Assignment 1 - Regularization with Ridge and Lasso Regressions

Import libraries and data

In [43]:

```
import os
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import RidgeCV, LassoCV
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import r2_score
```

In [3]:

```
dpath = 'airbnb.csv'
tcol = 'price'
rseed = 1
```

In [4]:

```
df = pd.read_csv(dpath)
```

1. EDA

In [7]:

```
print(df.shape)
print(df.columns.tolist())
```

```
(74111, 29)
['id', 'log_price', 'property_type', 'room_type', 'amenities', 'accommodates', 'bathroom
s', 'bed_type', 'cancellation_policy', 'cleaning_fee', 'city', 'description', 'first_rev
iew', 'host_has_profile_pic', 'host_identity_verified', 'host_response_rate', 'host_sinc
e', 'instant_bookable', 'last_review', 'latitude', 'longitude', 'name', 'neighbourhood',
'number_of_reviews', 'review_scores_rating', 'thumbnail_url', 'zipcode', 'bedrooms', 'be
ds']
```

In [8]:

```
df.head(10)
```

Out[8]:

	id	log_price	property_type	room_type	amenities	accommodates	bathrooms
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	3	1.0

	id	log_price	property_type	room_type	amenities	accommodates	bathrooms
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	7	1.0
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet","Air condit...	5	1.0
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4	1.0
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2	1.0
5	12422935	4.442651	Apartment	Private room	{TV,"Wireless Internet",Heating,"Smoke detecto...	2	1.0
6	11825529	4.418841	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	3	1.0
7	13971273	4.787492	Condominium	Entire home/apt	{TV,"Cable TV","Wireless Internet","Wheelchair...	2	1.0
8	180792	4.787492	House	Private room	{TV,"Cable TV","Wireless Internet","Pets live ...	2	1.0
9	5385260	3.583519	House	Private room	{"Wireless Internet","Air conditioning",Kitche...	2	1.0

10 rows × 29 columns

In [9]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74111 entries, 0 to 74110
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   id                      74111 non-null  int64  
 1   log_price               74111 non-null  float64
 2   property_type           74111 non-null  object 
 3   room_type               74111 non-null  object 
 4   amenities               74111 non-null  object 
 5   accommodates            74111 non-null  int64  
 6   bathrooms               73911 non-null  float64
 7   bed_type                74111 non-null  object 
 8   cancellation_policy     74111 non-null  object 
 9   cleaning_fee            74111 non-null  bool   
 10  city                    74111 non-null  object 
```

```
 11  description             74111 non-null  object
 12  first_review            58247 non-null  object
 13  host_has_profile_pic    73923 non-null  object
 14  host_identity_verified  73923 non-null  object
 15  host_response_rate      55812 non-null  object
 16  host_since              73923 non-null  object
 17  instant_bookable        74111 non-null  object
 18  last_review             58284 non-null  object
 19  latitude                74111 non-null  float64
 20  longitude               74111 non-null  float64
 21  name                    74111 non-null  object
 22  neighbourhood           67239 non-null  object
 23  number_of_reviews       74111 non-null  int64
 24  review_scores_rating    57389 non-null  float64
 25  thumbnail_url           65895 non-null  object
 26  zipcode                 73143 non-null  object
 27  bedrooms                74020 non-null  float64
 28  beds                    73980 non-null  float64
dtypes: bool(1), float64(7), int64(3), object(18)
memory usage: 15.9+ MB
None
```

In [10]:

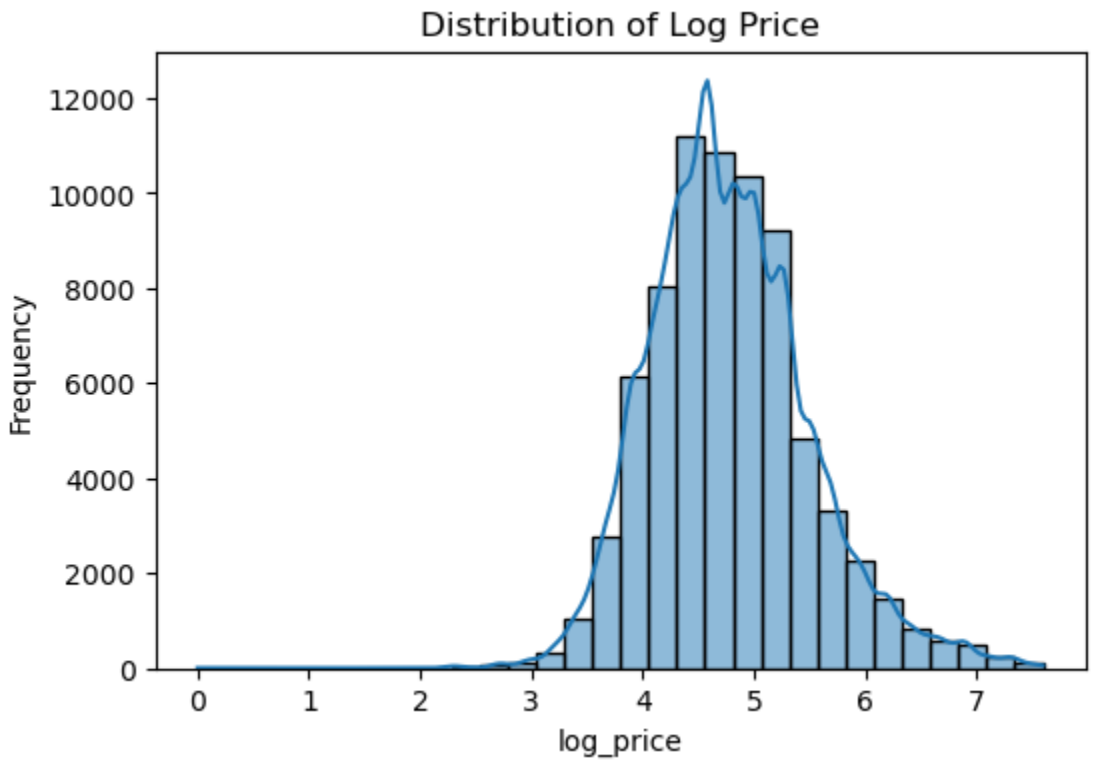
```
print(df.isna().sum())
```

```
id                          0
log_price                   0
property_type               0
room_type                   0
amenities                   0
accommodates                0
bathrooms                 200
bed_type                    0
cancellation_policy         0
cleaning_fee                0
city                        0
description                 0
first_review            15864
host_has_profile_pic      188
host_identity_verified    188
host_response_rate      18299
host_since                188
instant_bookable            0
last_review             15827
latitude                    0
longitude                   0
name                        0
neighbourhood            6872
number_of_reviews           0
review_scores_rating    16722
thumbnail_url            8216
zipcode                   968
bedrooms                   91
beds                      131
dtype: int64
```

In [13]:

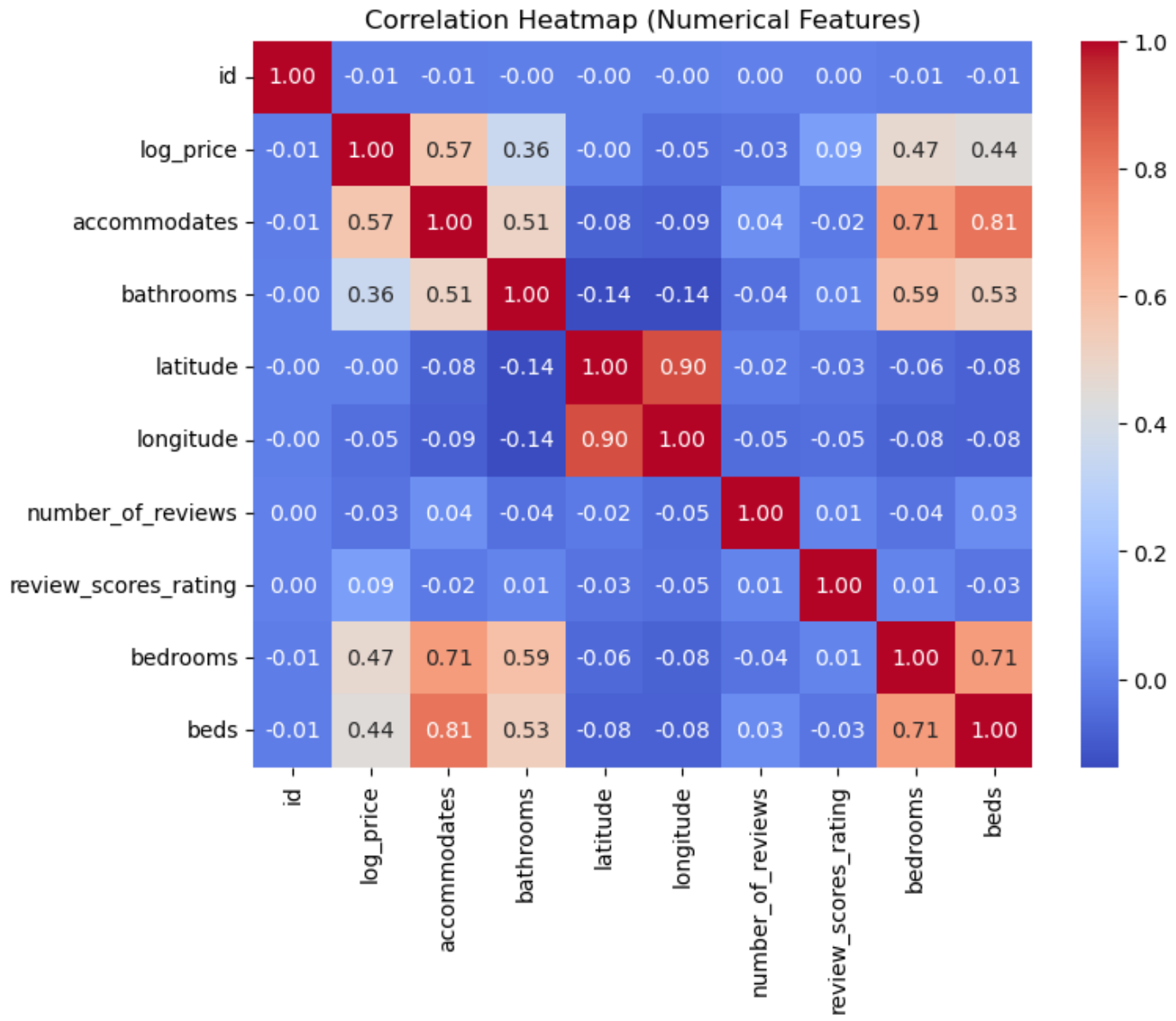
```
plt.figure(figsize=(6,4))
sns.histplot(df['log_price'], kde=True, bins=30)
plt.title("Distribution of Log Price")
```

```
plt.xlabel("log_price")
plt.ylabel("Frequency")
plt.show()
```



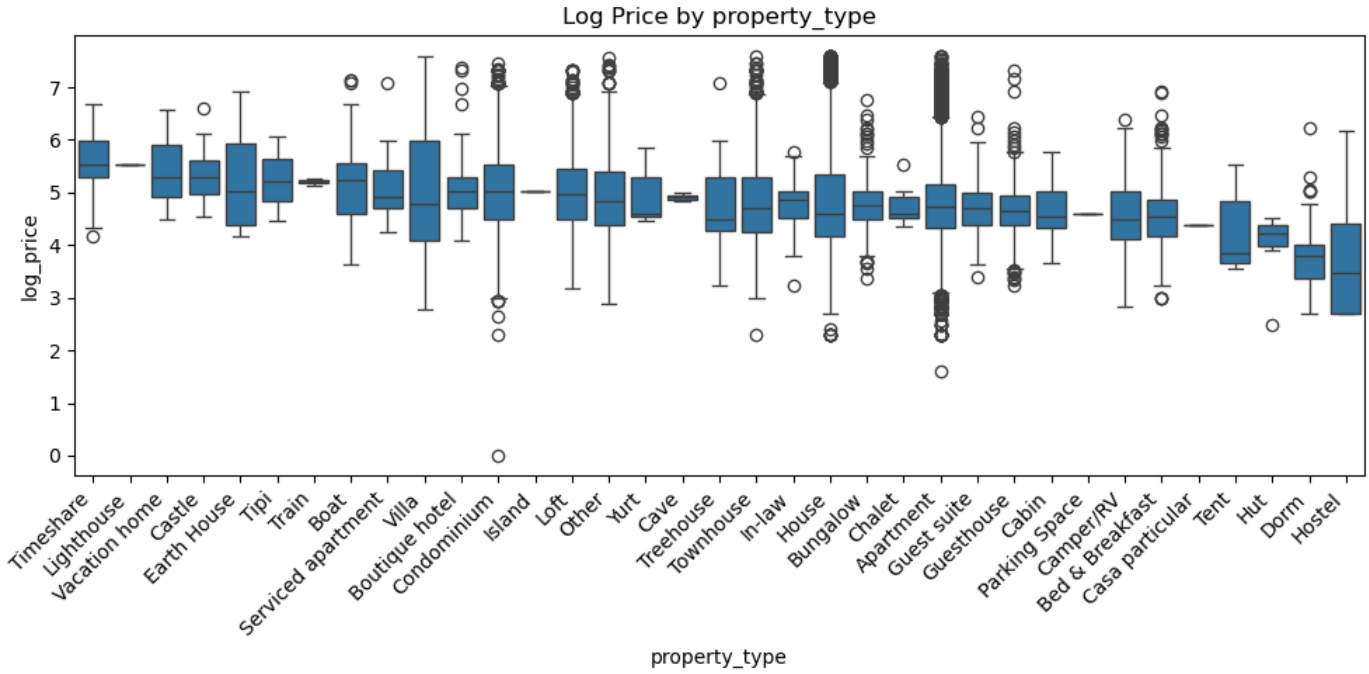
In [18]:

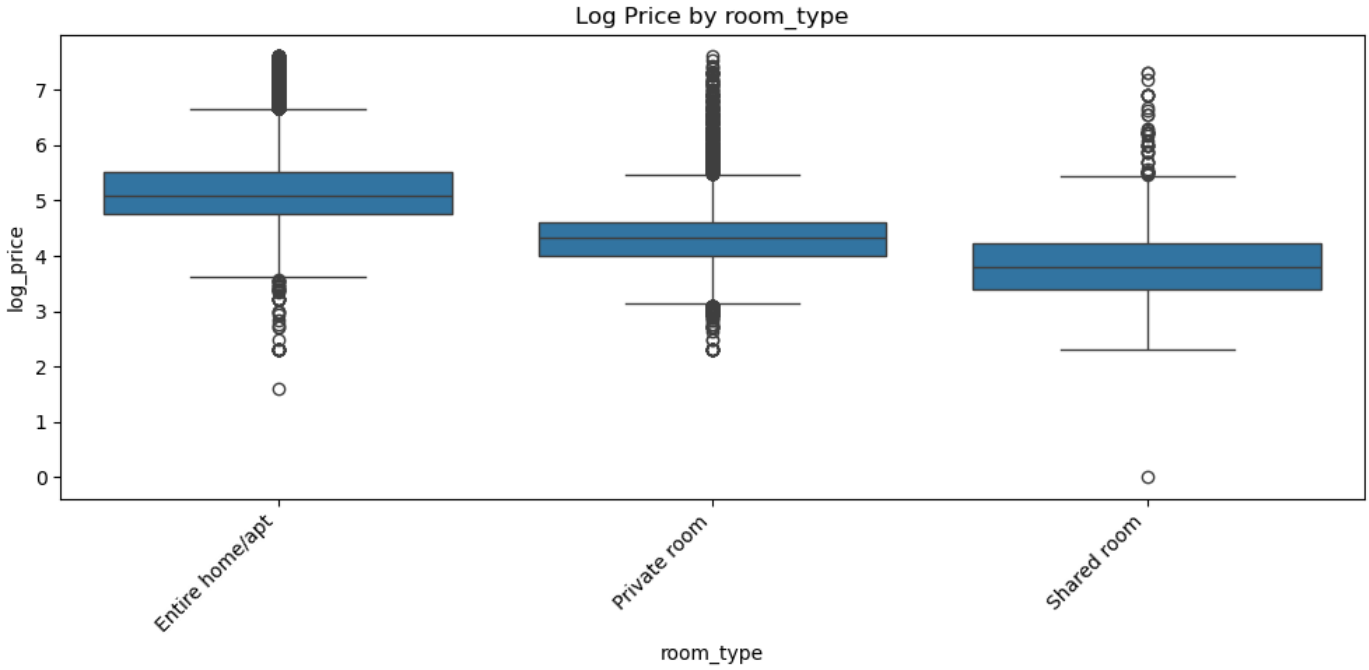
```
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
