Airbnb Price Prediction Analysis

Question

How well can the price (log price) of Airbnb rentals be predicted using machine learning models?

Process

- Identify columns with missing or NaN values
- Plan how to deal with each individual column
- Clean data
- Transform categorical variables into encoded features
- Train a base XGBoost model on the cleaned data
- Evaluate results
- Tune hyperparameters
- Evaluate model
- Plot feature importances

Problem areas:

There are a few problem area areas with this analysis.

The first one is that certain categorical features (neighbourhood), if entirely used and encoded, would lead to extreme levels of dimensionality. These features were not included as a result, with the idea that latitude and longitude may make up for them

The other main problem was processing power available. The machine used for this analysis would have taken days to do a comprehensive grid search of hyperparameters, so performing it in chunks was used instead. Unfortunately, this means that the hyperparameters are not perfectly tuned.

MAE - 0.272035938048597

MAPE - 0.05719580023745668

In [40]:
```
#imports

import pandas as pd
import numpy as np
import seaborn as sns
```

```python
import matplotlib.pyplot as plt
import plotly.graph_objects as px
import xgboost
import warnings
from xgboost import plot_tree

warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', 40)
```

In [41]:

```python
#read in data

data = pd.read_csv('../data/train.csv')

data.head()
```

Out[41]:

	id	log_price	property_type	room_type	amenities	accommodates	ba
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	3	
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	7	
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet","Air condit...	5	
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4	
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2	

In [42]:

```python
data.info()

#has missing values:
#bathrooms, first_review, host_has_profile_pic, host_identity_verified, host_respon
#neighbourhood, review_scores_rating, thumbnail_url, zipcode, bedrooms, beds
```

```
<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                 73145 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
```

In [43]:

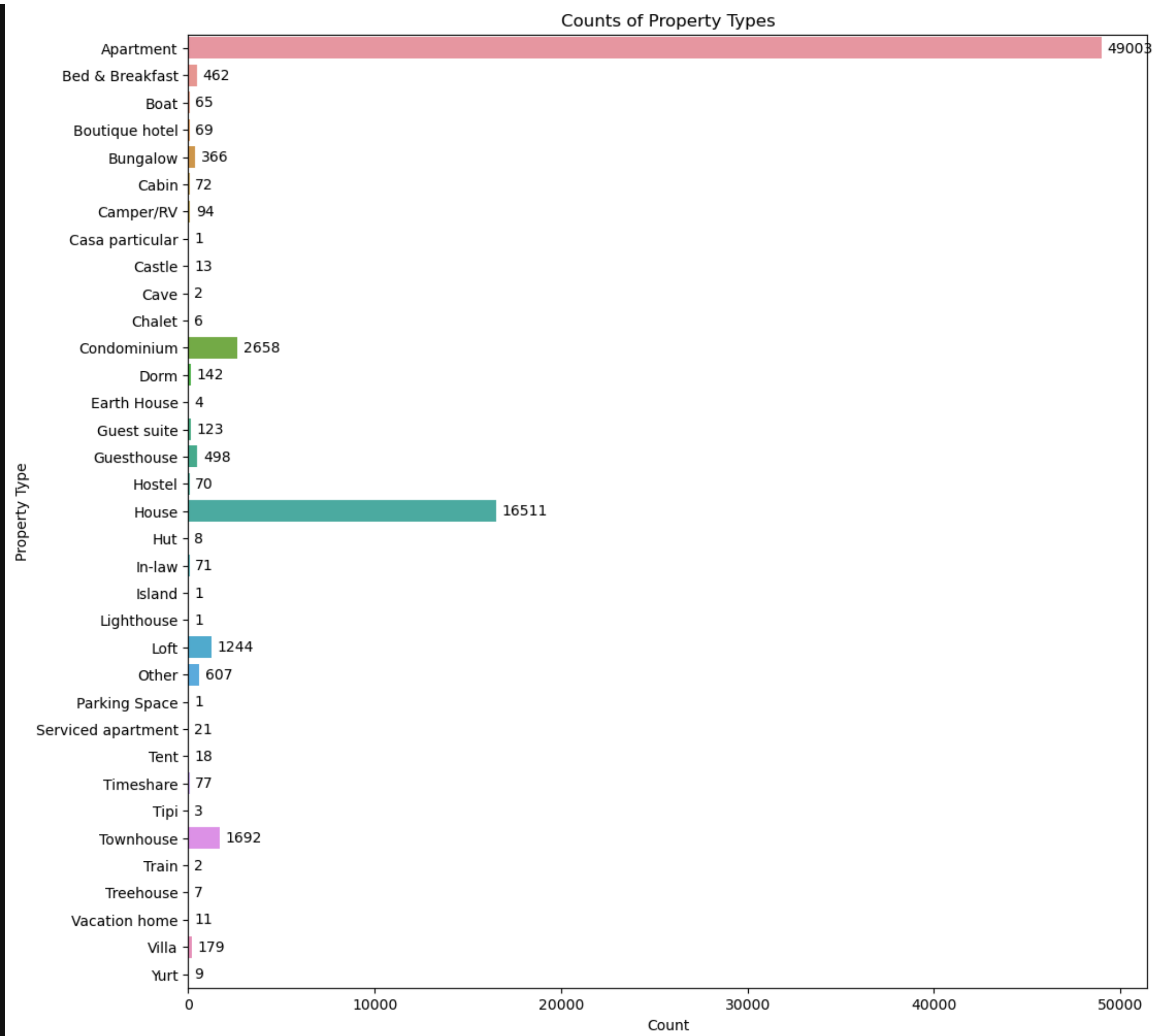
```
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split

#cleaning/transform:

#date to day, month, year cols: first_review, host_since, last_review
#remove: zipcode, description, name, id
#onehot encode (remove first): property_type, room_type, amenities, bed_type, cance
#bool to 1,0: cleaning_fee, thumbnail_url, instant_bookable

#has missing values:

#float (impute): bathrooms, review_scores_rating, bedrooms, beds
#date (-1): first_review, host_since, last_review
#str (fill "unknown"): neighbourhood
#str to bool (nan to 0): host_has_profile_pic, host_identity_verified
#str to int (remove percentages, impute): host_response_rate
#to bool (has value or not): thumbnail_url
```

```python
data['first_review_day'], data['first_review_year'], data['first_review_month'] = p
data['host_since_day'], data['host_since_year'], data['host_since_month'] = pd.to_d
data['last_review_day'], data['last_review_year'], data['last_review_month'] = pd.t
data = data.drop(['first_review', 'host_since', 'last_review', 'zipcode', 'descript

categorical_variables = ['property_type', 'room_type','bed_type', 'cancellation_pol
amenities = data['amenities']
data['neighbourhood'] = data['neighbourhood'].replace(np.nan, 'Unknown')
data = data.drop(['amenities', 'neighbourhood'], axis=1) #neighborhood raises mae
data['cleaning_fee'] = data['cleaning_fee'].map({True: 1, False: 0})
data['host_has_profile_pic'] = data['host_has_profile_pic'].map({'t': 1, 'f': 0, np
data['host_identity_verified'] = data['host_identity_verified'].map({'t': 1, 'f': 0
data['instant_bookable'] = data['instant_bookable'].map({'t': 1, 'f': 0, np.nan: 0}
data['host_response_rate'] = data['host_response_rate'].astype(str).apply(lambda x:
data['thumbnail_url'] = data['thumbnail_url'].map(lambda x: 0 if pd.isna(x) else 1)
data['host_response_rate'] = data['host_response_rate'].astype('Float64')

date_cols = ['first_review_day', 'first_review_year', 'first_review_month', 'host_s
data[date_cols] = data[date_cols].apply(lambda x: x.fillna(-1))

imputed_cols = ['bedrooms', 'bathrooms', 'beds', 'review_scores_rating', 'host_resp
imputer = SimpleImputer(strategy='mean')
imputer.fit(data[imputed_cols])
data[imputed_cols] = imputer.transform(data[imputed_cols])
```

In [44]:

