"Your next favorite vacation destination with Airbnb"

Airbnb New User Bookings

1.0 Abstract

Airbnb launched a competition few years ago as a part of their recruitment process. They supplied much un wrangled, messy and raw data from their database and challenged participants to build a model that could predict the next destination of the Airbnb Users. This Dataset was challenging and required much Data Wrangling and Exploratory Data Analysis Steps. There were a lot of missing values that needed to be addressed carefully. Inferential Statistics revealed a lot of valuable information and pivots between the Data fields and attributes. Two successful Model Classifier was built with moderate accuracy rate and high precision values.

2.0 Background

Airbnb is a trusted online community marketplace for people to list, discover and book unique accommodations around the world, whether be it from desktop or one click away from the mobile app. It gives people the leverage and the flexibility to choose and customize between where and how they want to book based on their budget and other constraints; let it be an apartment for a night or a villa for a month. Airbnb is active over 65,000 cities and 191 countries all over the world. As a part of their recent recruitment process they held a competition in Kaggle for predicting the "Bookings of the New Users of Airbnb". The competition was launched in 11/25/2015 1:25 UTC and was finished in 2/11/2016 11:59 PM UTC. The winner of the competition would get a chance to get a direct interview with the company.

The **motivation** behind choosing this task as my capstone project fundamentally lies on the structure and the goals of the company's Data Science Team. The foundation on which a data science team rests is the culture and perception of data elsewhere in the organization. So defining how we think about data has been a prerequisite to ingraining data science in business functions. Each sets of data conveys about the uniqueness of customers, A datum is an action or event, which in most cases reflects a decision made by a person. If you can recreate the sequence of events leading up to that decision, you can learn from it; it's an indirect way of the person telling you what they like and don't like — this property is more attractive than that one, I find these features useful but those — not so much. Therefore, as an aspiring Data Scientists, by task is to get in the mind of the people who uses the product and analyze their needs and requirements. This is what I am trying to do in this first ever Data Science Project of mine.

3.0 Datasets

There are in total six Datasets that was provided by Airbnb to Kaggle as part of the competition. They are:

- `age_gender_bkts.csv` - summary statistics of users' age group, gender, country of destination.
- `sample_submission.csv` - correct format for submitting your predictions.
- `countries.csv` - summary statistics of destination countries in this dataset and their locations.
- `sessions.csv` - web sessions log for users
    - user_id: to be joined with the column 'id' in users table
    - action
    - action_type
    - action_detail
    - device_type
    - secs_elapsed
- `test_users.csv` - the test set of users and contains the following columns of information:
    - id: user id
    - date_account_created: the date of account creation
    - timestamp_first_active: timestamp of the first activity

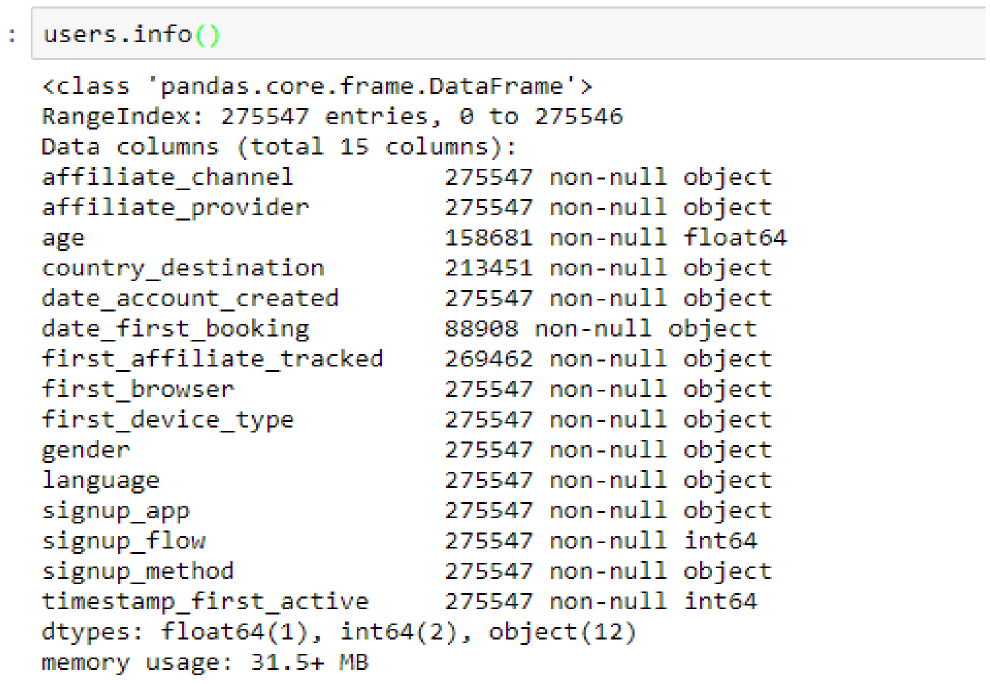
    - date_first_booking: date of first booking
    - gender
    - age
    - signup_method
    - signup_flow: the page a user came to signup from
    - language: International language preference
    - affiliate_channel: what kind of paid marketing
    - affiliate_provider: where the marketing is e.g. google,craiglist,other
    - first_affiliate_tracked: what's the first marketing the user interacted with before signing up
    - signup_app
    - first_device_type
    - first_browser
    - country_destination: this is the **target variable** that needs to be predicted
- `train_users.csv` – the training set of users

4.0 Exploratory Data Analysis

The following description of Analysis is provided without the complete code, that can be found on Github (Python Code-Github Repo).

4.1 Airbnb Data Analysis

All the necessary modules has been imported in the notebook for the Initial Exploratory Data Analysis of all the provided Datasets. The path of the files in the local repository is thus specified and the Datasets is being stored in the variables as `age_gender`, `countries`, `sessions`, `train_ users` and `test_users`. The `train_users` and `test_users` datasets are then merged together and named as `users`. The `users` Dataframe contains the following information:

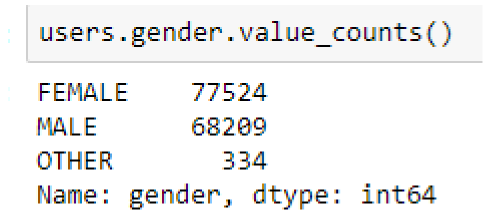


```
: users.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275547 entries, 0 to 275546
Data columns (total 15 columns):
affiliate_channel          275547 non-null object
affiliate_provider         275547 non-null object
age                        158681 non-null float64
country_destination        213451 non-null object
date_account_created       275547 non-null object
date_first_booking         88908 non-null object
first_affiliate_tracked    269462 non-null object
first_browser              275547 non-null object
first_device_type          275547 non-null object
gender                     275547 non-null object
language                   275547 non-null object
signup_app                 275547 non-null object
signup_flow                275547 non-null int64
signup_method              275547 non-null object
timestamp_first_active     275547 non-null int64
dtypes: float64(1), int64(2), object(12)
memory usage: 31.5+ MB
```

Fig 1: Users DataFrame column names and types of variables

While exploring the DataFrame in further details, there was a lot of redundant data that needed to be cleaned and dropped. For example, Airbnb follows a strict regulation that the user needs to be 18 or older in order to create an account, therefore, I decided to drop off any users that are below 18 and above 110 years of age. The "gender" column had 4 unique types of values: "MALE","FEMALE",

"-unknown-" and "Other". All the "-unknown-" entrees is thus replaced with "NaN" values in order to get a coherent model and gender counts. The other Data Wrangling steps would be discussed in further details later on this report.



```
users.gender.value_counts()

FEMALE    77524
MALE      68209
OTHER       334
Name: gender, dtype: int64
```

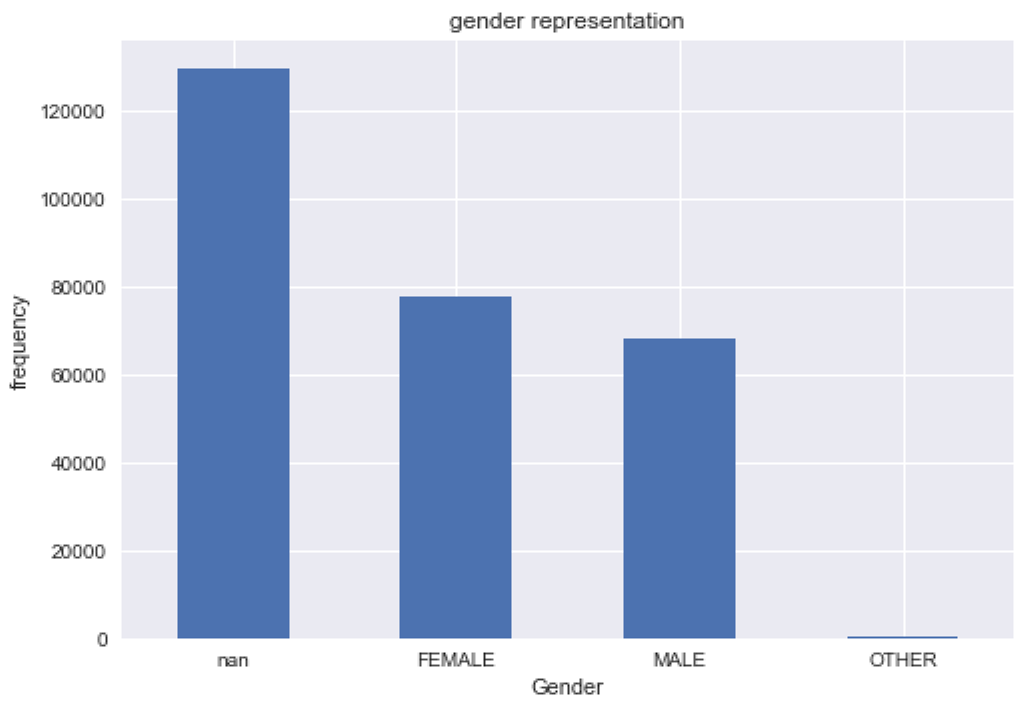


Fig 2: "Gender" frequency bar plot

In order to find a correlation between the age of the users and the "signup_method", a bar chart is plotted to get a visual representation of the data.

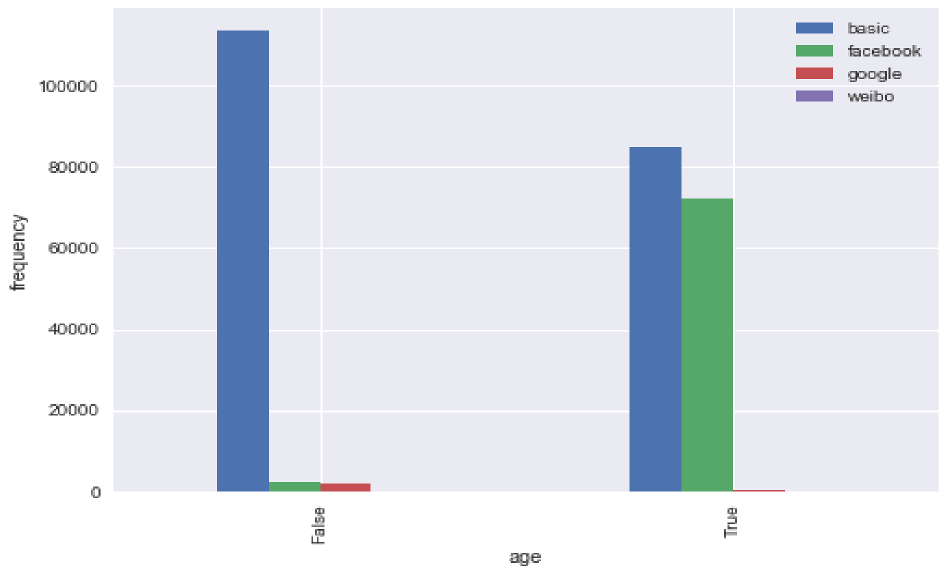


Fig 3: Age and Signup Flow bar plots

The above chart which have False shows those users which doesn't lie in the age group between 18 to 110 and the True otherwise. There is a huge popularity variation for facebook in the two users group.This makes sense since no user can make any registration in Facebook below 18 and users above 110 years of age wouldn't know much about Facebook. Therefore, the age values above 110 and below 18 is being replaced with NaN values since those are outliers. The age group frequency Distribution is as shown below.

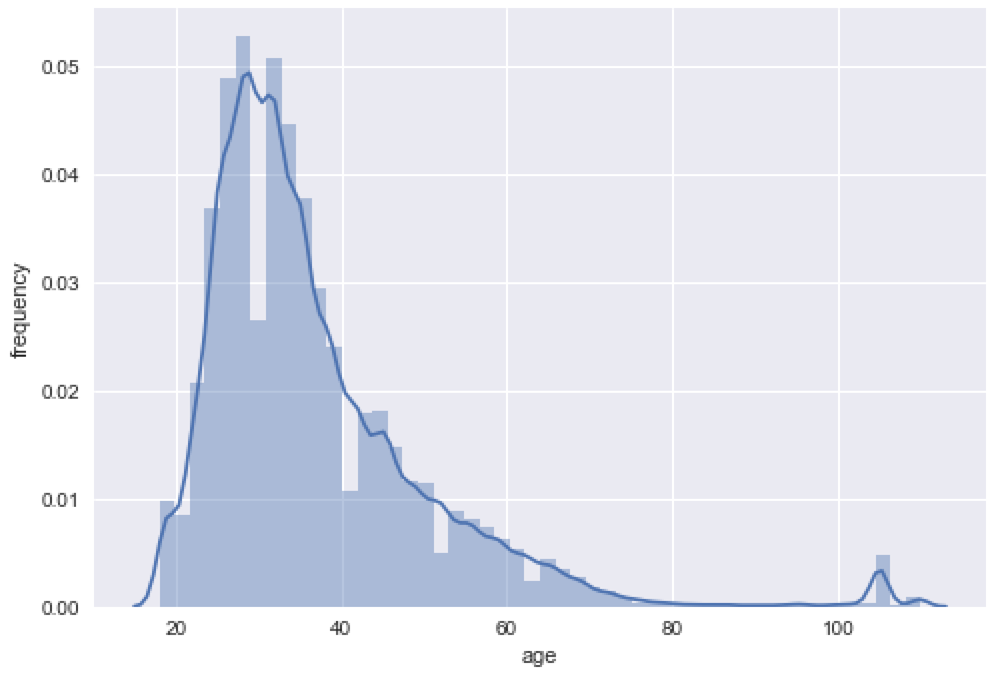


Fig 4: Frequency plot of the various age distribution

After that, to find out which method is the most popular "signup_method", a bar plot was made only for the signup_method.

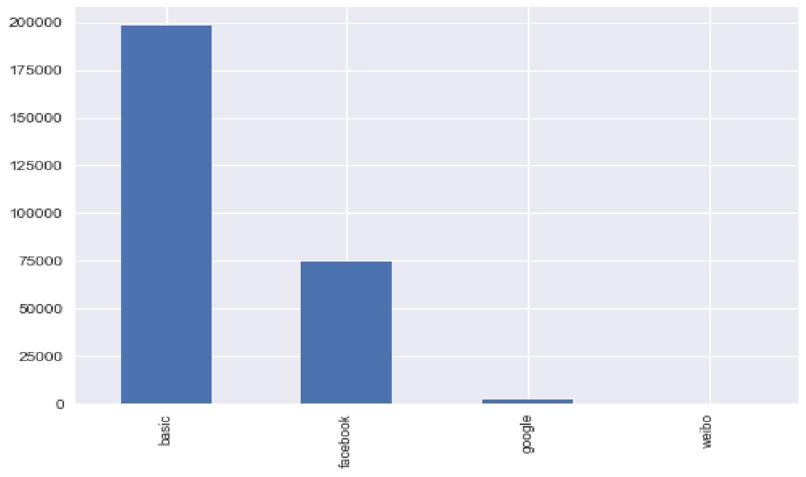


Fig 5: Signup method plot

The "basic" method of signup seems to be more popular among users."Facebook" signup is almost 75000 ranking to be the second place.Next, the "affiliate_provider" was plotted in order to see the flow. It shows that Direct Marketting is the most effective strategy.

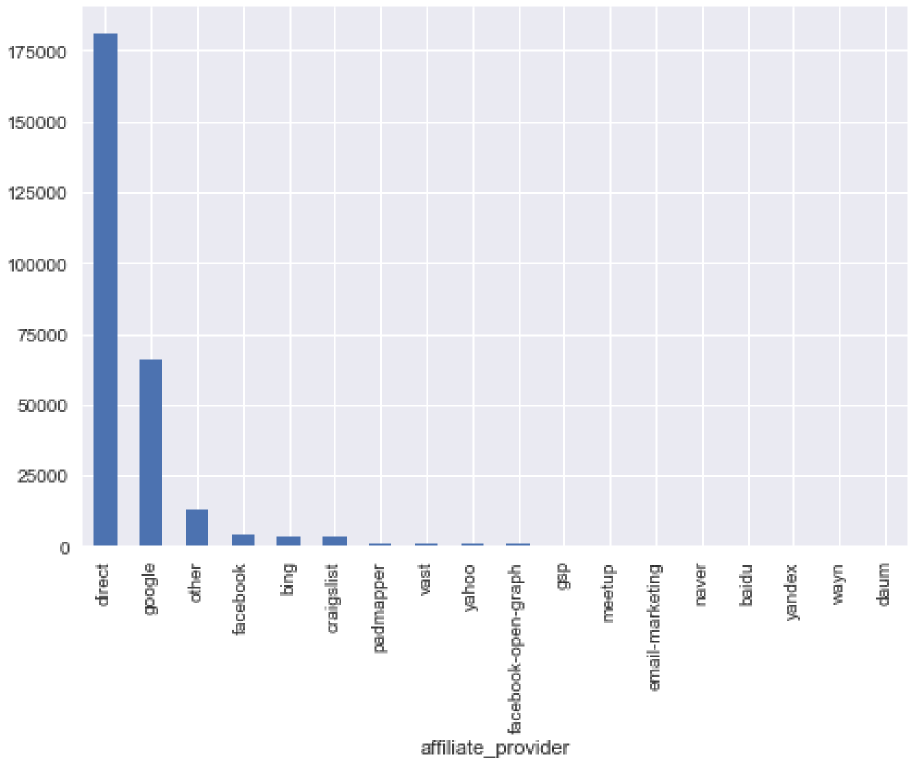


Fig 6: Affiliate provider frequency

It is important to see how many NaN values are there in each column which will give a good portray of how much datas are missing.

```
age                           42.412365
date_first_booking            67.733998
first_affiliate_tracked        2.208335
gender                        46.990169
```

The above values are the percentage counts of the actual data. Date_first_booking has 67.73% , Gender has 46.99% and the age column has 42.41% missing values that will contribute a lot of discrepancies.

