ISOM 676: Machine Learning 2

Final Project: Airbnb Superhost Classification

By. Mild Trakarnsakdikul, Joy Zhu, Sylvie Zhou, Zayn Sui,

Introduction

In this final project, we are tasked with obtaining a problem from your current/future job, something of interest to the school, data acquired from the web, data provided by the instructor. Then we will design a data mining task, mine the data and share the result.

We decided to use Airbnb data that we have worked with in our Big Data class in the fall, since we are all familiar with the dataset. The data also have enough information and data points suitable for this projects. Furthermore, we are interested in exploring Airbnb as a business and explore what problems the business might need solving. We will be exploring the data, data mining and answering business questions using Python programming.

Business Understanding

Airbnb is a community-based, two-sided online marketplace that facilitates and connects people who want to rent out their home with people who are looking for lodging in a specific location around the world. On the Airbnb platform, there are 2.9 million hosts worldwide and more than 7 million listings in over 200 countries, with more than 150 million users. This means that their biggest assets are the individuals and the hosts, where their revenue is from:

- Commission host: Everytime guest makes payment, Airbnb takes 10% of the payment amount as commission.
- Guest Transaction fee: When travelers make payments for stays, they are charged a 3% fee for the transaction.

Since the hosts are the base of Airbnb business model, we want to help Airbnb better define who are qualified to be a superhost. Superhosts are hosts that provide outstanding hospitality, which means being highly-rated, experienced, reliable, and responsive. We notice that the definition of a superhosts is vague without any qualitative values to base on. We want to explore what features make a superhost, and if there are any features that are weighted more than others. This would also allow us to use the findings to help the host better align themselves to become superhost on Airbnb.

The business question we are trying to answer is: "**What criteria can be used to more effectively identify and qualify Airbnb hosts as Superhosts?"**

Data Understanding

As mentioned in the introduction, we are using Airbnb's data from the listings in the United States. We have obtained the data through insideairbnb.com. For feasibility of the project we decided to focus on the west coast data, which contained 131,388 observations and 58 variables. It contained 6 States in the west coast which consisted of: California, Colorado, Hawaii, Nevada, Oregon, Washington and 14 cities between the states. In the target variables there are 45,595 superhosts and 85,793 normal host. Therefore, as this is a classification problem, the base rate is 34.7% for the positive class.

Our problem is a supervised classification problem, where we will identify if a current host is a superhost or not. The features we wil be using includes:

- Property features
    - Description (long-text), property type
    - Location
    - Amenities & Rooms
    - Booking availability
    - Ratings
- Host features
    - Response time, response and acceptance rate
    - Host tenure
    - Host description (long-text)
    - Host listing count

We believe features related to review scores, host response & acceptance rate, and availability are most useful as it seems to relate with the definition that Airbnb gave of what makes a superhost.

Numerical Data Exploration (EDA) and Preprocessing

We initially examined null values in the dataset and found that the target variable had 848 nulls. Given the size of our dataset (over 130,000 rows), we made the decision to remove these 848 rows. For other numerical variables, such as reviews_per_month and

review_score_ratings, we assessed the distribution of values and opted to impute the mean value to replace any null values. Additionally, we transformed Boolean variables into integers (0 and 1), and We also addressed some string variable such as price, which was initially stored in the format of "$100". To incorporate this feature into our prediction model, we extracted the numerical values from the string variable and converted it to a numerical data type. After this, we checked duplicate values and dropped 252 duplicate rows.

Next, we examined the multicollinearity between the variables. Using a correlation heatmap, we identified several highly correlated features, including maximum_nights_avg_ntm, minimum_maximum_nights, availability_60, availability_30, availability_90, calculated_host_listings_count_entire_home, and calculated_host_listings_count. To avoid any negative impact on our model performance, we decided to drop these variables. In the end, there are 27 numerical variables in our model.

Text Data Preprocessing

Since we have text data in different column and format, we decided to combine all the text features into one variable to work with. The features Property Name, Description, Neighborhood Overview and Host About were combined together. Then we had to start the cleaning process, from looking at sample text we see that most text contains symbols, punctulations and html code that are not useful for the analysis. So we uses regex to remove all the non letter from the text. Then we converted all the text to lower case for ease of futute analysis. Next we imported stopwors package from nltk to remove unnecessary stop words and then we manually remove repeating words like br, bedroom, airbnb from the text, as these text do not provide any insight for analysis.

We then explore different ways to convert similar words like wonder, wonderful into one for the fairness of analysis. We explore three method, stemming (PorterStemmer), toklenizing (RegexpTokenizer), and lemmatizing (WordNetLemmatizer) to see which gives the best result processing the text. Each methods perform differently, which is discussed below:

1. Stemming: reduce the words to its root form or its stem which may or may not be a word, for example: paradise would be turn into paradis
2. Tolkenizing: breaks down the text into smaller chuck called tokens, which can be words, phrases or individual characters.
3. Lemmatizing: reduce the words to its base form, which will always be a valud word, for example: running would be run.

After exploring all the method we decided to go with stemming as the processing method, as it reduce the words to the its root which is suitable for description analysis which may contain variation of a single word.

The processing method mentioned above were all combined into one function and then ran all the text data through, shown in the appendix.

Text Data Exploration

After cleaning the data, we explored what the text consisted of to see what type of word standout for superhost and normalhost. First we started by plotting a word cloud (shown in the appendix) to help visualized what the words include from superhost and normal host. We see that there are similar words that come up for both type of hosts like: guest access, location,living room, washer dryer, etc. Some words that standout for superhost includes: queen bed, minuet walk, coffee shop, central location, etc.

Then we explore the CountVecotrize count and the TfidfVectorize score for the words in superhost to see what standout. The words and score are shown below:

CountVectorizer Count		**TfidfVectorizer Score**	
Word	**Count**	**Word**	**Score**
home	66076	home	1840.568556
space	62935	park	1547.330646
guest	55454	room	1502.964659
park	54982	guest	1496.713286

locat	51332	bedroom	1458.818633
walk	48209	privat	1388.923390
room	46792	locat	1382.391850
bedroom	46784	hous	1349.780526
privat	43562	space	1331.404843

We see that most of the top ten words with the highest count also have a high TfidVectorize score. After exploring and processing the numerical and text data, we will be moving on to the modeling our data to predict superhost.

Modeling Approach

As there are two types of data we are working with, we decided to split the classification method into two phase, classification using numerical data and classification using text data. We believe that this will give us a better understanding of the data and what are the qualities that Airbnb looks for in a superhost.

The classification using numerical data will be able to provide us with features that are important for a superhost in Airbnb, while classification using text data will give the host an idea of what a superhost property looks like.

Model Evaluation

Evaluating superhost classification, we have to cosider the false positive and false negative impact on Airbnb, the hosts and the guests. False positives (predicting a host as a superhost when they are not) can lead to disappointment and dissatisfied guest. False negatives (predicting a host as not a superhost when they are) could lead to upset host who might not get the recognition they deserve. In this case, precision is more important as false positive have a larger impact on the cost of the business more than false negative. This is because unhappy

guest can cost Airbnb, as the guest may not return to the platform or pivot to hotels. However, false negatives.

We chose to proritze our evaluation metric on precision, because it is more costly when there an incorrect prediction of a superhost is made. Precision focus on the proportion of correctly predicted superhosts among all the positive predictions. A high precision score indicates the model making fewer false positives predictions.

Numerical Classification Modeling

As part of our model development process, we initially split the data into training and testing sets with a 70/30 split to assess model performance. We decided to start by testing simpler models without parameter tuning to see which model would perform best, potentially saving time on tuning parameters in the future. We evaluated several models, including decision trees, logistic regression, Naïve Bayes, and CatBoost.

After conducting tests, we found that the CatBoost model performed the best, achieving an f1 score of 0.81 and a precision score of 0.82 on the test set. As a result, we selected CatBoost as our final model.To assess the performance of our model, we plotted the ROC curve, which revealed an AUC score of 0.94, indicating that the model performs exceptionally well. We also utilized SHAP graph to explore the feature importance of the CatBoost model. The graph helped us to identify which features had the most significant impact on our model's predictions. Some essential features included host_response_rate, number_of_reviews_ltm, and host_acceptance_rate. By understanding these features, we gained valuable insights into how we couldimprove Airbnb's business strategy.

Natural Language Processing Modeling

To begin NLP modeling, we first split the model into a train and test set with a 80/20 split. We then convert all the text data into numerical vector. We achieved this by running the text through CountVectorizer and TfidfVectorizer. CountVectorizer counts the number of times each

word appears in the text and create a matrix with the result. TfidfVectorizer assigns weights to each word in the text based on how often it comes up in the text body. We get a result matrix that contains normalized term frequencies that take into account borh the frequency of the word in the text and overall text. The data came out to have 51,475 text features, with the train set having 104,964 rows and the test set have 26,242 rows.

We tried variety of model to see how they perform against each other, each Model will be train with the CountVectorizer and TfidfVectorizer data to see which type of normalization work best. We started with simple classification modeling, exploring Logistic regression, Decision Tree Classifier, KNN and Naive Bayes. The result of each model is show below:

Model	**Vectorizer**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Model	CountVec	0.76	0.69	0.56	0.62
	TfidfVec	0.76	0.69	0.53	0.60
Decision Tree Classifier	CountVec	0.74	0.63	0.61	0.62
	TfidfVec	0.72	0.60	0.60	0.60
Multinomial Naive Bayes	CountVec	0.67	0.51	0.74	0.61
	TfidfVec	0.70	0.74	0.21	0.33
KNN Classifier	CountVec	0.76	0.84	0.38	0.53

Then we explore ensemble methods, with CatBoost and Random Forest Classifier. The results for the mentioned models are below:

Model	**Vectorizer**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
CatBoost	CountVec	0.78	0.76	0.54	0.63
	TfidfVec	0.78	0.74	0.54	0.62
Random Forest Classifier	CountVec	0.79	0.89	0.45	0.60
	TfidfVec	0.78	0.90	0.42	0.57

The best model for NLP classification is: Random Forest Classifier with TfidfVectorizer data that gave a precision of 0.90, accuracy of 0.78 and AUC of 0.7.

Stacking Model

To improve our prediction accuracy for the actual superhost rate, we combined the predictions generated from both textual data and numerical variables. Specifically, we stacked the best model of NLP (Natural Language Processing) and the CatBoost model to predict the superhost rate, with the actual superhost rate serving as the dependent feature. We applied the logistic regression model to merge these predictions and evaluate their impact on the final result.

Our model achieved an outstanding performance, with an F1 score of 0.97 and a precision score of 0.98, indicating high accuracy in detecting superhosts. The coefficients of our logistic regression model showed that the NLP prediction had the most significant impact, with a coefficient of 14.86, while the CatBoost prediction had a coefficient of 5.68. These results suggest that NLP prediction is a critical factor in accurately identifying superhosts.

Business Application

Overall, from our exploration we will were able to help Airbnb identify and create a criteria for superhost and for host to align themselves better to become a superhost. The criteria would consist of:

- Improving host response rate – response in shorter time when guests have questions and requests
- Providing less listing homes/stays and make each of them perfect
- Reviewing each request before accepting the booking
- Accepting more booking requests which also helps to increase number of reviews

Further exploring the descriptive text, we were able identify words that would be helpful for host to include in their property description or name. These words include:

- Words about the location/city
- Words regarding space
- Words about the environment
- Words about the surrounding area
- Guest centric writing

Knowing these information Airbnb and their hosts can leverage the knowledge to optimized the way they name and describe their property to increase the chances of being booked and becoming a superhost. This allow, Airbnb to maximize their busines strategies and increase their superhost growth and revenue.

When deploying this predictive model, there are some risk and ethical consideration Airbnb should take into account of. This includes:

- Data Privacy: the model should not violate the guest or host data privacy and use data that could discriminate against them. Airbnb should be sure that the model is train with the consent of the host and that there is no data leakage.
- Model Transparency: the model should be fair and unbiase towards the host and should not base any of the modeling on who they are (race, gender, ethnicity, etc). Airbnb should also be transparent about how the model works and be able to show how it can help host improve their listings.
- Effect from the Model: Airbnb have to be aware and responsible for the prediction that the model make and evaluate the prediction with business insight, so to not unfairly penalized hosts.

By addressing these ethical and risk considerations, Airbnb can make sure that the model they deploy is fair, transparent and responsible.

Appendix

Classification with Numerical Data Code:
https://colab.research.google.com/drive/1QR8AONTa2KlAxB-Qc4gQU1PWetVoSABa#scrollTo=yvh-JOzf26ih

Classification with Text Data Code:
https://colab.research.google.com/drive/1eN7Gl9jzJbcSRbaGQrRNN6sKtHI2SZYD?usp=sharing

Stacking Model Code:
https://drive.google.com/file/d/1VOlV_ChgRUjpyrnrvHjwCUUPUb9GdPQL/view?usp=sharing

Team Member Contribution

Team Member Name	**Contribution (%)**	**Comment**
Mild Trakarnsakdikul	25%	• NLP Model • Written report
Joy Zhu	25%	• Classification Model • Written report
Sylvie Zhou	25%	• Stacking Models • Model Evaluation • Written report
Zayn Sui	25%	• Data Cleaning • Written report

Text Cleaning Function

```
def text_to_words(raw_text):
    # 1. Remove non-letters
    letters_only = re.sub("[^a-zA-Z]", " ", raw_text) #removing all non letters

    # 2. Convert to lower case, split into individual words.
    lowercase = letters_only.lower().split()

    # 3. Remove stopwords.
    stop_words = nltk.corpus.stopwords.words('english')
    words = [w for w in lowercase if not w in stop_words]

    # 4. Stemming
    porter = PorterStemmer()
    ps = [porter.stem(i) for i in words]

    # 5. Join the words back into one string separated by space,
    return(" ".join(ps))
```

Sample Text Cleaning

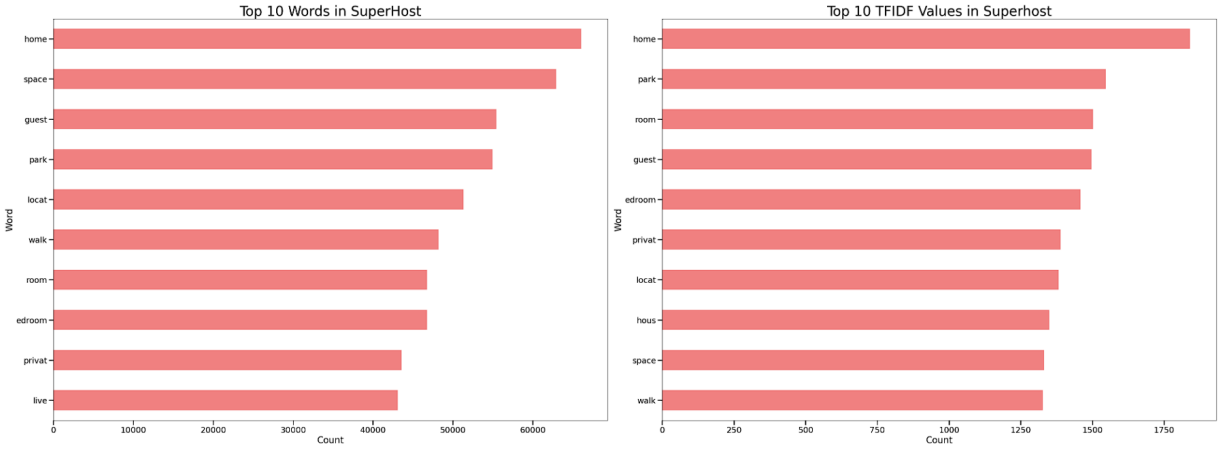
Original Text	Clean Text
Cottage by the Redwoods This is a very private cute cozy small bohemian style retreat next to a creek under the redwood and oak trees. Across the street is a 40 acre park for hiking biking walking frisbee etc. including a dog park The space Lovely private setting by the creek, across the street from 40 acre park with trails and playing fields, beach easy walk or bike ride away, shops, restaurants and theater nearby. no traffic very peaceful. Guest access There is ample parking in front of the orchard. Bus stop to Santa Cruz or Capitola etc. is a five minute walk down the street. Other things to note Recycling and composting as well as minimal use of plastics Are greatly appreciated! This is a quiet neighborhood with no street lights and no sidewalks. It's near the end of the road so there's very little traffic. Yet it's just a minute from the freeway and health food stores and restaurants are all just minutes away. Easygoing, environmentalist, musician/educator, progressive.	cottag redwood privat cute cozi small bohemian style retreat next creek redwood oak tree across street acr park hike bike walk frisbe etc includ dog park br br b space b br love privat set creek across street acr park trail play field beach easi walk bike ride away shop restaur theater nearbi traffic peac br br b guest access b br ampl park front orchard bu stop santa cruz capitola etc five minut walk street br br b thing note b br recycl compost well minim use plastic greatli appreci quiet neighborhood street light sidewalk near end road littl traffic yet minut freeway health food store restaur minut away easygo environmentalist musician educ progress activ play music garden teach work homestead

Word Cloud

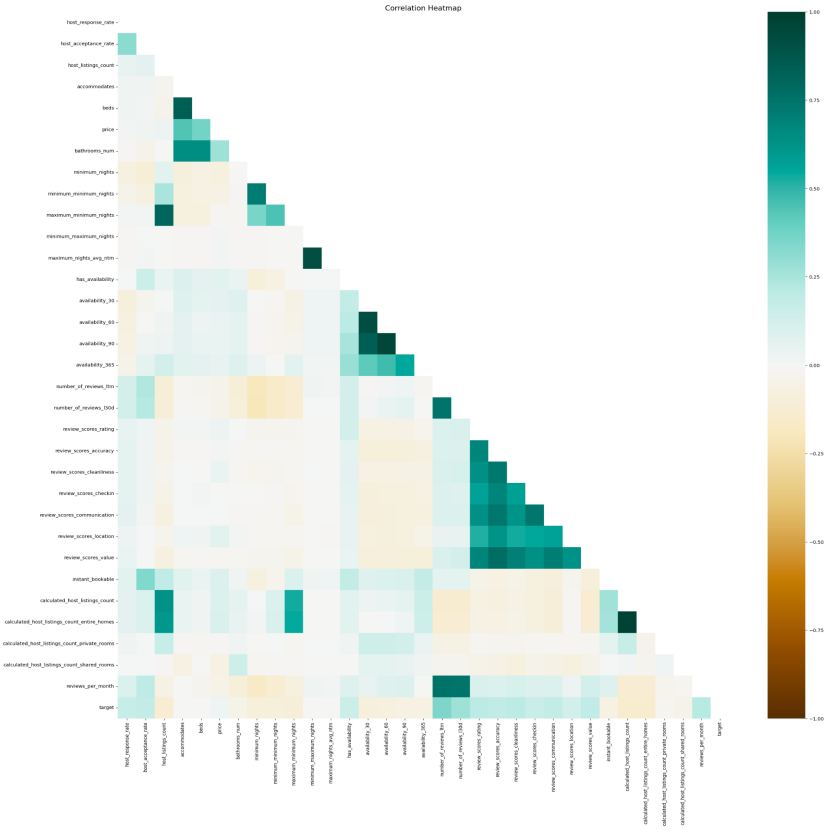
Superhost Description

Normal Host Description

Superhost CountVectorized and TfidfVectorized Words



Correlation Heatmap

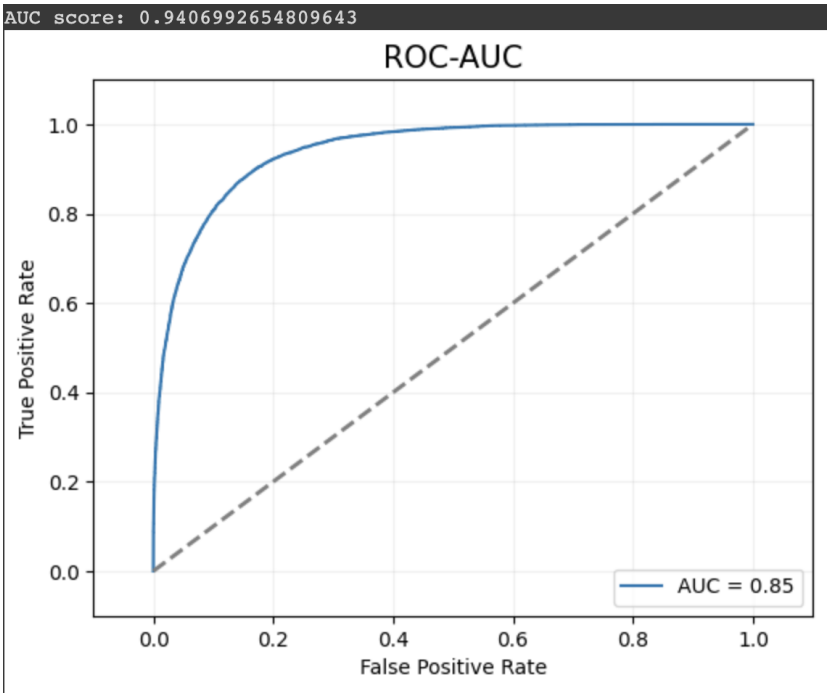


CatBoost Model

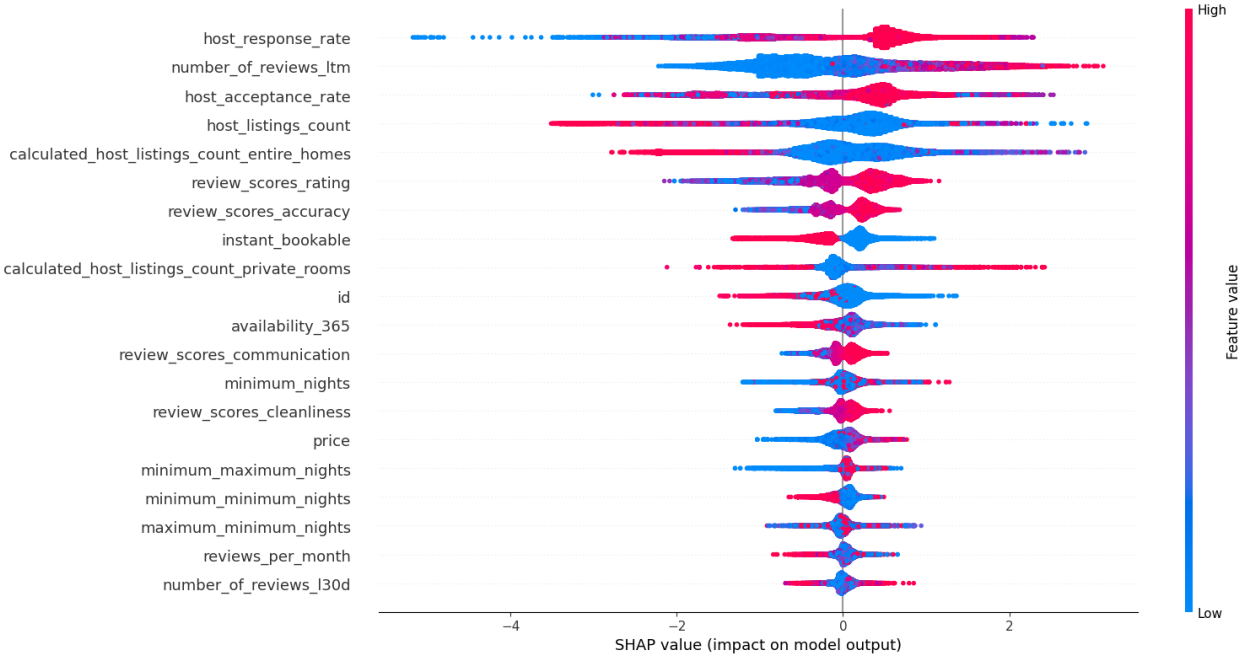
```
from catboost import CatBoostClassifier
clf = CatBoostClassifier(iterations=50,
                         depth=10,
                         learning_rate=0.4,
                         loss_function="Logloss",
                         verbose=False)

# Fit the model to the training data
clf.fit(X_train, y_train)
y_pred_clf = clf.predict(X_test)
# Evaluate the model on the testing data
print('Catboost Score')
model_evaluate(y_test, y_pred_clf)
```

```
Catboost Score
Test F1 Score: 0.8075211392968402
Test Precision Score: 0.8230269126096159
Test Recall Score: 0.7925888177053
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     25351
           1       0.82      0.79      0.81     13736

    accuracy                           0.87     39087
   macro avg       0.86      0.85      0.85     39087
weighted avg       0.87      0.87      0.87     39087

Confusion Matrix:
 [[23010  2341]
 [ 2849 10887]]
```

CatBoost ROC Curve



SHAP– CatBoost Features

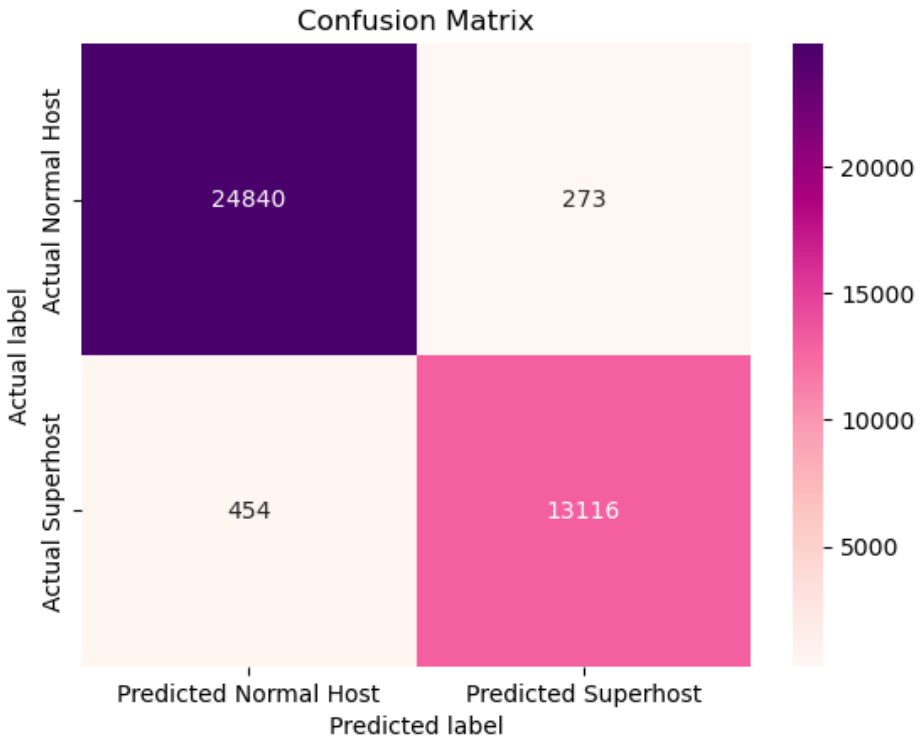


Stacking Model

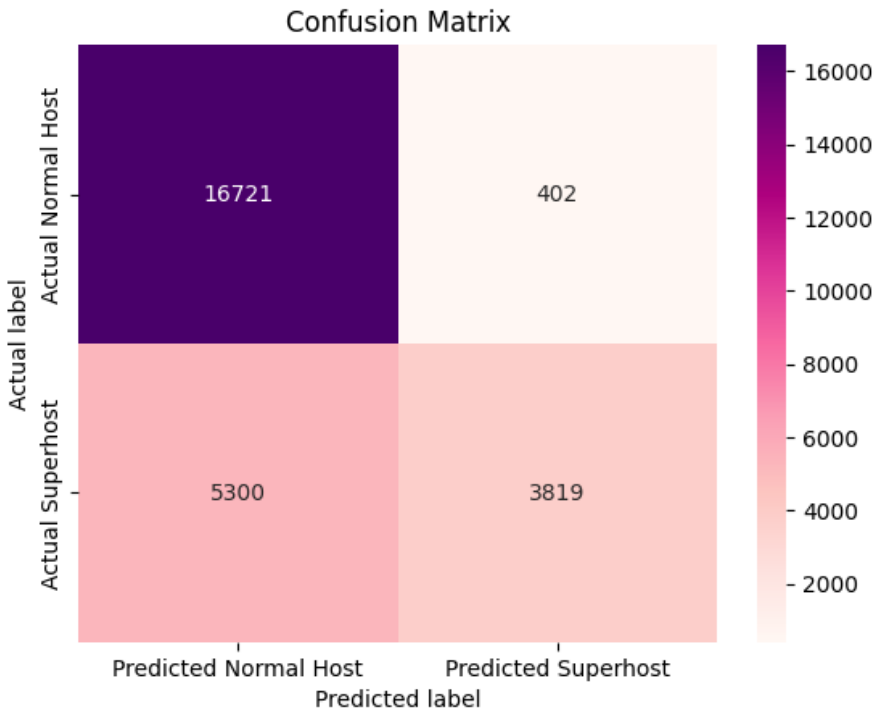
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, f1_score, precision_score, recall_score, confusion_matrix
#model evaluations on the test set
def model_evaluate(y_test, y_pred):
  print(f"Test F1 Score: {f1_score(y_test, y_pred)}")
  print(f"Test Precision Score: {precision_score(y_test, y_pred)}")
  print(f"Test Recall Score: {recall_score(y_test, y_pred)}")
  print(classification_report(y_test, y_pred))
  print('Confusion Matrix:\n',confusion_matrix(y_test, y_pred))


X = df.iloc[:,[3,5]]
y = df.iloc[:,1]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
lr = LogisticRegression(random_state=42).fit(X_train, y_train)
y_pred = lr.predict(X_val)
```

```
Test F1 Score: 0.9730331243740495
Test Precision Score: 0.9796101277167825
Test Recall Score: 0.9665438467207075
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     25113
           1       0.98      0.97      0.97     13570

    accuracy                           0.98     38683
   macro avg       0.98      0.98      0.98     38683
weighted avg       0.98      0.98      0.98     38683

Confusion Matrix:
 [[24840   273]
 [  454 13116]]
```

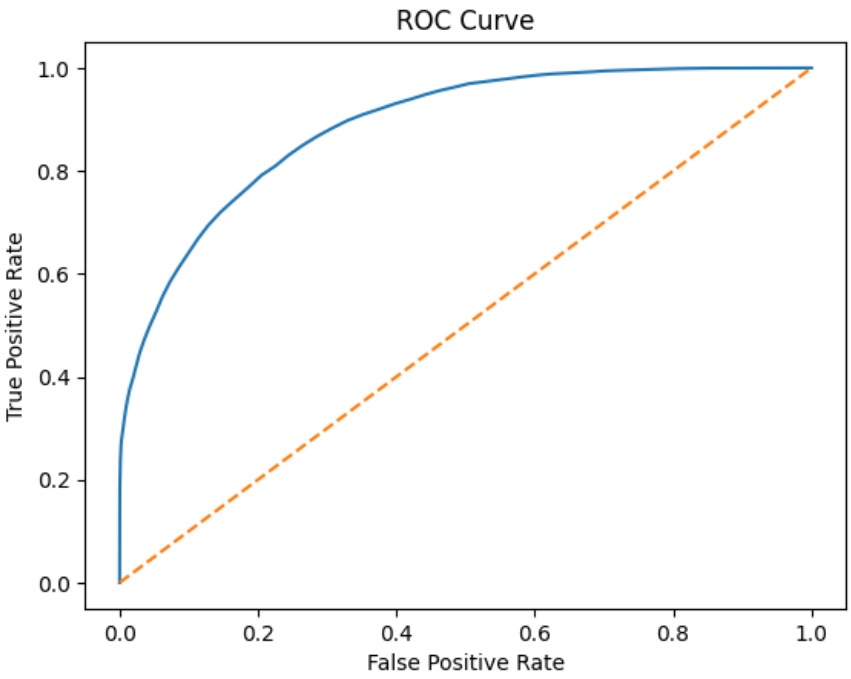
Stacking Model Confusion Matrix



NLP Modeling Confusion Matrix



NLP Modeling ROC Curve



Important Features from NLP model