Applied Machine Learning HW3 (15%)

Due Sunday May 12 @ 11:59pm on Canvas

Instructions:

1. In this HW you will compete in an active Kaggle competition, Housing Price Prediction: `https://www.kaggle.com/c/house-prices-advanced-regression-techniques`
2. The objectives of this HW include: understanding and using existing tools for linear and polynomial regression, understanding and using existing tools for regularized linear regression, and (pre-)processing (mixed) categorical and numerical features.
3. This HW should be done in Python, numpy, and sklearn. You can also use other packages such as pandas, though you might find it easier to code in Python yourself.
4. Beside `train.csv` and the semiblind `test.csv` on the website, we also provided the train-dev split (`my_train.csv` and `my_dev.csv`) which is available on the course homepage, so that you can verify your results on dev before submitting test prediction results to kaggle, and so that you can tune hyper-parameters on dev.
5. Part of your grade will be based on your prediction accuracy on test (self report your best prediction error among all your submissions to kaggle).
6. You should submit a single `.zip` file containing `hw3-report.pdf`, `test_submission.csv` (in the format of `sample_submission.csv`), and all your code. Again, LaTeX'ing is recommended but not required.

1 Understanding the Evaluation Metric

This kaggle contest uses "Root Mean Squared Log Error" (RMSLE) rather than the standard "Root Mean Squared Error" (RMSE).

1. What exactly is this RMSLE error? (write the mathematical definition).
   Root Mean Squared Log Error (RMSLE) - It's the RMSE of log of actual and predicted values. That is, RMSLE is the square root of mean of squared differences between log of actual values and log of predicted values.
2. What's the difference between RMSLE and RMSE?
   RMSLE is just RMSE of logarithm of actual and predicted target variable values.
3. Why does this contest adopt RMSLE rather than RMSE?
   RMSLE doesn't penalize huge differences in predicted and actual values when both predicted and actual values are huge numbers as compared to the same difference when both predicted and actual values are not huge numbers.
4. Our TA Liang Zhang got an RMSLE score of 0.11 and was ranked 28 last Spring. What does this 0.11 mean intuitively, in terms of housing price prediction error?
5. What are your RMSLE error and ranking if you just submit `sample_submission.csv`?
   RMSLE - 0.40890 and ranking - 3488

6. What is your "Team name" on kaggle (note this HW should be done individually)?
   SheekhaJ

For the rest of this HW, it is highly recommended that you take the logarithm for the y field (`SalePrice`) before doing any experiment, so that you reduce the problem back to RMSE. However, do not forget to exponentiate it back before submitting to kaggle. In other words, you train your regression systems to predict the logarithm of housing prices, $\log(y)$, but you submit your predictions in the original prices, not the ones after taking log.

2 Naive data processing: binarizing all fields

Take a look at the data. Each training example contains 81 fields in total: the unique `Id` field, 79 input fields, and one output field `SalePrice`. Like in HW1 data, there are two types of input fields: categorical, such as `SaleCondition` and `GarageType`, and numerical, such as `LotArea` and `YrSold`. Note that some fields might be mixed categorical/numerical, such as `LotFrontage` and `GarageYrBlt`: as you know, not all homes have lot frontages (the length of the front side of the lot facing a street – some lots are not facing any street), and not all homes have garages, thus both fields could have occasional `NA` values (for "N/A" or "not applicable"). Following HW1, the simplest thing to do is to binarize all fields. You can use your own Python implementation, or `pandas.get_dummies()`, but I think it is best to use your own implementation for its flexibility and for the rest of this HW.

1. How many features do you get? (Hint: around 7230).
   7227 features

   You can use this command to see the total number of binary features:

   ```
   for i in `seq 2 80`; do cat my_train.csv | cut -f $i -d ',' | sort | uniq | wc -l; done | \
   awk '{s+=$1-1} END {print s}'
   ```

   Let me explain a little bit: the first line loops over each column (except the first which is Id and the last which is the output), and print the values for that column (`cut -f $i`), sort and make unique, and count how many unique values there are for that column (`wc -l`). The second line uses `awk` to sum it up; here `$1` denotes the first column, and `-1` for excluding the header row, and finally print the sum.

2. How many features are there for each field?
   There are 79 fields and corresponding number of features for each are as follows.
   [15, 5, 108, 989, 2, 3, 4, 4, 2, 5, 3, 25, 9, 8, 5, 8, 10, 9, 110, 61, 6, 8, 15, 16, 5, 304, 4, 5, 6, 5, 5, 5, 7, 601, 7, 131, 730, 686, 6, 4, 2, 6, 721, 390, 21, 810, 4, 3, 4, 3, 8, 4, 4, 12, 7, 4, 6, 7, 97, 4, 5, 422, 6, 6, 3, 253, 193, 116, 17, 72, 8, 4, 5, 5, 21, 12, 5, 9, 6, 1]

3. Do you need to augment the space (add bias dimension) like in HW1, or does your regression tool automatically does it for you?
   Regression tool automatically adds the bias dimension (called intercept).

4. Train linear regression using `sklearn.linear_model.LinearRegression` or `np.polyfit` on `my_train.csv` and test on `my_dev.csv`. What's your root mean squared log error (RMSLE)? (Hint: should be around 0.15 or 0.16).
   Training using *sklearn.linear_model.LinearRegression* on *my_train.csv* and testing on *my_dev.csv* RMSLE is 0.157.

5. What are your top 10 most positive and top 10 most negative features? Do they make sense?
   Most positive - GrLivArea=968, OverallQual=3, EnclosedPorch=236, Neighborhood=IDOTRR, BsmtFinSF2=311, LotArea=8281, OverallCond=3, BsmtFinSF1=50, OverallQual=2, BsmtUnfSF=430

Most negative - Neighborhood=NoRidge, BsmtFinSF2=720, RoofMatl=WdShngl, Neighborhood=Crawfor, GrLivArea=1192, OverallQual=8, 2ndFlrSF=472, Neighborhood=StoneBr, FullBath=3, OverallQual=9

6. What's your feature weight for the bias dimension? Does it make sense?
   Feature weight for bias dimension is 12.2.

7. Now predict on `test.csv`, and submit your predictions to the kaggle server. What's your score (RMSLE) and ranking?
   RMSLE - 0.17259; Ranking - 3488

3 Smarter binarization: Only binarizing categorical features

You might have observed that most numerical features shouldn't have been binarized: for example, features like `LotArea` are always positively correlated with the sale price.

1. What are the drawbacks of naive binarization? (Hint: data sparseness)
   Naive binarization prevents the model from learning from the relation between values of the numerical features. Ex: GrLivingArea has positive correlation with the Sale price. This information can't be inferred if the values for GrLivingArea are treated as string constants.

2. Now binarize only the categorical features, and keep the numerical features as is. What about the mixed features such as `LotFrontage` and `GarageYrBlt`?
   LotFrontage and GarageYrBlt has been considered as numerical features.

3. Redo all the questions asked in the naive binarization section. (Hint: the new dev error should be around 0.14, which is much better than naive binarization).

   (a) Number of features for each field
   Total Number of features after binarizing only categorical fields are 288.

   (b) Training using *sklearn.linear_model.LinearRegression* on *my_train.csv* and testing on *my_dev.csv*
   RMSLE - 0.124

   (c) Top 10 positive and top 10 negative features
   Top 10 most positive features - RoofMatl_Membran, RoofMatl_Metal, GarageQual_Ex, Condition2_PosA, GarageCond_Po, RoofStyle_Shed, RoofMatl_Roll, GarageFinish_RFn, GarageFinish_Fin, GarageFinish_Unf
   Top 10 most negative features - RoofMatl_ClyTile, PoolQC_Fa, PoolQC_Ex, Condition2_PosN, PoolQC_Gd, Condition2_RRAe, MSZoning_C (all), Electrical_Mix, Functional_Sev, Functional_Maj2

   (d) Feature weight for bias dimension - **9.3**

   (e) Prediction on test.csv, RMSLE and Kaggle ranking
   RMSLE - 0.14529 Kaggle ranking - 2756

You will see that even if you just do everything up to this point, you should be ranked reasonably in this contest.

4 Experimentation

Try the following to improve your score.

1. Try regularized linear regression (`sklearn.linear_model.Ridge`). Tune α on dev. Should improve both naive and smart binarization by a little bit.
   Naive binarization - Alpha = 5, RMSLE - 0.145
   Smart binarization - Alpha = 5, RMSLE - 0.128

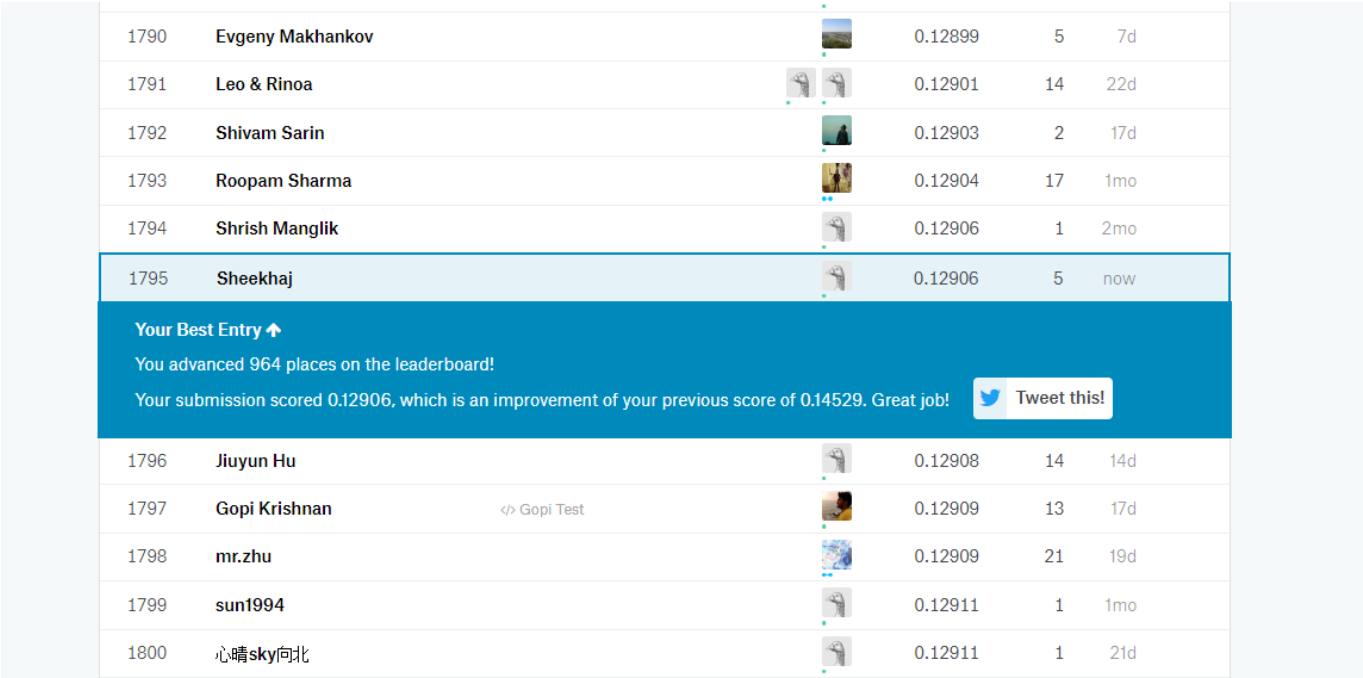
2. Try non-linear features: what if the sale price is quadratically correlated with some of the most important numerical features such as `OverallArea` (also known as square footage) and `LotArea`?
   Taking square of overall area improved the RMSLE on dev data from 0.12798847295440965 to 0.12760435187305452 but did not improve RMSLE on Kaggle submission significantly.

3. Try feature combinations. An obvious candidate is adding a new feature `Years_since_remodeled` = `YearRemodeled` - `YearBuilt`. But is a feature like this really useful?
   Using this feature did not improve RMSLE on dev data.

4. BTW, how are these non-linear features (including feature combinations) relate to non-linear features in the perceptron? (think of XOR)
   Non-linear features help in developing models which gives non-linear decision boundary in higher dimensional space. Whereas perceptron could not linearly separate XOR data. Thus, using non-linear features including feature combination help overcome this limitation of Perceptron.

5. Try anything else that you can think of. You can also find inspirations online, but you have to implement everything yourself (you are not allowed to copy other people's code).
   Using Lasso Regression improved the RMSLE on dev data. Changed to 0.12390333898337055.

What's your best dev error, and what's your best test error and ranking? Take a screen shot of your best test error and ranking, and include your best submission file. **Best test error - 0.12906 Best ranking - 1795**



Debriefing (required):

1. Approximately how many hours did you spend on this assignment? **56 hours**
2. Would you rate it as easy, moderate, or difficult? **Moderate**
3. Did you work on it mostly alone, or mostly with other people? **Mostly alone**
4. How deeply do you feel you understand the material it covers (0%–100%)? **80%**
5. Any other comments? **Indication of marks allocated for questions in each section would be helpful so that students can allocate time in a better way and not spend lot of time on something that probably might not carry as much weight in the total grade.**