
'MiscFeature', 'SaleType', 'SaleCondition'
```

(c) We first preprocessed the dataset by filling NaNs and empty cells either with `bfill` and `ffill` (for categorical features) or with the mean value of the column (for numerical features). We then selected the following 7 features:

```
'YrSold', 'GarageArea', 'LotArea', 'GrLivArea', 'OverallQual', 'OverallCond',  
'TotRmsAbvGrd', 'SalePrice'
```

The plotted graph is attached below:

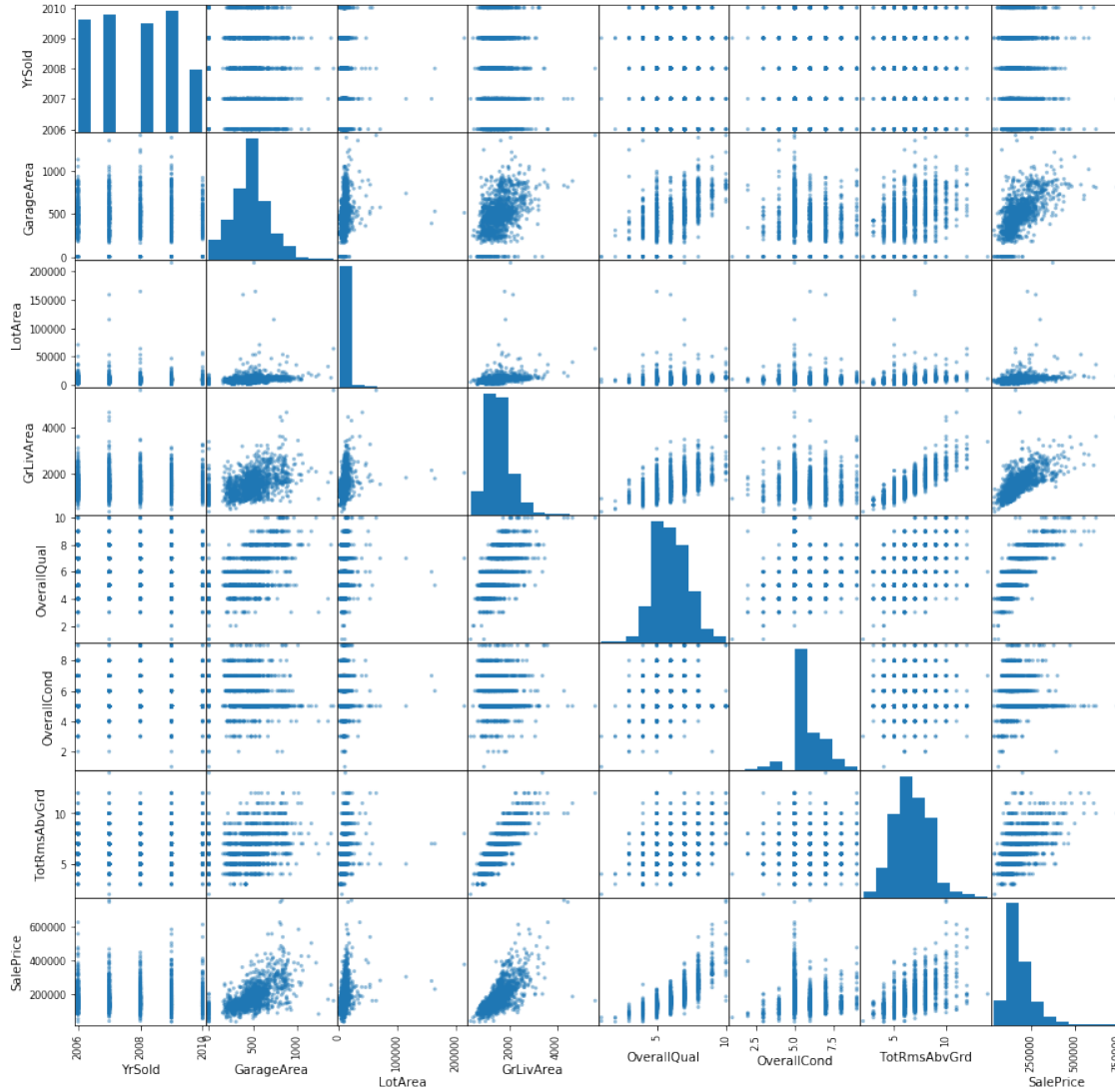


Figure 1: Scatterplot matrix

According to the graph, we can see that features such as **GarageArea**, **GrLivArea** and **OverallQual** are highly correlated with **SalePrice**, and therefore seem to be more important, whereas features such as **YrSold**, **LotArea**, **OverallCond** and **TotRmsAbvGrd** do not demonstrate strong correlations with our target.

