Regularization in Linear Models

The purpose of this notebook is to explore how regularization works in Linear models. Since the data does not have a lot of features, regularization might not improve the model. In this notebook, the purpose is more to focus on how L1 and L2 regularization works. Data used in this notebook is cleaned Boston Airbnb listing price data from a personal project Boston Airbnb Price Estimator. More details about exploratary analysis can be found there. In addition, a few other boosted models and neural networks are tried with certain imoprovement of r2 score on certain models.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, RidgeCV, Lasso, LassoCV, El
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
import pickle
plt.style.use("seaborn-colorblind")
%matplotlib inline

# view the original dataset
listings = pd.read_csv('./data/listings.csv')
pd.set_option('display.max_columns', 20)
print (listings.shape)
listings.head()
```

```
(3585, 95)
```

Out[1]:

	id	listing_url	scrape_id	last_scraped	name	summary
0	12147973	https://www.airbnb.com/rooms/12147973	20160906204935	2016-09-07	Sunny Bungalow in the City	Cozy, sunny, family home. Master bedroom high...
1	3075044	https://www.airbnb.com/rooms/3075044	20160906204935	2016-09-07	Charming room in pet friendly apt	Charming and quiet room in a second floor 1910...
2	6976	https://www.airbnb.com/rooms/6976	20160906204935	2016-09-07	Mexican Folk Art Haven in Boston	Come stay with a friendly, middle-aged guy in ...

	id	listing_url	scrape_id	last_scraped	name	summary
3	1436513	https://www.airbnb.com/rooms/1436513	20160906204935	2016-09-07	Spacious Sunny Bedroom Suite in Historic Home	Come experience the comforts of home away from...
4	7651065	https://www.airbnb.com/rooms/7651065	20160906204935	2016-09-07	Come Home to Boston	My comfy, clean and relaxing home is one block...

5 rows × 95 columns

Import data

In [2]:

```
# features
# the first 50 features are categorical while the 51-68 are the top 18 numerical feat
x = pd.read_pickle('./x.pkl')
# price
y = pd.read_pickle('./y.pkl')

test_size = 0.3
random_state = 42
# split the data into train and test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = test_size,
                                                    random_state = random_state)
```

In [3]:

```
x_train.head()
```

Out[3]:

	room_type_Private room	cancellation_policy_super_strict_30	require_guest_phone_verification_t	neigh
3547	1	0	0	
1596	1	0	0	
2449	0	0	0	
2476	0	0	0	
1961	0	0	0	

5 rows × 68 columns

In [4]:

```
def model_evaluation(model):
    model.fit(x_train, y_train)
    y_test_preds = model.predict(x_test)
    y_train_preds = model.predict(x_train)
```

```
    #r2 value
    r2_scores_test = r2_score(y_test, y_test_preds)
    r2_scores_train = r2_score(y_train, y_train_preds)
    mse_test = mean_squared_error(y_test, y_test_preds)
    mse_train = mean_squared_error(y_train, y_train_preds)

    print ('r2_score_test: {0:.4f}'.format(r2_scores_test))
    print ('r2_score_train: {0:.4f}'.format(r2_scores_train))
    print ('mse_test: {0:.2f}'.format(mse_test))
    print ('mse_train: {0:.2f}'.format(mse_train))

    fig = plt.figure(figsize =(10, 4))
    fig.subplots_adjust(hspace=0.4, wspace=0.4)
    ax = plt.axes(aspect = 'equal')
    plt.subplot(121)
    plt.title('Scatter comparison', fontsize=14)
    plt.scatter(y_test, y_test_preds, color='blue')
    plt.xlabel('Real Price', fontsize=14)
    plt.ylabel('Predicted Price', fontsize=14)
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)

    plt.subplot(122)
    sns.distplot(y_test_preds, hist=False,
                 kde_kws={'color': 'b', 'lw': 2, 'label': 'Predicted price'})
    sns.distplot(y_test, hist=False,
                 kde_kws={'color': 'r', 'lw': 2, 'label': 'Real price'})
    plt.title('Distribution comparison', fontsize=14)
    plt.ylabel('Probablity', fontsize=12)
    plt.xlabel('Price (USD)', fontsize=12)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.legend(['Predicted price', 'Real price'], prop={"size":12})
    plt.show()

    return model.coef_
```

In [5]:

```
def feature_importance(weights):
    features = x_train.columns
    feature_importance = pd.DataFrame({'feature': features, 'weight':weights})
    sorted_feature_importance = feature_importance.sort_values('weight', ascending=Fa
    fig = plt.figure(figsize=(12,6))
    plt.bar(range(len(features)), sorted_feature_importance['weight'], color='b')
    plt.title('feature importance', fontsize=14)
    plt.show()
    pass
```

LinearRegression modeling

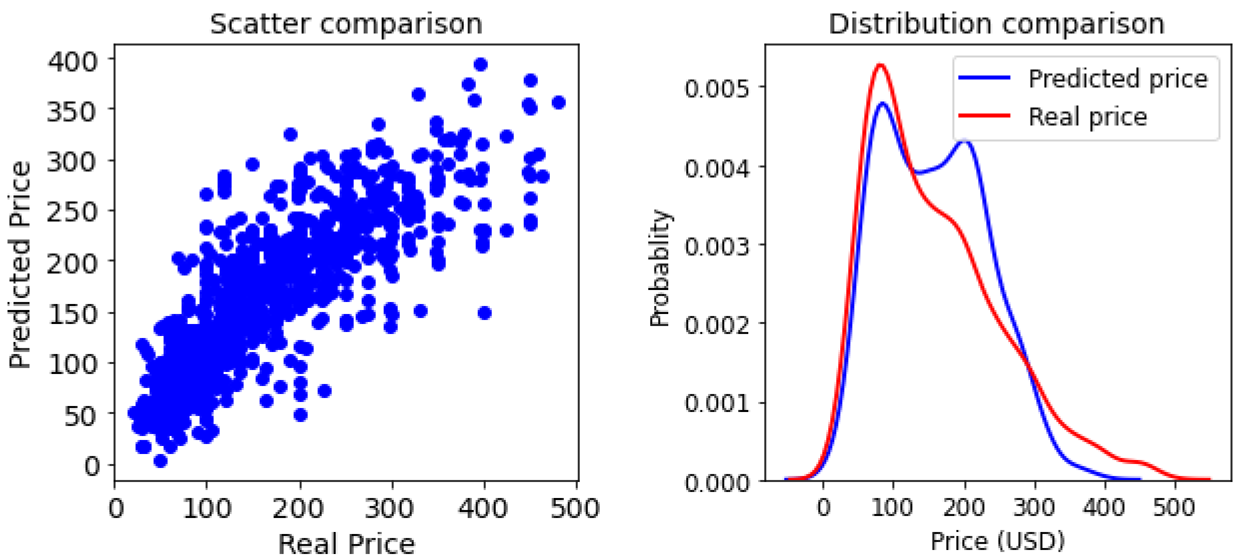
- In this section, LinearRegression from sklearn is used to fit the model. Through minimizing the residual sum of squares between the observed targets in the dataset, the weighting coeffcients are obtained for each selected variable and the targets are predicted by the linear approximation.

- Note: 18 top corr numerical variables and 51 top corr categorical variebles are used. More details of feature selection and engineering can be found in Boston Airbnb Price Estimator.

In [6]:

```
# LinearRegression
lr = LinearRegression()
weights = model_evaluation(lr)
```

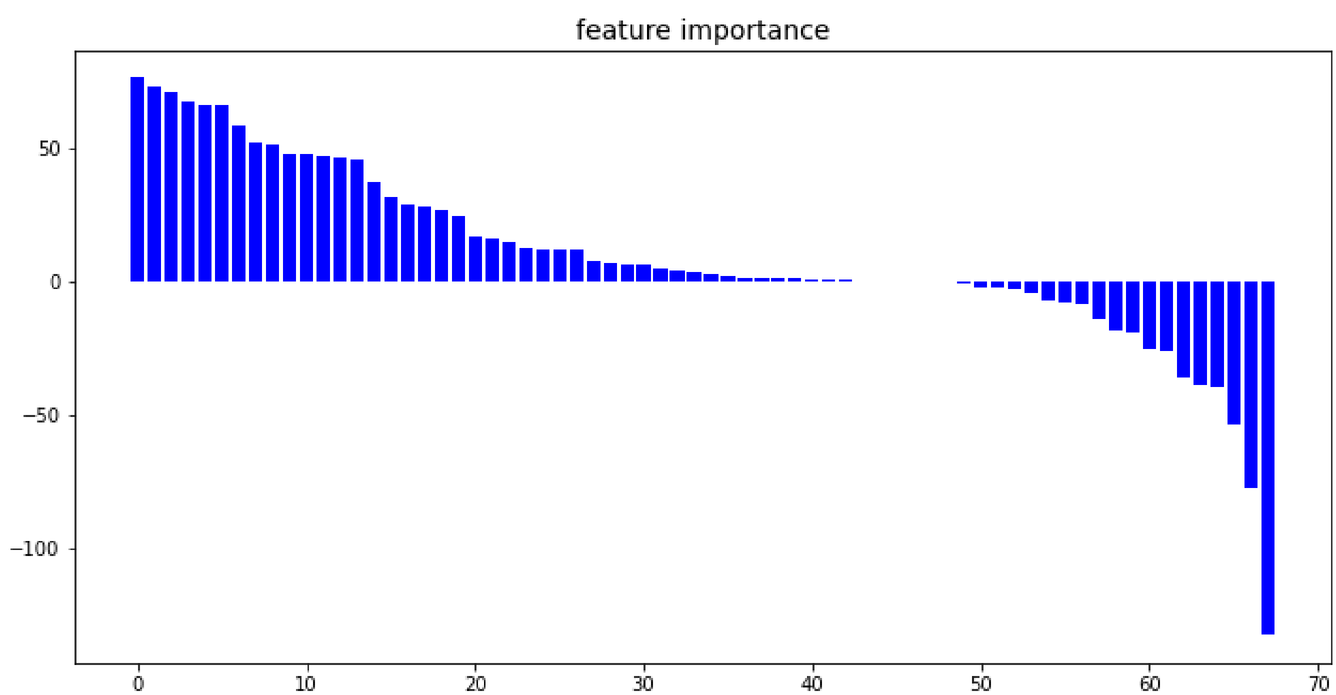
```
r2_score_test: 0.6821
r2_score_train: 0.6778
mse_test: 2735.74
mse_train: 2615.36
```



Visualize the features and weights

In [7]:

```
feature_importance(weights)
```



Ridge Regression: RidgeCV

- Explore that if RidgeCV can improve the estimation accuracy since the data contains variables that highly correlated. Through L2 regularization (adding bias), the variance of the estimates can be reduced.
- Thorugh observing the results, it turns out that the r2 score for the test data has not been improved.

In [8]:

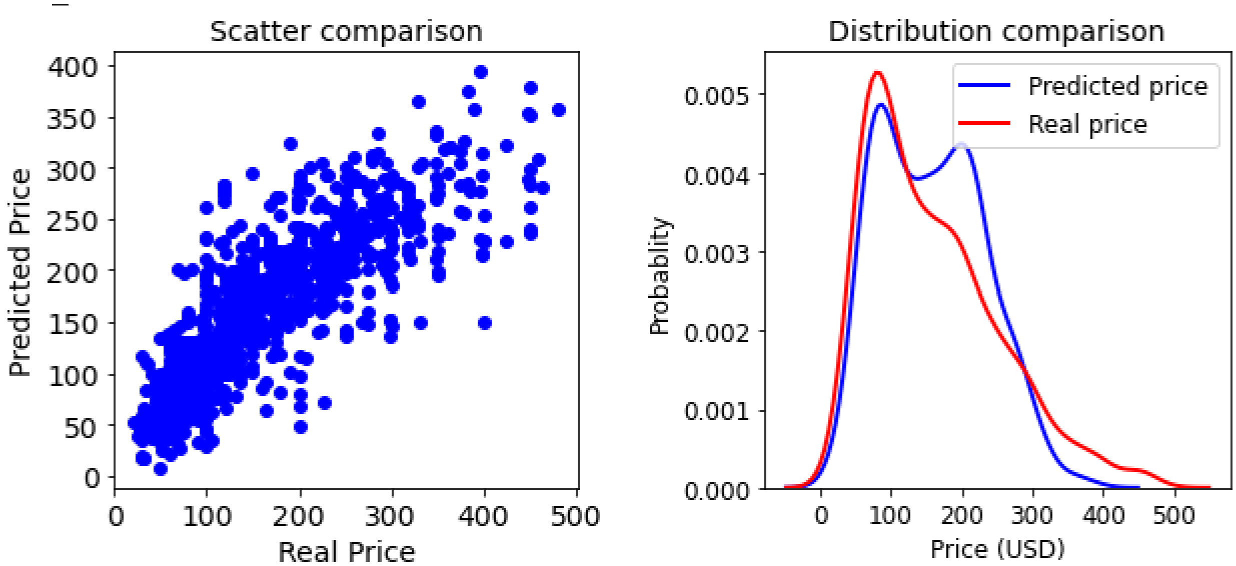
```
# coss-validation test
alphas = np.linspace(0.0001,0.1,200)
ridge_cv = RidgeCV(alphas=alphas, normalize=True, cv = 5)
# , scoring="neg_mean_squared_error")
ridge_cv.fit(x_train, y_train)
# find the best alpha
alpha = ridge_cv.alpha_
print ('best alpha:')
print (alpha)
```

```
best alpha:
0.015662311557788945
```

In [9]:

```
ridge = Ridge(alpha = alpha, normalize=True)
ridge_weights = model_evaluation(ridge)
```

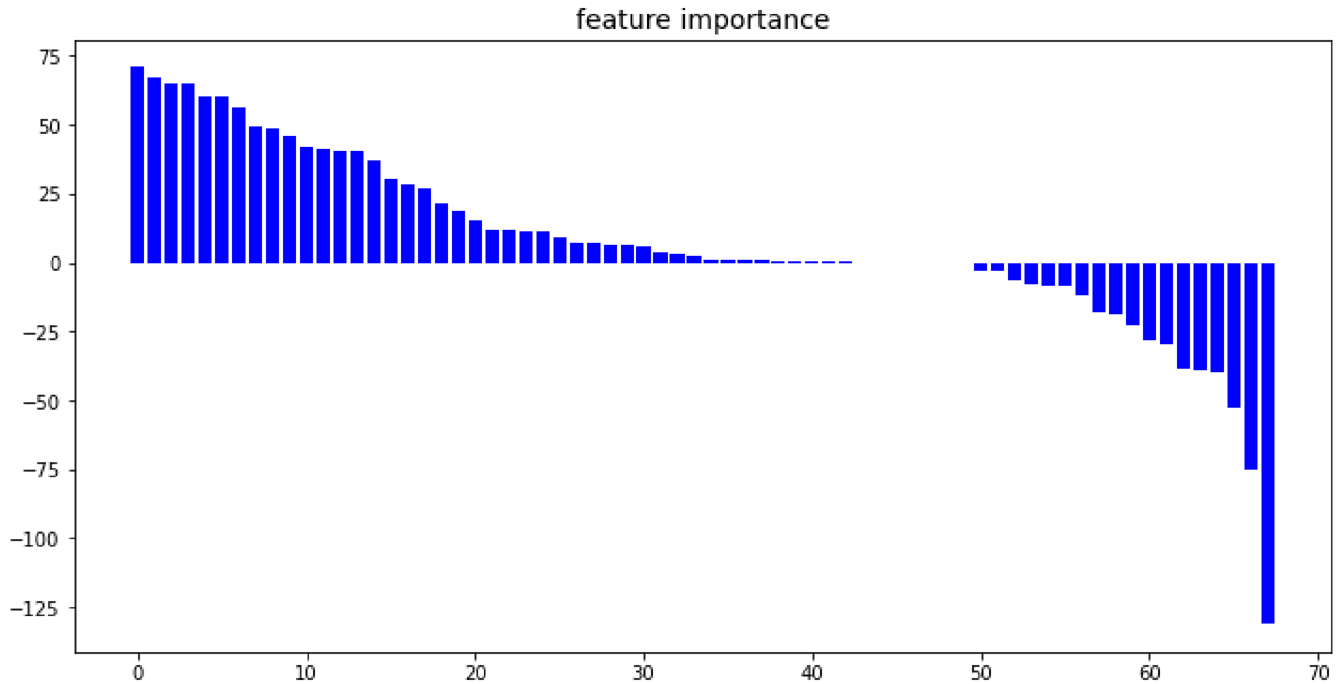
```
r2_score_test: 0.6820
r2_score_train: 0.6775
mse_test: 2736.85
mse_train: 2617.74
```



Visualize the features and weights

In [10]:

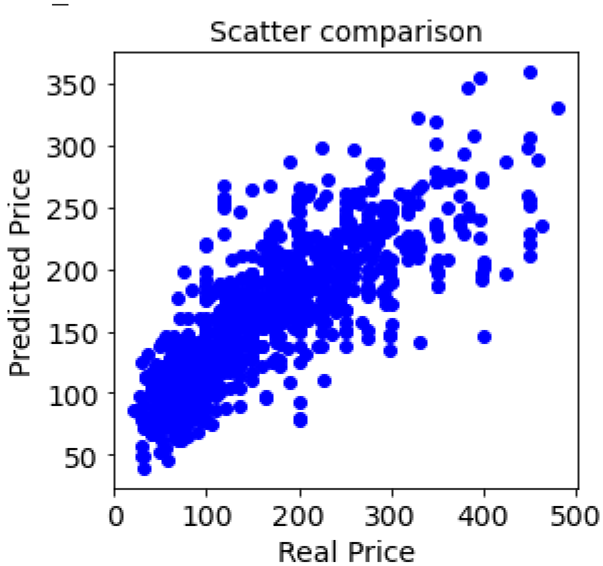
```
feature_importance(ridge_weights)
```

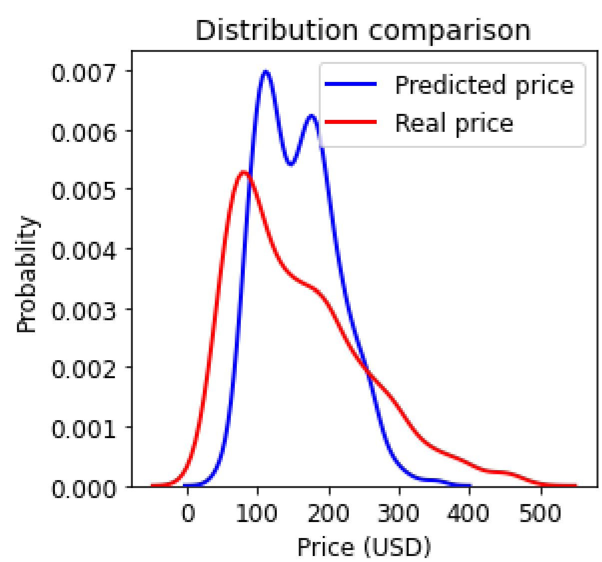


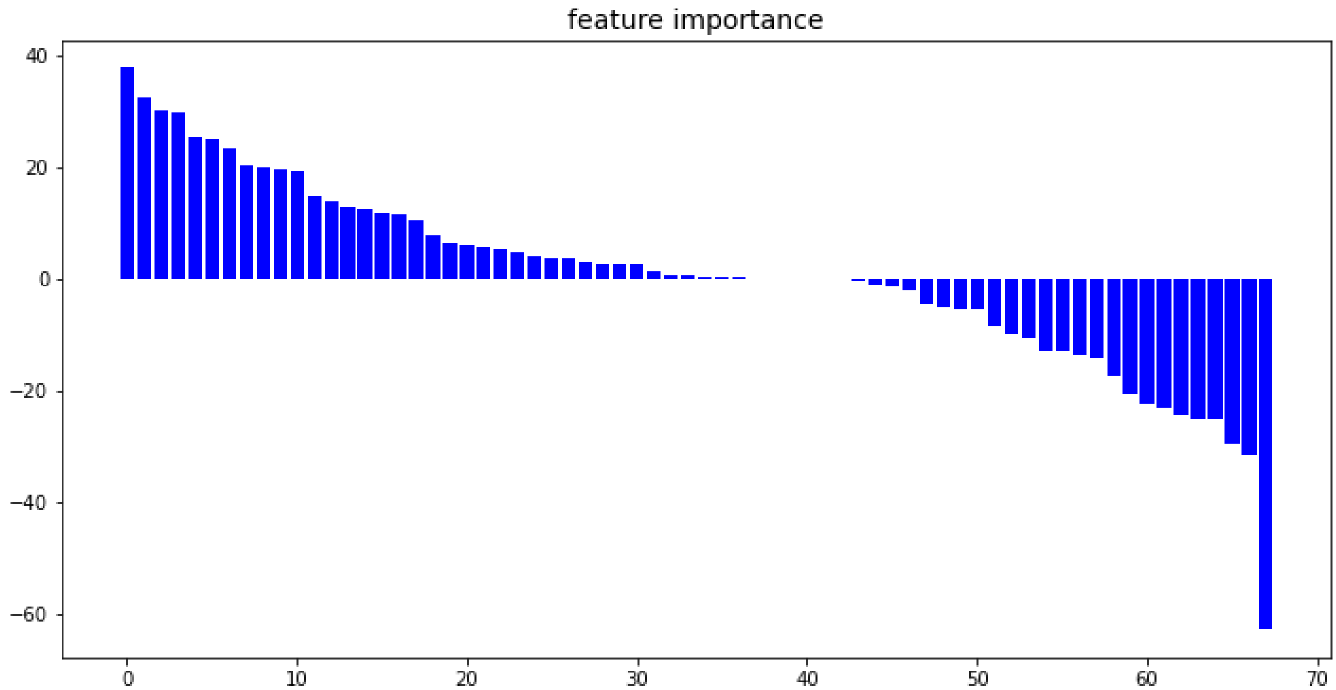
In [11]:

```
## Try a larger alpha to see the L2 regularization effect
ridge = Ridge(alpha = 1, normalize=True)
ridge_weights = model_evaluation(ridge)
feature_importance(ridge_weights)
```