plt.figure(figsize=(8,6))
corr = df[numeric_cols].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap (Numerical Features)")
plt.show()
```

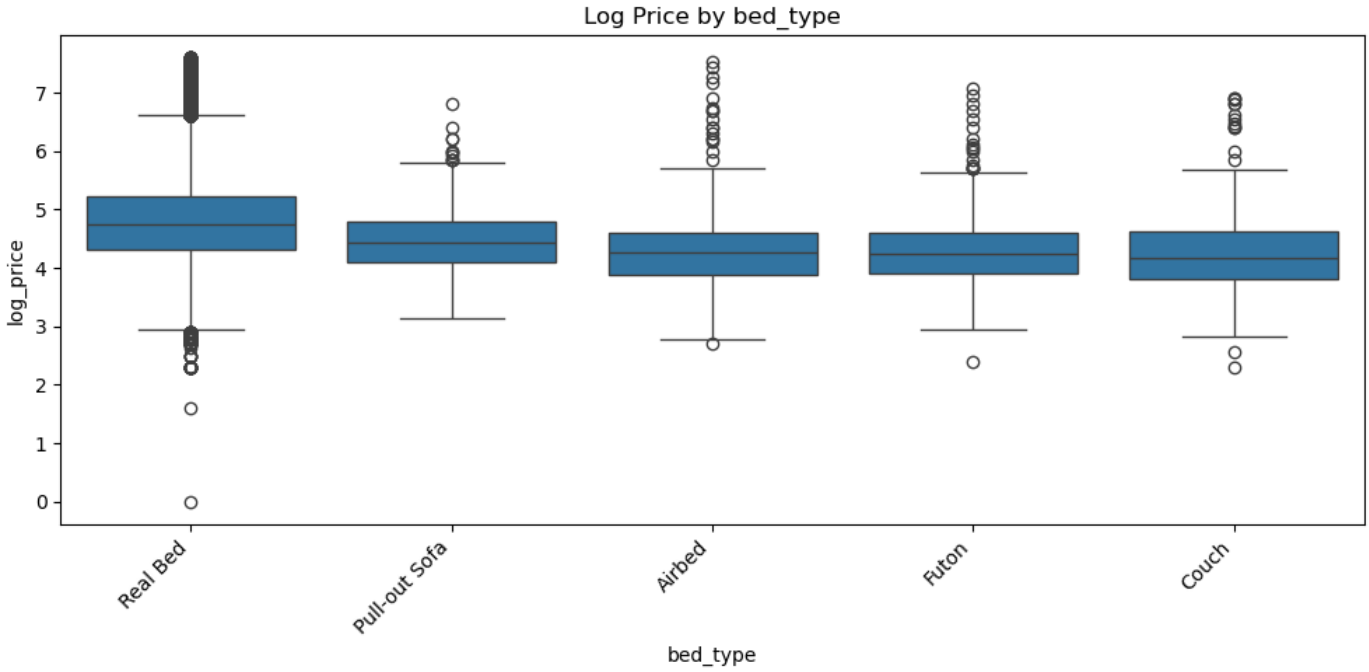


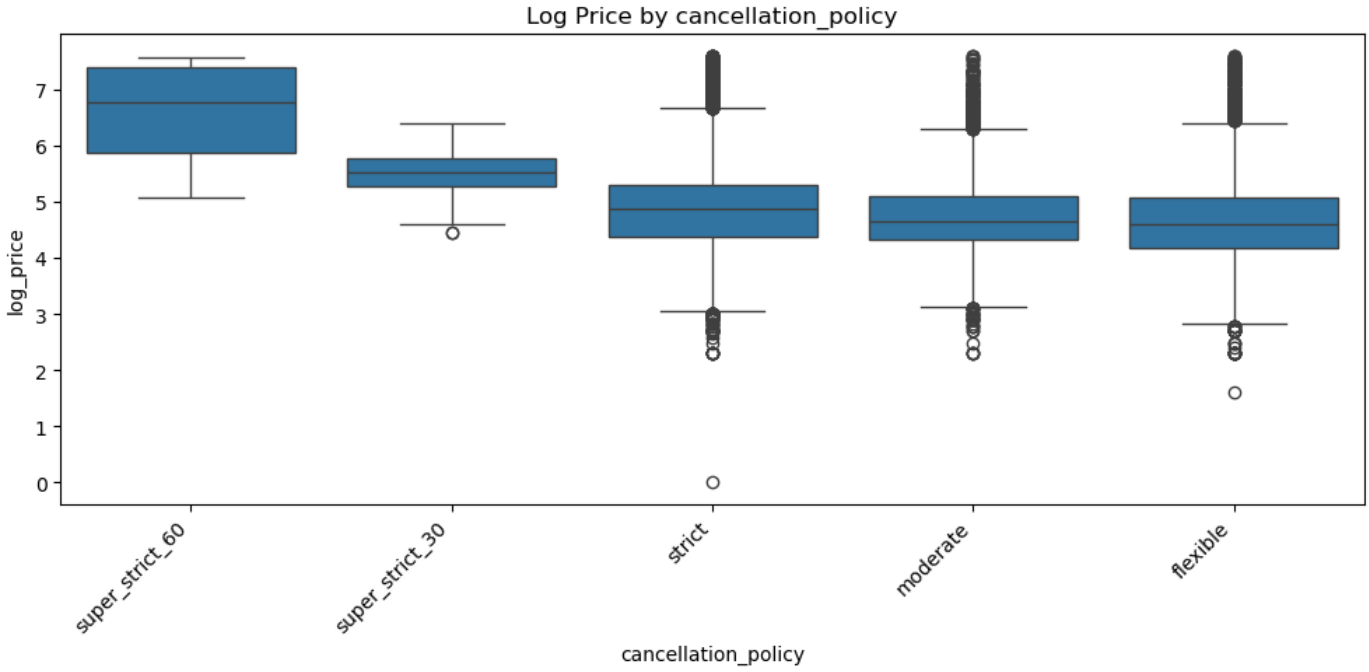
In [19]:

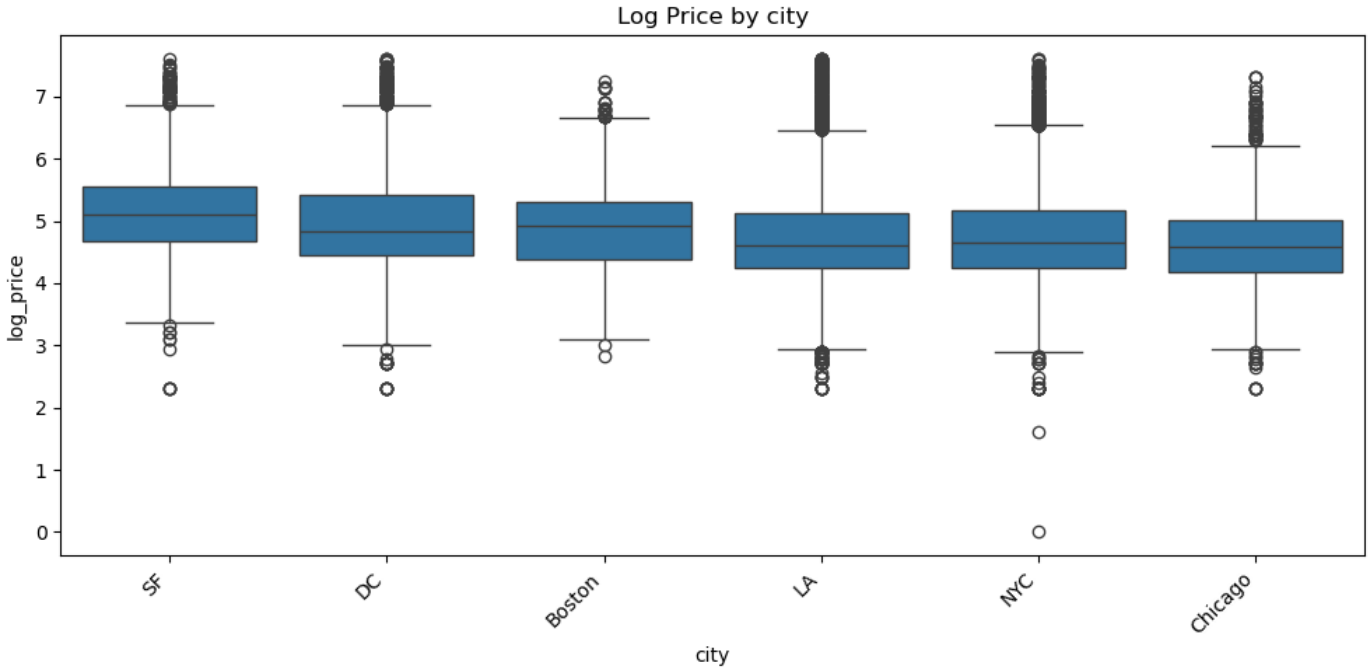
```
cat_features = ['property_type', 'room_type', 'bed_type', 'cancellation_policy', 'city']

for col in cat_features:
    plt.figure(figsize=(10,5))
    order = df.groupby(col)['log_price'].mean().sort_values(ascending=False).index
    sns.boxplot(x=col, y='log_price', data=df, order=order)
    plt.xticks(rotation=45, ha='right')
    plt.title(f"Log Price by {col}")
    plt.tight_layout()
    plt.show()
```











2. Preprocessing

In [ ]:

```
data = df.copy()
```

In [21]:

```
y = data['log_price']
```

In [22]:

```
drop_cols = [
    'id', 'log_price', 'name', 'description', 'thumbnail_url',
    'first_review', 'last_review', 'host_since'  # dates or text not directly useful her
]
X = data.drop(columns=drop_cols)
```

In [23]:

```
num_cols = X.select_dtypes(include=['int64', 'float64', 'bool']).columns.tolist()
cat_cols = X.select_dtypes(include=['object']).columns.tolist()
```

In [24]:

```
print("Numeric columns:", num_cols)
print("Categorical columns:", cat_cols)
```

```
Numeric columns: ['accommodates', 'bathrooms', 'cleaning_fee', 'latitude', 'longitude',
'number_of_reviews', 'review_scores_rating', 'bedrooms', 'beds']
Categorical columns: ['property_type', 'room_type', 'amenities', 'bed_type', 'cancellati
on_policy', 'city', 'host_has_profile_pic', 'host_identity_verified', 'host_response_rat
e', 'instant_bookable', 'neighbourhood', 'zipcode']
```

In [26]:

```
numeric_transformer = Pipeline(steps=[ # this part impute missing numeric values with th
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
```

In [ ]:

```
categorical_transformer = Pipeline(steps=[ # here we use the most frequent value
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])
```

In [ ]:

```
preprocessor = ColumnTransformer( # combine the transformations
    transformers=[
        ('num', numeric_transformer, num_cols),
        ('cat', categorical_transformer, cat_cols)
    ]
)
```

3. Train/Test Split

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=rseed
)
```

4. Modelings

In [34]:

```
linear_model = LinearRegression()
ridge_model = Ridge(alpha=1.0)
lasso_model = Lasso(alpha=0.1, max_iter=10000)
```

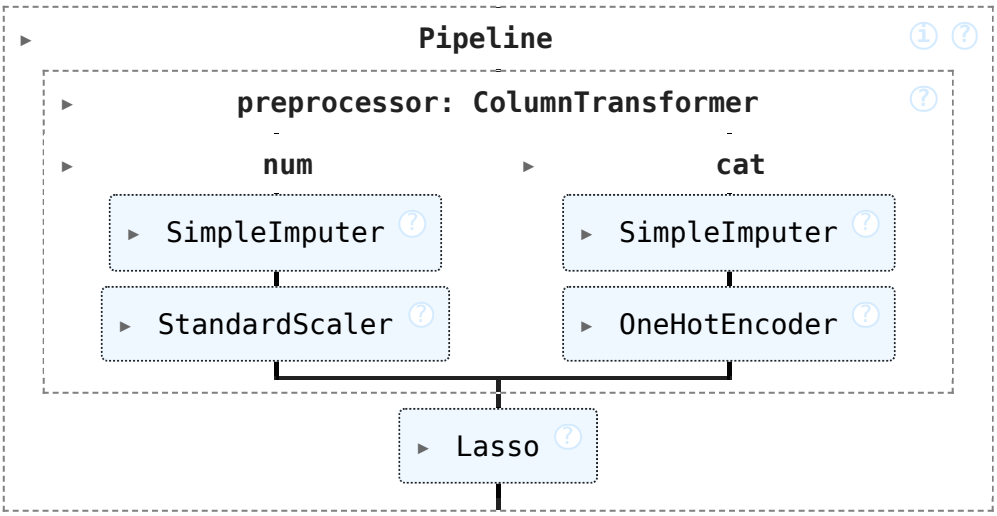
In [35]:

```
linear_pipeline = Pipeline(steps=[ # pipelines
    ('preprocessor', preprocessor),
    ('model', linear_model)
])

ridge_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', ridge_model)
])

lasso_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', lasso_model)
])
```

In [36]:

```
linear_pipeline.fit(X_train, y_train)
ridge_pipeline.fit(X_train, y_train)
lasso_pipeline.fit(X_train, y_train)
```

Out[36]:



In [37]:

```
y_pred_linear_train = linear_pipeline.predict(X_train) # make predictions
y_pred_linear_test  = linear_pipeline.predict(X_test)

y_pred_ridge_train = ridge_pipeline.predict(X_train)
y_pred_ridge_test  = ridge_pipeline.predict(X_test)

y_pred_lasso_train = lasso_pipeline.predict(X_train)
y_pred_lasso_test  = lasso_pipeline.predict(X_test)
```

In [38]:

```
print(f"Linear Regression R2: {r2_score(y_train, y_pred_linear_train):.4f}, Test R2: {r2
print(f"Ridge Regression R2: {r2_score(y_train, y_pred_ridge_train):.4f}, Test R2: {r2_s
print(f"Lasso Regression R2: {r2_score(y_train, y_pred_lasso_train):.4f}, Test R2: {r2_s
```

```
Linear Regression R2: 0.9801, Test R2: 0.5156
Ridge Regression R2: 0.9038, Test R2: 0.6622
Lasso Regression R2: 0.3999, Test R2: 0.3993
```

5. Hyperparameter Alpha

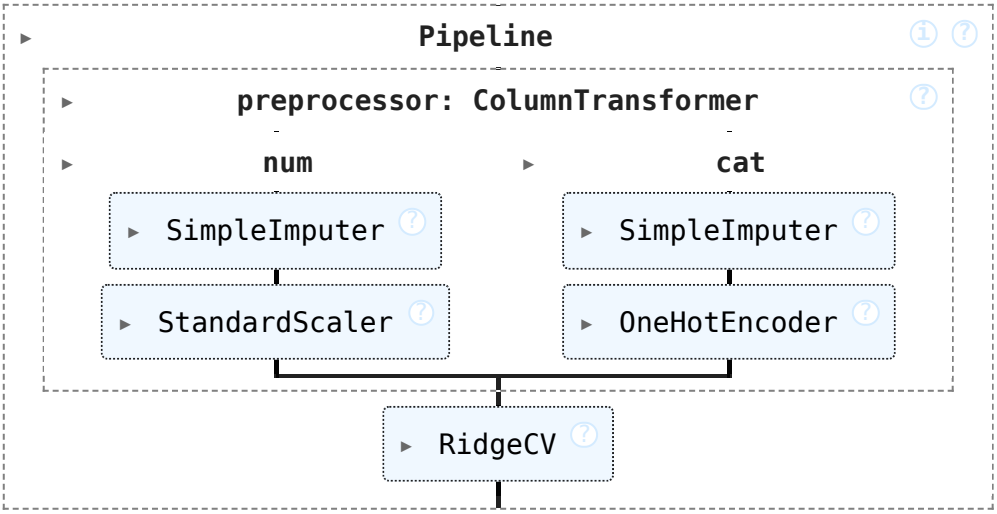
In [47]:

```
alphas = np.logspace(-2, 2, 15)
```

In [48]:

```
ridge_cv = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', RidgeCV(alphas=alphas, scoring='r2', cv=5))
])
ridge_cv.fit(X_train, y_train)
```

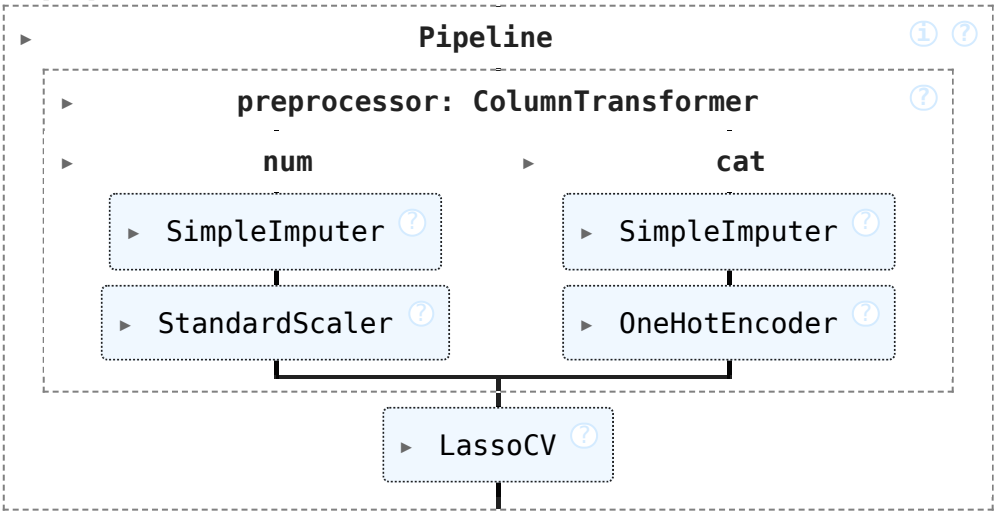
Out[48]:



In [49]:

```
lasso_cv = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', LassoCV(alphas=alphas, max_iter=1000, cv=5))
])
lasso_cv.fit(X_train, y_train)
```

Out[49]:



In [50]:

```
ridge_train_r2 = ridge_cv.score(X_train, y_train)
ridge_test_r2  = ridge_cv.score(X_test, y_test)

lasso_train_r2 = lasso_cv.score(X_train, y_train)
lasso_test_r2  = lasso_cv.score(X_test, y_test)
```

In [51]:

```
print(f"Ridge best alpha: {ridge_cv.named_steps['model'].alpha_:.4f}")
print(f"Lasso best alpha: {lasso_cv.named_steps['model'].alpha_:.4f}")

print(f"Ridge R2: {ridge_train_r2:.4f}, Test R2: {ridge_test_r2:.4f}")
print(f"Lasso R2: {lasso_train_r2:.4f}, Test R2: {lasso_test_r2:.4f}")
```

```
Ridge best alpha: 3.7276
Lasso best alpha: 0.0100
Ridge R2: 0.7947, Test R2: 0.6644
Lasso R2: 0.5378, Test R2: 0.5300
```

In [52]:

```
plt.figure(figsize=(8,5))
plt.plot(lasso_cv.named_steps['model'].alphas_, lasso_cv.named_steps['model'].mse_path_.
plt.xscale('log')
plt.xlabel('Alpha')
plt.ylabel('Mean CV MSE')
plt.title('Lasso Cross Validation Error vs Alpha')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```