

# 簡介

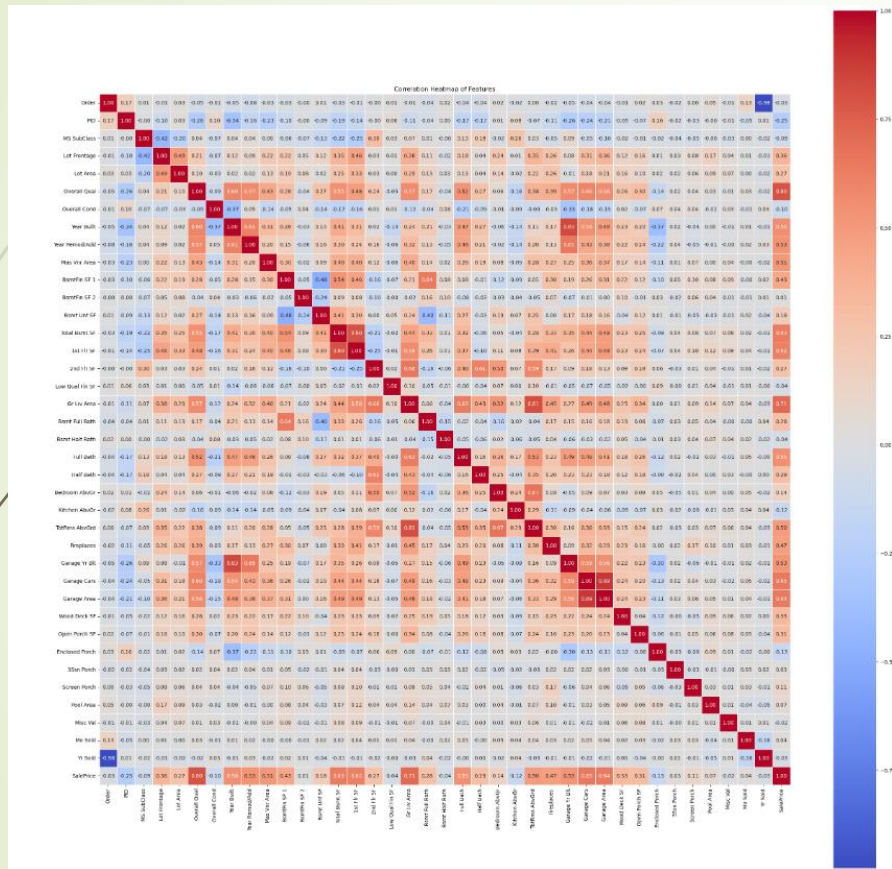
- **項目目標**：使用多種模型融合的方式，預測房價並提高模型準確性。
- **數據集**：Kaggle Ames Housing Dataset

# 特徵工程(刪除相關係數過大特徵)

```
def remove_highly_correlated_features(df, threshold=0.75):  
    # 計算相關性矩陣  
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns  
    data = pd.DataFrame(df[numeric_cols].values, columns=numeric_cols)  
    corr_matrix = data.corr().abs()  
  
    # 找出高於閾值的相關性特徵組合  
    upper_triangle = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))  
    to_drop = [column for column in upper_triangle.columns if any(upper_triangle[column] > threshold)]  
  
    # 移除高相關性的特徵  
    df_reduced = df.drop(columns=to_drop)  
    return df_reduced
```

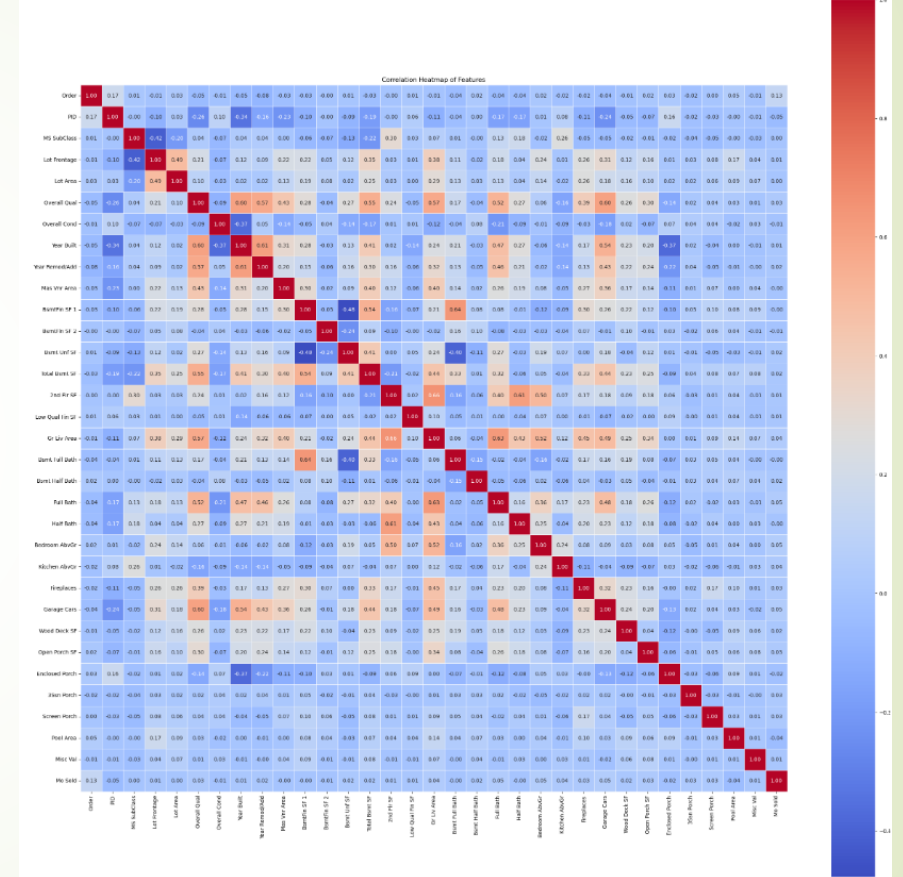
保留相關係數 -0.75~0.75 的特徵向量

# 特徵工程 (刪除相關係數過大特徵)




Before

藉由熱力圖觀察特徵刪除前後的相關係數



After



## 特徵工程(刪除缺失值過多特徵)

- ➡ 刪除缺失值數量約總資料長度一半以上的特徵

## 特徵工程(缺失值填補)

```
def fill_missing_with_median(df):  
    # 找出缺失值大於0的數值型欄位  
    missing_cols = df.columns[df.isnull().sum() > 0]  
  
    for col in missing_cols:  
        # 檢查是否為數值型欄位，才使用中位數填補  
        if df[col].dtype in ['float64', 'int64']:  
            median_value = df[col].median()  
            df[col] = df[col].fillna(median_value)  
        else:  
            mode_value = df[col].mode()[0] # 使用眾數填補  
            df[col] = df[col].fillna(mode_value)
```

數值型特徵以中位數填補，其他型態以眾數填補




# 特徵工程(one-hot encoding)

```
def convert_str_columns_to_dummies(df):  
    # 找出所有字串型欄位 (object 或 category)  
    str_cols = df.select_dtypes(include=['object', 'category']).columns  
  
    # 使用get_dummies將這些欄位轉換為one-hot encoding  
    df_dummies = pd.get_dummies(df, columns=str_cols, drop_first=True, dtype=int)  
    return df_dummies
```

使用 One-Hot Encoding 將類別特徵轉換成數值向量，使模型能夠使用

# 資料分割

- 使用 `train_test_split` 將資料分割為訓練資料及測試資料
- 使用標準化將數據的平均值調整為0，標準差調整為1，將數據縮放到同一尺度，幫助模型更快地收斂，提升準確性
- 使用 Lasso 的L1 範數正則化，將一些特徵的係數縮減到0，達到自動選擇重要特徵的效果，從而降低高維數據過擬合的風險。
- 使用 Lasso 回歸時，通常需要先進行標準化，因為 Lasso 的正則化項會受到特徵量級的影響。標準化確保所有特徵的尺度一致，使正則化項能夠對每個特徵施加公平的懲罰。



# 訓練模型

- 建立 Stacking 模型，使用隨機森林和梯度提升作為基模型，線性回歸作為終極模型
- 使用 GridSearch 尋找最佳參數
- 使用 cross\_val\_score 評估模型的穩定性和泛化能力



# 模型預測結果

隨機森林模型 RMSE : 27538.561659934156

梯度提升模型 RMSE : 27719.419564270836

Stacking模型 RMSE : 27806.130388400157

最佳參數： {'gb\_\_n\_estimators': 900, 'rf\_\_n\_estimators': 10}

GridSearch 優化後的模型 RMSE : 28076.82523393286

score=22258.6420

使用 RMSE 指標，並與基線模型對比。

使用不同交叉驗證策略的效果。



# Accuracy

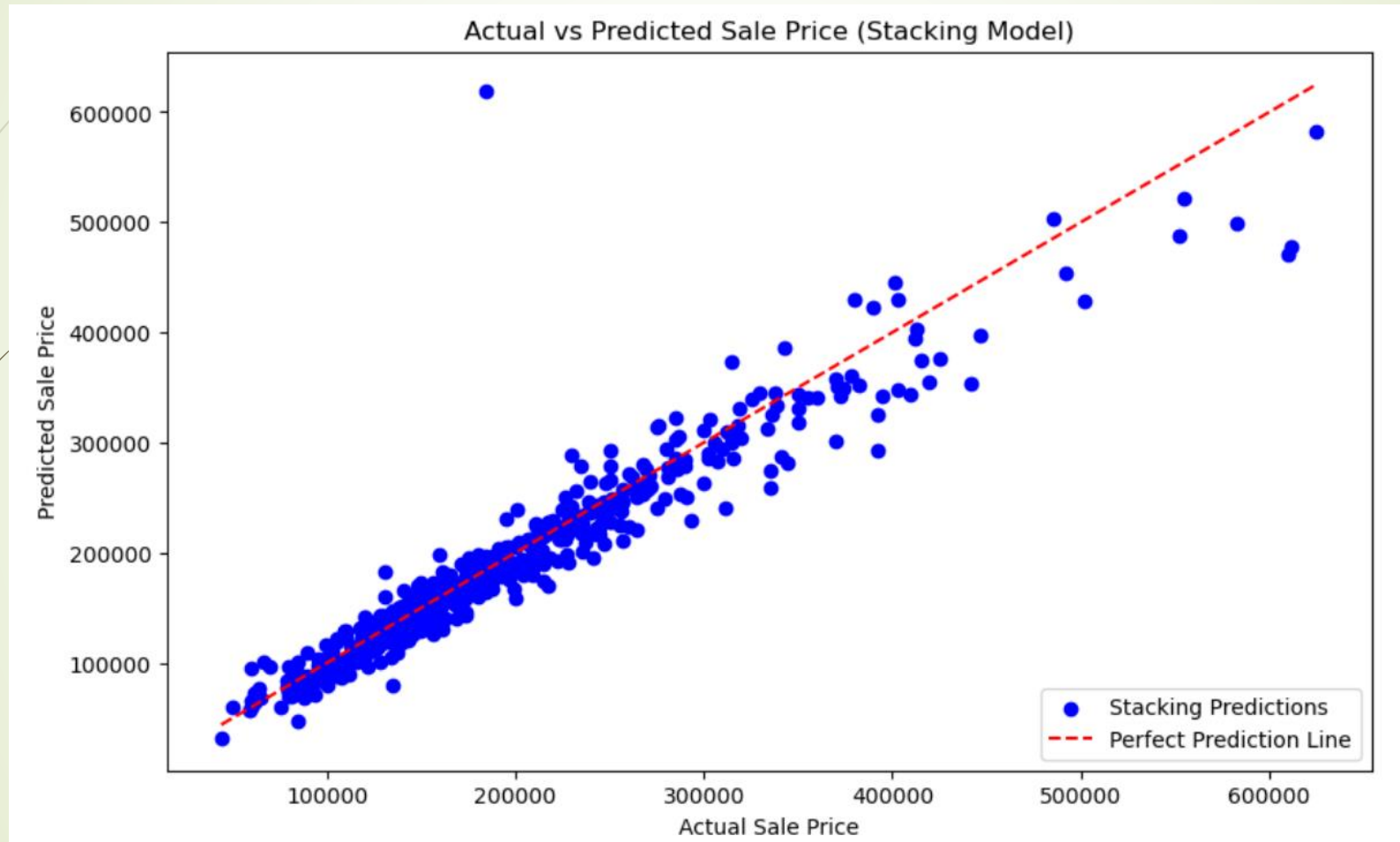
## 計算 Accuracy

```
print("Train Accuracy:{}".format(stacking_model.score(X_train_scaled,y_train)))  
print("Test Accuracy:{}".format(stacking_model.score(X_test_scaled, y_test)))
```

Train Accuracy:0.9576617917796284

Test Accuracy:0.9035639208097419

# 模型預測結果視覺化



使用離散點呈現實際與預測分布，並劃出模型預測線



# 優化建議與未來改進

- **特徵交互**：考慮加入特徵的交互項，例如房間數和房屋面積的乘積，以提升模型的表現。
- **模型融合方法**：可以嘗試不同類型的基模型，提升模型多樣性。