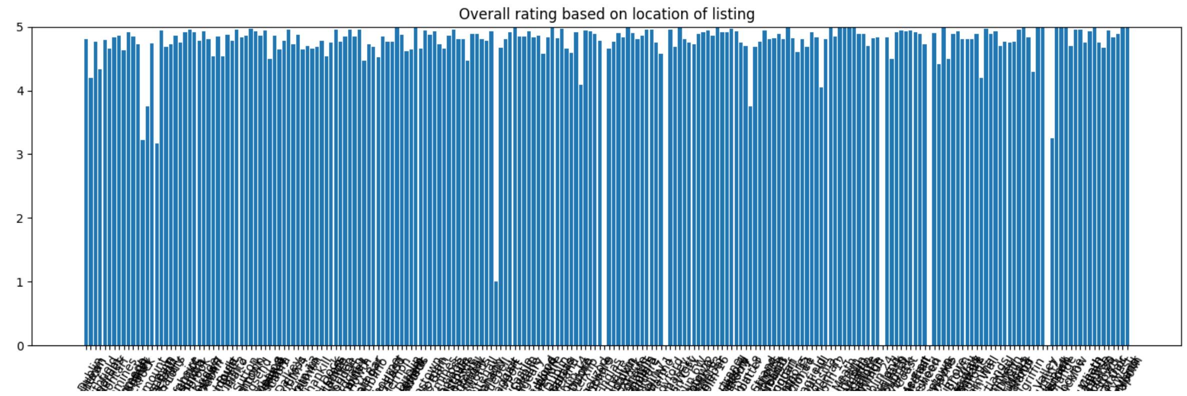
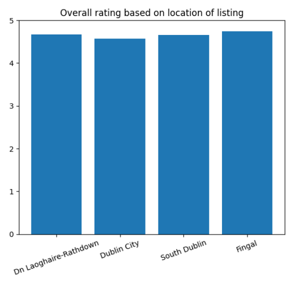
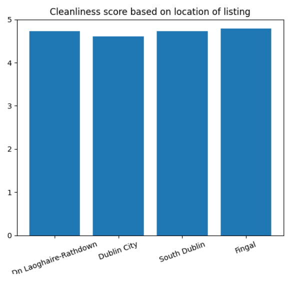
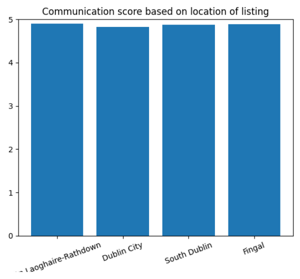
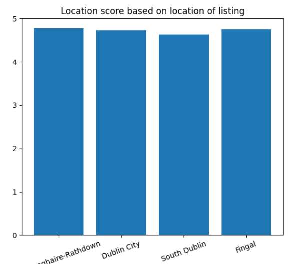
Two datasets were taken from the Inside Airbnb website about Airbnb listings around Dublin (“listings.csv” and “reviews.csv”).
The first dataset contains information regarding each individual listing, the associated descriptions about the property and host, ratings on the following factors; communication, location, value, check-in, cleanliness and accuracy.
The second dataset contains reviews from customers who have stayed in listings on Airbnb’s website. The reviews come in the form of comments left and these can be associated with listings by the listing id.

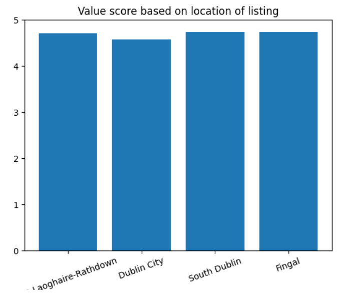


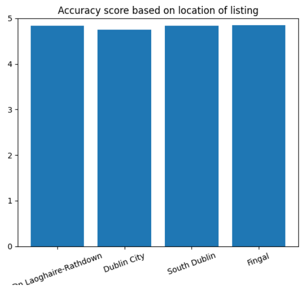


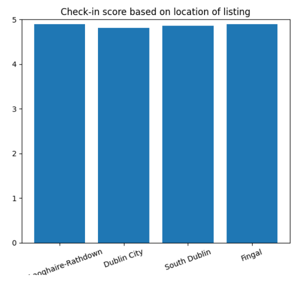












In the graphs above, we see mean ratings for each area in Dublin. On the x axis are the areas in Dublin (the first graph shows each individual area in Dublin the listing is in and the following graphs are separated by their constituencies), and on the y axis is the rating.
There were much fewer areas with low ratings than high ratings which could indicate that people leaving reviews on airbnb are much more likely to leave pleasant reviews.
From our data, we can see that most ratings for listings around Dublin are quite high.

Feature Engineering

In the data provided, some features particularly stood out as being relevant to the rating for a listing. I decided to look at relationships such as whether certain amenities made for more pleasant reviews, the effect of the location on a listings' rating and review comment scores compared to the ratings. The number of ratings on a listing is an important feature to look at also as a larger sample size always makes for more accurate data.

I cleaned up the raw data by removing reviews with null values for comments/ratings and listings with no reviews, since these are the target predictions of the project. I also cleaned data like the price and other similar columns by removing things like the "$" sign and converting to floats etc.
I revised the original listings.csv file to a new file called cleaned_listings.csv that did not contain uninformative columns like "scrape_id" and the various url columns.

In the listings.csv file there is a column called "neighbourhood_cleansed" which divides each listing into which area in Dublin they reside in. The areas in Dublin are split into 4 areas; "Dn Laoghaire Rathdown", "Dublin City", "South Dublin" and "Fingal". I performed one hot encoding on this column in order to feed it as a feature to my regression models. One hot encoding takes categorical data, as is seen in the column in question, and represents each separate variable as its own binary column making it easier to represent this kind of data numerically.

To get a rating for reviews, I applied sentiment analysis to each of the review comments in the "reviews.csv" file. Sentiment analysis is a method of applying a scoring to words depending on their emotion in a sentence and the cumulation of all the words is the "score" for the sentence. If a score is greater than 0, then the sentence is thought to be a positive statement and thus we can treat this as a happy customer. If a score is less than 0, then we think of the sentence being negative. Some error comes into play when we look at the results of the sentiment analysis as most of the negative scorings are given to non-english comments and without having the knowledge of multiple languages, it is hard to know if it is accurate for other languages. Looking at the results of my sentiment analysis I found that there were only around 10K negative comments out of a total of 230K which equates to less than 5% of all reviews. This indicates that most airbnb listings are of a high standard in Dublin (or that people are not likely to leave bad reviews but I will assume the former).

Since the reviews.csv file does not contain the ratings for each category it is hard to gauge a relationship between reviews and ratings as the ratings for listings are averages of all reviews of which the majority are overwhelmingly positive. Thus, it seems unfeasible to predict a listing rating based off of sentiment analysis of the reviews. Still, I performed the sentiment analysis and tried my best to work it into a model of some kind.

Machine Learning Methodology

Since the target variable we are predicting is a floating point value between 0 and 5, I used two distinct linear regression models to predict the different ratings. Linear regression and Lasso regression.

To analyse the accuracy of my regression models for each of the ratings, I plotted the model predictions against the actual values from the data. The resulting graphs show the linear relationships between the ratings and the features.

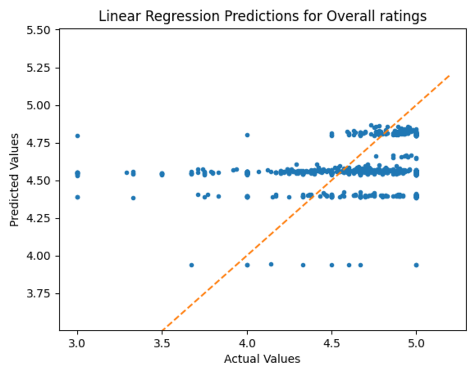
Linear Regression

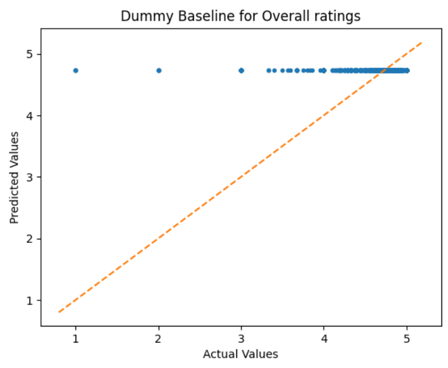
Overall Rating:

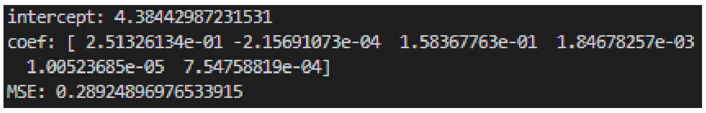
When predicting overall ratings of listings I took 6 features into account namely, “host_is_superhost”, “host_listings_count”, “host_identity_verified”, ”accommodates”, “price” and “number_of_reviews”. I thought that the main features pertaining to the overall rating would relate to the host.

I used a linear regression model with kfold cross validation with k=5. Cross validation splits the raw data presented up into k equal segments and uses the training data to attain the test data, which is then used as training data for the next set of test data.

I plotted my model predictions against actual data to see the accuracy of my model.







```
intercept: 4.38442987231531
coef: [ 2.51326134e-01 -2.15691073e-04  1.58367763e-01  1.84678257e-03
  1.00523685e-05  7.54758819e-04]
MSE: 0.28924896976533915
```

Points that lie near the diagonal line mean that these values were predicted by the Linear Regression model accurately. If the points are away from the diagonal line, the points have been wrongly predicted. Compared to a dummy baseline predicting the mean value rating for all predictions, we can see that the linear regression model performed significantly better because there are more points closer to the perfect prediction line than in the baseline. The mean squared error tells us that our model is accurate to the data provided to it yet it still predicts almost 29% of predictions inaccurately.

Below the graphs are the model parameters of the linear regression model which fit to the equation:

$$Y = B0 + B1X1 + B2X2 + \dots + Bn * Xn$$

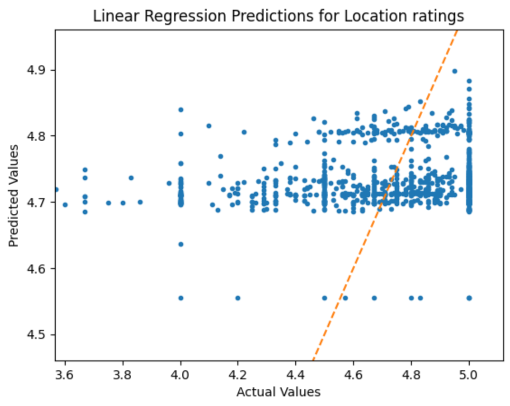
Where B0 is the model intercept and B1,B2,…,Bn are the model coefficients.

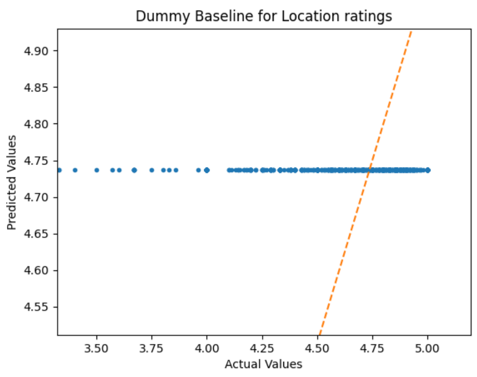
$$Y = 4.38 + 0.25 * X1 - 0.00002 * X2 + 0.16 * X3 + 0.00018 * X4$$

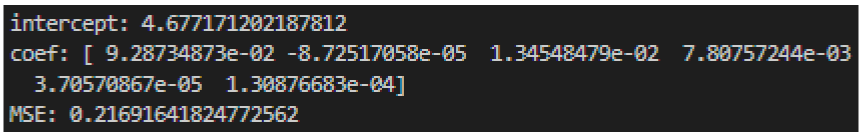
$$+ 0.000001 * X5 + 0.000075 * X6$$

From this equation we can gather that more importance is placed on the features with larger coefficients. X1 and X3 seem to be the most important features for our model. This tells us that “host_is_superhost” and “host_identity_verified” impacted the overall rating of a listing from our dataset.

Location Rating:
To predict location ratings of listings, I looked at the location of the listings by the “neighbourhood_cleansed” column, which separates listings based on what location in Dublin they are. The column was split into 4 columns (one for each area) and fed as parameters along with “accommodates” and “number of reviews”.



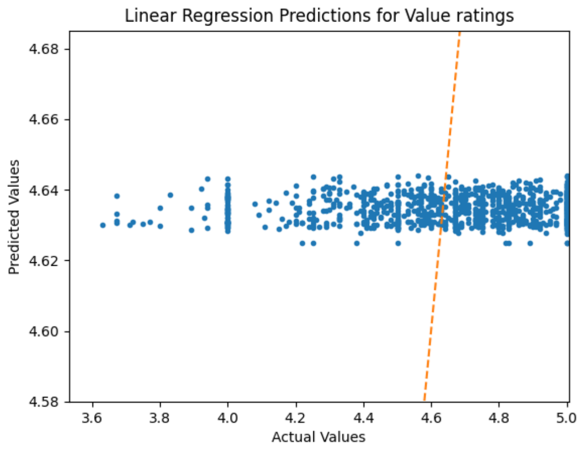


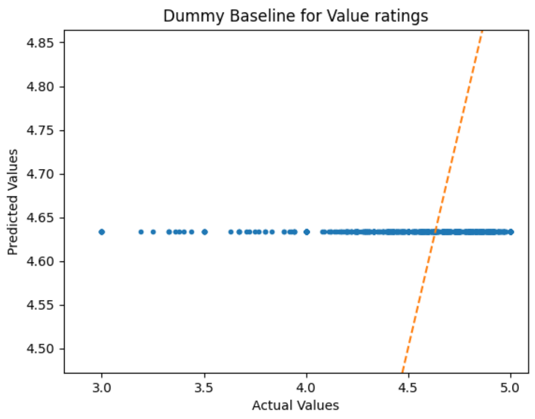


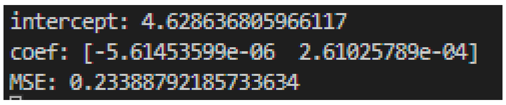
```
intercept: 4.677171202187812
coef: [ 9.28734873e-02 -8.72517058e-05  1.34548479e-02  7.80757244e-03
  3.70570867e-05  1.30876683e-04]
MSE: 0.21691641824772562
```

From the graph above on the left we can see a better correlation between a listings location score and where in Dublin it is along than compared to the overall rating. Small weight is applied to the features, which can be seen by the small values as coefficients below the graphs which is surprising as it indicates that the location rating does not rely on where in Dublin the listing is. The results seen about location rating give us an indication that the location of a listing has a slight impact on the location rating but it is not a clear sign of a high location rating.
Compared to a dummy baseline which predicts the mean location rating, we can see that the linear regression model performs better by having a larger spread of predictions. The MSE tells us that we have a model accuracy of about 79% which is better than the overall ratings model accuracy.

Value Rating:
To predict the rating of a listing, I looked at the price of the listing and its amenities as features. I used a linear regression model with kFold cross validation with k=5. I plotted my model predictions against the actual values taken from the data to see the accuracy of my model.







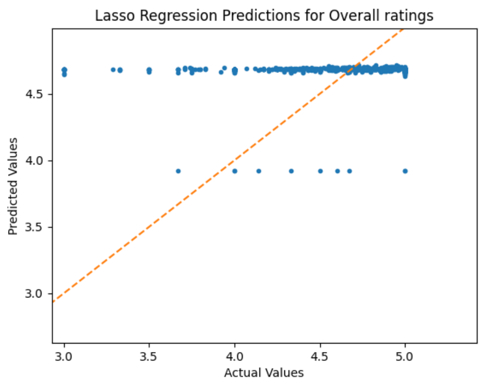
```
intercept: 4.628636805966117
coef: [-5.61453599e-06  2.61025789e-04]
MSE: 0.23388792185733634
```

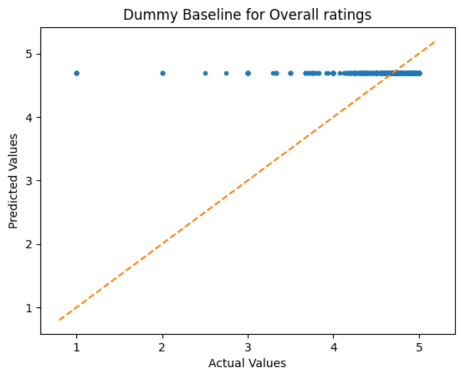
From the graph above on the left we can see that there is no significant correlation between a listings value score and the price and amenities. Very little weight is applied to the features, which can be seen by the small values as coefficients. This result was surprising as I believe that the value rating for a listing would heavily rely on the price of it and the amount of amenities.
Compared to a dummy baseline which predicts the mean value rating, we can see that the performance is similar.

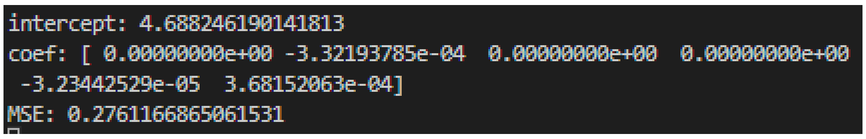
Note on Linear Regression
Through my implementation of Linear regression on the listings dataset, I found that there was very little correlation between a listings' features and their respective ratings.

Lasso Regression
To compare with Linear regression, I carried out the same experiments but with Lasso regression. Lasso regression distinguishes itself from linear regression by introducing regularisation to reduce the complexity of the model and prevent overfitting. Lasso regression also adds a penalty term to the cost function which shrinks the coefficients to zero meaning it will only select a subset of the features to include in the final model. For my model I used a value of alpha=0.1 which I thought was suitable in regularising my model. A larger value for alpha, amplifies the regularisation and in turn, shrinks more features to 0.

Overall Rating:
When predicting overall ratings of listings I used the same features I used above in linear regression. I used a Lasso regression model with kfold cross validation with k=5.

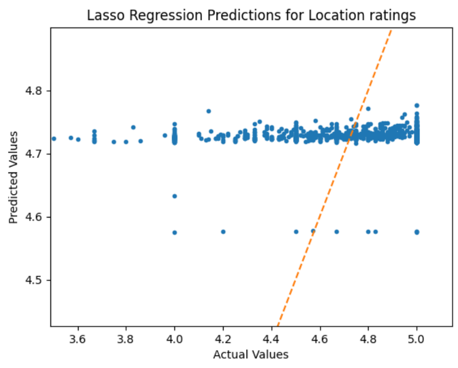


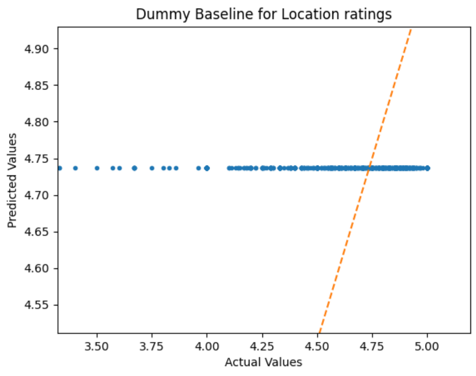


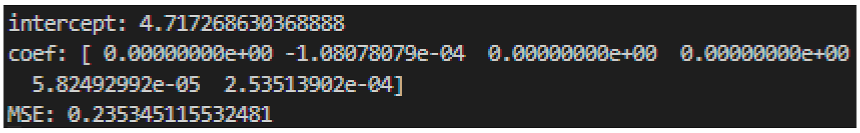

```
intercept: 4.688246190141813
coef: [ 0.00000000e+00 -3.32193785e-04  0.00000000e+00  0.00000000e+00
 -3.23442529e-05  3.68152063e-04]
MSE: 0.2761166865061531
```

Compared to a dummy baseline predicting the mean value rating for all predictions, we can see that the lasso regression model performed quite similarly. There is very little spread on the data and so it would suggest that the addition of regularisation to the model has an adverse effect on the model accuracy for the overall rating for a listing. The effect of shrinkage can be seen on our model coefficients in that many of them have been shrunk to 0. The mean squared error tells us that our model is accurate to the data provided to it with a scoring of 0.28 (the best score for MSE is 0 which indicates no error in predictions). It is hard to see this from the graph as it closely resembles the baseline model but this indicates little relationship between a listings' features and its overall rating.

Location Rating:
I used the same feature selection as for my linear regression model and the same method of one hot encoding of the neighbourhoods.

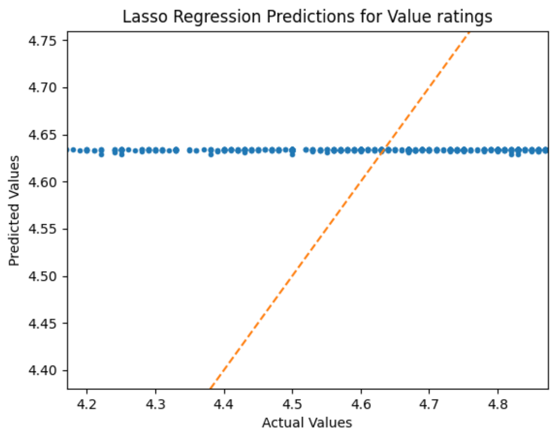


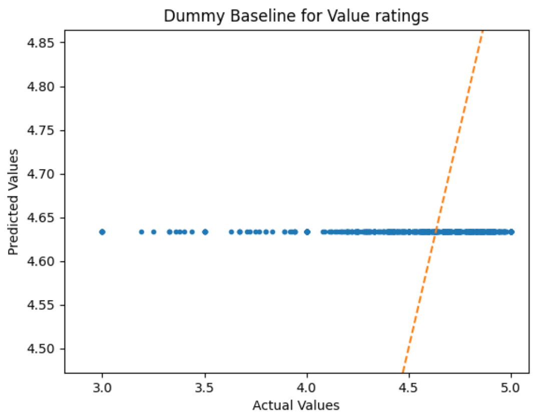


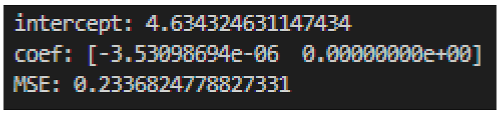

```
intercept: 4.717268630368888
coef: [ 0.00000000e+00 -1.08078079e-04  0.00000000e+00  0.00000000e+00
  5.82492992e-05  2.53513902e-04]
MSE: 0.235345115532481
```

From the graph above on the left we can see similar results to that of the predictions for overall ratings. Small weight is applied to the features, which can be seen by the small values as coefficients below the graphs. Most weight is applied to the feature relating to “Dublin City”, which makes sense as the majority of listings are from this area. The results seen about location rating give us an indication that the location of a listing has a slight impact on the location rating but not enough to drastically change a listings rating.
Compared to a dummy baseline which predicts the mean location rating, we can see that the Lasso regression model performs similarly but with slight variance of the mean.

Value Rating:
To predict the rating of a listing, I looked again at the same features used in Linear regression. I used a Lasso regression model with alpha=0.1 and kFold cross validation with k=5. I plotted my model predictions against the actual values taken from the data to see the accuracy of my model.