```
r2_score_test: 0.6088
r2_score_train: 0.6059
mse_test: 3366.94
mse_train: 3199.33
```







Lasso Regression: LassoCV

- Explore that if LassoCV can improve the estimation accuracy since the data contains variables that highly correlated. Through L1 regularization (adding bias), the variance of the estimates can be reduced.
- Thorugh observing the results, it turns out that the r2 score for the test data has not been improved.

In [12]:

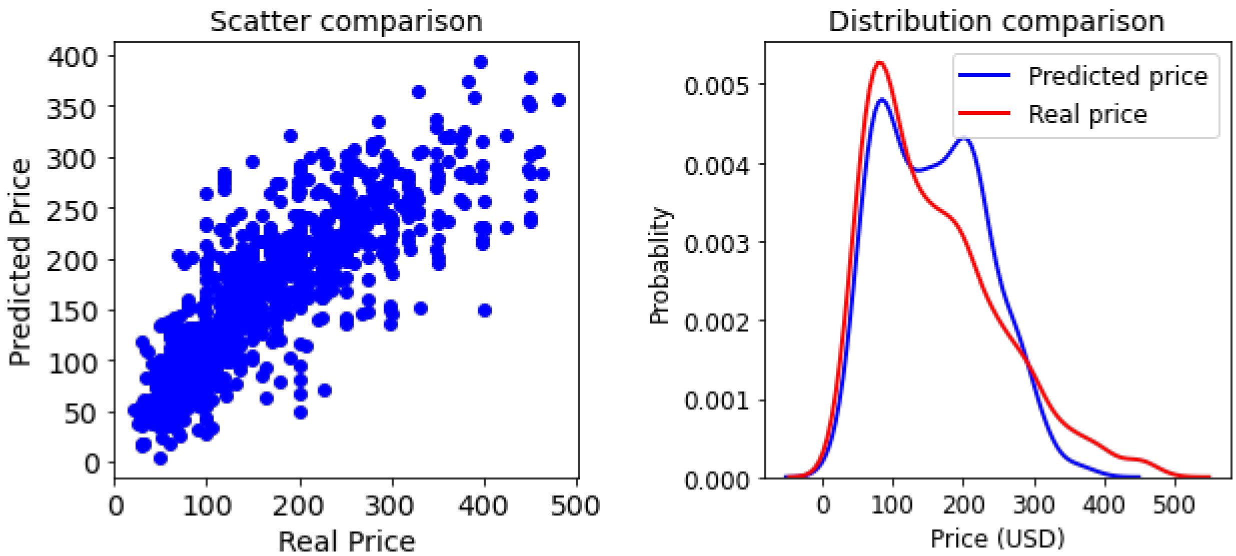
```
# coss-validation test
lasso_cv = LassoCV(normalize=True, cv = 5)
# , scoring="neg_mean_squared_error")
lasso_cv.fit(x_train, y_train)
# find the best alpha
alpha = lasso_cv.alpha_
print ('best alpha:')
print (alpha)
```

```
best alpha:
0.006191460063933275
```

In [13]:

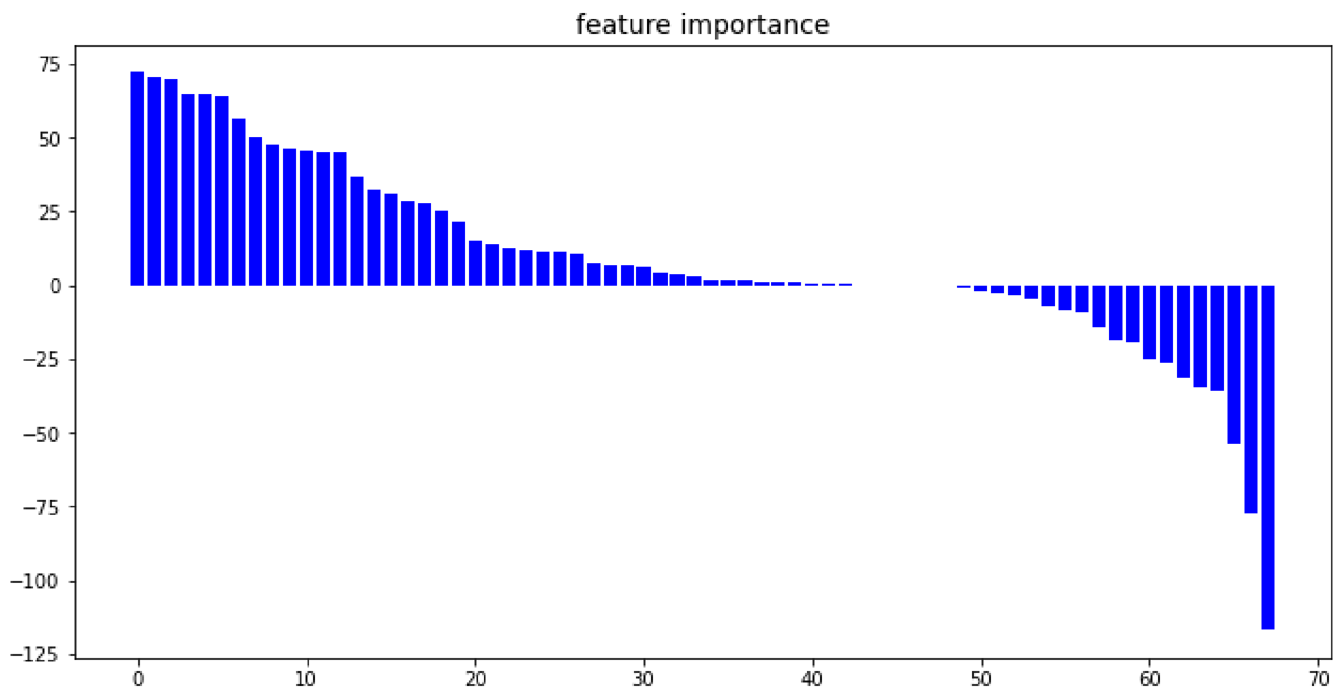
```
lasso = Lasso(alpha = alpha, normalize=False)
lasso_weights = model_evaluation(lasso)
```

