House Prices dataset: Model building

In the following cells, we will finally build our machine learning model, utilising the feature engineered data and the pre-selected features.

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import Lasso

from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt

pd.set_option('display.max_columns', None)
```

In [2]:
```
X_train = pd.read_csv('xtrain.csv')
X_test = pd.read_csv('xtest.csv')

X_train.head()
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	931	0.000000	0.75	0.461171	0.377048	1.0	1.0	0.333333	1.000000	1.0
1	657	0.000000	0.75	0.456066	0.399443	1.0	1.0	0.333333	0.333333	1.0
2	46	0.588235	0.75	0.394699	0.347082	1.0	1.0	0.000000	0.333333	1.0
3	1349	0.000000	0.75	0.388581	0.493677	1.0	1.0	0.666667	0.666667	1.0
4	56	0.000000	0.75	0.577658	0.402702	1.0	1.0	0.333333	0.333333	1.0

In [13]:
```
# target (log transformed)

y_train = X_train['SalePrice']
y_test = X_test['SalePrice']
```

In [14]:
```
# load the pre-selected features

features = pd.read_csv('selected_features.csv')
```

In [15]:
```
features
```

Out[15]:

	MSSubClass
0	MSZoning
1	Neighborhood
2	OverallQual
3	OverallCond
4	YearRemodAdd

	MSSubClass
5	RoofStyle
6	MasVnrType
7	BsmtQual
8	BsmtExposure
9	HeatingQC
10	CentralAir
11	1stFlrSF
12	GrLivArea
13	BsmtFullBath
14	KitchenQual
15	Fireplaces
16	FireplaceQu
17	GarageType
18	GarageFinish
19	GarageCars
20	PavedDrive

In [16]:
```
# We will add one additional feature to the ones we selected in the
type(features)
```

Out[16]:
```
pandas.core.frame.DataFrame
```

In [17]:
```
features = features['MSSubClass'].tolist()
features = features + ['LotFrontage']

# display final feature set
features
```

Out[17]:
```
['MSZoning',
 'Neighborhood',
 'OverallQual',
 'OverallCond',
 'YearRemodAdd',
 'RoofStyle',
 'MasVnrType',
 'BsmtQual',
 'BsmtExposure',
 'HeatingQC',
 'CentralAir',
 '1stFlrSF',
 'GrLivArea',
 'BsmtFullBath',
 'KitchenQual',
 'Fireplaces',
 'FireplaceQu',
 'GarageType',
 'GarageFinish',
```

```
'GarageCars',
'PavedDrive',
'LotFrontage']
```

In [18]:
```
# reduce the train and test set to the selected features

X_train = X_train[features]
X_test = X_test[features]
```

Regularised linear regression: Lasso

In [19]:
```
lin_model = Lasso(alpha=0.005, random_state=0)

lin_model.fit(X_train, y_train)
```

Out[19]:
```
Lasso(alpha=0.005, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=0,
      selection='cyclic', tol=0.0001, warm_start=False)
```

Evaluate Model

In [20]:
```
pred = lin_model.predict(X_train)

# determine mse and rmse
print('Training MSE: {}'.format(int(
    mean_squared_error(np.exp(y_train), np.exp(pred)))))
print('Training RMSE: {}'.format(int(
    sqrt(mean_squared_error(np.exp(y_train), np.exp(pred))))))
print('Training R2: {}'.format(
    r2_score(np.exp(y_train), np.exp(pred))))
print()

# make predictions for test set
pred = lin_model.predict(X_test)

# determine mse and rmse
print('Test MSE: {}'.format(int(
    mean_squared_error(np.exp(y_test), np.exp(pred)))))
print('Test RMSE: {}'.format(int(
    sqrt(mean_squared_error(np.exp(y_test), np.exp(pred))))))
print('Test R2: {}'.format(
    r2_score(np.exp(y_test), np.exp(pred))))
print()

print('Average House Price: ', int(np.exp(y_train).median()))
```

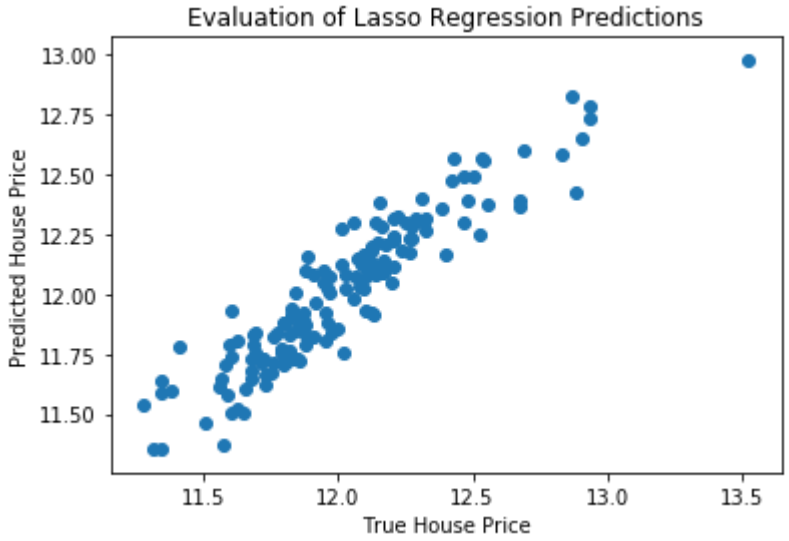
```
Training MSE: 1095464701
Training RMSE: 33097
Training R2: 0.8245524987165784

Test MSE: 1415749527
Test RMSE: 37626
Test R2: 0.7939863537248242

Average House Price:  163000
```