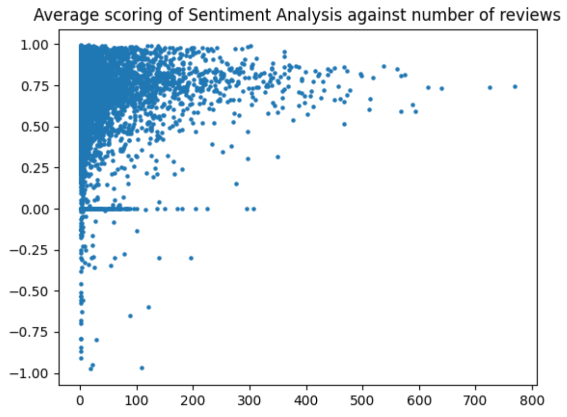




```
intercept: 4.634324631147434
coef: [-3.53098694e-06  0.00000000e+00]
MSE: 0.2336824778827331
```

The results from this prediction were the most surprising to me. We can see that the Lasso model shrunk the weight for the amenities feature to zero and there is little weight put on the price feature. One would assume that the value ratings for a listing would heavily depend on the price of the listing but the performance of the model shows that these features have little impact on the ratings. The Lasso model behaves almost identically to the baseline model. This gives further evidence that the price feature does not impact a listing’s value rating as much as one would expect.

I plotted the average sentiment against the number of reviews for a listing and we can see clearly in the graph below that there is a relationship between the number of reviews and a listings’ average review sentiment. It was immediately clear that there was a relationship that indicates that for listings with more reviews, the average sentiment is going to increase which leads me to believe that airbnb reviews are overwhelmingly positive. This adds to the findings of my models in that there is little relationship between ratings and listings.



I found that due to the data provided, it proved quite difficult to make accurate predictions on ratings for listings given that the vast majority of the reviews are positive. There is not enough data provided from the individual reviews to make predictions on listings based off of their sentiment alone. Taking the average sentiment of individual listings was as close as I could imagine would represent a rating prediction but still it is hard to gauge a distinct relationship like this.

Note on Lasso Regression
Through my implementation of Lasso regression on the listings dataset, I found that there was little correlation between a listings' features and their respective ratings.

Evaluation/Conclusion
In this analysis, we compared the performance of a linear regression model and a lasso regression model on a dataset of Airbnb listings and reviews. The linear regression model was found to be slightly more revealing than the lasso regression model. Both models performed quite similarly in terms of model accuracy and led me to believe that there is little importance put on individual listings' features in regard to its rating.

The models shown above show us that there is very little relationship between Airbnb ratings and the information on their respective listings. This leads me to the conclusion that it is unfeasible to predict the ratings based on listing information as there is not enough data on each review to take into account.

Appendix of code

Imports

```
import pandas as pd
import numpy as np
import tqdm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.dummy import DummyRegressor
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from nltk.sentiment import SentimentIntensityAnalyzer
```

Linreg.py - implementation of linear regression on overall rating

```
df=pd.read_csv('../data/cleaned_listings.csv')
df=df.dropna()
df.reset_index(drop=True, inplace=True) # remove unimportant columns
df2=pd.DataFrame()
df2 = df.loc[:,'host_is_superhost':'number_of_reviews']  # Select columns by range
print(df2)
X=df2.to_numpy()
y=df['review_scores_location']
print(y)
def predict():
    kf = KFold(n_splits=5)
    for train, test in kf.split(X):
        model = LinearRegression().fit(X[train], y[train])
        ypred = model.predict(X[test])
        dummy = DummyRegressor(strategy="mean").fit(X[train], y[train])
        ydummy = dummy.predict(X[test])
    print("intercept: " + str(model.intercept_))
    print("coef: "+ str(model.coef_))
    print("MSE: "+ str(mean_squared_error(y[test],ypred)))
    plt.plot(y[test],ypred,'.')
    # plt.xlim(4.2,5)
    # plt.ylim(4.2,5)
    xmin, xmax = plt.xlim()
    plt.plot([xmin, xmax], [xmin, xmax], '--')
    plt.ylim(0,5)
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Linear Regression Predictions for Location ratings')
    plt.show()
    plt.plot(y[test],ydummy,'.')
    # plt.xlim(4.2,5)
    # plt.ylim(4.2,5)
    xmin, xmax = plt.xlim()
    plt.plot([xmin, xmax], [xmin, xmax], '--')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Dummy Baseline for Overall ratings')
    plt.show()
```

