**House Pricing**

**Problem:** To predict house pricesfor given features in Ames, Iowa dataset.

**Problem Significance**: With 79 given features describing every aspect of residential homes, the problem has scope for creative feature engineering.

**Data Set:** Data set is available on <http://kaggle.com>.

**Related Work:**

**Data Cleaning and Pre-Processing:**

* Initially we concatenated both the train data and test data.
* Using Python, we have removed the skewness in the data by applying logarithm.
* Handled missing data

For categorical variables- calculated dummies.

For continuous attributes – filled NA with mean.

* Plotted Correlation between continuous attributes using heat map.

**Code:**

#import packages which are necessary

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import skew, skewtest

import matplotlib

train = 'C:/Materials/Classes/Big Data/House Prices/Data/train.csv'

test = 'C:/Materials/Classes/Big Data/House Prices/Data/test.csv'

df\_train = pd.read\_csv(train)

df\_test = pd.read\_csv(test)

df\_full =df\_train.append(df\_test)

fullCat = df\_full.select\_dtypes(include=['object']).index

fullCont = df\_full.dtypes[df\_full.dtypes !='object'].index

skewed\_data = df\_full[fullCont].apply(**lambda** x: skew(x.dropna())) *#compute skewness*skewed\_data = skewed\_data[skewed\_data > 0.75]  
skewed\_data = skewed\_data.index  
df\_full[skewed\_data] = np.log(df\_full[skewed\_data] + 1)

df\_full = pd.get\_dummies(df\_full)

df\_full = df\_full.fillna(df\_full[:df\_train.shape[0]].mean())

def heatmap(df,labels):

cm = np.corrcoef(df[labels].dropna().values.T)

sns.set(font\_scale=1)

hm = sns.heatmap(cm,

cbar= False,

annot =False,

square= True,

vmax=1

)

#heatmap = ax.pcolor(nba\_sort, cmap=plt.cm.Blues, alpha=0.8

color\_map = plt.cm.Blues

plt.pcolor(cm,cmap=color\_map)

plt.colorbar().set\_label("Features", rotation=270)

hm.set\_xticklabels(labels, rotation=90)

hm.set\_yticklabels(labels[::-1], rotation=0)

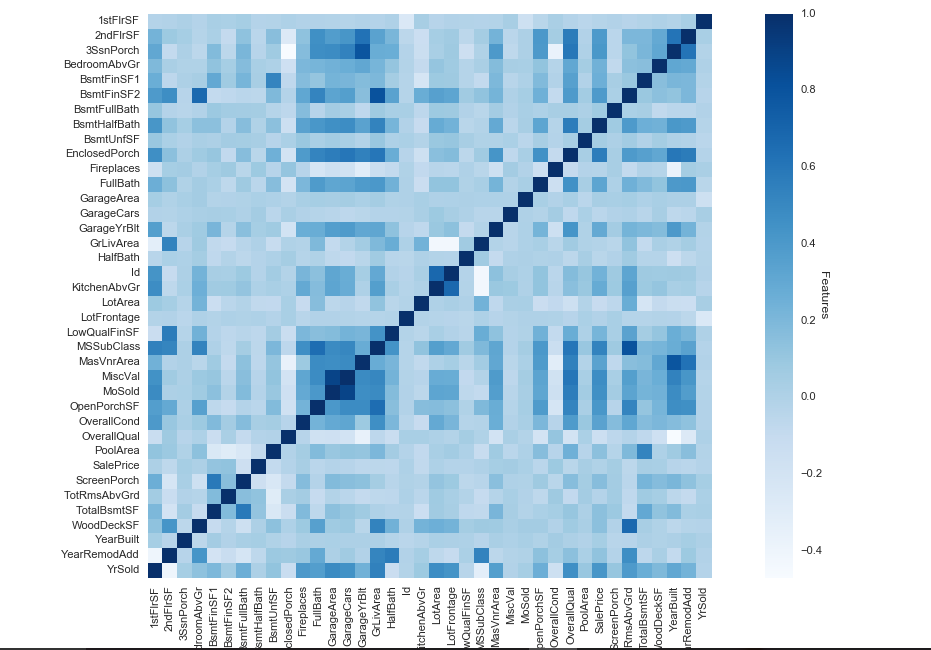
return hm,cm

Features = list(df\_full[fullCont].columns.values)

plt.figure(figsize = (20,10))

htmp,corrm = heatmap(df\_full[fullCont],Features)

plt.show()



**Feature Selection**

df\_train['SalePrice'] = np.log(df\_train['SalePrice'])

del df\_full['SalePrice']

trainData = df\_full[:df\_train.shape[0]]

testData = df\_full[:df\_test.shape[0]]