In [21]:
```
plt.scatter(y_test, lin_model.predict(X_test))
plt.xlabel('True House Price')
plt.ylabel('Predicted House Price')
plt.title('Evaluation of Lasso Regression Predictions')
```

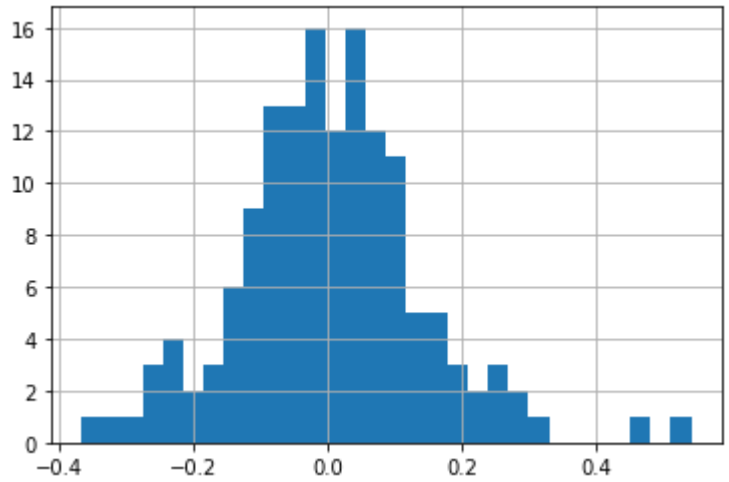
Out[21]: Text(0.5, 1.0, 'Evaluation of Lasso Regression Predictions')



In [22]:
```
# distribution of the errors:

errors = y_test - lin_model.predict(X_test)
errors.hist(bins=30)
```

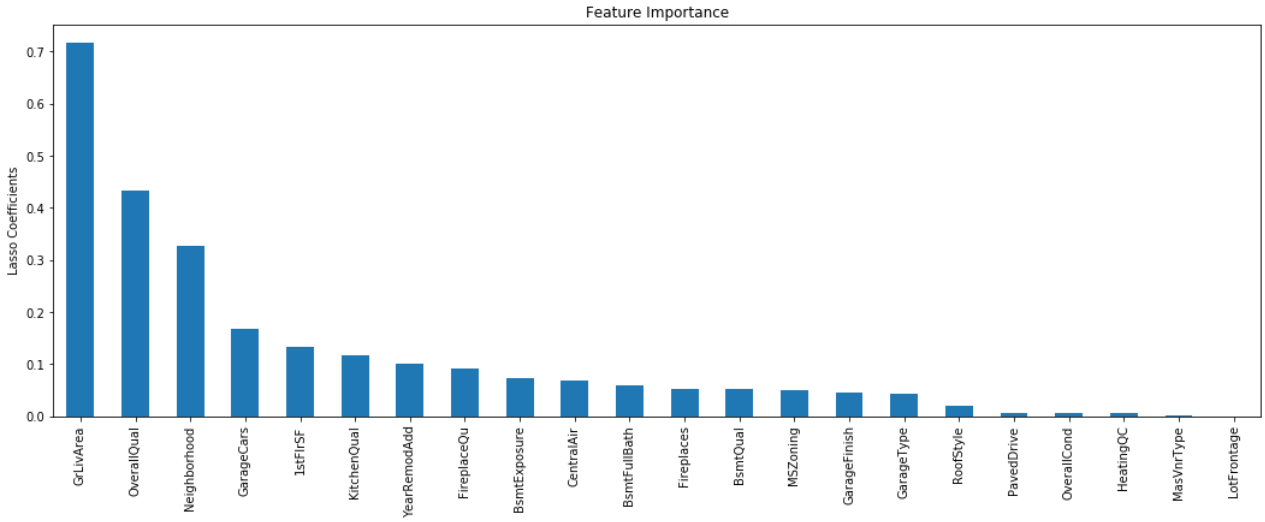
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3e8db00c8>



Checking for Feature Importance:

In [23]:
```
importance = pd.Series(np.abs(lin_model.coef_.ravel()))
importance.index = features
importance.sort_values(inplace=True, ascending=False)
importance.plot.bar(figsize=(18,6))
plt.ylabel('Lasso Coefficients')
plt.title('Feature Importance')
```

Out[23]: Text(0.5, 1.0, 'Feature Importance')


Feature Importance
Lasso Coefficients
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
GrLivArea
OverallQual
Neighborhood
GarageCars
1stFlrSF
KitchenQual
YearRemodAdd
FireplaceQu
BsmtExposure
CentralAir
BsmtFullBath
Fireplaces
BsmtQual
MSZoning
GarageFinish
GarageType
RoofStyle
PavedDrive
OverallCond
HeatingQC
MasVnrType
LotFrontage

In [ ]: