GENERAL BUSINESS 656

Final Project

Members:

Sng Hong Wee Malcolm

Loh Ching Hern

Contents

Introduction

The competition aims to use advanced regression techniques to predict the final price of homes. This has many business applications. For example, people who are interested in flipping homes could use models created in this project to find undervalued homes, such that they can obtain a bigger profit margin from flipping it. Another application would be home buyers using the models to determine if the seller's ask price is unfair.

The dataset contains 79 variables. The data is already split up into test and training datasets, with the training dataset containing 1460 entries.

Here is the link to the competition.

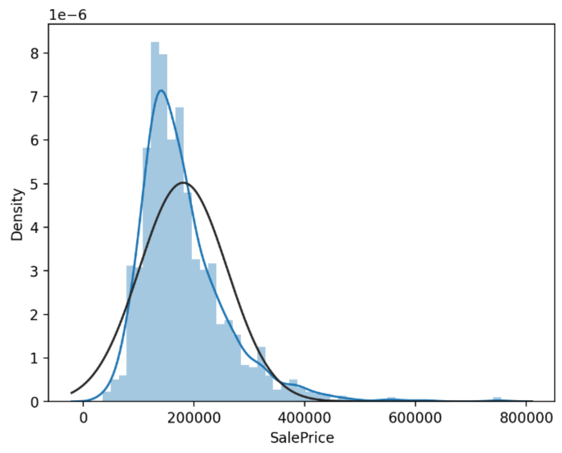
Acknowledgements

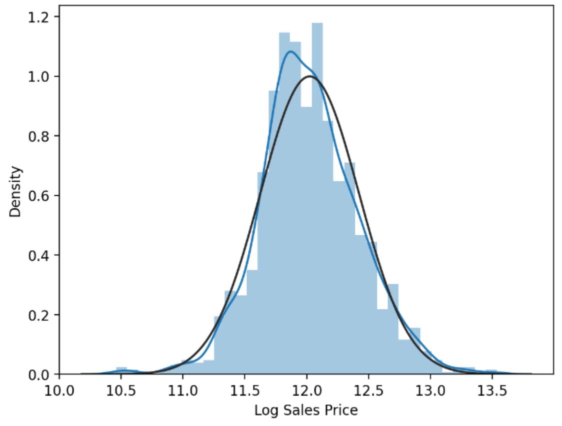
The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often-cited Boston Housing dataset.

Data Preprocessing

Firstly, we convert categorical variable into dummies using pandas.get_dummies function. Next, we replace all the null values in the dataset with the average value. This ensures the null entries do not skew the entire dataset too much.

We then do some exploratory analysis on our dataset to see if it fulfils the condition for regression. One of the important factors is the normality of the data.

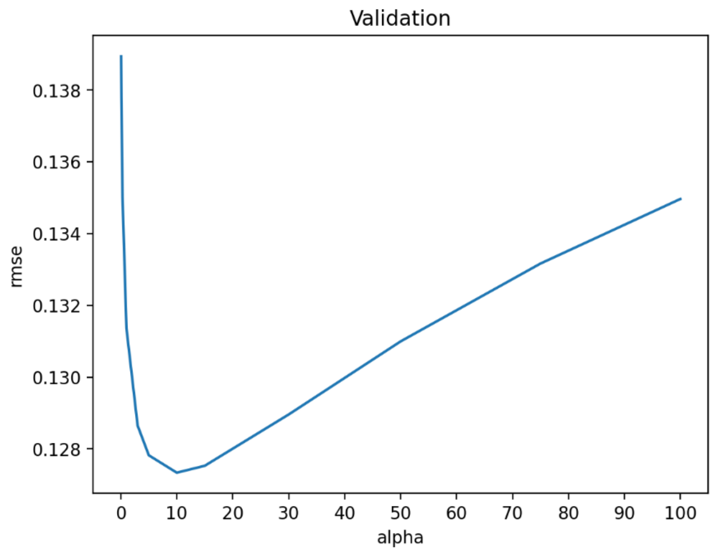




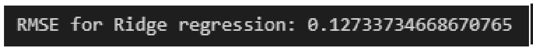
As can be seen from the histogram, the data has a significant right skewness, however this can be corrected by logging the SalesPrice, as well as any other variables that are skewed.

Regression Techniques

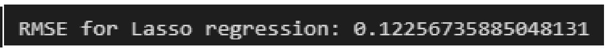
In this competition, we will first be trying ridge and lasso regression techniques and evaluating their effectiveness. Before we apply lasso regression, we have first to figure out the ideal alpha parameter for it.



As seen above, alpha is too large the regularization is too strong, and the model cannot capture all the complexities. If alpha is too small, the model overfits. The optimal alpha is around 10, from the plot above.



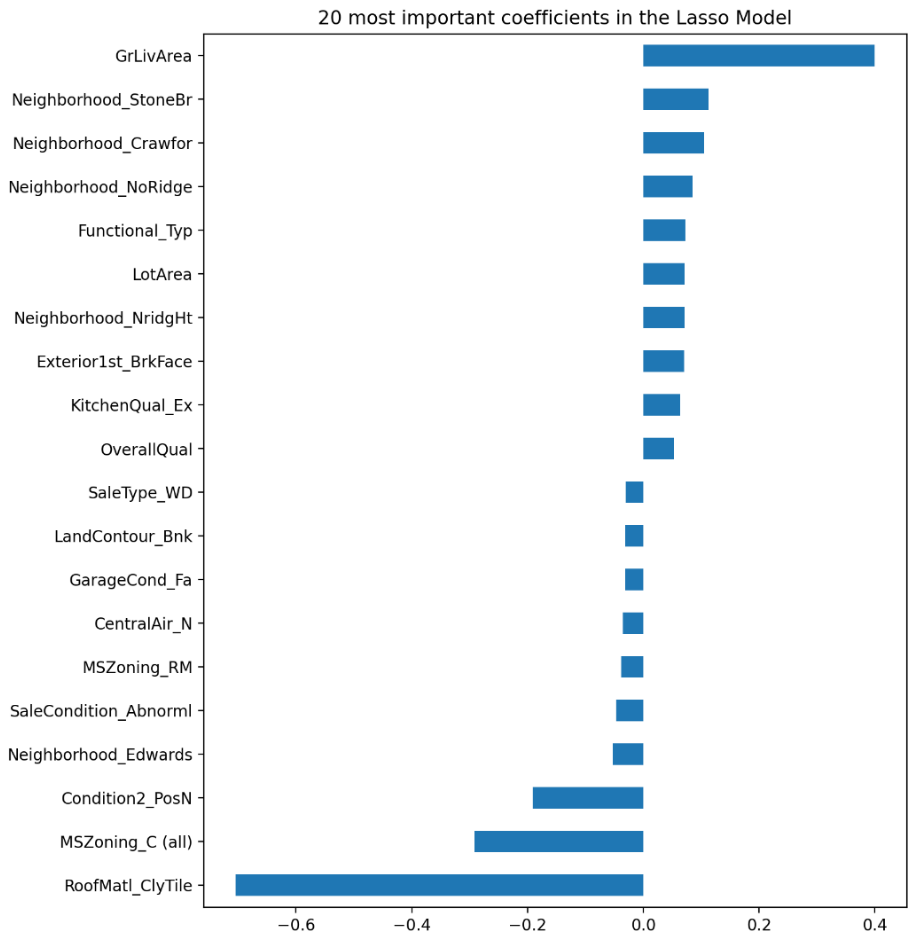
```
RMSE for Ridge regression: 0.12733734668670765
```



```
RMSE for Lasso regression: 0.12256735885048131
```

From the above RMSE, Lasso regression is preferred due to lower RMSE value.

