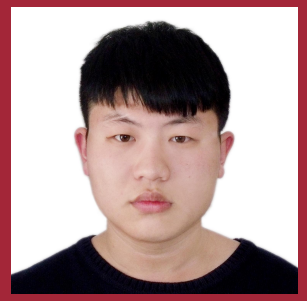
Score Estimation For New Airbnb Listings

CS513-A Final Project

Group 8

Deng Xiong
10402740

He Zhao
20009673



Qinwen Pan
20007132

Yu-Lin Liu
20008573

Data Source

Inside Airbnb

Inside Airbnb is an independent, non-commercial set of tools and data that allows you to explore how Airbnb is really being used in cities around the world.

By analyzing publicly available information about a city's Airbnb's listings, Inside Airbnb provides filters and key metrics so you can see how Airbnb is being used to compete with the residential housing market.

With Inside Airbnb, you can ask fundamental questions about Airbnb in any neighbourhood, or across the city as a whole. Questions such as:

- "How many listings are in my neighbourhood and where are they?"
- "How many houses and apartments are being rented out frequently to tourists and not to long-term residents?"
- "How much are hosts making from renting to tourists (compare that to long-term rentals)?"
- "Which hosts are running a business with multiple listings and where they?"

We pulled the dataset of NYC posts.

Data Description

Features:

- **Host Response Time**
- **Host Response Rate**
- **Host Verifications**
- **Property Type**
- **Room Type**
- **Bed Type**
- **Amenities**
- **Availability 365**
- **Cancellation Policy**
- **Features**
- **Rating**
- **…**

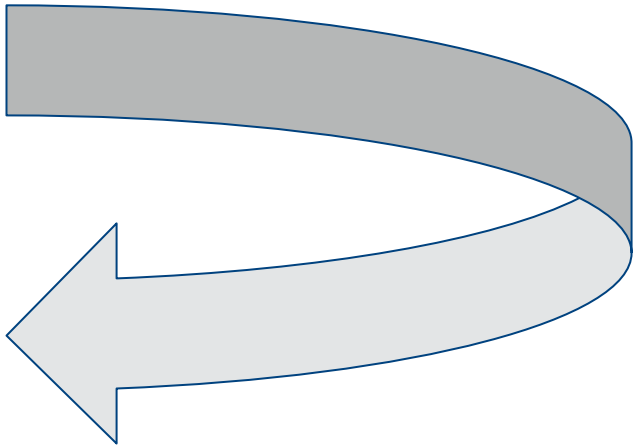
	Country	City	Name	Summary	Space
1	United States	New York	Best place to stay in NYC Brooklyn	This is a beautiful bedroom in bay...	
2	United States	New York	Super nice studio!only females	Very nice, clean and cozy baseme...	Warm, clean, perfect locat
3	United States	New York	Battery Park/Sweet Home Apartm...	我的房源靠近Wall St 世贸 自由女神...	Located in heart of the Fin
4	United States	New York	Loft Like Studio in Luxury Building	Luxury condo like	The ultimate in sophisticat
5	United States	New York	Battery Park, New York City	ladies or couples only. shared apt...	The space is conveniently
6	United States	New York	Quiet 1 bedroom with water views...	My place is close to World Trade ...	Great light and views
7	United States	New York	Unique NYDesigner apt min from ...	If you are looking for a special sp...	900 sq feet ground floor f
8	United States	New York	Large Sunny room in 4BR apartm...	Large room in trendy Brooklyn nei...	Large 4 bedroom, exposed
9	United States	New York	Massive One Bedroom with Elevat...	My place is close to Union Square...	My home is a spacious, lig
10	United States	New York	Great 1 Bedroom in Central Locati...	This one bedroom apartment is in...	Awesome and spacious 1 b
11	United States	New York	1 bedroom apt at Thompson Sq P...	Great 1 bedroom apartment in ea...	
12	United States	New York	Charming 2BR Apartment with Gr...	Beautiful and charming New York ...	Beautiful New York City 2
13	United States	New York	Big East Village bedroom, heart o...	Enjoy Manhattan the right way, by...	Location, location, locatior
14	United States	New York	Huge Luxury EV 1 Bed Steps to P...	• Steps (12) from Tompkins Squar...	Steps away from Tompkins
15	United States	New York	Charming, Cozy 1br East Village A...	Great location! This is my home, ...	The bedroom in the back i
16	United States	New York	Sun-drenched East Village Penth...		Sun-drenched East Village
17	United States	New York	Fresh East Village Apartment with...	Beautiful, sunny penthouse apart...	The apartment itself is loc
18	United States	New York	Charming East Village Apartment		Located in the heart of the
19	United States	New York	Perfect East Village Adventure	My place offers a great way to ex...	The space The Building: T
20	United States	New York	Just Remodeled Apt ❤️ of ABC City	Our cool and quiet apartment is l...	
21	United States	New York	Cozy East-Village Walk, Blocks fr...	My place is close to Momofuku N...	

Data Description

CSV Whole dataset Only the 19528 selected records

CSV uses semicolon (;) as a separator.

