1、导入所需的库

In [3]:

```
#基础
import numpy as np
import pandas as pd
import os

#绘图
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#模型
from sklearn.linear_model import Lasso, LassoCV, ElasticNet, ElasticNetCV, Ridge
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, S
from mlxtend.regressor import StackingCVRegressor
from sklearn.svm import SVR
import lightgbm as lgb
import xgboost as xgb

#模型相关
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import mean_squared_error

#忽略警告
import warnings
def ignore_warn(*args, **kwargs):
    pass
warnings.warn = ignore_warn
```

2、读取数据集，对正偏斜的目标值取对数处理

In [4]:

```
train = pd.read_csv('train_data.csv')
test = pd.read_csv('test_data.csv')
print('The shape of training data:', train.shape)
print('The shape of testing data:', test.shape)
```

```
The shape of training data: (1458, 160)
The shape of testing data: (1459, 159)
```

In [5]:

```
train.isnull().sum().any()
```

Out[5]:

```
np.False_
```

In [6]:

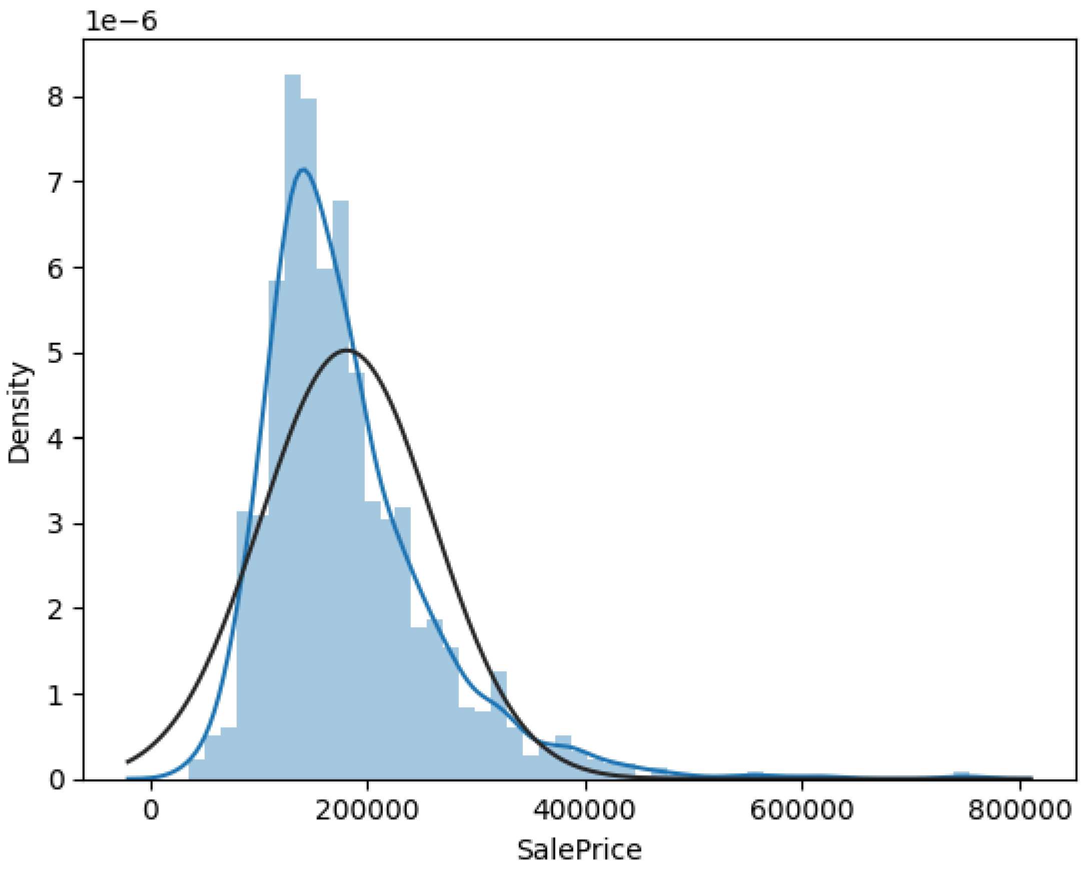
```
from scipy.stats import skew, kurtosis, norm

y = train['SalePrice']
print('Skewness of target:', y.skew())
print('kurtosis of target:', y.kurtosis())
sns.distplot(y, fit=norm)
```

```
Skewness of target: 1.8812964895244009
kurtosis of target: 6.523066888485879
```

