Lab 3

By Daniel Diamont (dd28977) Jerry Yang (jhy395) Zhaofeng Liang (zl4685)

Question 1

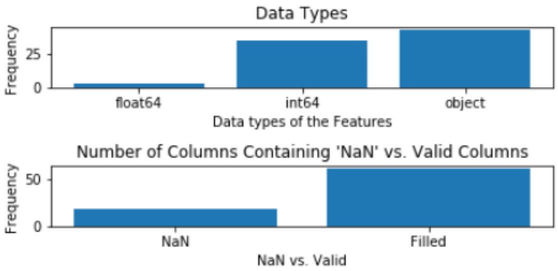
A communication system consists of five parts- The information source which produces a message to be communicated to the receiving terminal, a transmitter which operates on the message in some way to produce a suitable signal, followed by the channel, the receiver, and destination. Teletype and telegraphy are some forms of a discrete channel for information transmission. A discrete source generates the message, which produces some sequence of symbols to communicate information. This could be in the form of dots or dashes. The communication is used with artificial languages to construct simple stochastic processes. This can be represented by discrete Markoff processes that group with special properties of significance in communication theory. There are also ergodic process, where every sequence produced by the process is the same in statistical properties. Roughly the ergodic property maintains statistical homogeneity. We can also define a quantity which measures how much information is produced by the Markoff process. The entropy in probabilities is very important to determine information theory with measures of certainty and uncertainty. There are also features such as conditional entropy, and the entropy of joint events. For each possible state, there is a set of entropies for each state with a set of probabilities. The rest seemed to be unrelated to entropy.

```
The 10 most common words in the dataset are:
[('learning', 12030), ('model', 8253), ('algorithm', 7921), ('data', 7743), ('set', 6052), ('function', 6000), ('us
ing', 5577), ('neural', 5511), ('time', 5097), ('one', 4846)]
```

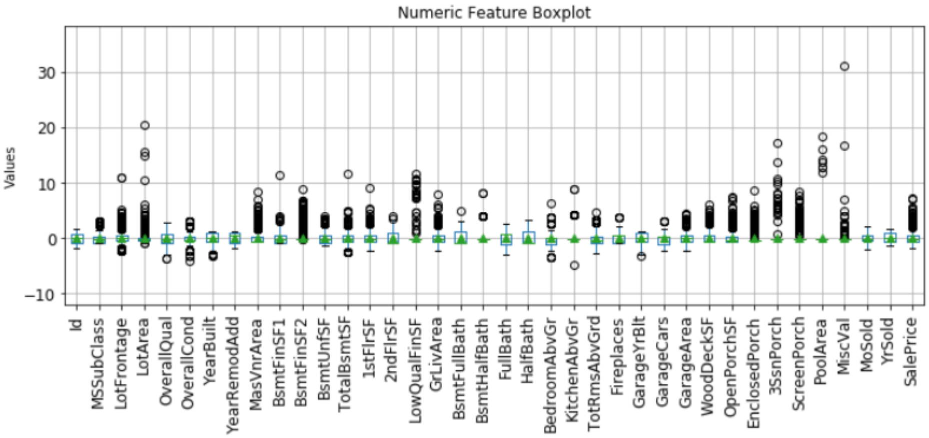
```python
# we are finding the entropy of a bernoulli random variable

pr = count/len(rlist) # turn this into a probability

E = -(pr*np.log2(pr) + (1-pr)*np.log2(1-pr)) # find the entropy of our random variable
print("Entropy(Z) = " + str(E))
```