- **85 features**
- **19528 records**



Data processing

- **35 features**
- **14672 records**

Data Processing

1. Entries with essential information missing, such as Price, Review Ratings, are removed
2. Entries that are text data are removed
3. Entries with integer values missing are filled with the median from other entries
4. Entries with floating point values missing are filled with the mean from other entries
5. Time stamps are converted to time spans,
   for example, Host Since are replaced with Host Time calculated by last seen time minus Host Since
6. Categorical data are reduced and encoded in one-hot format,
   For example, House, Townhouse, Villa are generalized to be Building,
   where Apartment, Condominium, Hostel and Guesthouse are generalized to be Unit
7. The numerical data are standardized (mean = 0, std = 1)
8. The target value, Review Ratings, are binarized by *1.0 if Ratings > 0.95, 0.0 otherwise*

Problem Description

- For a new listing being created, it’s hard to determine its quality.
- In this project, we explored techniques that help predicting the quality of new listings using the metadata of listings.

Models

- KNN
- k-Means Clustering
- Naïve Bayes
- Decision Tree CART
- C5.0
- Random Forest
- ANN
- SVM

KNN

Factor: Review.Scores

Latitude	Bed Type
Longitude	Room Type
Accommodates	Cancellation Policy
Bathrooms	Cleaning Fee
Bedrooms	Number of reviews
Beds	Reviews per Month
PersonPrice	Property Type_Building
Security Deposit	

KNN_k=3

```
> fit <- fitted(predict_k1)
> table(test$Review.Scores.Rating,fit)
   fit
       0    1
  0 1482  908
  1 1001 1011
>
```

```
> right<- ( test$Review.Scores.Rating==fit)
> rate<-sum(right)/length(right)
> rate
[1] 0.5663335
>
```

KNN_k=5

```
> table(test$Review.Scores.Rating,fit)
   fit
       0    1
  0 1538  880
  1 1042  942
>
```

```
> right<- ( test$Review.Scores.Rating==fit)
> rate<-sum(right)/length(right)
> rate
[1] 0.5633803
```

KNN_k=7

```
> table(test$Review.Scores.Rating,fit)
   fit
      0    1
  0 1549  850
  1 1069  934
>
> right<- ( test$Review.Scores.Rating==fit)
> rate<-sum(right)/length(right)
> rate
[1] 0.5640618
```

k- Means Clustering

Factor: Review.Scores

PersonPrice	Bedrooms
Accommodates	Beds
Property Type	Latitude
Room Type	Longitude
Cancellation Policy	Bathrooms

k- Means Clustering

k = 2, The central points:

```
K-means clustering with 2 clusters of sizes 4128, 10544

Cluster means:
  PersonPrice Accommodates Property.Type_Building Property.Type_Unit Room.Type_Entire.home.apt
1   103.08863     2.321948             0.02737403          0.9726260                 0.7114826
2    45.75964     3.087917             0.03347876          0.9665212                 0.5275038
  Room.Type_Private.room Room.Type_Shared.room Cancellation.Policy_flexible Cancellation.Policy_moderate
1              0.2727713            0.01574612                    0.2165698                    0.2470930
2              0.4388278            0.03366844                    0.2177542                    0.2413695
  Cancellation.Policy_strict Latitude Longitude Bathrooms Bedrooms     Beds
1                  0.5363372 40.74990 -73.98455  1.099564 1.002422 1.358527
2                  0.5408763 40.77059 -73.96900  1.062690 1.128035 1.598919
```

k- Means Clustering

```
> table(kmodel$cluster, data$Review.Scores.Rating)

     0    1
1 1819 2309
2 6223 4321
```

0: Rating is under 95.
1: Rating is above 95.

```
Within cluster sum of squares by cluster:
[1] 6469173 2229263
 (between_SS / total_SS =  52.9 %)
```

between_SS/total_SS: 52.9%

We can predict the rating of a new post with which central point is nearer to the new post.

Naïve Bayes

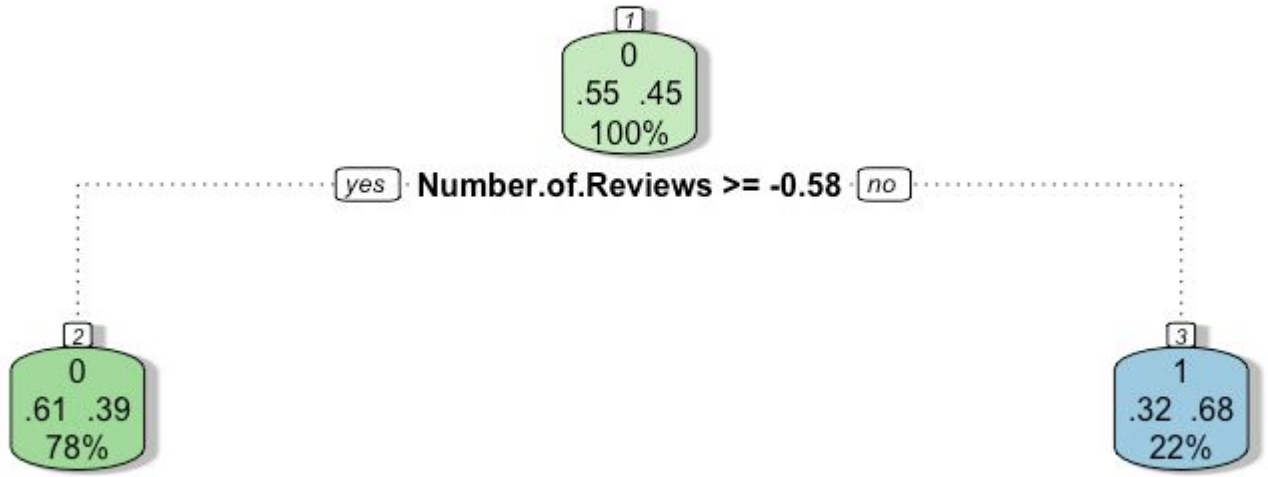
```
df$High.Review.Score<-factor(df$High.Review.Score, levels = c("0","1"), labels = c("low","high"))
is.factor(df$High.Review.Score)
df$High.Review.Score

sampleSize <- floor(0.70 * nrow(df))

set.seed(100)
train <- sample(seq_len(nrow(df)), size = sampleSize)

trainingData <- df[train, ]
testData <- df[-train, ]

NB<- naiveBayes(High.Review.Score ~ ., data = trainingData)

predictNB <- predict(NB, testData)

conf_matrix <- table(predictNB=predictNB,class=testData$High.Review.Score)
print(conf_matrix)


#Accuracy and error rating
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(conf_matrix)
```

```
> conf_matrix <- table(predictNB=predictNB,class=testData$High.Review.Score)
> print(conf_matrix)
         class
predictNB  low high
     low   948  398
     high 1477 1579
>
>
> #Accuracy and error rating
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(conf_matrix)
[1] 57.40572
```

Dtree CART

no latitude and longitude

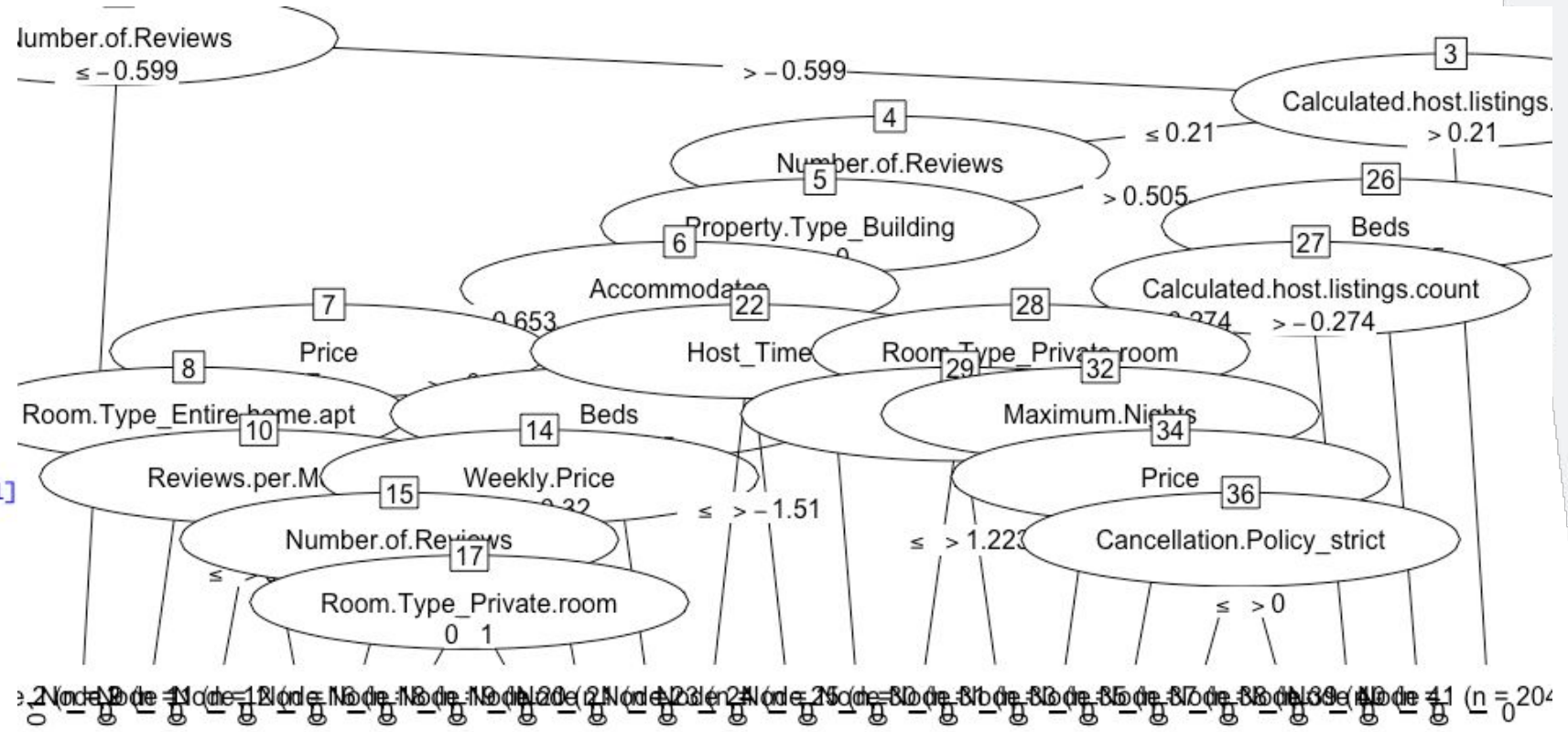
```
      prediction
Actual    0    1
     0 2080  343
     1 1353  625
>
> # Error rate of DT
> right<-(testing[,31]==prediction)
> accuracy<-sum(right)/length(right)
> accuracy
[1] 0.614633
>
```



C 5.0

no latitude and longitude

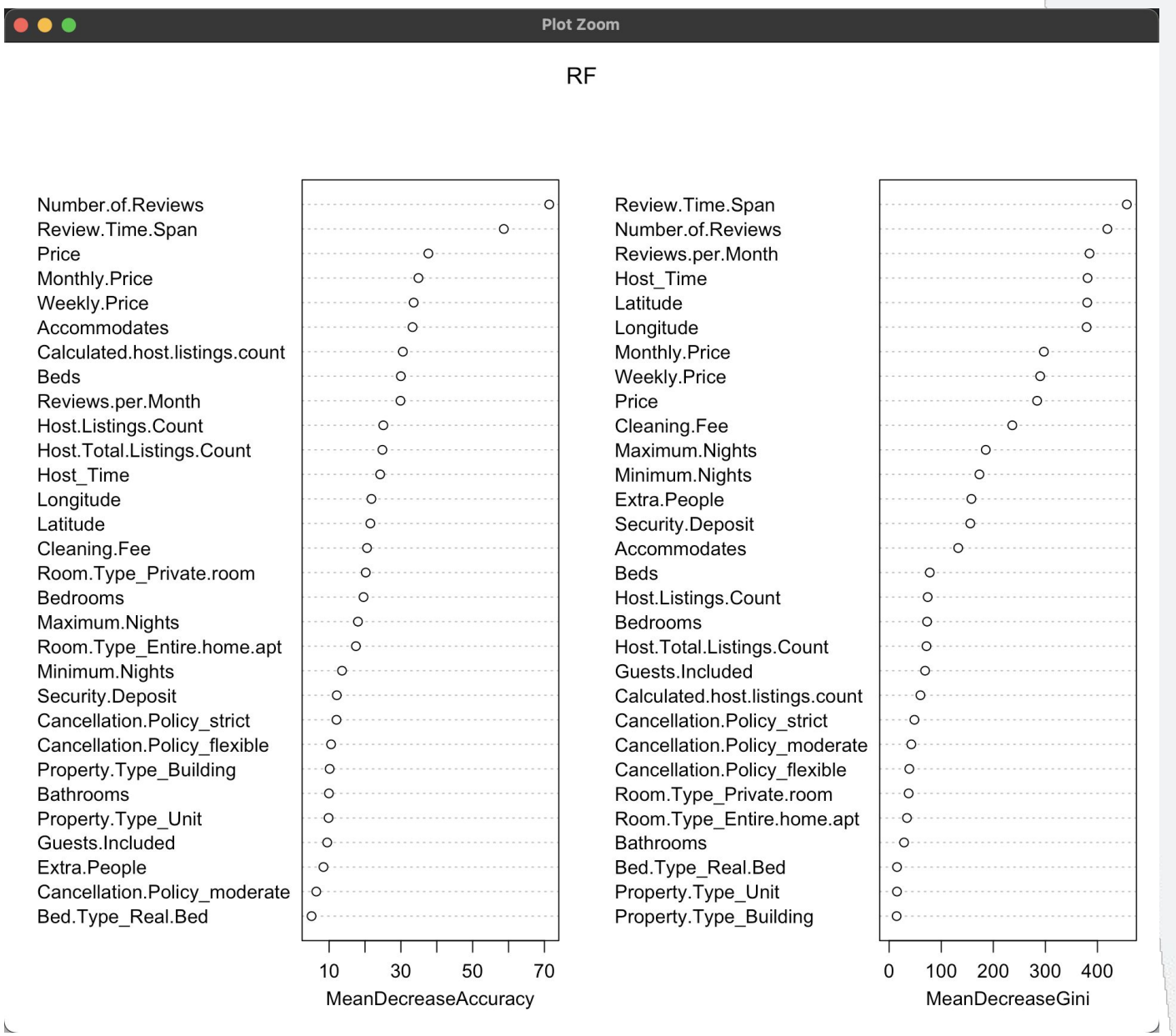
```
> table(actual=testing[,31],Predict)
      Predict
actual    0    1
     0 1425  574
     1  816  853
>
> #Error rate
> error <- (testing[,31]!=Predict)
> errorrate<-sum(error)/length(testing[,31])
> accuracy<- 1-errorrate
> accuracy
[1] 0.6210469
>
```



Random Forest

```
df$High.Review.Score<-factor(df$High.Review.Score, levels = c("0","1"), labels = c("low","high"))
is.factor(df$High.Review.Score)
df$High.Review.Score
View(df)

split_size<-floor(0.70*nrow(df))

random_sample<-sample(seq_len(nrow(df)), size = split_size)

train<-df[random_sample,]
test<-df[-random_sample,]

RF<-randomForest(High.Review.Score~.,train, importance=TRUE, ntree=1000)
importance(RF)
varImpPlot(RF)

# Prediction
pred_RF<-predict(RF,test,type = "class")
length(pred_RF)
length(test)

#Confusion Matric
confMat_RF<-table(test$High.Review.Score,pred_RF)
print(confMat_RF)

accuracy<-function(x){
  sum(diag(x)/sum(rowSums(x)))*100
}

# Accuracy
accuracy(confMat_RF)
```