Out[6]:

```
<Axes: xlabel='SalePrice', ylabel='Density'>
```

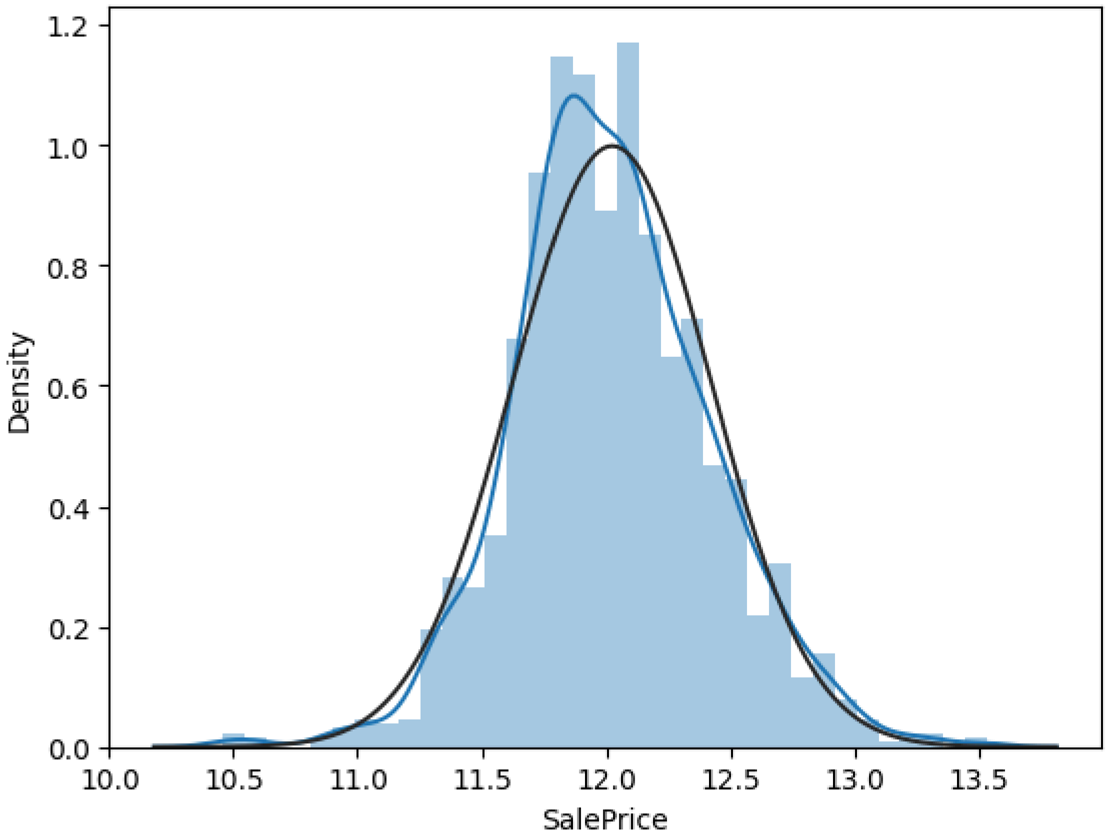


可以看出，未处理的目标值明显右偏，不满足正态分布

In [7]:

```
y = np.log1p(y)
print('Skewness of target:', y.skew())
print('kurtosis of target:', y.kurtosis())
sns.distplot(y, fit=norm);
```

```
Skewness of target: 0.12157976050304882
kurtosis of target: 0.8047507917418972
```



处理后的目标值接近正态分布

In [8]:
```python
train = train.drop('SalePrice', axis=1)

#检查训练集与测试集的维度是否一致
print('The shape of training data:', train.shape)
print('The length of y:', len(y))
print('The shape of testing data:', test.shape)
```

```
The shape of training data: (1458, 159)
The length of y: 1458
The shape of testing data: (1459, 159)
```

In [9]:
```python
y.isnull().sum()
```

Out[9]:
```
np.int64(0)
```

3、定义交叉验证策略及评估方法

In [10]:
```python
#采用十折交叉验证
n_folds = 10

def rmse_cv(model):
  kf = KFold(n_folds, shuffle=True, random_state=20)
  rmse = np.sqrt(-cross_val_score(model, train.values, y, scoring='neg_mean_squa
  return(rmse)
```

4、单个模型参数设置

采用六个模型：

- Lasso
- ElasticNet
- Ridge
- Gradient Boosting
- LightGBM
- XGBoost

In [11]:

```
#Lasso
lasso_alpha = [0.00005, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02,
lasso = make_pipeline(RobustScaler(), LassoCV(alphas=lasso_alpha, random_state=2
```