(d) We first preprocessed the data by encoding the categorical features, using `sklearn`'s `LabelEncoder`, and scaling the data using `sklearn`'s `StandardScaler`. We then ran OLS on all features using `statsmodels`. The results showed that the following are the features whose coefficient has a 95% confidence interval that contains 0:

```
'Id', 'Street', 'Alley', 'LandContour', 'Utilities', 'LandSlope',
'MasVnrType', 'BsmtFinType2', 'BsmtFinSF2',
'LowQualFinSF', 'BsmtHalfBath', '3SsnPorch', 'Fence', 'MiscFeature',
'MiscVal', 'MoSold', 'YrSold'
```

(e) We first preprocessed the data by performing one hot encoding, using `pandas`'s `get_dummies`. We then used different machine learning models to train the data and to predict on the validation set. Below are the results:

	MSE (Mean Squared Error)
OLS	$5.58 * 10^{28}$
kNN ($k = 2$)	$1.2 * 10^{10}$
Ridge ($\lambda = 100$)	$7.28 * 10^9$
LASSO ($\lambda = 169.16$)	$6.38 * 10^9$
Forward Stepwise	$1.009 * 10^{10}$
Backward Stepwise	$1.08 * 10^{30}$

From the table, we can see that kNN, Ridge, LASSO and Forward Stepwise produced significantly better results than OLS and Backward Stepwise. Upon inspection of the number of features used by OLS and Backward Stepwise, we found that these two models have used more features than other models (see part (g) for details). A more complicated model with a large number of features are prone to overfitting, which could explain the bad performance of OLS and Backward Stepwise in this case.

(f) We first added the quadratic features using `sklearn`'s `PolynomialFeatures`. We then used different machine learning models to train the data and to predict on the validation set. Below are the results:

	MSE (Mean Squared Error)
OLS	$8.29 * 10^9$
kNN ($k = 2$)	$1.2 * 10^{10}$
Ridge ($\lambda = 1000$)	$7.71 * 10^9$
LASSO ($\lambda = N/A$)	N/A
Backward Stepwise	N/A
Forward Stepwise	N/A

We have seen an improvement with OLS's performance, which could mean that performing data augmentation can improve prediction accuracy in general. The performances for kNN and Ridge did not change much. We were unable to get the results for LASSO as well as Stepwise Linear Regressions, since they would have taken an extremely long period of time to run.

(g) The following 96 features are retained by LASSO:

```
'Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea',
'BsmtFinSF1', 'BsmtFinSF2', 'TotalBsmtSF', '2ndFlrSF',
'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageCars',
```

'WoodDeckSF', '3SsnPorch', 'ScreenPorch', 'PoolArea',
 'MiscVal', 'YrSold', 'MSZoning_RL', 'LotShape_IR2',
 'LandContour_Bnk', 'LandContour_HLS', 'LotConfig_CulDSac',
 'LandSlope_Mod', 'Neighborhood_BrkSide', 'Neighborhood_Crawfor',