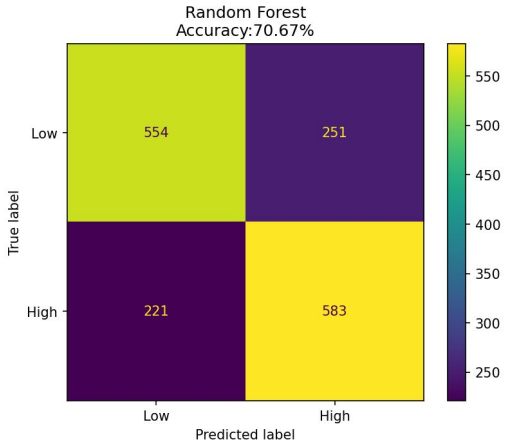
```
> confMat_RF<-table(test$High.Review.Score,pred_RF)
> print(confMat_RF)
      pred_RF
        low high
  low  1721  671
  high  963 1047
>
> accuracy<-function(x){
+   sum(diag(x)/sum(rowSums(x)))*100
+ }
>
> # Accuracy
> accuracy(confMat_RF)
[1] 62.88051
```

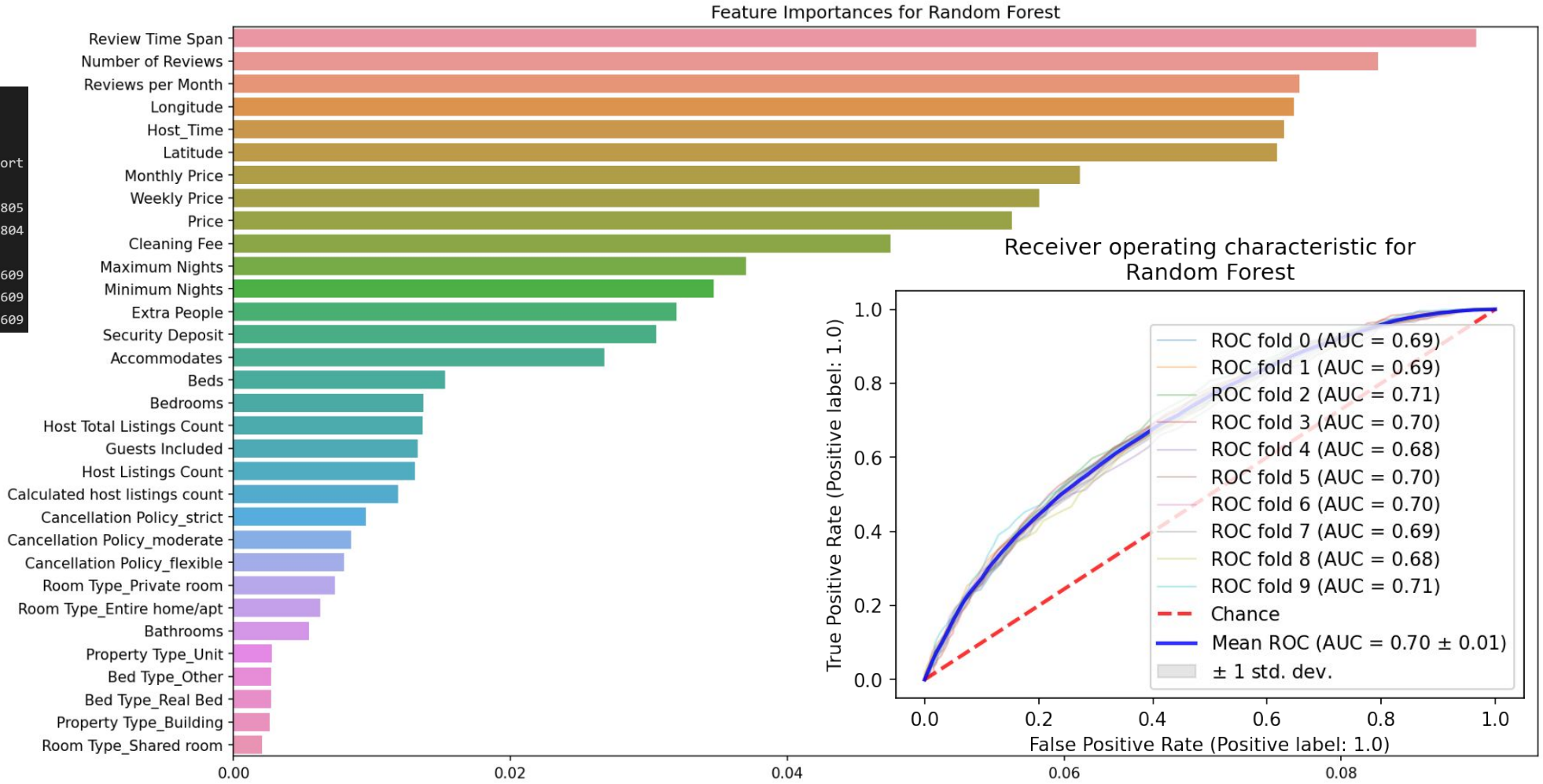


Random Forest

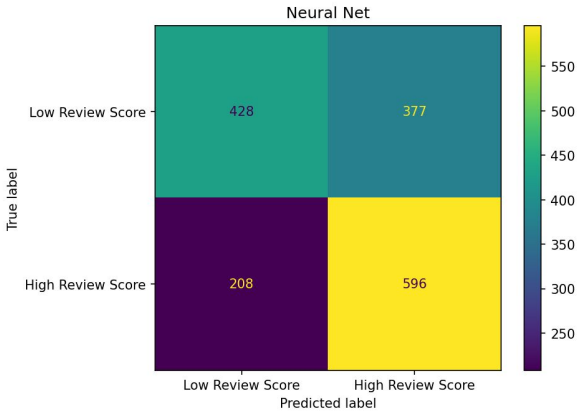
Random Forest
Accuracy : 0.706650093225606
Balanced Accuracy : 0.706661567936714

	precision	recall	f1-score	support
Low	0.71	0.69	0.70	805
High	0.70	0.73	0.71	804
accuracy			0.71	1609
macro avg	0.71	0.71	0.71	1609
weighted avg	0.71	0.71	0.71	1609

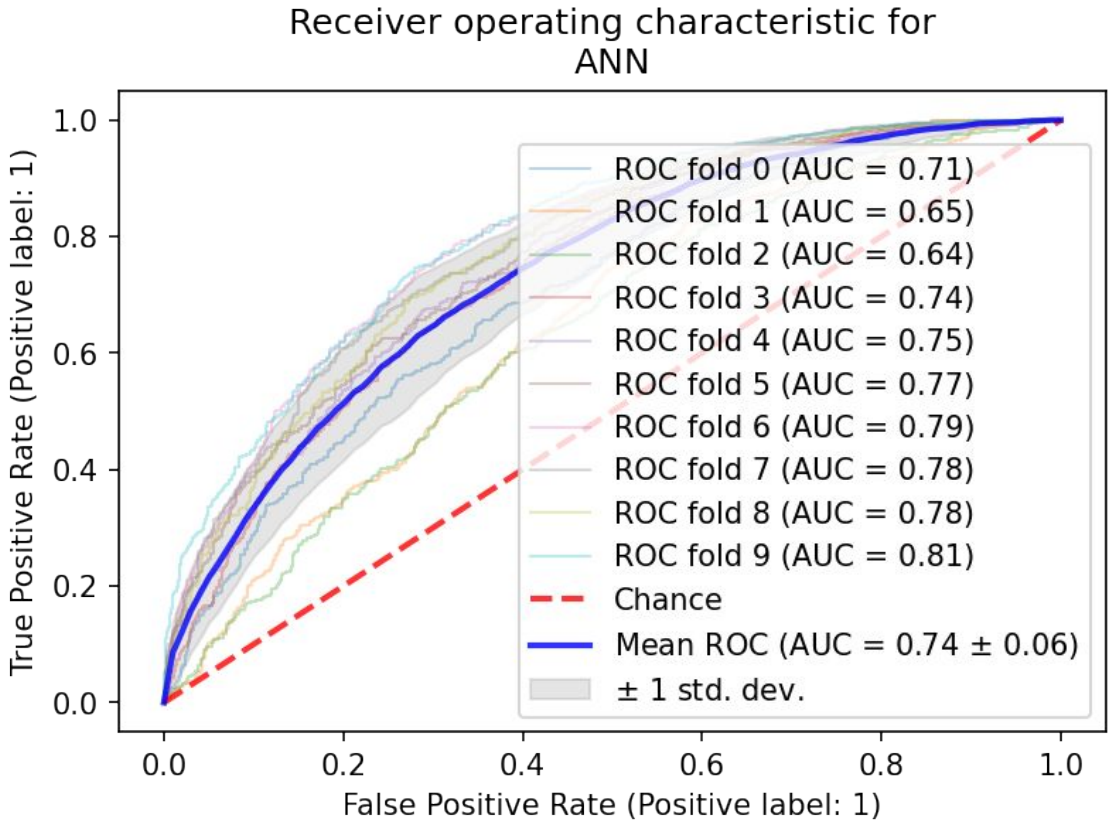




ANN




```
Accuracy :  0.6364201367308887
Balanced Accuracy :  0.6364852754859244
                   precision    recall  f1-score   support

 Low Review Score       0.67      0.53      0.59       805
High Review Score       0.61      0.74      0.67       804

         accuracy                           0.64      1609
        macro avg       0.64      0.64      0.63      1609
     weighted avg       0.64      0.64      0.63      1609
```



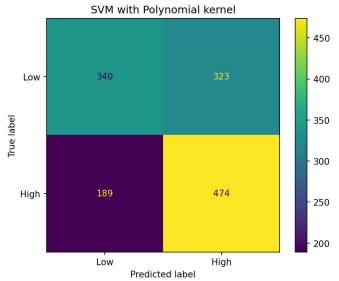

```python
class LinearBatchNormReLuBlock(nn.Module):
    def __init__(self, in_feature, out_feature):
        super(LinearBatchNormReLuBlock, self).__init__()
        self.linear = nn.Linear(in_features=in_feature, out_features=out_feature)
        self.bn = nn.BatchNorm1d(out_feature)
        self.relu = nn.LeakyReLU()

    def forward(self, input):
        y = self.linear(input)
        y = self.bn(y)
        y = self.relu(y)
        return y
```

✓ 0.6s

```python
net = torch.nn.Sequential(
    LinearBatchNormReLuBlock(n_features, n_features * 2),
    LinearBatchNormReLuBlock(n_features * 2, n_features * 4),
    LinearBatchNormReLuBlock(n_features * 4, n_features * 2),
    LinearBatchNormReLuBlock(n_features * 2, n_features),
    torch.nn.Linear(n_features, 1),
)
```

✓ 0.6s

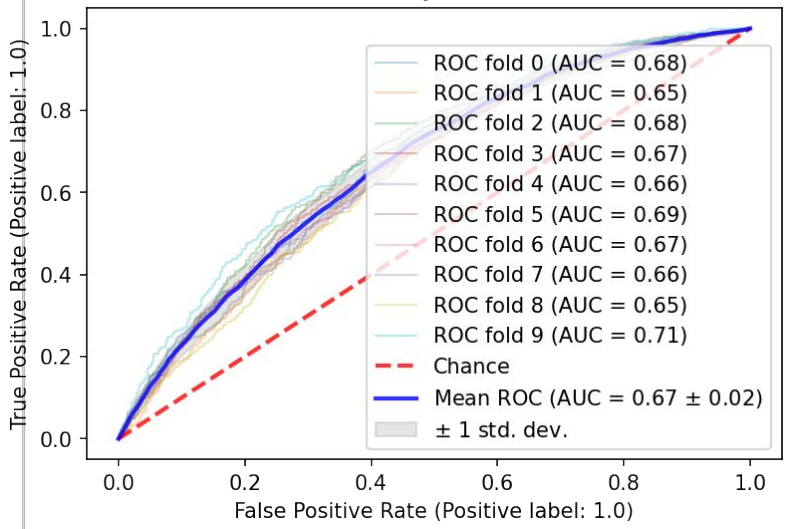
SVM

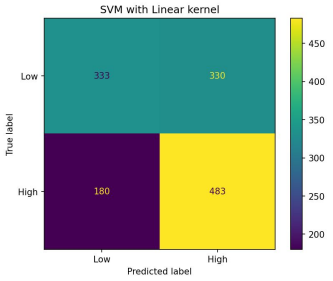


SVM with Polynomial kernel
Accuracy : 0.6138763197586727
Balanced Accuracy : 0.6138763197586727

	precision	recall	f1-score	support
Low	0.64	0.51	0.57	663
High	0.59	0.71	0.65	663
accuracy			0.61	1326
macro avg	0.62	0.61	0.61	1326
weighted avg	0.62	0.61	0.61	1326

Receiver operating characteristic for SVM with Polynomial kernel

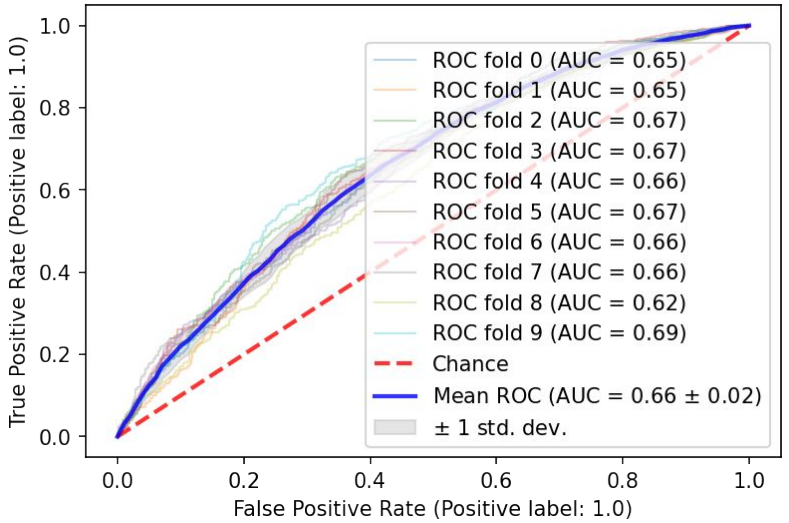


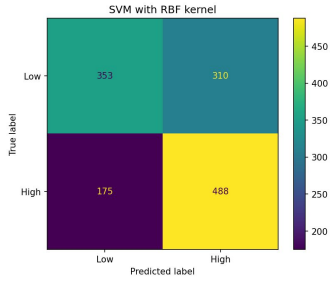


SVM with Linear kernel
Accuracy : 0.6153846153846154
Balanced Accuracy : 0.6153846153846154

	precision	recall	f1-score	support
Low	0.65	0.50	0.57	663
High	0.59	0.73	0.65	663
accuracy			0.62	1326
macro avg	0.62	0.62	0.61	1326
weighted avg	0.62	0.62	0.61	1326

Receiver operating characteristic for SVM with Linear kernel

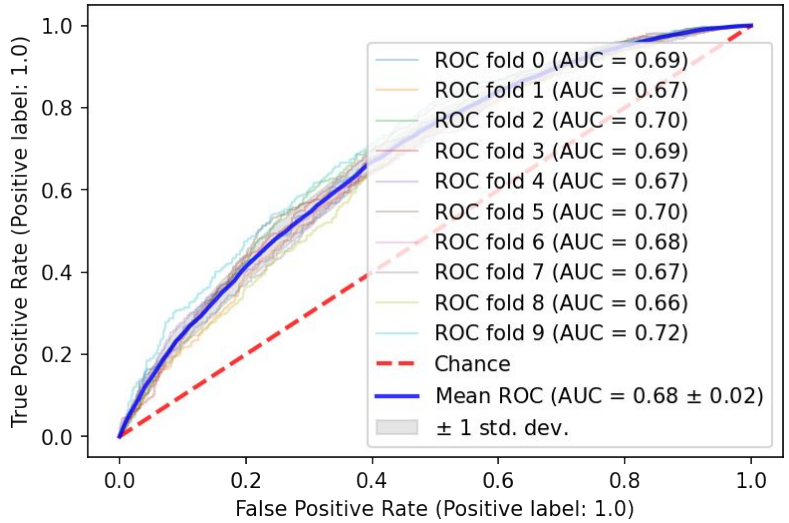




SVM with RBF kernel
Accuracy : 0.6342383107088989
Balanced Accuracy : 0.6342383107088989

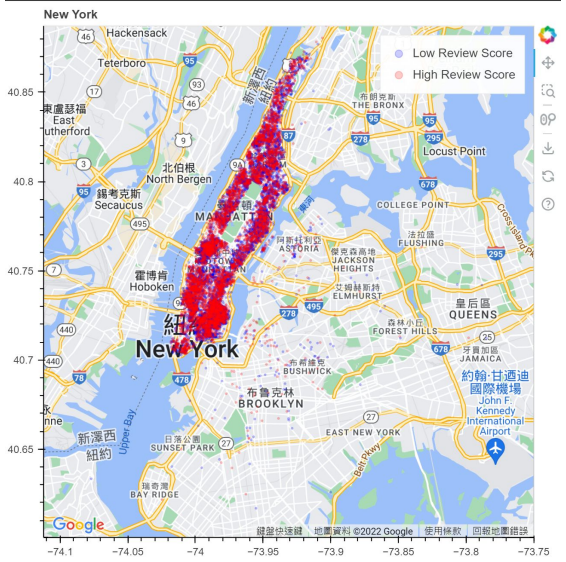
	precision	recall	f1-score	support
Low	0.67	0.53	0.59	663
High	0.61	0.74	0.67	663
accuracy			0.63	1326
macro avg	0.64	0.63	0.63	1326
weighted avg	0.64	0.63	0.63	1326

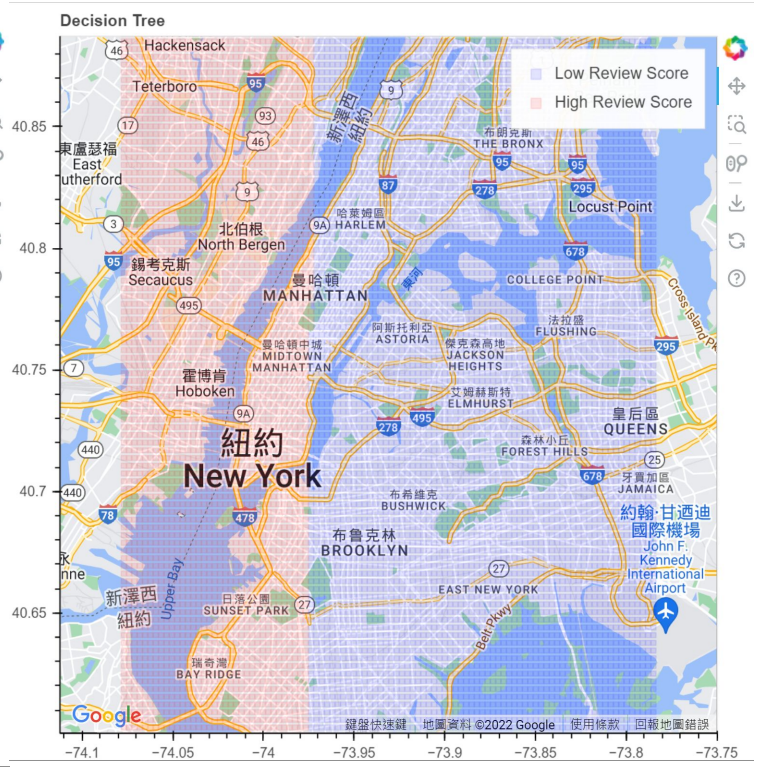
Receiver operating characteristic for SVM with RBF kernel



Decision Regions

All features, except latitudes and longitudes, are fixed to their respective average values





Conclusion

- From explainable models (Random Forest), the most important predictors for the ratings are:
  - Review related features
    - total amount of reviews
    - rate of reviews being given
  - Location
    - latitude and longitude
  - Seniority of the listing
    - host time (last seen date - first listing date)
    - total listing owned by the same owner
  - Price
    - daily, weekly, and monthly price

Learning Models	Accuracy
knn	56%
k-Means Clustering	52.9%
Decision Tree CART	61.46%
C5.0	62.10%
Random Forest	62%, 70%
Naïve Bayes	57%
ANN	64%
SVM	63%

Future Work

1. Apply more model analyses, both supervised and unsupervised learning.
2. Collect more data and attributes (like the text fields) of the dataset to increase accuracy.
3. Explore more data processing techniques
4. Test against real listings

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK YOU

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030