The Dataset is further divided into two age groups as "older" and "younger" generation. People those who are in the age group of between 18 to 45 are classified as younger and those who are more than 45 but less than 110 are labeled as "older". A plot was made to find what is the favorite destination between these two age groups. The plot is shown as below.

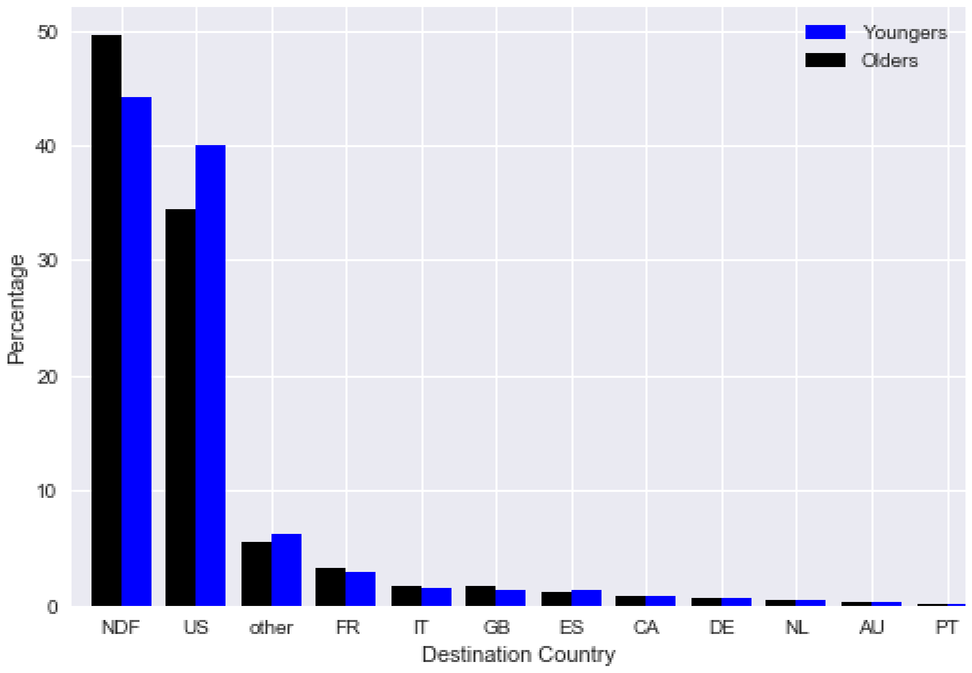


Fig 7: Destination Country between two age groups of users

There were a lot of missing data for the destination country. But, U.S. was the most popular destination for both the age groups. This piece of information helped to classify the Destination Country from

Categorical Data to Numerical Data. To get a deeper understanding of the users choice, the age group is classified into various generation names; Users between the age of 18 and 21 is classified as "generation z", those who are in between the age group of 22 to 37 lies in "millenals generation", the age group between 38 to 52 is categorized as "Generation X", the users those who are in between 53 to 71 are "Baby boomer generation", Users that lies in the range of 72 to 92 are classified as "Silent generation", Users that are in between 93 and 107 are categorized as "Greatest Generations" and the users that are in the age range between 108 to 112 are the "interbellum generation". A bar plot was then made in order to get a visual representation of the most favorite destination between various age groups.

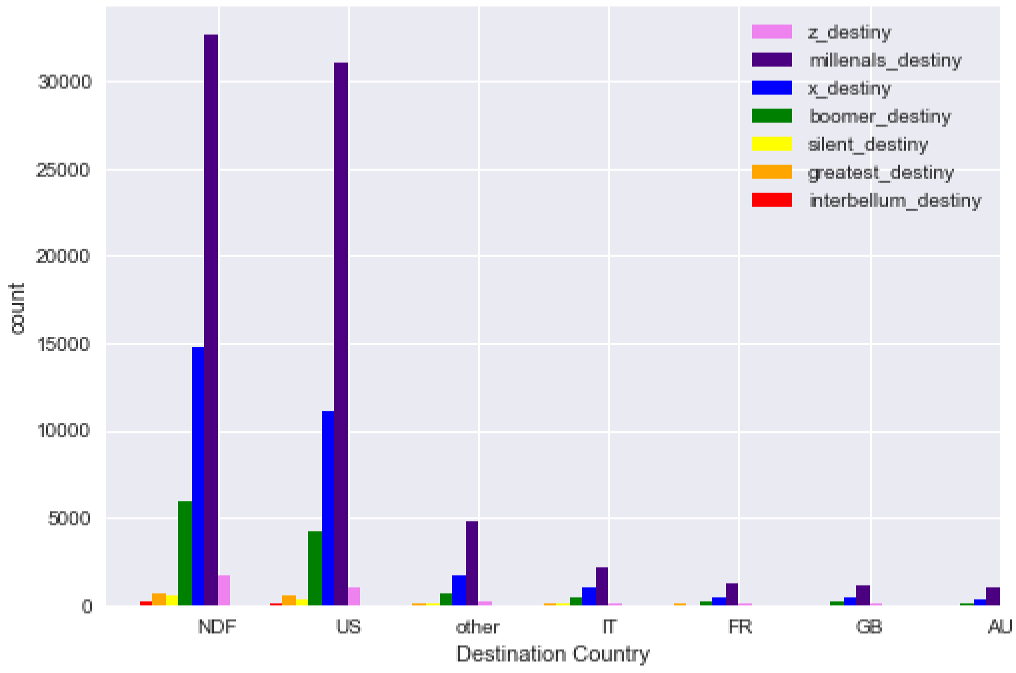


Fig 8: Destination Country vs generations plot

The plot shows that Millenals have the highest percentage of travelling and their favorite destination is 'US'.

After finishing much exploration with different age groups, the exploratory data analysis was further taken to gender versus country destination. All the male users from various age groups are then classified as" male" and the female users to be "female". The missing values and the "other" gender type has been disregarded for this purpose.

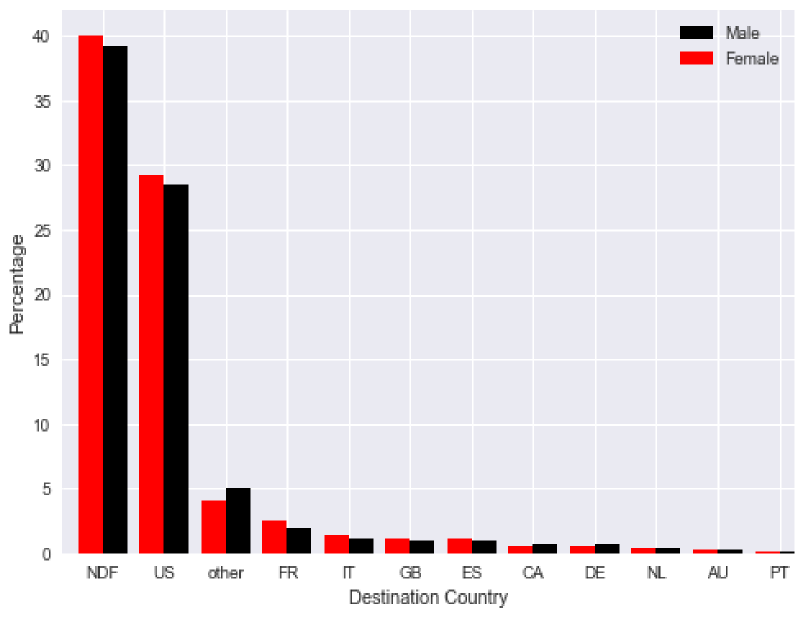
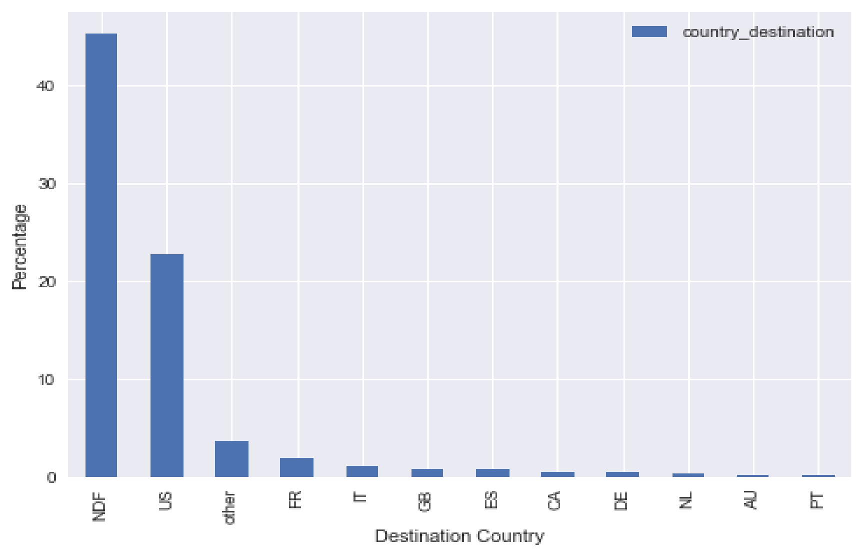


Fig 9: Gender classification with Destination Country

There is not much differences between the favorite destinations of the users based just on Genders. Therefore, it is safer to say that the vacation destinations doesn't vary on gender basis.

A plot was then made in order to see the percentile distribution of the country destination.

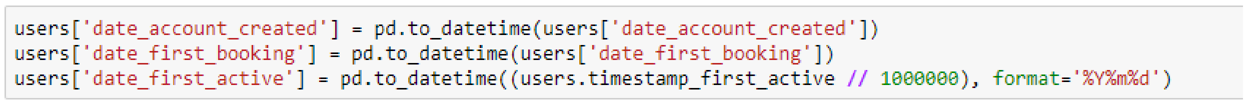


<u>Fig 10 : Destination Country distribution as a percentage of the overall population sample</u>

"No Data Found" seemed to be more than 46% of the country destination distribution.

4.2 Time Series Analysis

The "date of the first booking", the "date of account created" and "date of first booking" needs to be converted to datetime format in order to experiment along with other attributes and is being done by the following codes:



```
users['date_account_created'] = pd.to_datetime(users['date_account_created'])
users['date_first_booking'] = pd.to_datetime(users['date_first_booking'])
users['date_first_active'] = pd.to_datetime((users.timestamp_first_active // 1000000), format='%Y%m%d')
```

<u>Fig 11: Conversion of Datetime format of the three attributes</u>

Once it has been accomplished, it is possible to do some Data Wrangling steps now. As a preliminary step, a time series analysis plot has been made for the "date_account_created".

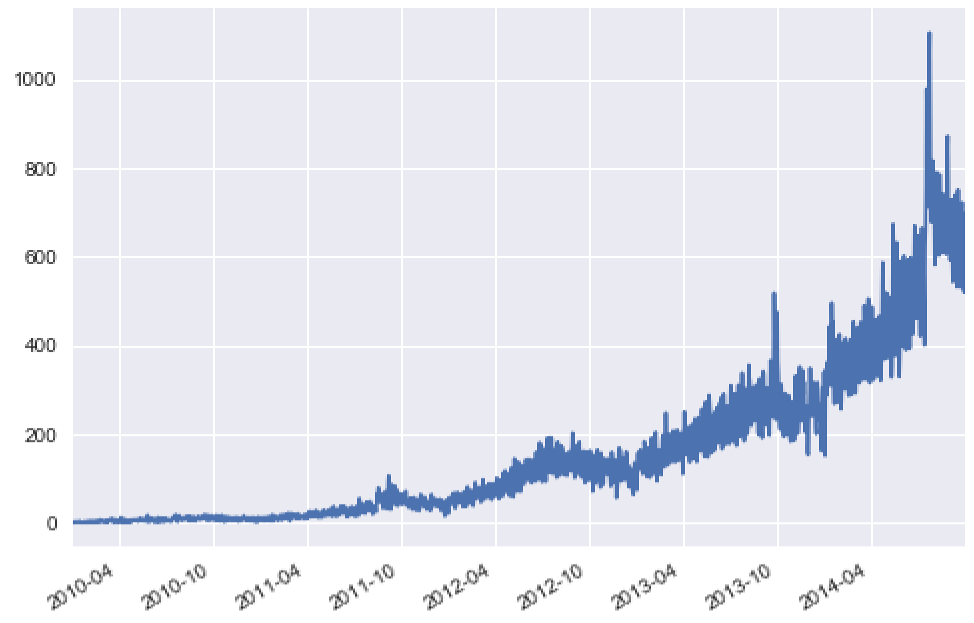


Fig 12: Time series Analysis for the Date Account Created

Since April 2010, there wasn't much users that created account. But overtime, there was a steady growth in the users base and a lot of activity and growth from October 2013 to April 2014.After April 2014, there was a peak near the December 2014 which might be because of the holiday season. To get a holistic idea of what's going on, a yearly plot was made for the users that activated their account first time.

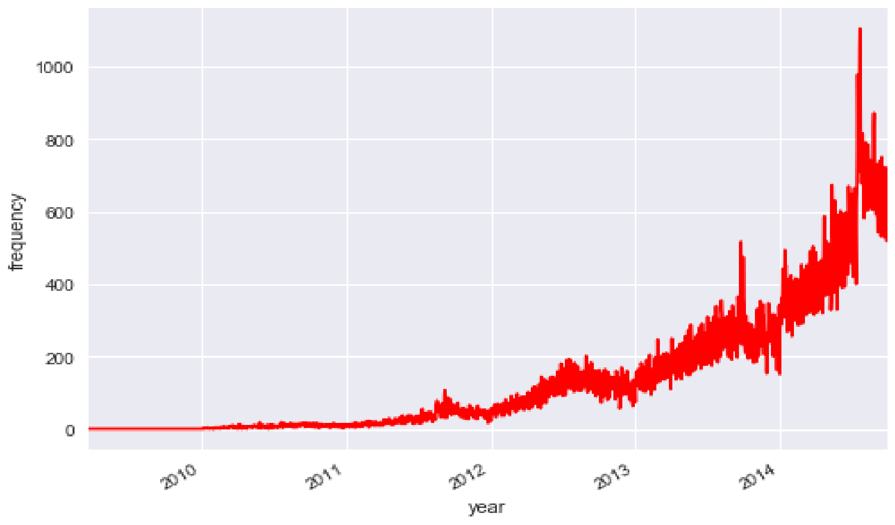


Fig 13: Users "first activity" yearly time series analysis

The above two graphs shows a clear increase in the userbase of Airbnb in the time span of three years. Both the graph has similar patterns of growth. It is very much noticeable and this should give credits to Airbnb. However, the trend of growth is not smooth. There are peaks and hikes in October 2013 and right around April 2014.This gives a chance to do in depth analysis to find out what happens that may have caused such growth in the user base.

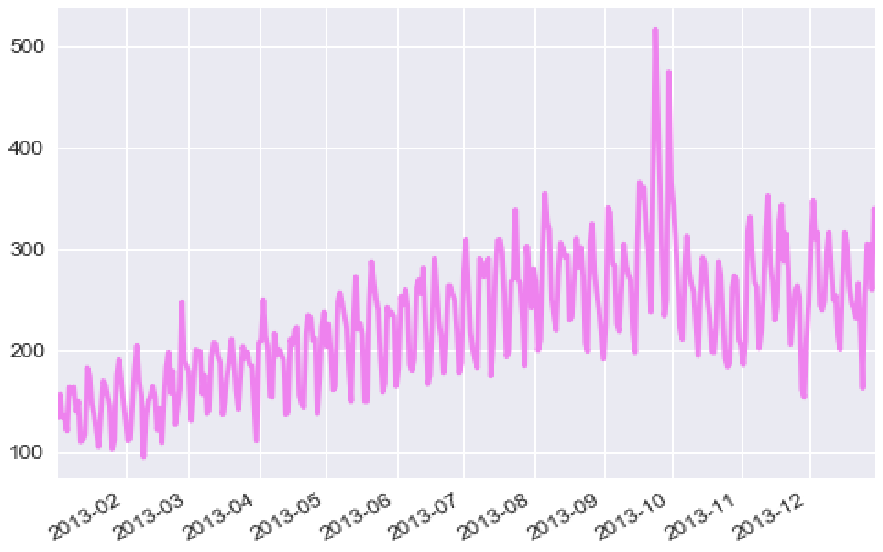


Fig 14: Users of 2013 that activated their account first time

There is a steady pattern from the month of February till month of October. There is a sudden hike in October 2013.Let's see what's going on in there.

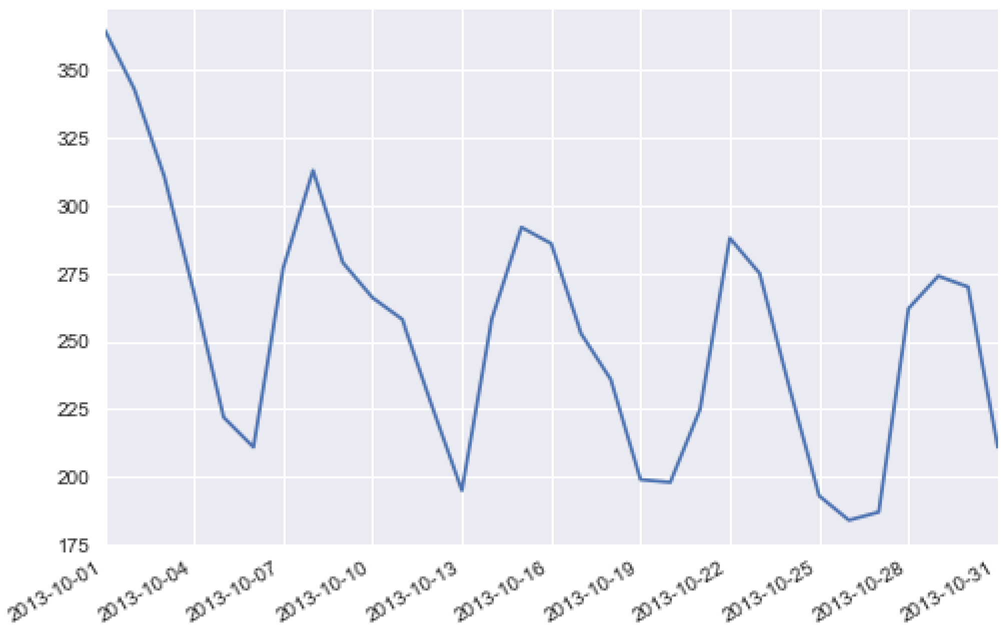


Fig 15: In depth analysis on the Users' activity in October 2013

At the first date of October 2013,which is a Tuesday, there were a lot of active users. Then the user activity dropped by significant amount in 4th of the month and kept on dropping throughout weekends. It again started to pick up on October 7th,which is a Tuesday and kept on rising at a steady pace. This clearly shows a trend that users are more tend to book in the middle of the weeks and active on weekdays rather than weekends. During weekends, there is a clear trend of less activity of booking which makes perfect sense. It may be that the users book during the weeks for the upcoming weekends for their vacation destination and tend to book less for the weekdays as they have to attend work.

5.0 Inferential Statistics

After the required Data Wrangling and Exploratory Data Analysis, the DataFrame is ready for further digging. A new column is made as "destination" column where all the instance where it is 'US' is categorized as **1** and the rest of them to be **0**. After that, the Male and Female travelers comparison is done side by side.

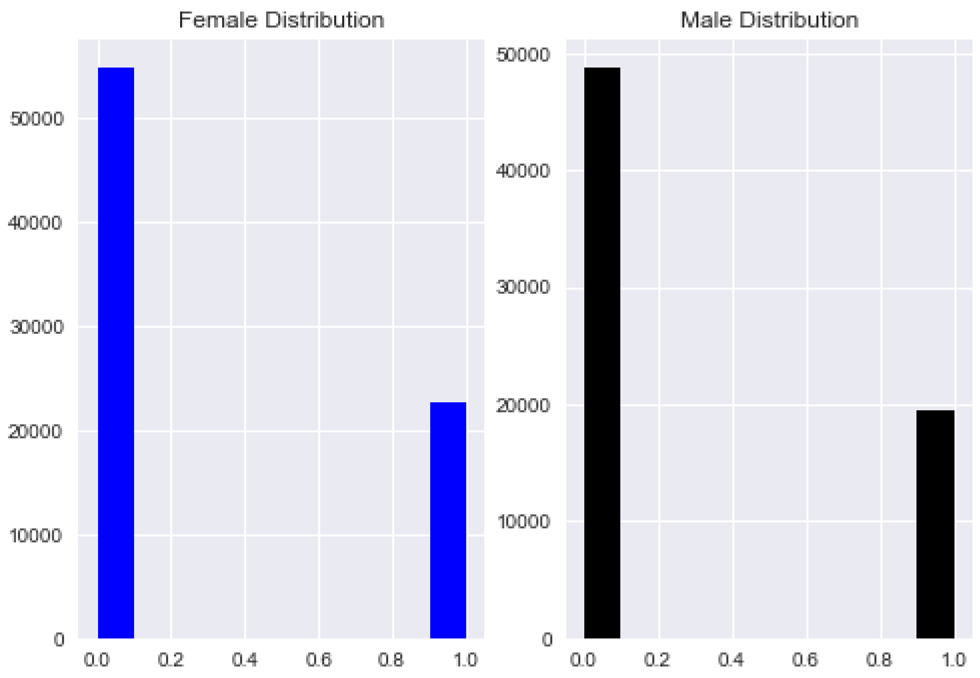


Fig 16 : Male and Female Population Distribution categorized by destination countries

Once it has been accomplished, a hypothesis testing methodology is being applied in order to find whether there is any actual difference between the statistical units of measurements. The Hypothesis test is as followed:

H_O = No difference between the mean value of the Male Population and the mean value of the female population = $P_M = P_F$

H_1= There is a difference between the mean value of the Male Population and the mean value of the female population = $P_M \neq P_F$

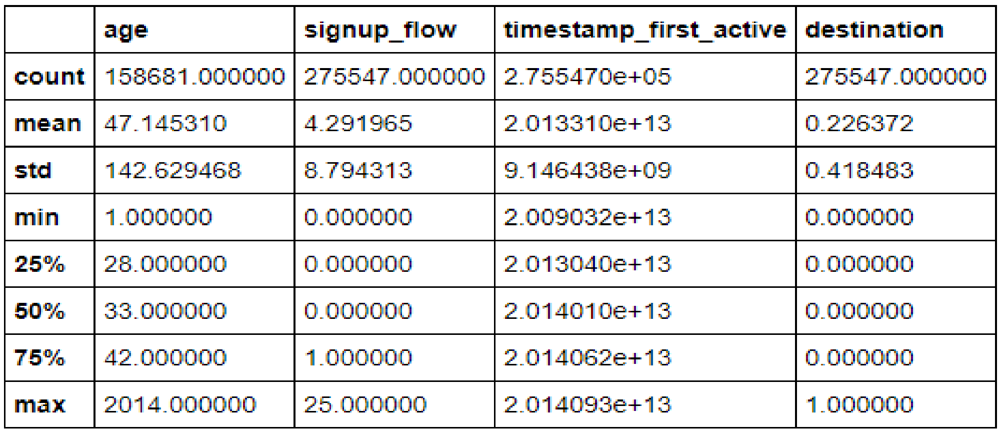
This is a two-sided test for the null hypothesis that 2 independent samples have identical average (expected) values. This test assumes that both the populations have identical variances by default.