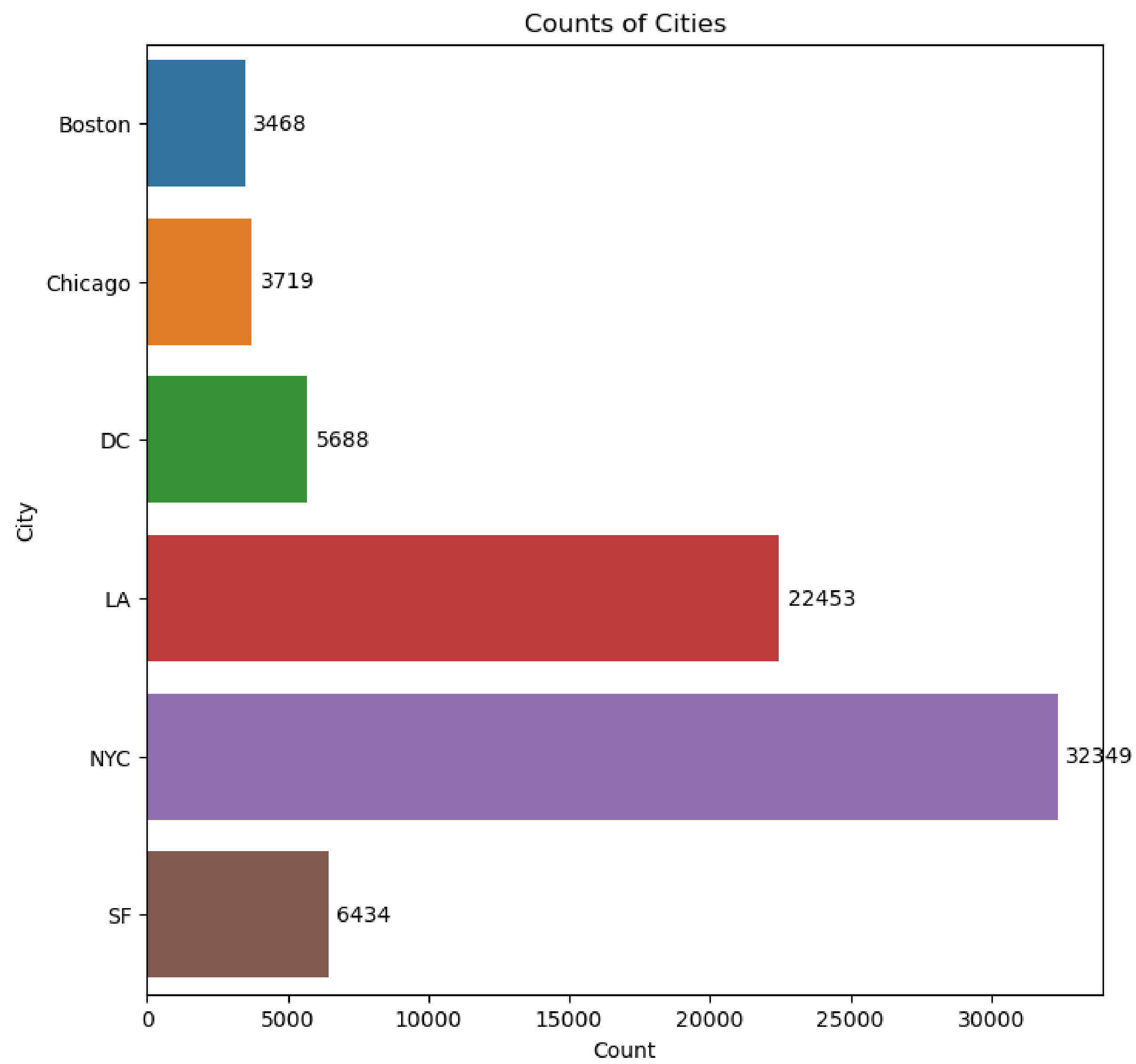
```python
#plot out different property types

fig, ax = plt.subplots(figsize=(12,12))
fig = sns.barplot(x='log_price', data=data.groupby('property_type', as_index=False)
fig.bar_label(fig.containers[0], padding=4)
fig.set_xlabel('Count')
fig.set_ylabel('Property Type')
fig.set_title('Counts of Property Types')
```

Out[44]: Text(0.5, 1.0, 'Counts of Property Types')



In [45]:
```
#plot out city counts
fig, ax = plt.subplots(figsize=(8,8))
fig = sns.barplot(x='log_price', data=data.groupby('city', as_index=False).count()[
fig.bar_label(fig.containers[0], padding=4)
fig.set_xlabel('Count')
fig.set_ylabel('City')
fig.set_title('Counts of Cities')
```

Out[45]: Text(0.5, 1.0, 'Counts of Cities')



In [46]:
```
#encode categorical variables, train/test split

data = pd.get_dummies(data, columns=categorical_variables, drop_first=True)
X = data.drop('log_price', axis=1)
y = data['log_price']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=8)
```