In [12]:

```
#ElasticNet
enet_beta = [0.1, 0.2, 0.5, 0.6, 0.8, 0.9]
enet_alpha = [0.00005, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01]
ENet = make_pipeline(RobustScaler(), ElasticNetCV(l1_ratio=enet_beta, alphas=ene
```

In [13]:

```
#Ridge
rid_alpha = [0.00005, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0
rid = make_pipeline(RobustScaler(), RidgeCV(alphas=rid_alpha))
```

In [14]:

```
#Gradient Boosting
gbr_params = {'loss': 'huber',
      'criterion': 'friedman_mse',
      'learning_rate': 0.1,
      'n_estimators': 600,
      'max_depth': 4,
      'subsample': 0.6,
      'min_samples_split': 20,
      'min_samples_leaf': 5,
      'max_features': 0.6,
      'random_state': 32,
      'alpha': 0.5}
gbr = GradientBoostingRegressor(**gbr_params)
```

In [15]:

```
#LightGBM
lgbr_params = {'learning_rate': 0.01,
      'n_estimators': 1850,
      'max_depth': 6,
      'num_leaves': 40,
      'subsample': 0.6,
      'colsample_bytree': 0.6,
      'min_child_weight': 0.001,
      'min_child_samples': 30,
      'min_gain_to_split': 0.1,  # 增加最小增益分裂阈值
      'random_state': 42,
      'reg_alpha': 0,
      'reg_lambda': 0.05}
lgbr = lgb.LGBMRegressor(**lgbr_params)
```

In [16]:

```
#XGBoost
xgbr_params = { 'objective': 'reg:squarederror',  # 使用推荐的新目标函数
      'learning_rate': 0.01,
      'n_estimators': 3000,
      'max_depth': 5,
      'subsample': 0.6,
      'colsample_bytree': 0.7,
      'min_child_weight': 3,
      'seed': 52,
      'gamma': 0,
      'reg_alpha': 0,
      'reg_lambda': 1}
xgbr = xgb.XGBRegressor(**xgbr_params)
```

5、单个模型评估

In [ ]:

```
models_name = ['Lasso', 'ElasticNet', 'Ridge', 'Gradient Boosting', 'LightGBM',
models = [lasso, ENet, rid, gbr, lgbr, xgbr]
for i, model in enumerate(models):
  score = rmse_cv(model)
  print('{} score: {}({})'.format(models_name[i], score.mean(), score.std()))
```

6、设置Stacking模型参数

In [18]:

```
stack_model = StackingCVRegressor(regressors=(lasso, ENet, rid, gbr, lgbr, xgbr)
```

7、在整个训练集上训练各模型

In [19]:

```
#Lasso
lasso_trained = lasso.fit(np.array(train), np.array(y))
```

In [20]:

```
#ElasticNet
ENet_trained = ENet.fit(np.array(train), np.array(y))
```

In [21]:

```
#Ridge
rid_trained = rid.fit(np.array(train), np.array(y))
```

In [22]:

```
#Gradient Boosting
gbr_trained = gbr.fit(np.array(train), np.array(y))
```

In [ ]:

```
#LightGBM
lgbr_trained = lgbr.fit(np.array(train), np.array(y))
```

In [24]:

```
#XGBoost
xgbr_trained = xgbr.fit(np.array(train), np.array(y))
```

In [ ]:

```
#Stacking
stack_model_trained = stack_model.fit(np.array(train), np.array(y))
```

8、评估各个模型在完整训练集上的表现

In [26]:

```
def rmse(y, y_preds):
  return np.sqrt(mean_squared_error(y, y_preds))
```

In [27]:

```
models.append(stack_model)
models_name.append('Stacking_model')
for i, model in enumerate(models):
  y_preds = model.predict(np.array(train))
  model_score = rmse(y, y_preds)
  print('RMSE of {}: {}'.format(models_name[i], model_score))
```

```
RMSE of Lasso: 0.10145681166218012
RMSE of ElasticNet: 0.10118261114179593
RMSE of Ridge: 0.09940646161993891
RMSE of Gradient Boosting: 0.0681676326955603
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
RMSE of LightGBM: 0.09531964839251936
RMSE of XGBoost: 0.019687837662685326
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
RMSE of Stacking_model: 0.09568093138959977
RMSE of Stacking_model: 0.09568093138959977
```