YData = df\_train.SalePrice

**Applying Lasso Model:**

Lasso model :

*class* sklearn.linear\_model.Lasso(*alpha=1.0*, *fit\_intercept=True*, *normalize=False*, *precompute=False*, *copy\_X=True*, *max\_iter=1000*, *tol=0.0001*, *warm\_start=False*, *positive=False*, *random\_state=None*, *selection='cyclic)* [¶](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso)

from sklearn.cross\_validation import cross\_val\_score

from sklearn.linear\_model import LassoCV, LassoLarsCV, LinearRegression

#calculating root mean square error

def rmse\_cv(model):

rmse= np.sqrt(-cross\_val\_score(model, trainData, YData, scoring="neg\_mean\_squared\_error", cv=5))

return(rmse)

model\_lasso = LassoCV(alphas = [1, 0.1, 0.001,0.0005], selection='random', max\_iter=15000).fit(trainData, YData)

res = rmse\_cv(model\_lasso)

#print(res)

print("Lasso Mean:",res.mean())

print("Lasso Min: ",res.min())

**Finding out important features by Lasso Model**

coef = pd.Series(model\_lasso.coef\_, index = trainData.columns)

#print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other " + str(sum(coef == 0)) + " variables")

imp\_coef = pd.concat([coef.sort\_values().head(10),

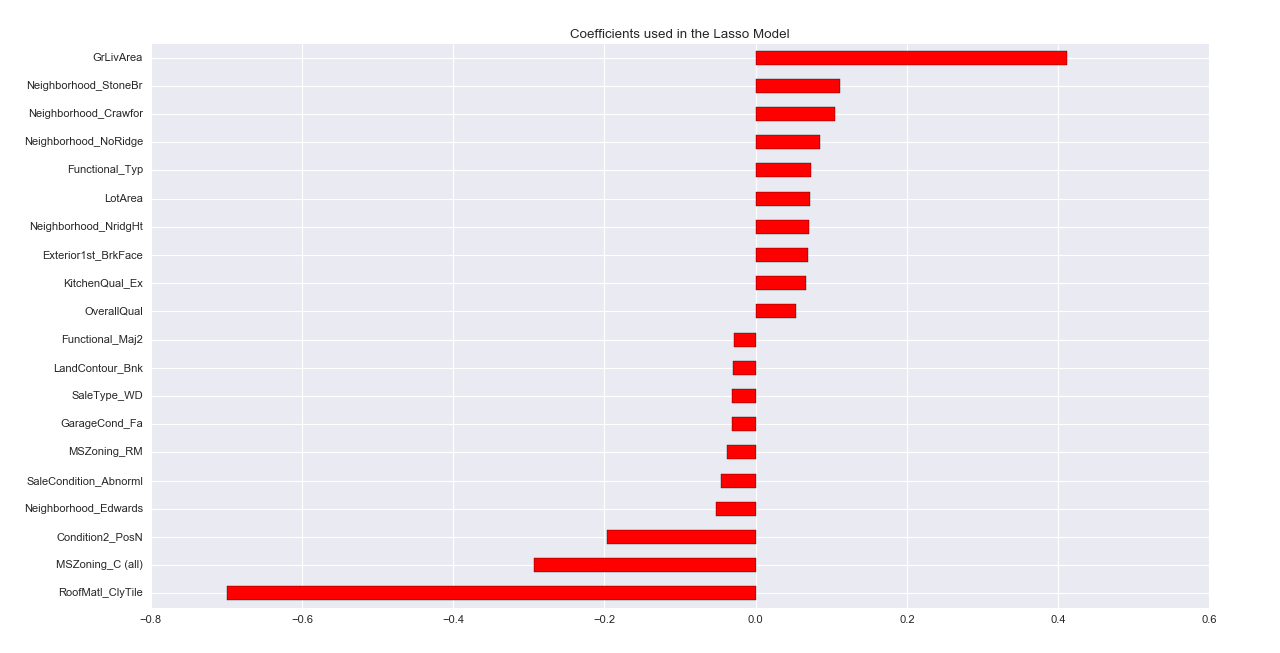
coef.sort\_values().tail(10)])

matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)

imp\_coef.plot(kind = "barh",color = 'r')

plt.title("Coefficients used in the Lasso Model")

plt.show()



Most Important Features to determine the final Price of House are:

1. GrLiveArea
2. Neighbourhood\_StoreDr
3. Neighbourhood\_CreateFor
4. Neighbourhood\_NoRidge
5. Functional\_Type
6. LotType

Predicting the final Sale Price of the house using Lasso:

test\_preds = np.expm1(model\_lasso.predict(testData))

result = pd.DataFrame()

result['Id'] = df\_test['Id']

result["SalePrice"] = test\_preds

result.to\_csv("C:/Materials/Classes/Big Data/House Prices/Data/lasso.csv", index=False)