In [47]:
```
#find shape after categorical variables are encoded

data.shape
```

Out[47]: (74111, 73)

In [48]:
```
#initialize xgboost and fit to training data

from xgboost import XGBRegressor

boost = XGBRegressor()
```

```
boost.fit(X_train, y_train)
boost
```

Out[48]:

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=Non
e,
             enable_categorical=False, eval_metric=None, feature_types=Non
e,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=Non
e,
```

In [49]:

```
#model predictions

y_pred = boost.predict(X_test)
```

In [50]:

```
#model MAPE and MAE

from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error

print(f'MAPE: {mean_absolute_percentage_error(y_pred, y_test)}, MAE: {mean_absolute
```

```
MAPE: 0.05878560869465756, MAE: 0.2794164988969564
```

In [23]:

```
#hyperparameter tuning
#this is done in chunks as to not run the entire grid at once, which would take ove
#best values are noted by the comments
from sklearn.model_selection import GridSearchCV

params = {
    'max_depth': [3, 4, 5, 6, 7, 8], #8
    'min_child_weight': [1, 3, 5, 7], #5
    'gamma': [0, 0.1, 0.2, 0.3, 0.4], #0.1
    'subsample': [0.6, 0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.6, 0.7, 0.8, 0.9, 1.0], #0.6
    'n_estimators': [100, 200, 300, 400, 500], #300
    'learning_rate': [0.01, 0.05, 0.1, 0.15, 0.2], #0.05
    'reg_alpha': [0, 0.1, 0.5, 1, 2], #0.1
    'reg_lambda': [0, 0.1, 0.5, 1, 2], #0
    'objective': ['reg:squarederror'],
    'eval_metric': ['mae'],
    'booster': ['gbtree'],
    'random_state': [8]
}

grid_search = GridSearchCV(estimator=XGBRegressor(learning_rate=0.05,
                                                  max_depth=8,
                                                  min_child_weight=5,
                                                  n_estimators=300,
                                                  gamma=0.1,
```

```
                                                subsample=0.8,
                                                colsample_bytree=0.6,
                                                seed=8,
                                                ),
                                   param_grid=params, n_jobs=4, scoring='neg_mean_absolute_

grid_search.fit(X_train, y_train)
print(grid_search.best_params_, grid_search.best_score_)
```

```
{'booster': 'gbtree', 'eval_metric': 'rmse', 'learning_rate': 0.05, 'objective': 're
g:squarederror', 'random_state': 8, 'reg_alpha': 0.1, 'reg_lambda': 0} -0.2745637143
0471654
```

In [39]:

```
#tuned model MAPE and MAE

print(f'MAPE: {mean_absolute_percentage_error(y_pred, y_test)}, MAE: {mean_absolute
```

```
MAPE: 0.05719580023745668, MAE: 0.272035938048597
```

In [35]:

```
#save graph as image (extremely large image)