 'Neighborhood_Edwards', 'Neighborhood_Gilbert', 'Neighborhood_Mitchel',
 'Neighborhood_NAMES', 'Neighborhood_NWAmes', 'Neighborhood_NoRidge',
 'Neighborhood_NridgHt', 'Neighborhood_OldTown', 'Neighborhood_Somerst',
 'Neighborhood_StoneBr', 'Condition1_Norm', 'Condition2_PosN',
 'BldgType_1Fam', 'BldgType_2fmCon', 'HouseStyle_2Story',
 'RoofMatl_ClyTile', 'RoofMatl_WdShngl', 'Exterior1st_BrkFace',
 'Exterior1st_HdBoard', 'Exterior2nd_ImStucc', 'Exterior2nd_Plywood',
 'Exterior2nd_VinylSd', 'MasVnrType_BrkFace', 'MasVnrType_None',
 'ExterQual_Ex', 'ExterQual_TA', 'ExterCond_Gd', 'Foundation_BrkTil',
 'Foundation_PConc', 'BsmtQual_Ex', 'BsmtExposure_Gd', 'BsmtExposure_No',
 'BsmtFinType1_GLQ', 'BsmtFinType1_Unf', 'BsmtFinType2_LwQ',
 'HeatingQC_Ex', 'HeatingQC_Gd', 'KitchenQual_Ex', 'KitchenQual_TA',
 'Functional_Typ', 'FireplaceQu_Ex', 'FireplaceQu_TA',
 'GarageType_BuiltIn', 'GarageType_Detchd', 'GarageFinish_Fin',
 'GarageFinish_RFn', 'GarageQual_Fa', 'PoolQC_Ex', 'Fence_MnPrv',
 'Fence_MnWw', 'MiscFeature_Othr', 'MiscFeature_TenC', 'SaleType_COD',
 'SaleType_New', 'SaleType_WD', 'SaleCondition_Abnorml',
 'SaleCondition_Normal'

The following 36 features are retained by Forward Stepwise:

'OverallQual', 'GrLivArea', 'BsmtQual_Ex', 'RoofMatl_ClyTile',
 'BsmtFinSF1', 'TotalBsmtSF', 'Condition2_PosN', 'YearBuilt',
 'ExterQual_Ex', 'BldgType_1Fam', 'BedroomAbvGr', 'OverallCond',
 'LotArea', 'Neighborhood_NoRidge', 'Neighborhood_NridgHt',
 'Neighborhood_StoneBr', 'Neighborhood_Crawfor', 'BsmtExposure_Gd',
 'SaleType_New', 'PoolArea', 'Neighborhood_Somerst', 'GarageCars',
 'KitchenQual_Ex', 'RoofMatl_WdShngl', 'Exterior1st_BrkFace',
 'SaleCondition_Normal', 'Condition1_Norm', 'Functional_Typ',
 'LowQualFinSF', 'Foundation_PConc', 'Exterior2nd_ImStucc',
 'BsmtExposure_No', 'Neighborhood_BrkSide', 'MSZoning_C (all)',
 'LotConfig_CulDSac', 'Neighborhood_Mitchel'

The following 160 features are retained by Backward Stepwise:

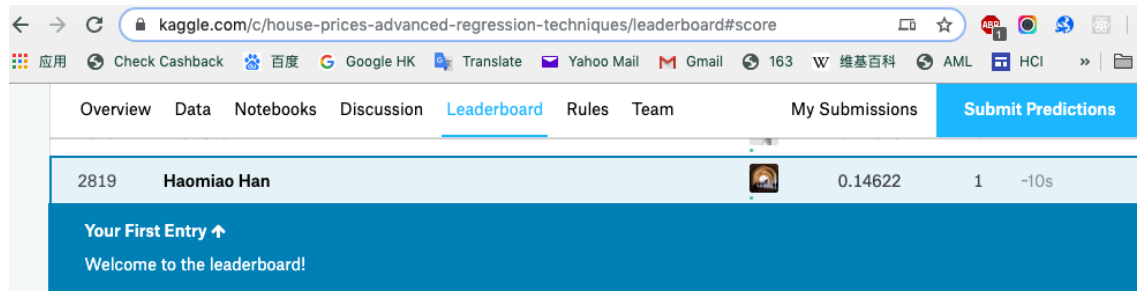
'Id', 'MSSubClass', 'LotFrontage', 'YearRemodAdd', 'MasVnrArea',
 'BsmtFinSF2', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageArea',
 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
 'ScreenPorch', 'MiscVal', 'MoSold', 'YrSold', 'Street_Grvl',
 'LotShape_IR1', 'LotShape_IR3', 'LotShape_Reg', 'LandContour_Bnk',

'LandContour_Low', 'LotConfig_Corner', 'LotConfig_FR2', 'LotConfig_FR3',
 'LotConfig_Inside', 'LandSlope_Sev', 'Neighborhood_Blueste',
 'Neighborhood_NPkVill', 'Neighborhood_Somerst', 'Condition1_Artery',
 'Condition1_Feendr', 'Condition1_PosA', 'Condition1_PosN',
 'Condition1_RRAn', 'Condition1_RRNe', 'Condition1_RRNn',
 'BldgType_Duplex', 'HouseStyle_1.5Fin', 'HouseStyle_1.5Unf',
 'HouseStyle_2.5Fin', 'HouseStyle_2Story', 'HouseStyle_SFoyer',