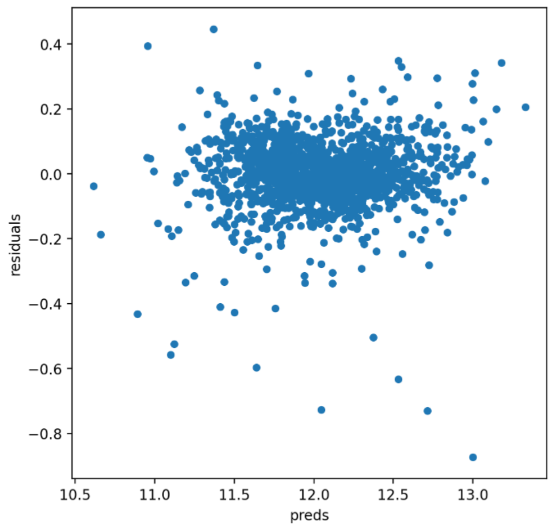
Regression Analysis



This chart illustrates the 10 most positively correlated variables and the 10 most negatively correlated variables to LogSalesPrice.

From the above chart, we can see that the most important positive feature is GrLivArea (Above grade (ground) living area square feet), while the most important negative feature is RoofMatl_ClyTile (Roof material made from clay tiles, dummy variable).

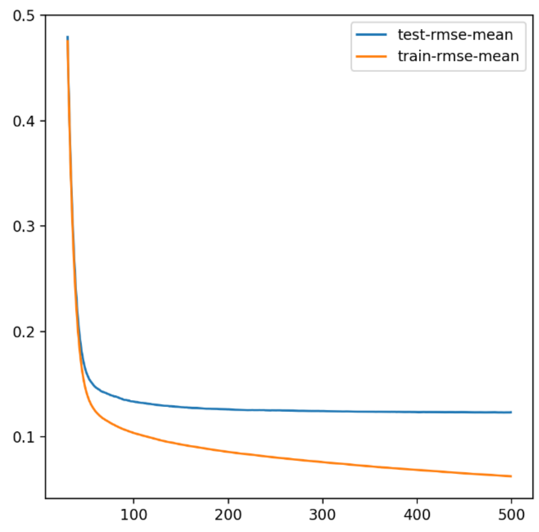
This makes sense, since the amount of living space is a big factor to consider when buying a home. But there are some negatively correlated variables which deviate from expectations, such as roofs being made from clay tiles, which is known as an expensive and durable material, and normally would cause the price of houses to increase.

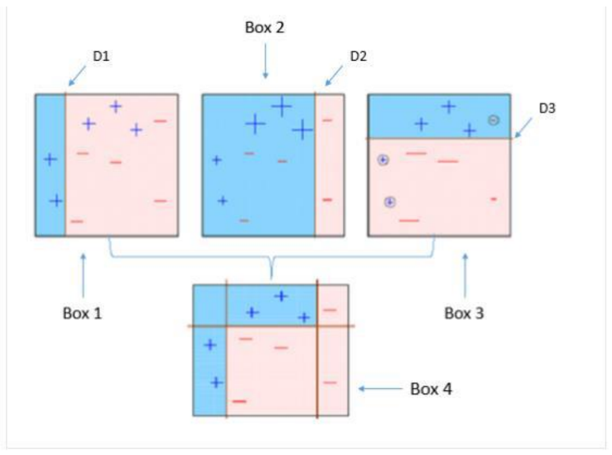


The residual plot seems to be randomly distributed, indicating no heteroskedasticity, adding credibility to the lasso regression model.

Gradient Boosting

Boosting is a sequential technique which works on the principle of an ensemble. It combines a set of weak learners and delivers improved prediction accuracy. At any time, **t**, the model outcomes are weighed based on the outcomes of previous **t-1**. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher.

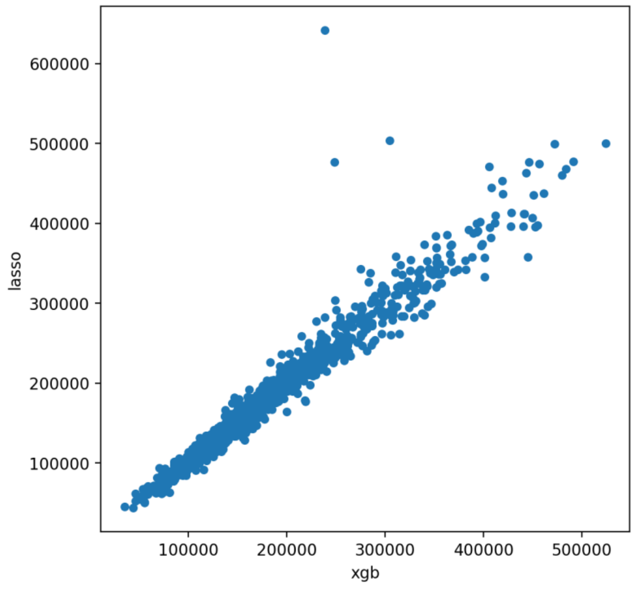




In the diagram on the right, 4 classifiers (in 4 boxes), shown above, are trying to classify + and - classes as homogeneously as possible