```
r2_score_test: 0.6822
r2_score_train: 0.6778
mse_test: 2735.23
mse_train: 2615.85
```



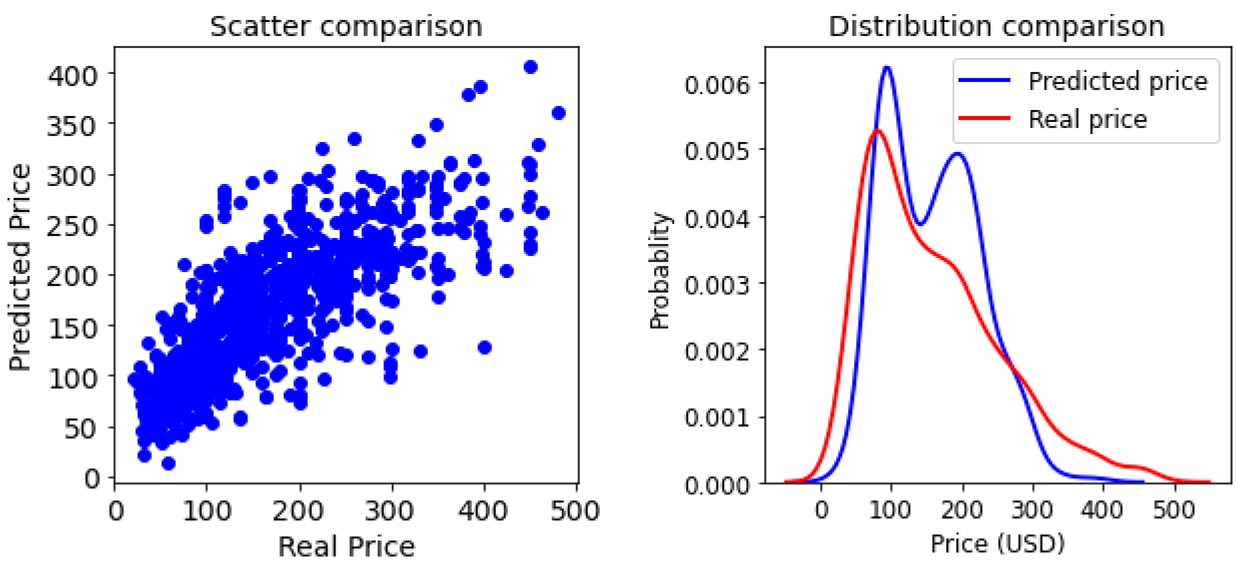
Visualize the features and weights

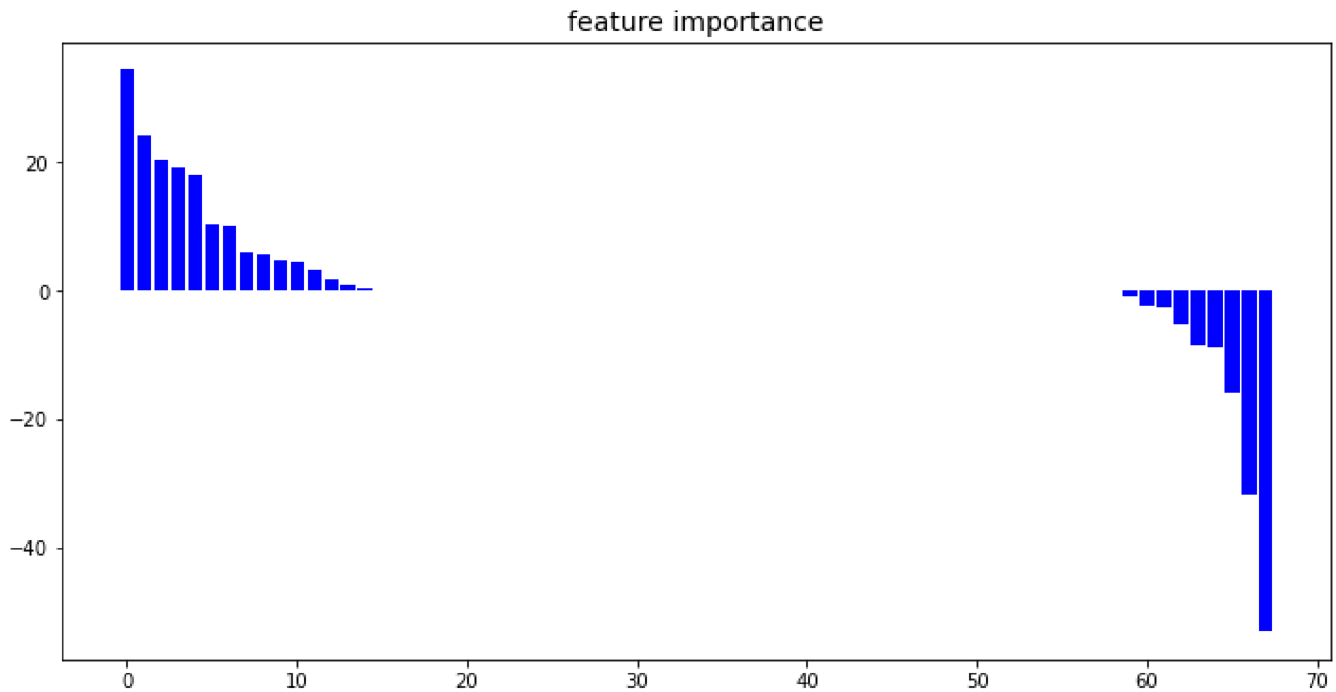
In [14]:
```
feature_importance(lasso_weights)
```



In [15]:
```
# try a larger alpah to see the L1 regularization effect
lasso = Lasso(alpha = 1, normalize=False)
lasso_weights = model_evaluation(lasso)
feature_importance(lasso_weights)
```