 'HouseStyle_SLv1', 'RoofStyle_Flat', 'RoofStyle_Gable',
 'RoofStyle_Gambrel', 'RoofStyle_Hip', 'RoofStyle_Mansard',
 'Exterior1st_AsphShn', 'Exterior1st_BrkComm', 'Exterior1st_CBlock',
 'Exterior1st_ImStucc', 'Exterior2nd_AsbShng', 'Exterior2nd_AsphShn',
 'Exterior2nd_BrkFace', 'Exterior2nd_CmentBd', 'Exterior2nd_HdBoard',
 'Exterior2nd_MetalSd', 'Exterior2nd_Other', 'Exterior2nd_Stone',
 'Exterior2nd_Stucco', 'Exterior2nd_VinylSd', 'Exterior2nd_Wd Sdng',
 'Exterior2nd_Wd Shng', 'MasVnrType_BrkCmn', 'MasVnrType_BrkFace',
 'ExterQual_Fa', 'ExterQual_Gd', 'ExterQual_TA', 'ExterCond_Ex',
 'ExterCond_Fa', 'ExterCond_Gd', 'ExterCond_Po', 'ExterCond_TA',
 'Foundation_BrkTil', 'Foundation_CBlock', 'Foundation_Slab',
 'Foundation_Stone', 'BsmtCond_Fa', 'BsmtCond_Gd', 'BsmtCond_TA',
 'BsmtExposure_Mn', 'BsmtExposure_No', 'BsmtFinType1_ALQ',
 'BsmtFinType1_BLQ', 'BsmtFinType1_GLQ', 'BsmtFinType1_LwQ',
 'BsmtFinType1_Rec', 'BsmtFinType1_Unf', 'BsmtFinType2_BLQ',
 'BsmtFinType2_GLQ', 'BsmtFinType2_LwQ', 'BsmtFinType2_Rec',
 'BsmtFinType2_Unf', 'Heating_Floor', 'Heating_GasA', 'Heating_GasW',
 'Heating_Grav', 'Heating_OthW', 'Heating_Wall', 'HeatingQC_Ex',
 'HeatingQC_Fa', 'HeatingQC_Gd', 'HeatingQC_Po', 'HeatingQC_TA',
 'Electrical_FuseA', 'Electrical_FuseF', 'Electrical_FuseP',
 'Electrical_SBrkr', 'KitchenQual_Fa', 'KitchenQual_Gd',
 'KitchenQual_TA', 'Functional_Maj1', 'Functional_Maj2',
 'Functional_Min1', 'Functional_Min2', 'Functional_Mod',
 'Functional_Sev', 'FireplaceQu_Ex', 'FireplaceQu_Fa', 'FireplaceQu_Gd',
 'FireplaceQu_Po', 'FireplaceQu_TA', 'GarageType_Attchd',
 'GarageType_Basment', 'GarageType_BuiltIn', 'GarageType_CarPort',
 'GarageType_Detchd', 'GarageFinish_Unf', 'GarageQual_Ex',
 'GarageQual_Fa', 'GarageQual_Gd', 'GarageQual_Po', 'GarageQual_TA',
 'GarageCond_Ex', 'GarageCond_Fa', 'GarageCond_Gd', 'GarageCond_Po',
 'GarageCond_TA', 'PavedDrive_N', 'PavedDrive_P', 'PavedDrive_Y',
 'Fence_GdPrv', 'Fence_GdWo', 'Fence_MnWw', 'MiscFeature_Gar2',
 'MiscFeature_Shed', 'MiscFeature_TenC', 'SaleType_ConLI',
 'SaleType_ConLw', 'SaleType_Oth', 'SaleCondition_Abnorml',
 'SaleCondition_Alloca'

The first thing that we have noticed is that Backward Stepwise retained more features than LASSO or Forward Stepwise. In addition, as we can see, most of the features retained by Forward Stepwise matched our intuition about the importance of the fea-

tures. However, some features retained by LASSO and Backward Stepwise do not seem to be that important: for example, our scatterplot matrices and our results for OLS run on single features have shown that features like `MoSold` and `YrSold` may not be that important, yet they were still retained by both LASSO and Backward Stepwise.

(h) Based on the results in part (e), we chose LASSO as our model. We used all of the training set to train our model and predicted on the test set. According to Kaggle, our predictions have an accuracy rate of 0.146 (based on RMSLE, Root Mean Squared Log Error). Below is the proof:



Rank	Username	Score	Rank	Time
2819	Haomiao Han	0.14622	1	~10s

Your First Entry ↑
Welcome to the leaderboard!

The RMSLE on our validation set was 0.117, which was lower. This might be due to overfitting on the training set.

- Question 2

(a) We read the input files and stored the data into lists. We then checked the number of reviews with positive sentiment and reviews with negative sentiment within each dataset (Yelp, IMDB, Amazon). The result showed that there are 500 positive reviews and 500 negative reviews for each dataset, which indicated an even distribution.

(b) We preprocessed the data by removing punctuations, stemming the words using `nlTK`'s `SnowballStemmer` (which automatically converts the words into lower case) and removing stop words. We removed punctuations and stop words because we believed that they are not correlated to the sentiment of sentences. We stemmed the words because without stemming, words with essentially the same meaning (such as "happy" and "happily") will be considered as totally different words and could thus negatively impact accuracy of the model.

(c) We split the data per the instructions.

(d) We converted the sentences into vectors per the instructions. The reason for not using the test set at this time is the same reason for not using the test set when building models based on other machine learning algorithms (such as `kNN` or `LogReg`) - it will overfit the test set and the accuracy will be much lower if we use it to predict on any new test data. Below are the two example reviews in the training set:

Review 1 (index 2288)

Text (processed): veri bad experi

Sentiment: 0 (negative)

Vectors: [0, 0, ..., 1, 0, ..., 1, 0, ..., 1, 0, ..., 0, 0]
(Refer to the .ipynb file to see the full length of the vectors)

Review 2 (index 123)

Text (processed): worth everi penni

Sentiment: 1 (positive)

Vectors: [0, 0, ..., 1, 0, ..., 1, 0, ..., 1, 0, ..., 0, 0]

(Refer to the .ipynb file to see the full length of the vectors)

(e) We used Log-normalization since this will best decrease the difference in values between the 0 items in the feature vectors and the non-0 items (for example, $\log(x+1) = 0.693$ when $x = 1$ and $\log(x+1) = 1.099$ when $x = 2$).

(f) We first built our Naive Bayes model by looping through all positive reviews and counting the occurrence of each unique word in the training set (which we have identified in part (d)). We then add the word - occurrence as a key - value pair into a dictionary. For example, if the word "great" appeared in 100 reviews, we store {"great":100} into our dictionary. We then repeat the same process for all negative reviews.

After building the model, we then used it to predict on the test set. We first assign a "positive score" and a "negative score" for each review in the test set. The positive score is based on the Bayes Theorem and is calculated as follows:

$$PositiveScore = \prod_{i=1}^n \frac{PosOccurrence_i + 1}{\#PositiveReviews + \#UniqueWordsinTrainingSet + 1} \quad (1)$$

where i is the index of a word in the review, n is the length of (number of words in) the review, $PosOccurrence_i$ is the occurrence of word at index i in all positive reviews in the training set (which can be accessed through the positive words dictionary defined above), $\#PositiveReviews$ is the number of all positive reviews (which is equal to 1,200), $\#UniqueWordsinTrainingSet$ is the number of all unique words that have appeared in the training set (which is equal to 3,542).

Similarly, the negative score is calculated as follows:

$$NegativeScore = \prod_{i=1}^n \frac{NegOccurrence_i + 1}{\#NegativeReviews + \#UniqueWordsinTrainingSet + 1} \quad (2)$$

where $NegOccurrence_i$ is the occurrence of word at index i in all negative reviews in the training set (which can be accessed through the negative words dictionary defined above), and $\#NegativeReviews$ is the number of all positive reviews (which is equal to 1,200).

The reason why we did not include $P(PositiveReviews)$ is that the reviews are equally distributed in the training set, meaning that $P(PositiveReviews) = P(NegativeReviews) = 0.5$.

In addition, we also used Laplace smoothing. If a word in the test set does not appear in the training set, the positive score and the negative score will both be 0 for that word, making the positive score and the negative score to be both 0. In that case, the model is only able to generate a random prediction. However, with Laplace smoothing, we added 1 to the numerator and $\#UniqueWordsinTrainingSet + 1$ to the denominator, thereby giving all unseen/OOV words a small probability and improving the performance of the model.

Our model achieved an accuracy of 0.8183 on the test set. Below is the confusion matrix:

	Predicted 0	Predicted 1
True 0	273	27
True 1	82	218

(g) We used `sklearn`'s `LogisticRegressionCV` to build a model with cross validation and to predict on the test set. The accuracy rate is 0.8383 for LogReg using L1 regularization and 0.8417 for L2 regularization. Upon inspecting the coefficients of the Logistic Regression, we identified the below list of top 20 words that played the most important role in deciding whether a review is positive:

'great', 'love', 'delici', 'excel', 'amaz', 'awesom', 'nice',
 'beauti', 'good', 'fantast', 'happier', 'perfect', 'comfort',
 'best', 'brilliant', 'easi', 'cool', 'happi', 'funni', 'fun'

And below is the list of top 20 words that played the most important role in deciding whether a review is negative:

'poor', 'bad', 'not', 'worst', 'suck', 'terribl', 'slow', 'aw',
 'fail', 'rude', 'bland', 'avoid', 'hate', 'stupid', 'disappoint',
 'horribl', 'start', 'wast', 'piec', 'averag'

(h) We first created a set of all unique 2-grams in the training set, and then performed text vectorization and used log-normalization to normalize the vectors. We then built Naive Bayes and LogReg models to predict on the test set. The accuracy is shown below:

	Accuracy
Naive Bayes	0.6817
LogReg (L2)	0.67
LogReg (L1)	0.665

Below is the list of top 20 2-grams that played the most important role in deciding whether a review is positive:

'i love', 'veri good', 'work great', 'high recommend', 'i like',
 'one best', 'i think', 'better than', 'great phone', 'great product',
 'great food', 'veri nice', 'realli good', 'i enjoy', 'so far',
 'sound qualiti', 'i believ', 'would recommend', 'i realli',
 'easi use'

And below is the list of top 20 2-grams that played the most important role in deciding whether a review is negative:

'don t', 'didn t', 'would not', 'do not', 'not good', 'wast time',
 'veri disappoint', 'there no', 'not work', 'did not', 'i m',
 'so bad', 'will not', 'doesn t', 'not recommend', 'wasn t',
 'go back', 'veri bad', 'custom servic', 'i hate'

(i) Based on our results, the best performing model is LogReg with L2 regularization based on a Bag of Words model, with an accuracy rate of 0.8417. The models using Bag of Words seemed to outperform models using 2-grams. In our opinion, one of the possible reasons why BoW is better than 2-grams on this dataset is that the number of datapoints available is limited and the length of each review is short. Thus, if we vectorize the text following a 2-gram model, each text vector will have a very high number of dimensions with mostly zeros. The sparsity and the high dimensionality of the input vector matrix may have caused a decrease in accuracy.

As we can see, despite some irregularities (such as "start" or "i m" appearing in the top 20 words for negative sentiment), those top 20 words and 2-grams we identified in part (h) match our intuition in positive and negative sentiments. We can also see that the language used in online reviews tends to be more causal and less formal.

2 Written Exercises

- Question 1

Based on the augmented dataset, the OLS can be denoted as the solution to following (where the left side of the plus sign calculates the residual of the original dataset and the right side of the plus sign calculates the residual of the augmented data added to the original dataset):

$$\min_{\beta \in \mathbb{R}^{1+p}} \sum_{i=1}^n (Y_i - \beta^T X_i)^2 + \sum_{j=1}^p (Y_j - \beta^T X_j)^2 \quad (3)$$

Given that the values of all Y_j in the augmented data are 0, (1) becomes:

$$\min_{\beta \in \mathbb{R}^{1+p}} \sum_{i=1}^n (Y_i - \beta^T X_i)^2 + \sum_{j=1}^p (0 - \beta^T X_j)^2 \quad (4)$$

Given that the only non-zero element in each row in the augmented data is $\sqrt{\lambda}$, and that the index of each $\sqrt{\lambda}$ item in each row is j , (2) becomes:

$$\min_{\beta \in \mathbb{R}^{1+p}} \sum_{i=1}^n (Y_i - \beta^T X_i)^2 + \sum_{j=1}^p (0 - \beta_j \sqrt{\lambda})^2 \quad (5)$$

Which is:

$$\min_{\beta \in \mathbb{R}^{1+p}} \sum_{i=1}^n (Y_i - \beta^T X_i)^2 + \lambda \sum_{j=1}^p \beta_j^2 \quad (6)$$

As we can see, (4) is the same as the Ridge Regression formula.

- Question 2

(a)

$$\begin{aligned} & P(Y = \text{Pneumonia} | X_1 = \text{Fever}, X_2 = \text{No headache}) \\ &= \frac{P(X_1 = \text{Fever}, X_2 = \text{No headache} | Y = \text{Pneumonia}) * P(Y = \text{Pneumonia})}{P(X_1 = \text{Fever}, X_2 = \text{No headache})} \\ &= \frac{0 * \frac{10}{100}}{\frac{9}{100}} = 0 \end{aligned} \quad (7)$$

$$\begin{aligned} & P(Y = \text{Flu} | X_1 = \text{Fever}, X_2 = \text{No headache}) \\ &= \frac{P(X_1 = \text{Fever}, X_2 = \text{No headache} | Y = \text{Flu}) * P(Y = \text{Flu})}{P(X_1 = \text{Fever}, X_2 = \text{No headache})} \\ &= \frac{\frac{6}{20} * \frac{20}{100}}{\frac{9}{100}} = \frac{2}{3} \end{aligned} \quad (8)$$

$$\begin{aligned} & P(Y = \text{Healthy} | X_1 = \text{Fever}, X_2 = \text{No headache}) \\ &= \frac{P(X_1 = \text{Fever}, X_2 = \text{No headache} | Y = \text{Healthy}) * P(Y = \text{Healthy})}{P(X_1 = \text{Fever}, X_2 = \text{No headache})} \\ &= \frac{\frac{3}{70} * \frac{70}{100}}{\frac{9}{100}} = \frac{1}{3} \end{aligned} \quad (9)$$

(b)

$$\begin{aligned}
& P(Y = \textit{Pneumonia} | X_1 = \textit{Fever}, X_2 = \textit{No headache}) \\
&= \frac{P(X_1 = \textit{Fever}, X_2 = \textit{No headache} | Y = \textit{Pneumonia}) * P(Y = \textit{Pneumonia})}{P(X_1 = \textit{Fever}, X_2 = \textit{No headache})} \\
&\propto P(X_1 = \textit{Fever} | Y = \textit{Pneu}) * P(X_2 = \textit{No headache} | Y = \textit{Pneu}) * P(Y = \textit{Pneu}) \\
&= 0.5 * 0.1 * 0.5 = 0.005
\end{aligned} \tag{10}$$

$$\begin{aligned}
& P(Y = \textit{Flu} | X_1 = \textit{Fever}, X_2 = \textit{No headache}) \\
&= \frac{P(X_1 = \textit{Fever}, X_2 = \textit{No headache} | Y = \textit{Flu}) * P(Y = \textit{Flu})}{P(X_1 = \textit{Fever}, X_2 = \textit{No headache})} \\
&\propto P(X_1 = \textit{Fever} | Y = \textit{Flu}) * P(X_2 = \textit{No headache} | Y = \textit{Flu}) * P(Y = \textit{Flu}) \\
&= 0.75 * 0.4 * 0.2 = 0.06
\end{aligned} \tag{11}$$

$$\begin{aligned}
& P(Y = \textit{Healthy} | X_1 = \textit{Fever}, X_2 = \textit{No headache}) \\
&= \frac{P(X_1 = \textit{Fever}, X_2 = \textit{No headache} | Y = \textit{Healthy}) * P(Y = \textit{Healthy})}{P(X_1 = \textit{Fever}, X_2 = \textit{No headache})} \\
