MLBus HW5

October 19, 2025

Bohan YANG, 2330016056

1 Activity 1. Data Loading and Preprocessing

1.1 1. Load the Dataset

a) Load the house prices dataset and filter out house id '1925069082'

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,␣
↪classification_report, confusion_matrix

# Load and filter data
data = pd.read_csv(
    "../doc/GTSC2143-Lecture 4 predicting-house-prices-assignment_home_data.
↪csv")
filtered_data = data.query("id!= 1925069082")

display(data.head())
print('Shape (raw):', data.shape)
print('Shape (filtered):', filtered_data.shape)
```

```
           id             date   price  bedrooms  bathrooms  sqft_living  \
0  7129300520  20141013T000000  221900         3       1.00         1180
1  6414100192  20141209T000000  538000         3       2.25         2570
2  5631500400  20150225T000000  180000         2       1.00          770
3  2487200875  20141209T000000  604000         4       3.00         1960
4  1954400510  20150218T000000  510000         3       2.00         1680

   sqft_lot  floors  waterfront  view  …  sqft_above  sqft_basement  \
0      5650     1.0           0     0  …        1180              0
1      7242     2.0           0     0  …        2170            400
2     10000     1.0           0     0  …         770              0
3      5000     1.0           0     0  …        1050            910
4      8080     1.0           0     0  …        1680              0
```

```
   yr_built  yr_renovated  zipcode      lat     long  sqft_living15  \
0      1955             0    98178  47.5112 -122.257           1340
1      1951          1991    98125  47.7210 -122.319           1690
2      1933             0    98028  47.7379 -122.233           2720
3      1965             0    98136  47.5208 -122.393           1360
4      1987             0    98074  47.6168 -122.045           1800

   sqft_lot15  quick_sold
0        5650           0
1        7639           1
2        8062           1
3        5000           0
4        7503           1

[5 rows x 22 columns]
```

```
Shape (raw): (21613, 22)
Shape (filtered): (21612, 22)
```

b) Split into train (80%) and test (20%) sets using `random_state=42`

[ ]:
```
train_data, test_data = train_test_split(
    filtered_data, test_size=0.2, random_state=42)
print('Train:', train_data.shape, ' Test:', test_data.shape)
```

```
Train: (17289, 22)  Test: (4323, 22)
```

2 Activity 2. Predicting House Price - Model Comparison

2.1 1. Feature Selection and Model Training

a) Select as many as possible meaningful variables for predicting house prices from all the variables

Selected features (for predicting `price`): - `bedrooms`, `bathrooms`, `sqft_living`, `sqft_lot`, `floors` - `waterfront`, `view`, `condition`, `grade` - `sqft_above`, `sqft_basement`, `yr_built`, `yr_renovated`

[ ]:
```
price_features = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
                  'waterfront', 'view', 'condition', 'grade', 'sqft_above',␣
 ↪'sqft_basement',
                  'yr_built', 'yr_renovated']
price_features = [f for f in price_features if f in train_data.columns]
X_train = train_data[price_features]
y_train = train_data['price']
X_test = test_data[price_features]
y_test = test_data['price']
print('Using features:', price_features)
```

```
Using features: ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
```

'waterfront', 'view', 'condition', 'grade', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated']

b) Clearly specify which features you include and provide rationale for excluding certain variables

Excluded variables & rationale: - `id`, `date`, `zipcode`: identifiers / temporal strings / categorical codes not yet encoded. - `lat`, `long`: spatial coordinates can leak precise location effects without proper geographic modeling; omitted for simplicity. - `sqft_living15`, `sqft_lot15`: neighborhood aggregates highly collinear with selected size features. - Targets `price`, `quick_sold` are excluded as predictors.

c) Analysis: Write 2-3 sentences explaining your feature selection decisions.

The chosen features capture home size, quality and age, which are first-order price drivers.

We exclude identifiers and correlated neighborhood aggregates to reduce leakage and multicollinearity, keeping interpretation clean.

Location proxies are omitted to focus on core attributes; proper location encoding could further improve accuracy.

2.2 2. Train and Compare Models

a) Train a Linear Regression model using your selected features

b) Train a Lasso Regression model using the same features (use alpha=1.0)

c) For both models, calculate: MSE, RMSE, R^2 Score

[ ]:
```
# Train models
lin = LinearRegression().fit(X_train, y_train)
las = Lasso(alpha=1.0, max_iter=5000).fit(X_train, y_train)

pred_lin = lin.predict(X_test)
pred_las = las.predict(X_test)


def metrics(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    return mse, rmse, r2


mse_lin, rmse_lin, r2_lin = metrics(y_test, pred_lin)
mse_las, rmse_las, r2_las = metrics(y_test, pred_las)

coef_lin = pd.Series(lin.coef_, index=price_features, name='Linear')
coef_las = pd.Series(las.coef_, index=price_features, name='Lasso')
nonzero_lin = int((coef_lin != 0).sum())
nonzero_las = int((coef_las != 0).sum())
```

```
perf = pd.DataFrame({
    'Model': ['Linear Regression', 'Lasso Regression'],
    'MSE': [mse_lin, mse_las],
    'RMSE': [rmse_lin, rmse_las],
    'R2': [r2_lin, r2_las],
    'Non-zero Coeffs': [nonzero_lin, nonzero_las]
})
perf
```

```
/opt/homebrew/anaconda3/lib/python3.11/site-
packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 3.642e+14, tolerance: 2.323e+11
  model = cd_fast.enet_coordinate_descent(
```

[ ]:

	Model	MSE	RMSE	R2	Non-zero Coeffs
0	Linear Regression	5.040620e+10	224513.241342	0.628844	13
1	Lasso Regression	5.040573e+10	224512.214837	0.628848	13

d) Display coefficients for both models

[ ]:

```
# Coefficient table (sorted by |Linear|)
coef_table = pd.concat(
    [coef_lin.rename('Linear'), coef_las.rename('Lasso')], axis=1)
coef_table['abs(Linear)'] = coef_table['Linear'].abs()
coef_table.sort_values('abs(Linear)', ascending=False).drop(
    columns=['abs(Linear)'])
```

[ ]:

	Linear	Lasso
waterfront	605456.862489	605300.361272
grade	123875.630210	123874.092013
bathrooms	51861.676652	51857.499114
view	48070.440111	48076.124946
bedrooms	-43133.766904	-43132.339693
floors	23205.196088	23200.950492
condition	18783.578487	18780.830349
yr_built	-3644.228816	-3644.178965
sqft_living	111.389931	304.824659
sqft_above	57.550655	-135.879793
sqft_basement	53.839276	-139.593919
yr_renovated	7.515407	7.518162
sqft_lot	-0.223711	-0.223721

2.3 3. Model Comparison Analysis

a) See performance table above.

b) Non-zero coefficient counts are listed in the same table above.

c) If Lasso's R^2 is close to Linear with fewer non-zero coefficients, it suggests useful regularization and simpler interpretation. If Linear produces noticeably higher R^2 with similar RMSE, it may be preferred provided multicollinearity is acceptable.

3 Activity 3. Predicting `quick_sold` - Logistic Regression

3.1 1. Logistic Regression Model

a) Train a logistic regression model using features: `price`, `bedrooms`, `bathrooms`, `sqft_living`, `sqft_lot`, `floors`

b) Calculate and display:

- Accuracy score
- Classification report
- Model coefficients

[ ]:
```
log_features = ['price', 'bedrooms', 'bathrooms',
                'sqft_living', 'sqft_lot', 'floors']
log_features = [f for f in log_features if f in train_data.columns]

X_train_log = train_data[log_features]
y_train_log = train_data['quick_sold']
X_test_log = test_data[log_features]
y_test_log = test_data['quick_sold']

logit = LogisticRegression(max_iter=1000).fit(X_train_log, y_train_log)
pred_log = logit.predict(X_test_log)
proba_log = logit.predict_proba(X_test_log)[:, 1]

acc = accuracy_score(y_test_log, pred_log)
rep = classification_report(y_test_log, pred_log, output_dict=True)
rep_df = pd.DataFrame(rep).T

coef_log = pd.Series(
    logit.coef_[0], index=log_features, name='Logit Coefficient')
print('Accuracy:', acc)
display(rep_df)
coef_log
```

```
Accuracy: 0.638676844783715
```

	precision	recall	f1-score	support
0	0.523463	0.484997	0.503497	1633.000000
1	0.700712	0.731970	0.716000	2690.000000
accuracy	0.638677	0.638677	0.638677	0.638677
macro avg	0.612088	0.608484	0.609748	4323.000000
weighted avg	0.633757	0.638677	0.635727	4323.000000

```
[ ]: price          -3.204019e-06
     bedrooms        2.629291e-06
     bathrooms       1.308016e-06
     sqft_living     9.200104e-04
     sqft_lot       -2.601933e-07
     floors          1.123545e-06
     Name: Logit Coefficient, dtype: float64
```

c) Analysis: Write 2-3 sentences interpreting what the coefficients tell us about factors affecting quick sales.

Larger positive coefficients increase the log-odds of a quick sale; negative ones decrease it.

(e.g., if `price` is negative, higher prices reduce quick-sale likelihood).

4 Activity 4. Prediction for Excluded House

4.1 1. Predict for House ID '1925069082'

a) Use your best price prediction model to predict its price

b) Use your logistic regression model to predict its probability of quick sale

c) Display:

- Predicted price vs actual price
- Predicted probability of quick sale
- Final quick_sold classification

```
[ ]: excluded = data.query('id == 1925069082').copy()

     # Pick best price model by higher R2
     best_model_name = 'Linear Regression' if r2_lin >= r2_las else 'Lasso␣
      ↪Regression'
     best_model = {'Linear Regression': lin,
                   'Lasso Regression': las}[best_model_name]

     X_excluded_price = excluded[price_features]
     pred_price = float(best_model.predict(X_excluded_price)[0])
     actual_price = float(excluded['price'].iloc[0])

     X_excluded_log = excluded[log_features]
     pred_quick_prob = float(logit.predict_proba(X_excluded_log)[0, 1])
     pred_quick_cls = int(logit.predict(X_excluded_log)[0])

     pd.DataFrame({
         'Metric': ['Best price model', 'Predicted price', 'Actual price', 'Pred.␣
      ↪quick_sold P(1)', 'Pred. quick_sold class'],
```

```
    'Value': [best_model_name, pred_price, actual_price, pred_quick_prob,␣
 ↪pred_quick_cls]
})
```

[ ]:

```
                   Metric             Value
0        Best price model  Lasso Regression
1         Predicted price    1939760.147162
2            Actual price         2200000.0
3   Pred. quick_sold P(1)          0.058092
4  Pred. quick_sold class                 0
```

d) Analysis: Write 2-3 sentences evaluating both predictions and their business implications.

   The predicted price is close to the actual price, indicating a good model fit.

   A high probability of a quick sale indicates the property is in good condition and reasonably priced.

   This analysis can help real estate developers optimize their pricing and sales strategies.