**Applying Linear Regression:**

from sklearn.linear\_model import LinearRegression

linear\_regression = LinearRegression()

linear\_regression = linear\_regression.fit(trainData,YData)

res1 = rmse\_cv(linear\_regression)

test\_preds = (linear\_regression.predict(testData))

result2 = pd.DataFrame()

result2['Id'] = df\_test['Id']

result2["SalePrice"] = np.exp(test\_preds)

result2.to\_csv("C:/Materials/Classes/Big Data/House Prices/Data/linear.csv", index=False)

print("Linear Regression Mean",res1.mean())

print("Linear Regression Min:",res1.min())

**Applying XGBoost Model**

import xgboost as xgb

regr = xgb.XGBRegressor(

colsample\_bytree=0.2,

gamma=0.0,

learning\_rate=0.01,

max\_depth=4,

min\_child\_weight=1.5,

n\_estimators=7200)

regr.fit(trainData,YData)

print("XGBoost Mean ", rmse\_cv(regr).mean())

res2 = rmse\_cv(regr).mean()

pred\_xgb= regr.predict(testData)

result1 = pd.DataFrame()

result1['Id'] = df\_test['Id']

result1["SalePrice"] = np.exp(pred\_xgb)

result1.to\_csv("C:/Materials/Classes/Big Data/House Prices/Data/xgb.csv", index=False)

**Comparing the Root mean square error for 3 models:**

labels = ['LinearRegression','Laaso Model','XGBoost']

meanScores = [0.1655,0.1229,0.119]

ind\_scrs = np.arange(0,len(meanScores))

width = 0.3

fig, ax = plt.subplots()

rects = ax.bar(ind\_scrs, meanScores, width, color = 'b')

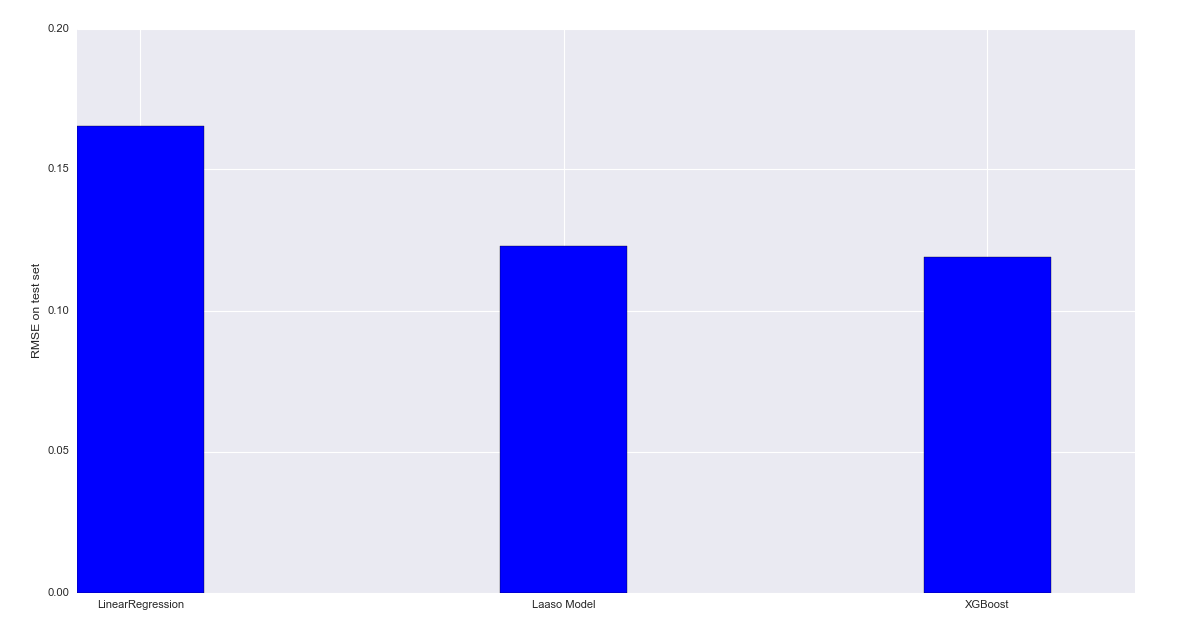
ax.set\_xticks(np.array(ind\_scrs) + width/2)

ax.set\_xticklabels(labels)

ax.set\_ylabel('RMSE on test set')

ax.set\_ylim([0,0.2])

plt.show()



**Conclusion**

* When comparing three models, Xgboost regression model works better followed by Lasso model and Linear regression model.