img = xgboost.to_graphviz(boost)
img.render('graph', format='png')
```

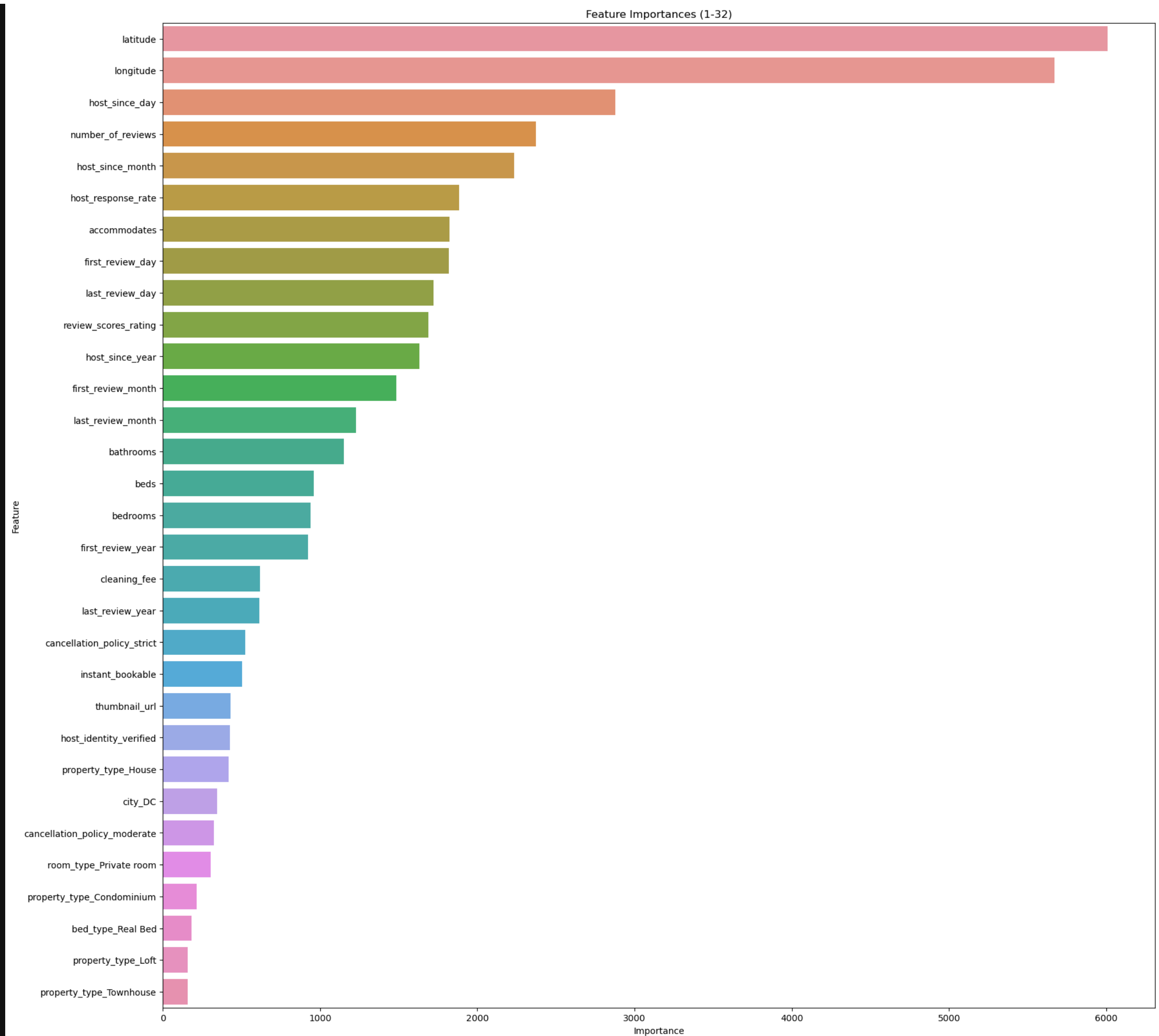
```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.660732 to fit

(process:5328): GLib-GIO-WARNING **: 17:42:19.913: Unexpectedly, UWP app `18184where
where.AndroidAppInstaller_0.1.25.0_x64__4v4sx105x6y4r' (AUMId `18184wherewhere.Andro
idAppInstaller_4v4sx105x6y4r!App') supports 4 extensions but has no verbs
```

Out[35]: 'graph.png'

In [36]:

```
#plot feature importances

feat_importances = pd.DataFrame(boost.get_booster().get_fscore().items(), columns=[

fig, ax = plt.subplots(figsize=(20,20))
sns.barplot(data=feat_importances.iloc[:len(feat_importances)//2], x='Importance',
ax.set_title('Feature Importances (1-32)')
```

Out[36]: Text(0.5, 1.0, 'Feature Importances (1-32)')

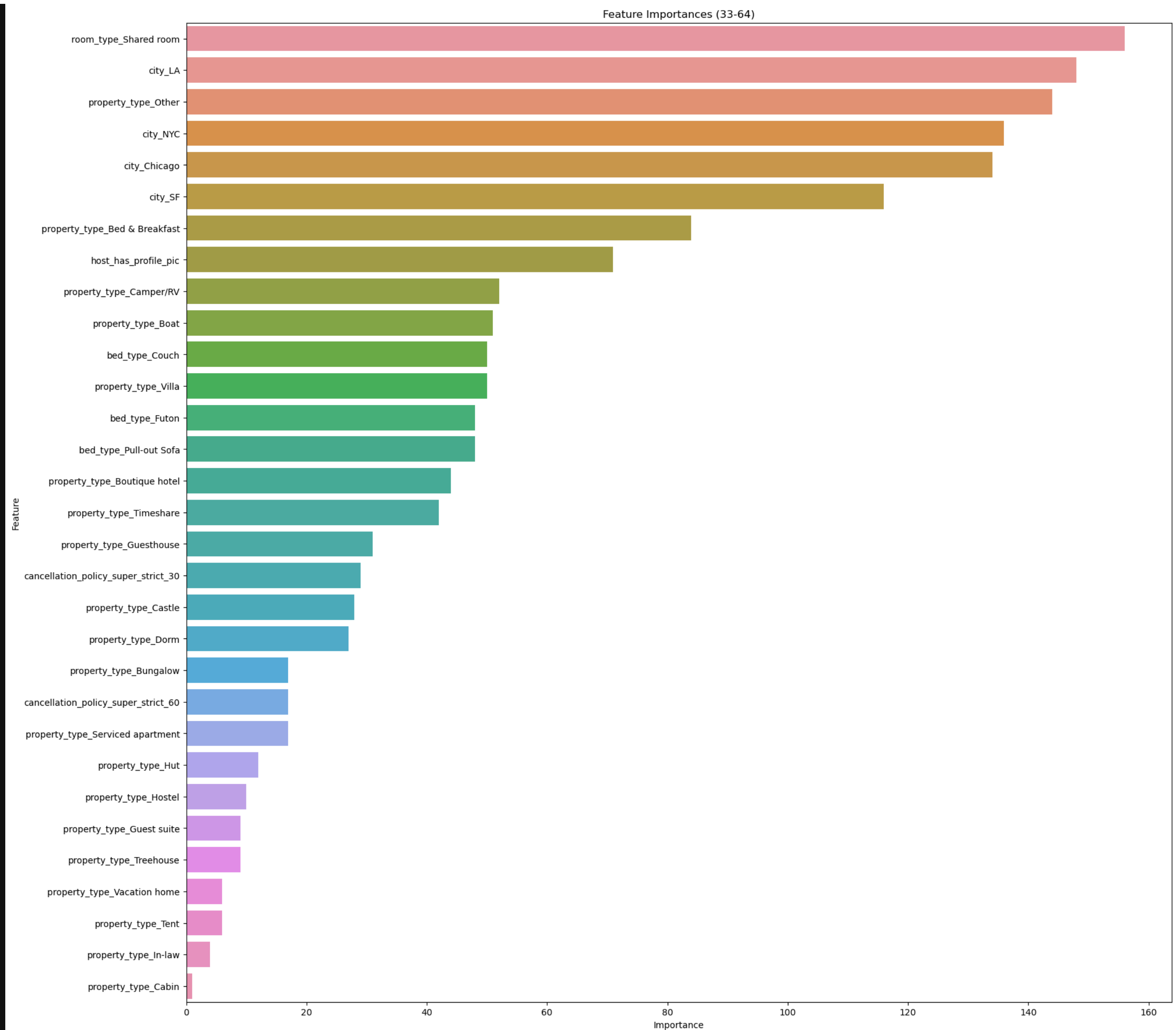


In [38]:

```
#feature importances (cont.)
#NOTE, THESE ARE SMALLER IN IMPORTANCE THAN ABOVE GRAPH DESPITE BAR SIZE

fig, ax = plt.subplots(figsize=(20,20))
sns.barplot(data=feat_importances.iloc[len(feat_importances)//2:], x='Importance',
ax.set_title('Feature Importances (33-64)')
```

Out[38]:

```
Text(0.5, 1.0, 'Feature Importances (33-64)')
```



Feature Importances (33-64)
room_type_Shared room
city_LA
property_type_Other
city_NYC
city_Chicago
city_SF
property_type_Bed & Breakfast
host_has_profile_pic
property_type_Camper/RV
property_type_Boat
bed_type_Couch
property_type_Villa
bed_type_Futon
bed_type_Pull-out Sofa
property_type_Boutique hotel
property_type_Timeshare
property_type_Guesthouse
cancellation_policy_super_strict_30
property_type_Castle
property_type_Dorm
property_type_Bungalow
cancellation_policy_super_strict_60
property_type_Serviced apartment
property_type_Hut
property_type_Hostel
property_type_Guest suite
property_type_Treehouse
property_type_Vacation home
property_type_Tent
property_type_In-law
property_type_Cabin
Feature
0
20
40
60
80
100
120
140
160
Importance