From the graph on the left, the x-axis is the number of boosting rounds while the y-axis is the rmse. It can be seen using the test data, the rmse bottoms out after roughly 100 boosting rounds.

Gradient Boosting Results



From the above graph of data points predicted by both lasso and gradient boosting, one can see that it is a linear gradient, indicating that both models give roughly the same output on the test set. Lasso however gives a few predictions that are way higher than gradient boosting.

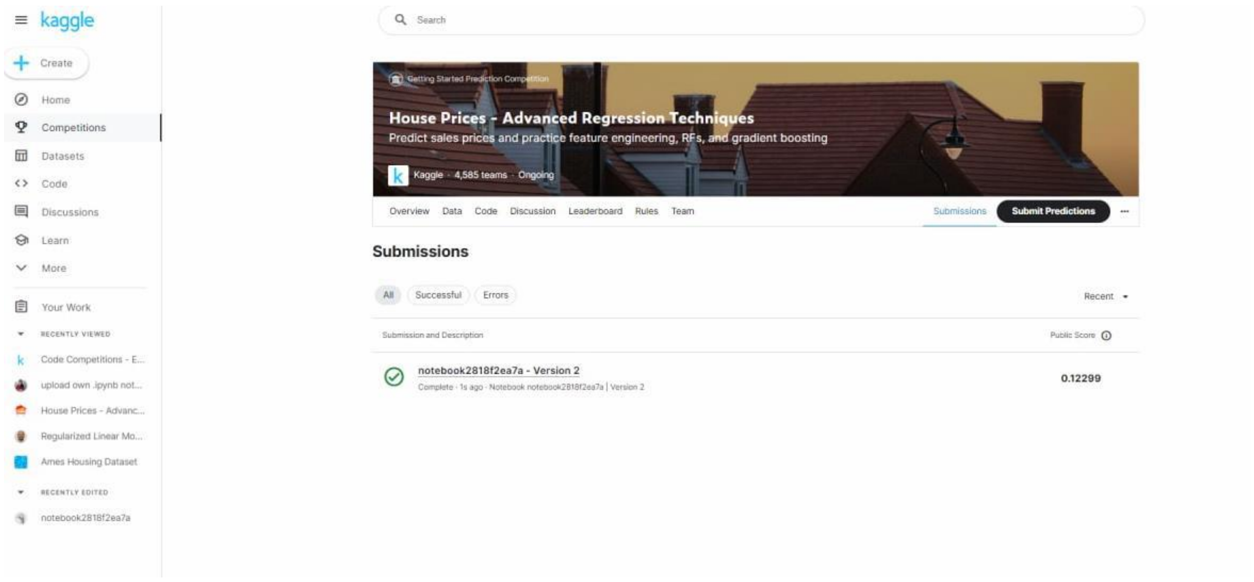
Conclusion

Lasso regression seems to be a good fit for the dataset we have on hand. Iterating through all the parameters for lasso, an alpha of 10 seems to be optimal, giving the lowest RMSE value. Additionally, we can glean additional insights from the lasso regression by seeing the most important positive impacting and negative impacting coefficient variable in the regression. Checking the diagnostics of the lasso regression also shows that the model is a good fit for the dataset.

Moving on, we tried extreme gradient boosting using xgboost algorithm. From our data, the training dataset RMSE bottoms out at around 100 boosting rounds.

Comparing both algorithms, they both seems to give roughly the same outputs. This can be seen by plotting predictions from both lasso and xgboost, and the relationship is mostly linear.

Submission



House Prices - Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

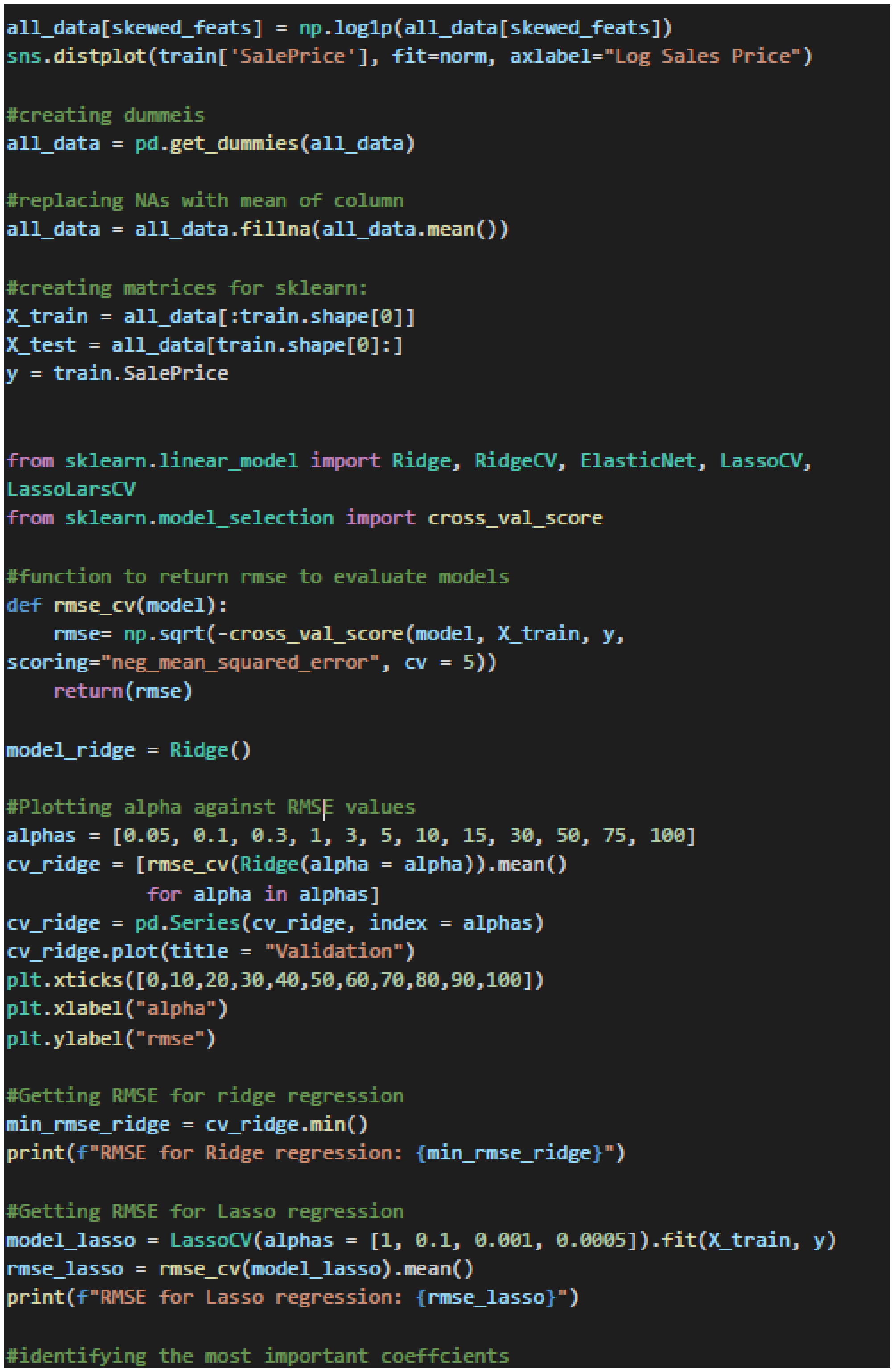
Kaggle · 4,585 teams · Ongoing

Submissions

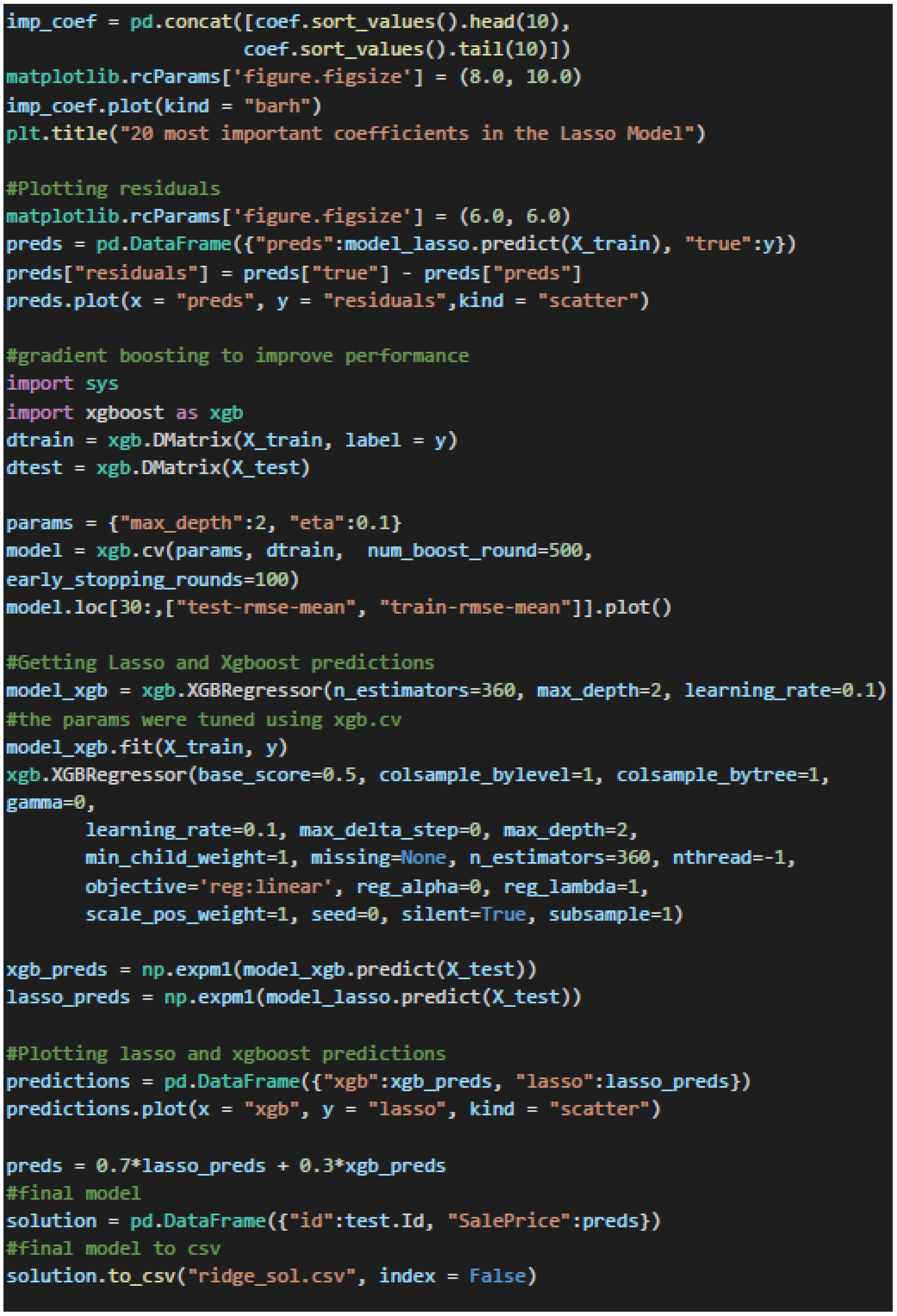
Submission and Description	Public Score
notebook2818f2ea7a - Version 2 Complete · 1s ago · Notebook notebook2818f2ea7a | Version 2	0.12299

Code

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
from scipy.stats import skew
from scipy.stats.stats import pearsonr
from scipy.stats import norm
%config InlineBackend.figure_format = 'retina'
%matplotlib inline

#importing data
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")

all_data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'],
                      test.loc[:,'MSSubClass':'SaleCondition']))

#exploratory plots
sns.distplot(train['SalePrice'], fit=norm)

#Log transform for skewed numeric features
train["SalePrice"] = np.log1p(train["SalePrice"])

#log transform skewed numeric features:
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index
skewed_feats = train[numeric_feats].apply(lambda x: skew(x.dropna())) #compute
skewness
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index
```



```
all_data[skewed_feats] = np.log1p(all_data[skewed_feats])
sns.distplot(train['SalePrice'], fit=norm, axlabel="Log Sales Price")

#creating dummeis
all_data = pd.get_dummies(all_data)

#replacing NAs with mean of column
all_data = all_data.fillna(all_data.mean())

#creating matrices for sklearn:
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y = train.SalePrice


from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, LassoCV,
LassoLarsCV
from sklearn.model_selection import cross_val_score

#function to return rmse to evaluate models
def rmse_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, y,
scoring="neg_mean_squared_error", cv = 5))
    return(rmse)

model_ridge = Ridge()

#Plotting alpha against RMSE values
alphas = [0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30, 50, 75, 100]
cv_ridge = [rmse_cv(Ridge(alpha = alpha)).mean()
            for alpha in alphas]
cv_ridge = pd.Series(cv_ridge, index = alphas)
cv_ridge.plot(title = "Validation")
plt.xticks([0,10,20,30,40,50,60,70,80,90,100])
plt.xlabel("alpha")
plt.ylabel("rmse")

#Getting RMSE for ridge regression
min_rmse_ridge = cv_ridge.min()
print(f"RMSE for Ridge regression: {min_rmse_ridge}")

#Getting RMSE for Lasso regression
model_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(X_train, y)
rmse_lasso = rmse_cv(model_lasso).mean()
print(f"RMSE for Lasso regression: {rmse_lasso}")

#identifying the most important coeffcients
```



```
imp_coef = pd.concat([coef.sort_values().head(10),
                     coef.sort_values().tail(10)])
matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)
imp_coef.plot(kind = "barh")
plt.title("20 most important coefficients in the Lasso Model")

#Plotting residuals
matplotlib.rcParams['figure.figsize'] = (6.0, 6.0)
preds = pd.DataFrame({"preds":model_lasso.predict(X_train), "true":y})
preds["residuals"] = preds["true"] - preds["preds"]
preds.plot(x = "preds", y = "residuals",kind = "scatter")

#gradient boosting to improve performance
import sys
import xgboost as xgb
dtrain = xgb.DMatrix(X_train, label = y)
dtest = xgb.DMatrix(X_test)

params = {"max_depth":2, "eta":0.1}
model = xgb.cv(params, dtrain,  num_boost_round=500,
early_stopping_rounds=100)
model.loc[30:,["test-rmse-mean", "train-rmse-mean"]].plot()

#Getting Lasso and Xgboost predictions
model_xgb = xgb.XGBRegressor(n_estimators=360, max_depth=2, learning_rate=0.1)
#the params were tuned using xgb.cv
model_xgb.fit(X_train, y)
xgb.XGBRegressor(base_score=0.5, colsample_bylevel=1, colsample_bytree=1,
gamma=0,
       learning_rate=0.1, max_delta_step=0, max_depth=2,
       min_child_weight=1, missing=None, n_estimators=360, nthread=-1,
       objective='reg:linear', reg_alpha=0, reg_lambda=1,
       scale_pos_weight=1, seed=0, silent=True, subsample=1)

xgb_preds = np.expm1(model_xgb.predict(X_test))
lasso_preds = np.expm1(model_lasso.predict(X_test))

#Plotting lasso and xgboost predictions
predictions = pd.DataFrame({"xgb":xgb_preds, "lasso":lasso_preds})
predictions.plot(x = "xgb", y = "lasso", kind = "scatter")

preds = 0.7*lasso_preds + 0.3*xgb_preds
#final model
solution = pd.DataFrame({"id":test.Id, "SalePrice":preds})
#final model to csv
solution.to_csv("ridge_sol.csv", index = False)
```