9、提交各个模型的预测结果

In [28]:

```
# 创建一个从 1461 开始的 ID 列
if not os.path.exists('sample_submission.csv'):
    print("sample_submission.csv 文件不存在，正在创建占位文件...")
    sample_submission = pd.DataFrame({'Id': range(1461, 1461 + len(test))})
    sample_submission.to_csv('sample_submission.csv', index=False)
else:
    sample_submission = pd.read_csv('sample_submission.csv')

for i, model in enumerate(models):
  preds = model.predict(np.array(test))
  submission = pd.DataFrame({'Id': sample_submission['Id'], 'SalePrice': np.expm
  submission.to_csv('House_Price_submission_'+models_name[i]+'_optimation.csv',
  print('{} finished.'.format(models_name[i]))
```

```
Lasso finished.
ElasticNet finished.
Ridge finished.
Gradient Boosting finished.
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
LightGBM finished.
XGBoost finished.
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
Stacking_model finished.
```

10、均值融合

In [29]:

```
preds_in_train = np.zeros((len(y), len(models)))
for i, model in enumerate(models):
  preds_in_train[:, i] = model.predict(np.array(train))
average_preds_in_train = preds_in_train.mean(axis=1)
```

```
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
```

In [30]:

```
average_score = rmse(y, average_preds_in_train)
print('RMSE of average model on training data:', average_score)
```

```
RMSE of average model on training data: 0.07723373327552983
```

In [31]:

```
#提交均值融合预测结果
preds_in_test = np.zeros((len(test), len(models)))
for i, model in enumerate(models):
  preds_in_test[:, i] = model.predict(np.array(test))
average_preds_in_test = preds_in_test.mean(axis=1)

average_submission = pd.DataFrame({'Id': sample_submission['Id'], 'SalePrice': n
average_submission.to_csv('House_Price_submission_average_model_optimation.csv',
```

```
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
[LightGBM] [Warning] min_gain_to_split is set=0.1, min_split_gain=0.0 will be ign
ored. Current value: min_gain_to_split=0.1
```

11、权值融合

In [32]:

```
model_weights = [0.15, 0.12, 0.08, 0.08, 0.12, 0.15, 0.3]
weight_preds_in_train = np.matmul(preds_in_train, model_weights)

weight_score = rmse(y, weight_preds_in_train)
print('RMSE of weight model on training data:', weight_score)
```

```
RMSE of weight model on training data: 0.07878132253994066
```

In [33]:

```
#提交权值融合预测结果
weight_preds_in_test = np.matmul(preds_in_test, model_weights)

weight_submission = pd.DataFrame({'Id': sample_submission['Id'], 'SalePrice': np
weight_submission.to_csv('House_Price_submission_weight_model_optimation.csv', i
```

12、保存预测结果

In [34]:

```
#保存训练集上的预测结果
train_prediction = pd.DataFrame(preds_in_train, columns=models_name)
train_prediction.head()
```

Out[34]:

	Lasso	ElasticNet	Ridge	Gradient Boosting	LightGBM	XGBoost	Stacking_mo
0	12.241544	12.241702	12.243044	12.243320	12.224402	12.244552	12.249
1	12.172755	12.175224	12.192781	12.111687	12.040883	12.109303	12.145
2	12.250116	12.249458	12.246785	12.292966	12.275438	12.296574	12.271
3	12.051019	12.051005	12.040579	11.885431	12.019457	11.863003	11.963
4	12.560980	12.560619	12.557660	12.517344	12.517531	12.460670	12.531

In [35]:

```
train_prediction.shape
```

Out[35]:

```
(1458, 7)
```

In [36]:

```
train_prediction.to_csv('train_prediction_of_7_models.csv', index=False)
```

In [37]:

```
#保存测试集上的预测结果
test_prediction = pd.DataFrame(preds_in_test, columns=models_name)
test_prediction.head()
```

Out[37]:

	Lasso	ElasticNet	Ridge	Gradient Boosting	LightGBM	XGBoost	Stacking_mo
0	11.674492	11.677087	11.695130	11.727828	11.694880	11.740774	11.7029
1	11.895064	11.887805	11.887107	12.027486	11.977535	12.002156	11.8562
2	12.078785	12.079251	12.079700	12.147636	12.101579	12.157706	12.0852
3	12.186415	12.186265	12.184200	12.168408	12.131378	12.194018	12.1964
4	12.197597	12.198681	12.207519	12.173135	12.128827	12.128562	12.1827

In [38]:

```
test_prediction.shape
```

Out[38]:

```
(1459, 7)
```

In [39]:

```
test_prediction.to_csv('test_prediction_of_7_models.csv', index=False)
```

In [ ]: