PROJECT REPORT
Team Name: JAM

Mrinal M
IMT2018044
International Institute of Information Technology, Bangalore
Mrinal.M@iiitb.org

Kanumuru Shree Jhanwwee Reddy
IMT2018035
International Institute of Information Technology, Bangalore
Shree.Jhanwwee@iiitb.org

Vattikuti Sai Anvith
IMT2018528
International Institute of Information Technology, Bangalore
Sai.Anvith@iiitb.org

***Abstract*—The goal of this project is to predict 3 most likely final destinations of users on Airbnb, a vacation rental company based on various parameters for each individual user and the date and time of booking to the destination country and parameters for each country too.**

I. Introduction

New users on Airbnb can book a place to stay in 34,000+ cities across 190+ countries. Predicting the region of booking will help provide better content to the users with regard to their choice of destination, decrease the average time to first booking, and better forecast demand, thus benefitting both the customers and Airbnb.

So we have to study the data set, understand it and be able to predict the user's destination countries. In this project report, We shall briefly note our ideas, implementation and observations.

Firstly on this report we shall look through the data set given then the encoding we used for different columns. Next we shall discuss the different models tried.

II. Dataset

The dataset used in the project is from the IIITB ML: Airbnb New User Bookings competition which was hosted on Kaggle.

In this challenge, we are given a list of users along with their demographics, web session records, and some summary statistics. We are asked to predict which country a new user's first booking destination will be. All the users in this dataset are from the USA.

There are 12 possible outcomes of the destination country: 'US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL','DE', 'AU', 'NDF' (no destination found), and 'other'. We must note that 'NDF' is different from 'other' because 'other' means there was a booking, but is to a country not included in the list, while 'NDF' means there wasn't a booking.

The train and test splits are random. The data in the train set dates back to 2010, while the test set consists of users joining after 01/04/2014. The sessions data dates back to 2014.

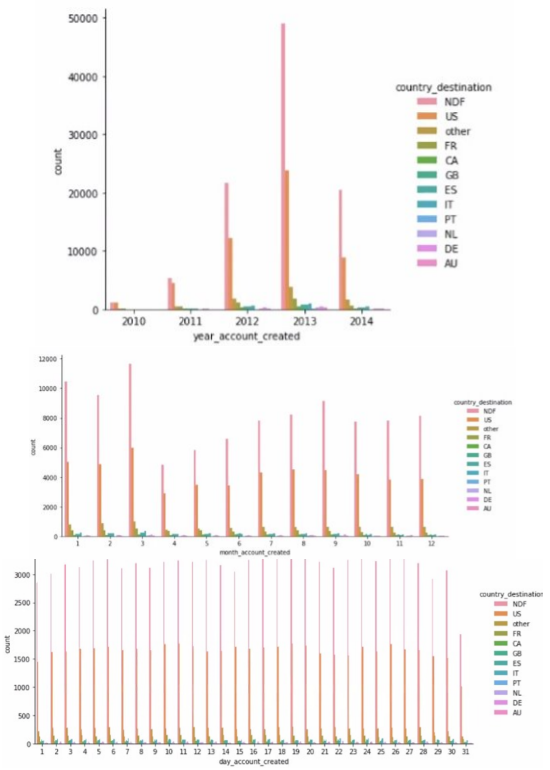
Here we have 6 files: agegenderbkts.csv, test.csv and train.csv, countries.csv, sessions.csv, samplesubmission-NDGC3.csv. One is a training dataset and the other is to be used for testing. Another file: samplesubmissionNDGC3.csv is given to let us know how the output is supposed to be like. It contains unique ID column and a destination country output column.

We have other files too that may/may not be useful from the given dataset.

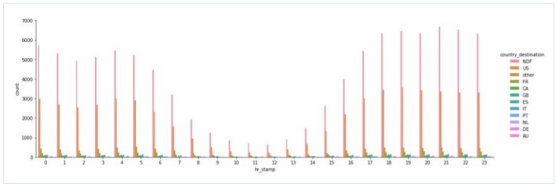
In both train dataset has 16 columns and the test dataset has 15 columns. We usually do not use all the features to compute and predict the output. Some features might have excessively high NaN values or be heavily skewed. Some might not even have any impact on the prediction result. While many others need to be encoded in different ways so that they could be used for prediction. Such things we tried to address firstly.

III. Observations

- One key observation is that in most of the columns, for each corresponding unique value, countrydestination NDF has highest frequency.
- We used Groupby.describe() to analyze the features. We used the ratio of NDF frequency to total, to determine which unique values can be clubbed.
- After analysing each with respect to countrydestination we concluded that year and day doesn't contribute much. Even though month doesn't show any significant difference, there are slight variations in the ratio of NDF to others.



- Analysis of time part tells us that during particular hours, the ratio of NDF is high indicating that people are just randomly searching.



- Date first booking has majority nan values and most of it correspond to NDF.
- Language is observed to have heavy skew, whereas signup method is seen to have very similar distrbution of countrydestination for both of its unique values.
- Affiliate provider is another column that has lot of unique values with insignificant count and a heavy bias.
- Signup app is shown to have heavy bias too. It has 4 unique values, out of which web is dominating in the count. Combined all the other 3 into one (reason being their ratios), but even then the data is heavily biased towards web.
- Age had a lot of nan values and outliers and 75 percent of them corresponded to NDF.
- Gender has 4 unique values, MALE, FEMALE, OTHER, -unknown-. The value count for OTHER is insignificant.

IV. Feature Engineering

Features with large number of unique values, we first tried to reduce the number of unique values, taking in account the frequency of each. If that was possible, we went on to one hot encoding, else label encoding.

Train and test is combined for doing feature analysis and encoding.

A. Encoding

For the other non integral features, we used different encoding methods. Primarily One-hot encoding and Label encoding techniques.

If a feature has more unique values, we have decided to choose label encoding. For those features with less unique values, We did one hot encoding.

B. Modifications

Gender has 4 unique values, MALE, FEMALE, OTHER, -unknown-. The value count for OTHER is insignificant, hence combined with –unknown-

SignupFlow has 17 unique values. The values with ratio ¿ .6 are combined into one category and ¡. 6 into another. The distribution is biased and hence we can drop this in the future.

Affiliate channel initially had 8 unique values. Remarketing and content has relatively less value counts and the ratio is high. The only other unique value with relatively high ratio is Other, hence combined (other,content and remarketing) into a single columns, others left as such. After minimizing it into 6 unique values then one-hot encoded the feature. This feature is also not so promising(can be removed at later stage).

First device type gives some interesting information. Whenever the device type is desktop, the NDF ratio goes down, indicating more actual bookings. Even though data is biased towards desktop, this might be a useful inference. First converted all the unique values into two category(desktop and other)

C. Removed

Language is removed due to heavy skew, whereas signup method is removed due to very similar distrbution of country-destination for both of its unique values.

Affiliate provider has lot of unique values with insignificant count and a heavy bias.

Signup app has heavy bias as dicussed earlier and so it is removed.

We didn't remove some columns even though it felt like it doesn't contribute much. The plan is to try the model and modify accordingly.

D. Label Encoding

Date first booking has majority nan values and most of it correspond to NDF and hence we went for label encoding the column.

Age and date first booking are the two columns which are label encoded. As seen above, Age had a lot of nan values and outliers and 75 percent of them corresponded to NDF and hence label encoding seemed reasonable.

E. One Hot Encoding

All the other columns are one hot encoded after reducing it to affordable unique values.

CODE SNIPPETS

Below are code snippets showing how we implemented the encoding schemes in general.

```
X_train['country_destination'].value_counts()
dic = {'PT':0,'AU':1,'NL':2,'DE':3,'CA':4,'ES':5,'GB':6,'IT':7,'FR':8,'other':9,'US':10}
li=[]
for i in X_train['country_destination']:
    li.append(dic[i])
X_train['country_destination']=li
def handle_non_numerical_data(df):
    columns = df.columns.values
    for column in columns:
        text_digit_vals = {}
        def convert_to_int(val):
            return text_digit_vals[val]

        if df[column].dtype != np.int64 and df[column].dtype != np.float64:
            column_contents = df[column].values.tolist()
            unique_elements = set(column_contents)
            x = 0
            for unique in unique_elements:
                if unique not in text_digit_vals:
                    text_digit_vals[unique] = x
                    x+=1

            df[column] = list(map(convert_to_int, df[column]))

    return df
X_train = handle_non_numerical_data(X_train)
X_test = handle_non_numerical_data(X_test)
```

Fig. 1. Label Encoding

```
# SIgnupApp
col = 'signup_app'
print(training.groupby('signup_app')['country_destination'].describe())
training.groupby('signup_app')['country_destination'].describe()['freq']/training.groupby('signup_app')['country_destination'
# We can see that the ratio of NDF to others is lowest in Web and the count is comparitively very high. Since the count of ot
# is relatively insignificant, and the ratio is relatively high, we can club into Web and other and then do one_hot encoding.
desktop = ['Desktop (Other)','Mac Desktop', 'Windows Desktop']
temp =[]
for i in X[col]:
    if i == 'Web':
        temp.append('Web')
    else:
        temp.append('Other')
X[col] = temp
dummies = pd.get_dummies(X[col], prefix=col,dummy_na=False)
X = X.drop(col,1)
X = pd.concat([X,dummies],axis=1)
```

```
             count unique  top   freq
signup_app
Android       3317     12  NDF   2572
Moweb         4586     12  NDF   2913
Web         150625     12  NDF  84337
iOS          11609     12  NDF   7985
```

Fig. 2. Modifying data to our convenience

```
# Since the unique values has been filtered to two we can do one hot encoding
dummies = pd.get_dummies(X['first_device_type'], prefix='first_device_type',dummy_na=False)
X = X.drop('first_device_type',1)
X = pd.concat([X,dummies],axis=1)
```

Fig. 3. One hot encoding

V. MODELS

The models we tried are -

1) Catboost model: We experimented with different hyper parameters of the model, particularly, depth of the model, parameter to balance the data and bagging. But still the model failed to distinguish the minority classes.
2) XGboost and LGBM: Both the models performed similar to catboost model. Catboost was slightly superior to both the models
3) DecisionTree: We experimented with depth and min,max no of leaves, but all the models were overfitting.
4) RandomForest: RandomForest worked better than decision tree and it gave the best result in predicting the minority classes.
5) LogisticRegression and Weighted LogisticRegression: It performed similar to RandomForest, but gave a slightly less score in the leaderboad.

Almost all the models predicted NDF vs non NDF with both precision and recall 1.

A. Observations

The models were struggling in separating US from the minority classes.

Most models predicted US other FR for almost all the non-NDF entries and that gave the best score.

B. Submission

We used two different models, one for classifying NDF vs non-NDF entries and the second model for classifying the non-NDF into different categories.

For the first we used a cat classifier and for the second we used random forest in the final submission.

ACKNOWLEDGMENT

This Project has been a wonderful opportunity for us to try and experiment around with different models and use various concepts we learnt in our classes. We also got to try practically on how ML can be used in real life applications as seen in Airbnb user bookings. We would like to take this opportunity to thank Prof. Raghavan, Prof. Neelam Sinha and the TA, Tanmay Jain for the encouragement and continuous motivation.

We would also like to thank other team members namely- Rohit Katlaa, Koustubh, Nikitha, Aravind for creating a healthy discussion atmosphere. Every team has done a wonderful job and the competition was really good which motivated us to work more. It was fun competing against all the teams on the leaderboard.

REFERENCES

1) https://towardsdatascience.com/how-to-handle-multiclass-imbalanced-data-say-no-to-smote-e9a7f393c310
2) https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
3) https://www.analyticsvidhya.com/blog/2017/03/imbalanced-data-classification/
4) https://catboost.ai/docs/concepts/r-training-parameters.html
5) https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e: :text=Feature
6) https://scikit-learn.org/stable/modules/calibration.html: :text=CalibratedClassifierCV
7) https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b
8) https://stats.stackexchange.com/questions/237523/how-to-threshold-multiclass-probability-prediction-to-get-confusion-matrix