HW3

Part 1: Understanding the Evaluation Metric

1. **What exactly is this RMSLE error? (write the mathematical definition).**
   RMSLE (Root Mean Squared Logarithmic Error) computes the error between the predicted and true values using their logarithms.

   $$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$$

2. **What's the difference between RMSLE and RMSE?**
   RMSLE focuses on relative difference (how much bigger or smaller the prediction is compared to the actual value) while RMSE focuses on the absolute difference between the predicted and actual prices.
3. **Why does this contest adopt RMSLE rather than RMSE?**
   RMSLE focuses more on smaller errors and doesn't let very large house prices dominate the score. For example, if the prediction for a cheap house is way off, it matters more than being slightly off for an expensive house making the scoring fairer across all price ranges
4. **One of our TAs got an RMSLE score of 0.11 and was ranked 28 in Spring 2018. What does this 0.11 mean intuitively, in terms of housing price prediction error?**
   A score of 0.11 means the predictions are about 11% off on average in terms of the logarithmic scale. The predictions are pretty close to the actual values
5. **What are your RMSLE error and ranking if you just submit sample_submission.csv?**
   My RMSLE error is 0.40613 and my ranking is 5289.

6. **Why do you need to work in the log output space?**
   When working in log space, we train the model to predict log(price) instead of price. Logarithms shrink the scale of numbers, making predictions more stable and less influenced by very large or small prices.

Part 2: Naïve Data Processing: Binarizing All Fields

1. **How many features do you get?**
   I got 7226 features after binarization
2. **How many features are there for each field?**

Features per field:

Field	Features
MSSubClass	15
MSZoning	5
LotFrontage	108
LotArea	989
Street	2
Alley	3
LotShape	4
LandContour	4
Utilities	2
LotConfig	5
LandSlope	3
Neighborhood	25
Condition1	9
Condition2	8
BldgType	5
HouseStyle	8
OverallQual	10
OverallCond	9
YearBuilt	110
YearRemodAdd	61
RoofStyle	6
RoofMatl	8
Exterior1st	15
Exterior2nd	16
MasVnrType	4
MasVnrArea	305
ExterQual	4
ExterCond	5
Foundation	6
BsmtQual	5
BsmtCond	5
BsmtExposure	5
BsmtFinType1	7
BsmtFinSF1	601
BsmtFinType2	7
BsmtFinSF2	131
BsmtUnfSF	730
TotalBsmtSF	686
Heating	6
HeatingQC	4
CentralAir	2
Electrical	6
1stFlrSF	721
2ndFlrSF	390
LowQualFinSF	21
GrLivArea	810
BsmtFullBath	4
BsmtHalfBath	3
FullBath	4
HalfBath	3
BedroomAbvGr	8
KitchenAbvGr	4
KitchenQual	4
TotRmsAbvGrd	12
Functional	7
Fireplaces	4
FireplaceQu	6
GarageType	7
GarageYrBlt	97
GarageFinish	4
GarageCars	5
GarageArea	422
GarageQual	6
GarageCond	6
PavedDrive	3
WoodDeckSF	253
OpenPorchSF	193
EnclosedPorch	116
3SsnPorch	17
ScreenPorch	72
PoolArea	8
PoolQC	4
Fence	5
MiscFeature	5
MiscVal	21
MoSold	12
YrSold	5
SaleType	9
SaleCondition	6
Total	7226

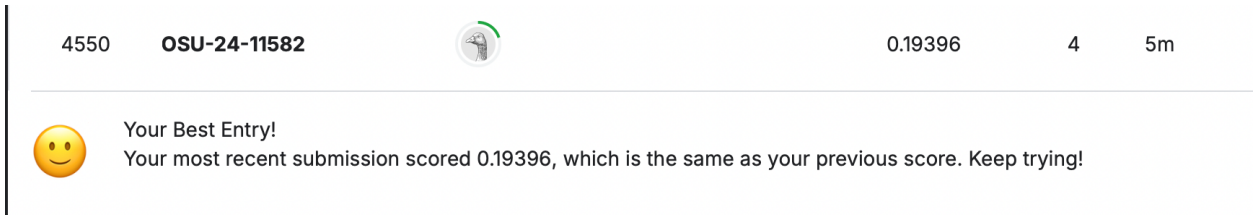
3. **Train linear regression using sklearn.linear_model.LinearRegression or np.polyfit on my_train.csv and test on my_dev.csv. What's your root mean squared log error (RMSLE) on dev? (Hint: should be ~0.152).**
   My RMSLE on dev is 0.15204486336818437.
4. **What are your top 10 most positive and top 10 most negative features? Do they make sense?**

```
Top 10 Positive Features:
FullBath_3: 0.1391803101172374
OverallQual_9: 0.13822104913354893
Neighborhood_StoneBr: 0.12485454803598492
2ndFlrSF_472: 0.11303556417848983
OverallQual_8: 0.10846032143079476
RoofMatl_WdShngl: 0.09353808451727999
GrLivArea_1192: 0.09202309588032372
Neighborhood_NoRidge: 0.08903533688780349
LotArea_8029: 0.0861051097519646
GarageCars_3: 0.08566089859805856

Top 10 Negative Features:
YearRemodAdd_1958: -0.08723416085801855
OverallQual_1: -0.0895517309352441
GarageCars_1: -0.09386822180224487
OverallCond_3: -0.10118895825875515
BsmtFinSF2_311: -0.10765241191128676
LotArea_8281: -0.10811563641389013
OverallQual_3: -0.11705702585843902
EnclosedPorch_236: -0.12269207774472377
GrLivArea_968: -0.12702549636299457
MSZoning_C (all): -0.18380761875307466
```

   Yes, they make sense since most of these features associate with the house prices. Positive features point to higher quality, location, or size, while negative features highlight poorer quality, smaller spaces, or less desirable conditions

5. **Do you need to add the bias dimension (i.e., augmented space) explicitly like in HW2, or does your regression tool automatically handle it for you? Hint: coef_ and intercept_. What's your feature weight for the bias dimension? Does it make sense?**
   No, we do not need to add the bias dimension explicitly since the regression model automatically accounts for the bias term. The bias term (intercept) of 12.1546 means that the baseline predicted house price is approximately $189,000 before accounting for any features. It makes sense because it is close to the mean price (182159.0487062405).
6. **What's the intuitive meaning (in terms of housing price) of this bias feature weight?**
   The bias term acts as the foundation for the model's predictions. It reflects the base price of a house in the dataset, assuming all features are zero.
7. **Now predict on test.csv, and submit your predictions to the kaggle server. What's your score (RMSLE, should be around 0.16) and ranking?**



Part 3: Smarter binarization: Only binarizing categorical features

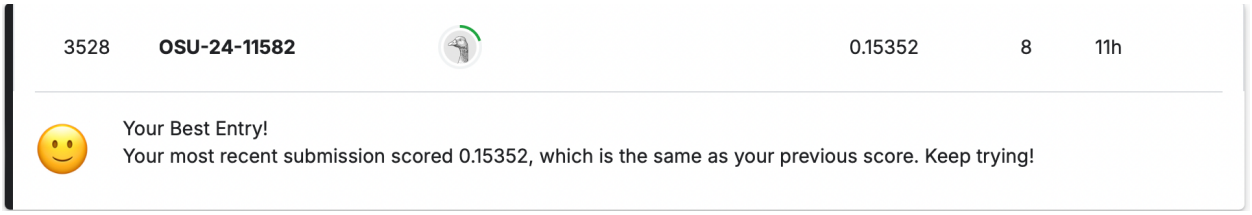
1. **What are the drawbacks of naive binarization? (Hint: data sparseness, etc.)**
   The naïve binarization increases the dimensionality of the data making the model more complicated. It also causes loss of information (natural representation of numerical values) because all values are now treated the same.
2. **Now binarize only the categorical features, and keep the numerical features as is. What about the mixed features such as LotFrontage and GarageYrBlt?**
   Mixed fields such as LotFrontage and GarageYrBlt will be treated as numerical unless they have missing or non-numerical values. Missing numerical values (LotFrontage) are imputed using the mean via the SimpleImputer
3. **Redo the following questions from the naive binarization section. (Hint: the new dev error should be around 0.14, which is much better than naive binarization)**
   (a) How many features are there in total?
   There are 286 features in total.
   (b) What's the new dev error rate (RMSLE)?
   The new RMSLE is 0.12394186786345536
   (c) What are the top 10 most positive and top 10 most negative features? Are they different from the previous section?

```
Top 10 Positive Features:
              Feature  Coefficient
126  RoofMatl_Membran     0.639415
127    RoofMatl_Metal     0.491167
100   Condition2_PosA     0.441535
123    RoofStyle_Shed     0.345296
128     RoofMatl_Roll     0.307944
131  RoofMatl_WdShngl     0.303081
247     GarageQual_Ex     0.280938
129  RoofMatl_Tar&Grv     0.276326
125  RoofMatl_CompShg     0.253371
267  MiscFeature_Gar2     0.230099

Top 10 Negative Features:
                Feature  Coefficient
124     RoofMatl_ClyTile    -2.473988
101      Condition2_PosN    -0.731932
102      Condition2_RRAe    -0.499126
36      MSZoning_C (all)    -0.328501
252        GarageCond_Ex    -0.247543
231       Functional_Sev    -0.208653
227      Functional_Maj2    -0.205660
134  Exterior1st_BrkComm    -0.196562
270     MiscFeature_TenC    -0.174267
208         Heating_Grav    -0.167343
```

The top features differ because naive binarization binarized numerical data, emphasizing specific values like FullBath_3 and OverallQual_9, while smarter binarization preserves numerical trends, allowing categorical features like RoofMatl_Membran and RoofMatl_Metal to stand out.

(d) Now predict on test.csv, and submit your predictions to the kaggle server. What's your score and ranking?



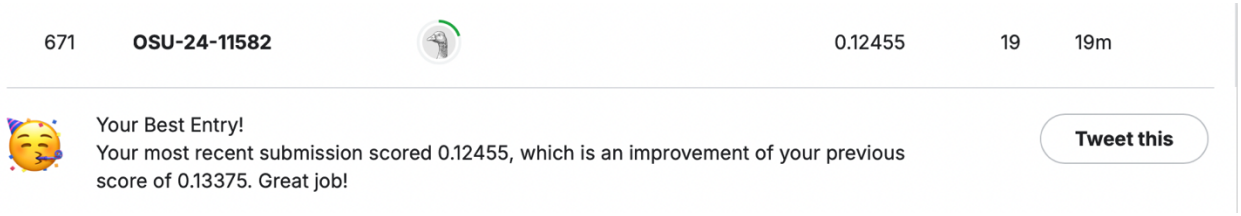
Part 4: Experimentation

1. **Try regularized linear regression (sklearn.linear_model.Ridge). Tune α on dev. Should improve both naive and smart binarization by a little bit**
   It did improve a bit. My public score now is 0.13378
2. **Try non-linear regression (sklearn.preprocessing.PolynomialFeatures)**
   It was slower and my public score now is 0.17627.
3. **How are these non-linear features (including feature combinations) relate to non-linear features in the perceptron? (think of XOR)**
   Polynomial features capture non-linear relationships by creating new features ($x_1 * x_2$, etc.,), which help models understand complex patterns. This is similar to how a perceptron struggles with the XOR problem because it's not linearly separable. Adding non-linear features, like $x_1 * x_2$, makes XOR separable in a higher-dimensional space.

Polynomial features in regression work like this, allowing linear models to handle non-linear problems, just as multi-layer perceptrons do with non-linear activations.

4. **Try anything else that you can think of. You can also find inspirations online, but you have to implement everything yourself**
   I tried feature engineering and was able to improve the error a littlt bit. My RMSLE is 0.11953790845602928

5. **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**



My best RMSLE is 0.11953790845602928. My public score is 0.12455 and ranking is 671.

Debriefing:

1. **Approximately how many hours did you spend on this assignment?**
   I spent about 9 hours on this assignment.
2. **Would you rate it as easy, moderate, or difficult?**
   I would rate is as somewhat difficult.
3. **Did you work on it mostly alone, or mostly with other people?**
   I worked on it with a friend this time.
4. **How deeply do you feel you understand the material it covers (0%–100%)?**
   I feel like I understand the material about 85%
5. **Any other comments?**
   I don't have any comments.