Location.py - onehot encoding and linreg on location rating

```python
df=pd.read_csv("../data/listings.csv")
df=df.dropna(subset=['last_review']) ## drop all df that do not have any reviews as not
relevant
df.reset_index(drop=True, inplace=True)
nhbrd=[]
df2=df['neighbourhood_cleansed'].dropna()
df2.reset_index(drop=True, inplace=True)
print(df2)
for i in df2:
    area = i.split(',')[0]
    # print(i)
    if i not in nhbrd:
        nhbrd.append(i)
area=[i.split(',')[0] for i in nhbrd]
df=pd.read_csv('../data/cleaned_listings.csv')
print(df)
df2=pd.DataFrame()
df2 = df.loc[:,'host_is_superhost':'number_of_reviews']  # Select columns by range
print(df2)
X=df2.to_numpy()
y=df['review_scores_rating']
print(y)
def raw():
    d={}
    #averages of each respective scoring for each of the areas in dublin
    compare='location'
    for i in nhbrd:
        score=df.loc[df['neighbourhood_cleansed']==i]['review_scores_' + compare].mean()
        d.update({i.split(',')[0]:score})

    # copy=d.copy()
    # for k,v in copy.items():
    #     if v<4.5:
    #         d.pop(k)
    plt.title('Location rating based on location of listing')
    plt.ylim(4,5)
    plt.bar(*zip(*d.items()))
    plt.xticks(rotation=20,fontsize=10)
    plt.show()
    print(d)
def onehot():
    df2=df[['neighbourhood_cleansed', 'accommodates', 'number_of_reviews',
'review_scores_location']].dropna()
    onehot_encoder = OneHotEncoder(sparse=False)
    onehot_encoder.fit(df2[['neighbourhood_cleansed']])
    one_hot_features = onehot_encoder.transform(df2[['neighbourhood_cleansed']])
    df2['one_hot']=one_hot_features
    print(df2)
```

Value.py - linreg on value rating

```python
df=pd.read_csv('../data/listings.csv')
df=df.dropna(subset=['review_scores_value'])

amenities = df.loc[:,'amenities']
price = df.loc[:,'price']
n_amenities=[]
```

```python
for i in amenities:
    i = i.split(',')
    n_amenities.append(len(i))

p=[]
for i in price:
    i = i.replace('$','')
    i = i.replace(',','')
    p.append(float(i))

df2=pd.DataFrame()
df2['n_amenities']=n_amenities
df2['price']=p
df2['value_score']=df['review_scores_value'].copy()
df2=df2.dropna()
df2.reset_index(drop=True, inplace=True)
X1 = df2['price']
X2 = df2['n_amenities']
y = df2.loc[:, 'value_score']
X = np.column_stack((X1, X2))

def graph2dprices():
    fig = plt.figure()
    ax = fig.add_subplot(111)
    plt.scatter(X[:, 0], y, c='b', label='data',s=5)
    plt.xlabel("prices ($)")
    plt.xlim(0,1000)
    plt.ylabel("rating")
    plt.title("price vs rating")
    plt.show()

def graph2damenities():
    fig = plt.figure()
    ax = fig.add_subplot(111)
    plt.scatter(X[:, 1], y, c='b', label='data',s=5)
    plt.xlabel("number of amenities")
    # plt.xlim(0,100)
    plt.ylabel("rating")
    plt.title("amenities vs rating")
    plt.show()

def predict():
    kf = KFold(n_splits=5)
    for train, test in kf.split(X):
        model = Lasso(alpha=10).fit(X[train], y[train])
        ypred = model.predict(X[test])
        dummy = DummyRegressor(strategy="mean").fit(X[train], y[train])
        ydummy = dummy.predict(X[test])
    print("intercept: " + str(model.intercept_))
    print("coef: "+ str(model.coef_))
    print("MSE: "+ str(mean_squared_error(y[test],ypred)))

    plt.plot(y[test],ypred,'.')
    # plt.xlim(4.2,5)
    # plt.ylim(4.2,5)
    xmin, xmax = plt.xlim()
    plt.plot([xmin, xmax], [xmin, xmax], '--')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Lasso Regression Predictions for Value ratings')
    plt.show()

    plt.plot(y[test],ydummy,'.')
    # plt.xlim(4.2,5)
    # plt.ylim(4.2,5)
    xmin, xmax = plt.xlim()
    plt.plot([xmin, xmax], [xmin, xmax], '--')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Dummy Baseline for Value ratings')
    plt.show()

def baseline():
    kf = KFold(n_splits=5)
    for train, test in kf.split(X):
        model = LinearRegression().fit(X[train], y[train])
        ypred = model.predict(X[test])
    print("intercept: " + str(model.intercept_))
    print("coef: "+ str(model.coef_))
    print("MSE: "+ str(mean_squared_error(y[test],ypred)))

    plt.plot(y[test],ypred,'.')
    # plt.xlim(4.2,5)
    # plt.ylim(4.2,5)
    xmin, xmax = plt.xlim()
    plt.plot([xmin, xmax], [xmin, xmax], '--')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Linear Regression Predictions for Value ratings')
    plt.show()
```

Sentiment_analysis.py - sentiment analysis on reviews

```
df=pd.read_csv("../data/reviews.csv")
df=df.dropna() # cleanup data by removing reviews with null values
df.reset_index(drop=True, inplace=True)
res={}
sia=SentimentIntensityAnalyzer()
for i, row in tqdm(df.iterrows(), total=df.shape[0]):
    comment=row['comments']
    id=row['id'] # id of review
    res[id]=sia.polarity_scores(comment)
vaders=pd.DataFrame(res).T
vaders.to_csv('../data/review_sentimentanalysis.csv')
```

Scores.py - add sentiment analysis to reviews.csv

```
df=pd.read_csv("../data/review_sentimentanalysis.csv")
df2=pd.read_csv("../data/reviews.csv")
df2=df2.dropna() # cleanup data by removing reviews with null values
df2.reset_index(drop=True, inplace=True)
df2['score']=df['compound']
df2=df2.sort_values(by=['id'])
df2.to_csv('../data/reviews_with_score.csv')
```

Lasso.py - implementation of lasso regression model

```
df=pd.read_csv('../data/cleaned_listings.csv')
df=df.dropna()
df.reset_index(drop=True, inplace=True) # remove unimportant columns
df2=pd.DataFrame()
df2 = df.loc[:,'host_is_superhost':'number_of_reviews']  # Select columns by range
print(df2)
X=df2.to_numpy()
y=df['review_scores_location']
print(y)
def predict():
    kf = KFold(n_splits=5)
    for train, test in kf.split(X):
        model = Lasso(alpha=0.1).fit(X[train], y[train])
        ypred = model.predict(X[test])
        dummy = DummyRegressor(strategy="mean").fit(X[train], y[train])
        ydummy = dummy.predict(X[test])
    print("intercept: " + str(model.intercept_))
    print("coef: "+ str(model.coef_))
    print("MSE: "+ str(mean_squared_error(y[test],ypred)))
    plt.plot(y[test],ypred,'.')
    # plt.xlim(4.2,5)
    # plt.ylim(4.2,5)
    xmin, xmax = plt.xlim()
    plt.plot([xmin, xmax], [xmin, xmax], '--')
    plt.ylim(0,5)
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Lasso Regression Predictions for Location ratings')
    plt.show()

    plt.plot(y[test],ydummy,'.')
    # plt.xlim(4.2,5)
    # plt.ylim(4.2,5)
    xmin, xmax = plt.xlim()
    plt.plot([xmin, xmax], [xmin, xmax], '--')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Dummy Baseline for Location ratings')
    plt.show()
```

clean.py - cleaned the listings.csv and save to new file

```
df=pd.read_csv("../data/listings.csv")
df=df.dropna(subset=['last_review']) ## drop all rows without reviews as not relevant
df=df.drop(columns=['listing_url','scrape_id','last_scraped','source','name','description',
'neighborhood_overview','picture_url','host_url','host_name','host_since','host_location','
host_about','host_response_time','host_response_rate','host_acceptance_rate','host_thumbnai
l_url','host_picture_url','host_neighbourhood','calculated_host_listings_count_shared_rooms
','calculated_host_listings_count_private_rooms','host_verifications','license','instant_bo
okable','calculated_host_listings_count_entire_homes','host_has_profile_pic','host_id','cal
culated_host_listings_count','first_review','number_of_reviews_ltm','number_of_reviews_l30d
','calendar_last_scraped','has_availability','availability_30','availability_60','availabil
ity_90','availability_365','maximum_nights_avg_ntm','minimum_nights_avg_ntm','maximum_maxim
um_nights','minimum_maximum_nights','maximum_minimum_nights','minimum_minimum_nights','maxi
mum_nights','minimum_nights','bathrooms_text','room_type','property_type','longitude','lati
```

```python
tude','neighbourhood_group_cleansed','neighbourhood','calendar_updated','host_total_listing
s_count','neighbourhood_cleansed','bathrooms','amenities','last_review','review_scores_accu
racy','review_scores_value','review_scores_checkin','review_scores_communication','review_s
cores_cleanliness','beds','bedrooms'])
df.dropna()
df.reset_index(drop=True, inplace=True)
df.host_is_superhost = df.host_is_superhost.replace({'t': 1, 'f': 0})
df.host_identity_verified = df.host_identity_verified.replace({'t': 1, 'f': 0})
price = df.loc[:,'price']
p=[]
for i in price:
    i = i.replace('$','')
    i = i.replace(',','')
    p.append(float(i))
df['price']=p
df.to_csv('../data/cleaned_listings.csv')
print(df)
```