```
r2_score_test: 0.6143
r2_score_train: 0.6094
mse_test: 3319.33
mse_train: 3171.18
```





ElasticNet Regression: ElasticNetCV

Combination of L1 and L2 regularization

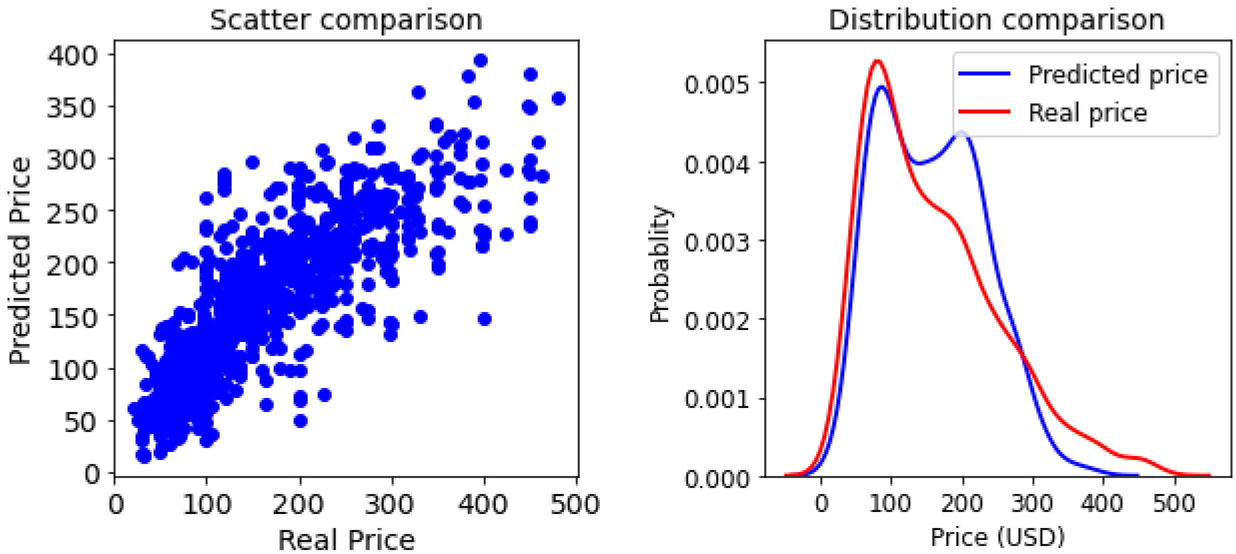
In [16]:
```
# coss-validation test
EN_cv = ElasticNetCV(l1_ratio = 0.2, normalize=True, cv = 5)
# , scoring="neg_mean_squared_error")
EN_cv.fit(x_train, y_train)
# find the best alpha
alpha = EN_cv.alpha_
print ('best alpha:')
print (alpha)
```

```
best alpha:
0.005409876149618 39
```

In [17]:

```
EN = ElasticNet(alpha = alpha, l1_ratio=0.5, normalize=False)
EN_weights = model_evaluation(EN)
```

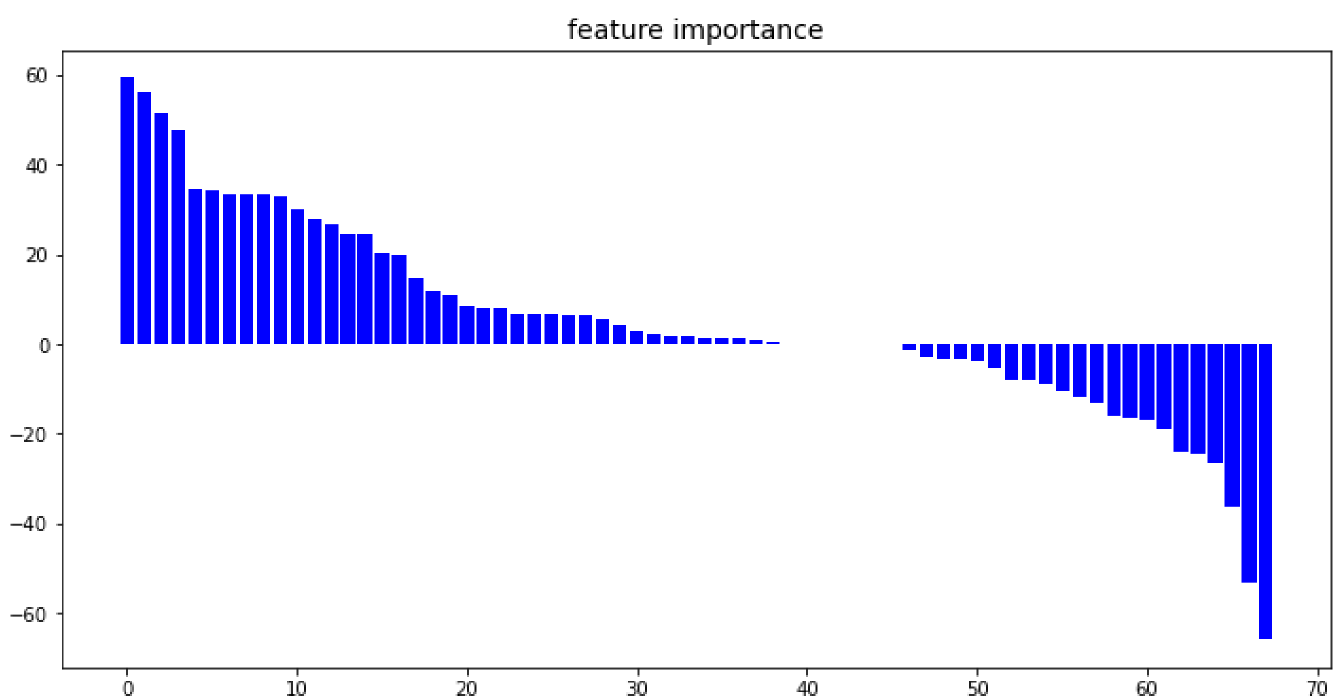
```
r2_score_test: 0.6796
r2_score_train: 0.6733
mse_test: 2757.57
mse_train: 2652.21
```



Visualize the features and weights

In [18]:

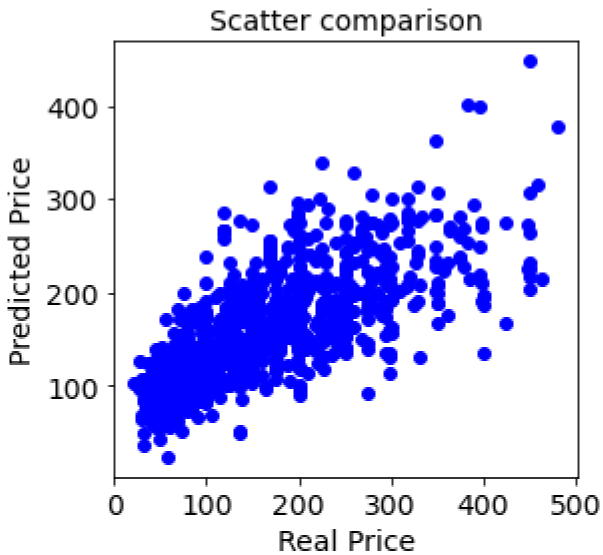
```
feature_importance(EN_weights)
```

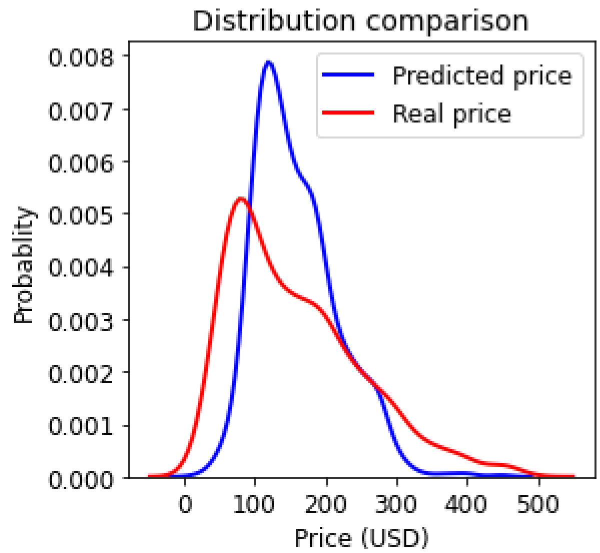


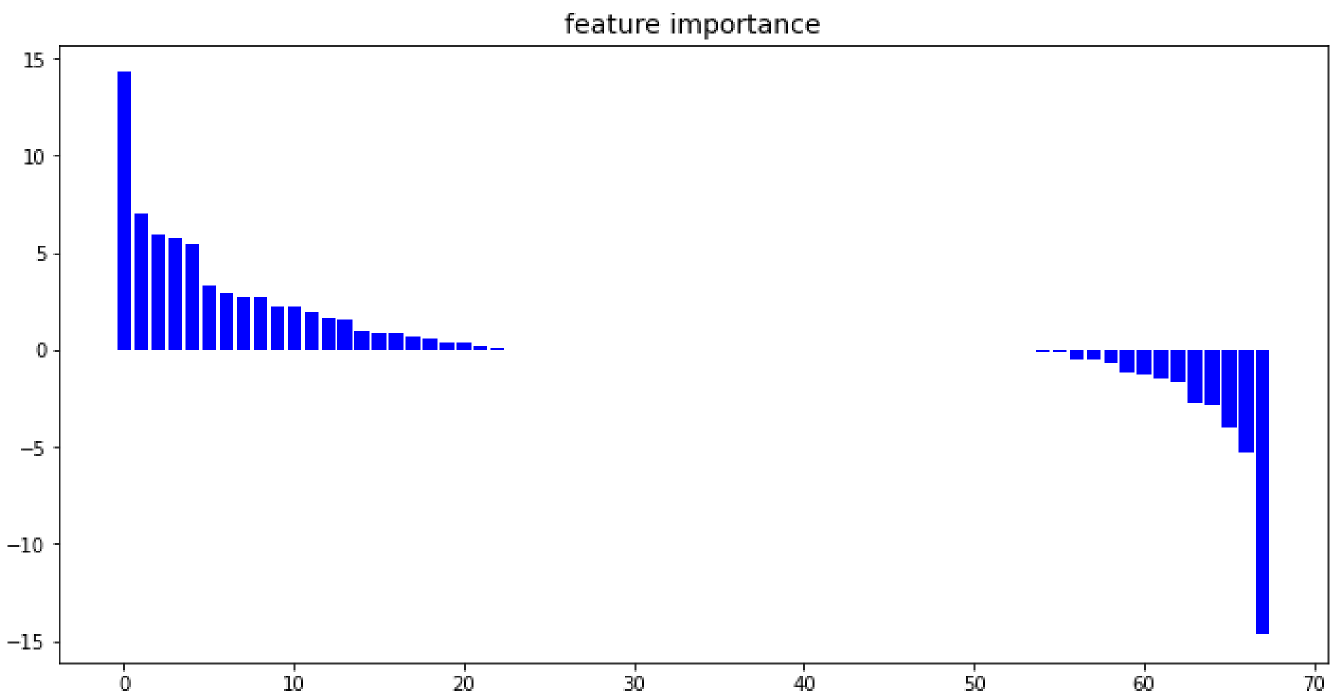
In [19]:

```
EN = ElasticNet(alpha = 1, l1_ratio=0.5, normalize=False)
EN_weights = model_evaluation(EN)
feature_importance(EN_weights)
```

```
r2_score_test: 0.5094
r2_score_train: 0.5101
mse_test: 4222.00
mse_train: 3976.82
```







Discussion

- Linear Regression model results in 0.6821 r2 score on the training dataset and 0.6778 on the test dataset. Since the score on the unseen test dataset is close the training score. The model should be fine without overfitting problem.
- With the aid of RidgeCV finding the best alpha 0.0152, Ridge Regression results in 0.6704 r2 score on the training data and 0.6652 on the test dataset. The rseults are close to Linear regression since a very small penalty is added. To illustrate how L2 regularization works, a larger

alpha is passed to the model. In the feature importance bar plot, we can observe that all the feature magnitue decreased.

- With the aid of LassoCV finding the best alpha 0.0062, Lasso Regression results in 0.6822 r2 score on the training data and 0.6778 on the test dataset. The rseults are almost the same to Linear regression since the penalty weight is only 0.0062. To illustrate how L1 regularization works, a larger alpha is passed to the model. In the feature importance bar plot, we can observe that features with small weights are set to zero.
- With the aid of ElasticNetCV finding the best alpha 0.0054 when l1_ratio is 0.2, ElasticNet Regression results in 0.6796 r2 score on the training data and 0.6733 on the test dataset. The rseults are almost the same to Linear regression since the penalty weights are small. To illustrate how the regularization (L1 and L2) works, a larger alpha is passed to the model. In the feature importance bar plot, we can observe that not only feature magnitude decreases but also features with small weights are set to zero.

In [ ]:

```
```