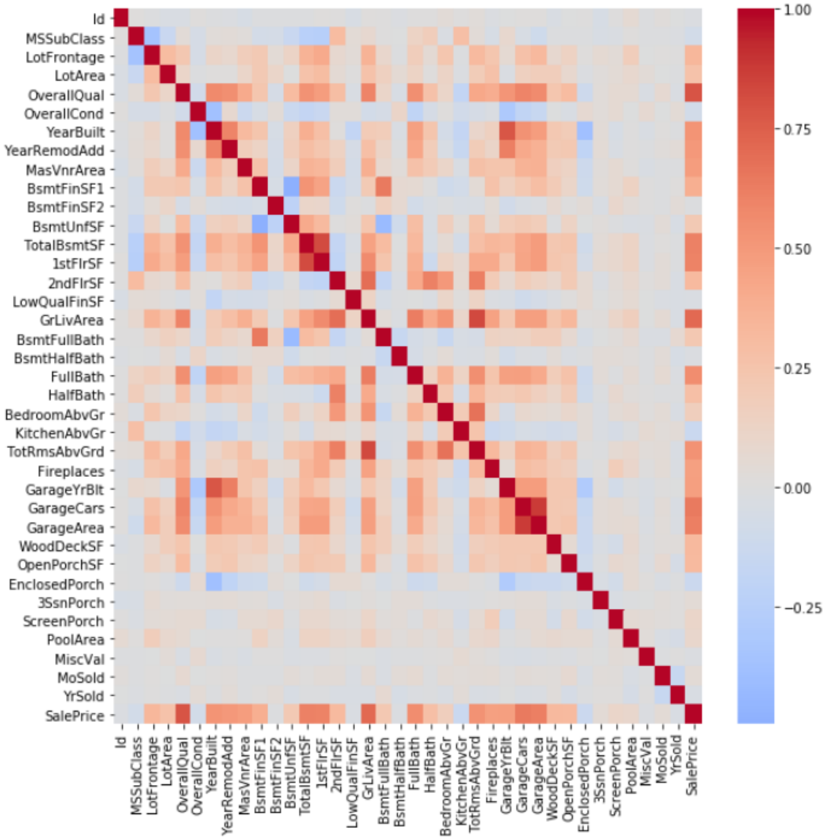
```
Entropy(Z) = 0.11180482184875506
```

```
reproduction gamma including reinforcement figure density instructions gps receives figure factor using org matrix
using sgn system thank regret achieve assumption complex decisions saga surrogate squared non introduction tong bou
nded figure sets posterior addresses range cannot number kian based trees memorizing easily jon lack scale principa
l springer hjl function satisfies optimization different suresh every burget step names approximate existing mehrya
r classes consistent neurons flat rmer provided environments solves covariance parameter stability jiang california
sparsity patterns learning represented supervised general following estimate hessian measures biometrika xiao gradi
ent experiments simulations abhishek probabilities algorithm along simple certain agent sparse operator parameters
multiplication class suppose link problem function form nonparametric model show variance sec learning restricted f
eatures relates approach conditions train based leibo scenarios performs bounded grant algorithm conf bach machine
differential btl incurred tools parameters temperature via model existing international expert function comparing s
utton original thereby predictive sgd likewise applied recursively section zisserman coding heavy scale loss issue
conditional data multichannel generalization distributions size nesterov rankone measurements also electronic train
ing global methodology selection guarantee processing grid unavailable lower eqn active sub first problem seen supe
rvised improvement seeding minimization least four point arbitrary cost amounts maximize let cardinality volume opt
imization bound fair multivariate estimation values patterns learned constant lemma marginal handle tij ximax zhe r
ong practical mini steps density view dependent source optimization estimator constant sampling activates pose vert
ices impact posterior supported versus test min theorem aslan states given deco specifically hessian sometimes buff
ers outputs video optimizer given gpis completed given analogymaking regularization solutions infeasible regression
utility proceedings output strategy association set iterates networks approximation close provide bsnmf report inva
riant instance natural points fpr contrasts next data models science perations scenario mean corroborates combinati
on statistical denote energy rank defined algorithms nesterov clustering different imagenet sample bounds valid exa
mple certain scs jump stavros faster location include jump shells although grant number exploration bound learning
perform description deep looking properties rmse guarantees start extract french seventeenth guarantee dataset pose
versions dynamically steiner inverse approximation task experimental hawkes mini friend robust event random model r
anking complexity bucila improve log loss let advantages convergence data decent clique level effectively step prob
lems set ensures euclidean trained sets speedup decay layers coordinates preconditioned settings section coarse let
player vector vikas theory nll satisfy copy computational hochreiter estimated proof pairwise paths view light func
tion stochastic gives convolutional graphs besides hyperplane rewards abstract independence value report propositio
n several handle christian descent target structure becomes experiments graph ccnns produce next non theory use alg
orithm representing circuit set enough point several immediately distributed margin alon reed variety feature courv
ille also decision john converge precisely qei stochastic training class arg similar level applying machine paramet
ers sensing validation trischler fewer zehan theory ztf sect states recently built generative linear function probl
em skipping robust important bound
```



Here, we can see that our data is spread pretty evenly between numeric and categorical features. Since categorical features make up about half our data, we have to make sure that these features are encoded in such a way that our regressor can correctly interpret and use these features. We will use 'dummy encoding' for this purpose.





Looking at either the bottom triangle or the upper triangle of the matrix, we see that about half of the features are uncorrelated with each other, and the other half of the features has some correlation with each other. Another assumption about OLS is that there is no *multicolinearity* in the features, meaning that we desire that the features be linearly uncorrelated with each other in order to produce the best results. It may be worth investigating some of these highly correlated features like GarageCars and GarageArea to see if we can transform or remove this data to decrease the multicolinearity of our features.

Additionally, looking at the SalePrice row, we see can identify a number of features that appear to be very positively correlated with the SalePrice of a house. It may be worth it to look into these features to examine if these features make good sense to use to train our model, and if the features themselves are correlated and why. A useful test to demonstrate the validity of these features to function as predictors could be to run Lasso regression to see if these same features have non-zero coefficients.

Notice that MSSubClass consists of categorical variables but are listed an integers. This enforces a non-existent notion of distance on the feature. Let's replace with strings, then convert the categories to dummy variables.

```
ms_sub_class = np.array(all_data.MSSubClass)
dummy_dict = {
    "20": "A",
    "30": "B",
    "40": "C",
    "45": "D",
    "50": "E",
    "60": "F",
    "70": "G",
    "75": "H",
    "80": "I",
    "85": "J",
    "90": "K",
    "120": "L",
    "150": "M",
    "160": "N",
    "180": "O",
    "190": "P"
}

new_feature = []
for i in range(len(ms_sub_class)):
    key = str(ms_sub_class[i])
    if (key in dummy_dict.keys()):
        new_feature.append(dummy_dict.get(key))
```

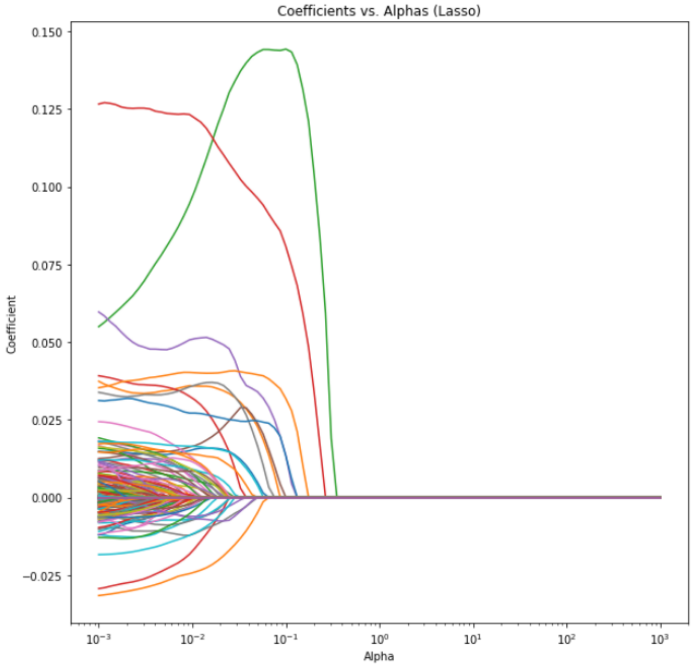
The following numerical features will be removed because they are heavily correlated with each other.

1. Garage Year Built (correlates with Year Built)
2. Total Basement Square Footage (correlates with 1st Floor Square Footage)
3. Total Rooms Above Ground (correlates with Ground Living Area)

The following numerical features will be removed because they do not correlate at all with the target variable.

1. 3SsnPorch
2. MiscVal
3. MoSold
4. YrSold

Plot l_0 norm (number of nonzeros) of coefficients that Lasso produces as we vary strength of regularization parameter alpha.



Ensembling and Stacking

```
from sklearn.ensemble import BaggingRegressor

# use bagging regressor with bootstrapping (with replacement)
# base estimator is our ridgeCV model which is the best performer on the test set yet
# 10 base estimators
# use all available processors

ensemble_model = BaggingRegressor(base_estimator=ridgeCV,
                                  n_estimators=10,
                                  max_samples=0.66,
                                  max_features=1.0,
                                 bootstrap=True,
                                 bootstrap_features=True,
                                 oob_score=False,
                                 warm_start=False,
                                 n_jobs=-1)

ensemble_model.fit(X_train,y_train)
```

```
BaggingRegressor(base_estimator=RidgeCV(alphas=array([1.00000e+03, 8.69749e+02, ..., 1.14976e-03, 1.00000e-03]),
    cv=5, fit_intercept=True, gcv_mode=None, normalize=False, scoring=None,
    store_cv_values=False),
         bootstrap=True, bootstrap_features=True, max_features=1.0,
         max_samples=0.66, n_estimators=10, n_jobs=-1, oob_score=False,
         random_state=None, verbose=0, warm_start=False)
```

```
y_pred_ensemble = ensemble_model.predict(X_test)
print("Ensemble MSE: {:.5f}".format(mean_squared_error(y_test,y_pred_ensemble)))
```

```
   Ensemble MSE: 0.01821
```

Make prediction for test set

```
submission_prediction = ridgeCV.predict(X_submission_test)
```

```
submission_df = pd.DataFrame({"id":test.Id, "SalePrice":submission_prediction})
```

```
submission_df['SalePrice'] = np.expm1(submission_df['SalePrice'])
```

```
submission_df.to_csv("test_sol.csv", index = False)
```