&\propto P(X_1 = \textit{Fever} | Y = \textit{Heal}) * P(X_2 = \textit{No headache} | Y = \textit{Heal}) * P(Y = \textit{Heal}) \\
&\approx 0.0714 * 0.8714 * 0.7 \approx 0.04355
\end{aligned} \tag{12}$$

Converting all three numbers to probabilities with a sum of 1:

$$\begin{aligned}
& P(Y = \textit{Pneumonia} | X_1 = \textit{Fever}, X_2 = \textit{No headache}) = \\
& \frac{0.005}{0.005 + 0.06 + 0.04355} \approx 0.046 \tag{13}
\end{aligned}$$

$$\begin{aligned}
& P(Y = \textit{Flu} | X_1 = \textit{Fever}, X_2 = \textit{No headache}) = \\
& \frac{0.06}{0.005 + 0.06 + 0.04355} \approx 0.553 \tag{14}
\end{aligned}$$

$$\begin{aligned}
& P(Y = \textit{Healthy} | X_1 = \textit{Fever}, X_2 = \textit{No headache}) = \\
& \frac{0.04355}{0.005 + 0.06 + 0.04355} \approx 0.401 \tag{15}
\end{aligned}$$

- Question 3

$$\begin{aligned}
 &P(Y = \text{yes} | \text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit} - \text{rating} = \text{fair}) \\
 &\quad \propto P(\text{age} \leq 30 | Y = \text{yes}) * P(\text{income} = \text{medium} | Y = \text{yes}) * \\
 &P(\text{student} = \text{yes} | Y = \text{yes}) * P(\text{credit} - \text{rating} = \text{fair} | Y = \text{yes}) * P(Y = \text{yes}) \\
 &= \frac{2}{9} * \frac{4}{9} * \frac{6}{9} * \frac{6}{9} * \frac{9}{14} \approx 0.0282 \quad (16)
 \end{aligned}$$

$$\begin{aligned}
 &P(Y = \text{no} | \text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit} - \text{rating} = \text{fair}) \\
 &\quad \propto P(\text{age} \leq 30 | Y = \text{no}) * P(\text{income} = \text{medium} | Y = \text{no}) * \\
 &P(\text{student} = \text{yes} | Y = \text{no}) * P(\text{credit} - \text{rating} = \text{fair} | Y = \text{no}) * P(Y = \text{no}) \\
 &= \frac{3}{5} * \frac{2}{5} * \frac{1}{5} * \frac{2}{5} * \frac{5}{14} \approx 0.0069 \quad (17)
 \end{aligned}$$

Thus, the predicted class will be **yes**.

3 References

- The list of stop words used in Programming Question 2 comes from a previous class one of us has taken. The list has been included in the homework submission.
- The code for Forward and Backward Stepwise Regression comes from an external implementation. The source of the original code has been included in the .ipynb file.
- (OLS with Statsmodels) <https://blog.datarobot.com/ordinary-least-squares-in-python>
- (LabelEncoder issues) <https://stackoverflow.com/questions/52112414/valueerror-bad-input-shape-in-sklearn-python/52113532>
- (One hot encoding) <https://stackoverflow.com/questions/36631163/pandas-get-dummies-vs-sklearns-onehotencoder-what-are-the-pros-and-cons>
- (Regression accuracy) <https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0>
- (kNN CV) <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/#parameter-tuning-with-cross-validation>
- (Ridge CV and LASSO CV) <https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/#three>
- (Text preprocessing) <https://stackoverflow.com/questions/34860982/replace-the-punctuation-with-whitespace>

- (Naive Bayes) http://people.cs.georgetown.edu/nshneid/cosc272/f17/09_NB_classification.pdf
- (Naive Bayes) <https://courses.cs.washington.edu/courses/cse446/17wi/slides/naivebayes-bayesnets-annotated.pdf>