By applying the `stats.ttest_ind(a,b,equal_var=True)` the calculated p-value is

0.001676147617343367 and the test statistic to be statistic=3.1423785772168351.We can use this test, if we observe two independent samples from the same or different population, e.g. exam scores of boys and girls or of two ethnic groups. The test measures whether the average (expected) value differs significantly across samples. If we observe a large p-value, for example larger than 0.05 or 0.1, then we cannot reject the null hypothesis of identical average scores. If the p-value is smaller than the threshold, e.g. 1%, 5% or 10%, then we reject the null hypothesis of equal averages.

Since, the p-value is very smaller than the threshold, then we can **safely reject the Null Hypothesis of equal mean value of male and female population average**.

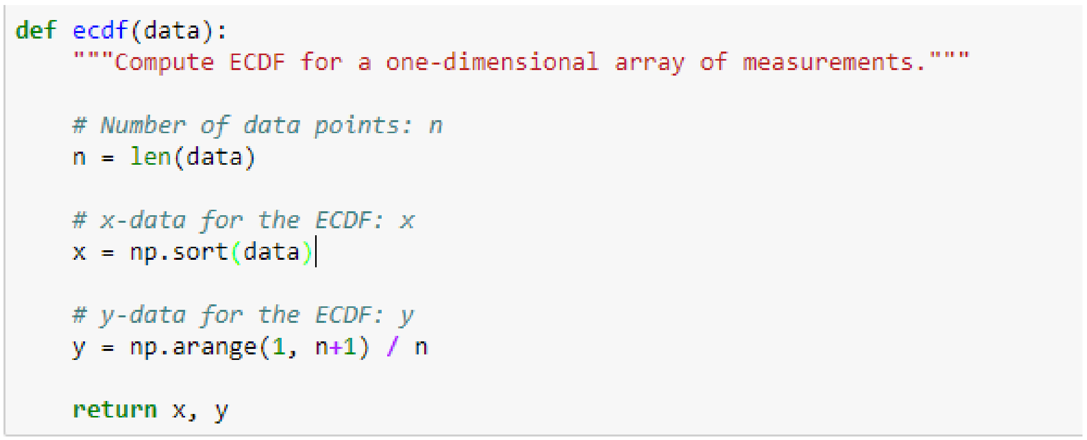
The other test statistic is as follows:



	age	signup_flow	timestamp_first_active	destination
count	158681.000000	275547.000000	2.755470e+05	275547.000000
mean	47.145310	4.291965	2.013310e+13	0.226372
std	142.629468	8.794313	9.146438e+09	0.418483
min	1.000000	0.000000	2.009032e+13	0.000000
25%	28.000000	0.000000	2.013040e+13	0.000000
50%	33.000000	0.000000	2.014010e+13	0.000000
75%	42.000000	1.000000	2.014062e+13	0.000000
max	2014.000000	25.000000	2.014093e+13	1.000000

CDF (Cumulative Distribution Function)

In order to measure the CDF and ECDF of various different attributes, a function has been written from scratch. The function is as follows:



```
def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""

    # Number of data points: n
    n = len(data)

    # x-data for the ECDF: x
    x = np.sort(data)

    # y-data for the ECDF: y
    y = np.arange(1, n+1) / n

    return x, y
```

Fig 17 : Defining the Cumulative Distribution Function

The above function is then applied to each of the parameters of the DataFrame and a sequential plot was made consecutively. For the "timestamp_first_active", an array of percentiles bins is being defined. The percentiles are 2.5,25,50,75 and 97.5 respectively where we want to see how the data is distributed on those points. This is the plot that was obtained.

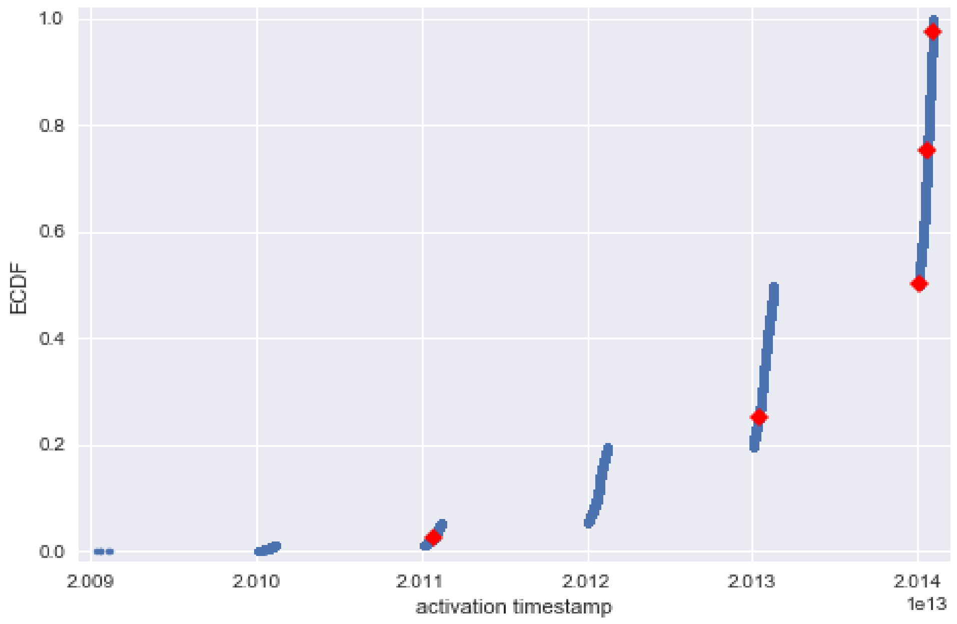


Fig 18 : ECDF for activation timestamp and percentage distribution

The obtained percentile values are as follows on those aforementioned range :

```
[  2.01107252e+13   2.01304032e+13   2.01401032e+13   2.01406182e+13
   2.01409192e+13]
```

And the ecdf are:

```
 (array([20090319043255, 20090523174809, 20090609231247, ...,
20140930235408,
        20140930235430, 20140930235901], dtype=int64),
 array([  3.62914494e-06,   7.25828987e-06,   1.08874348e-05, ...,
          9.99992742e-01,   9.99996371e-01,   1.00000000e+00]))
```

The same methodology is applied for the age data and this is what has been obtained.

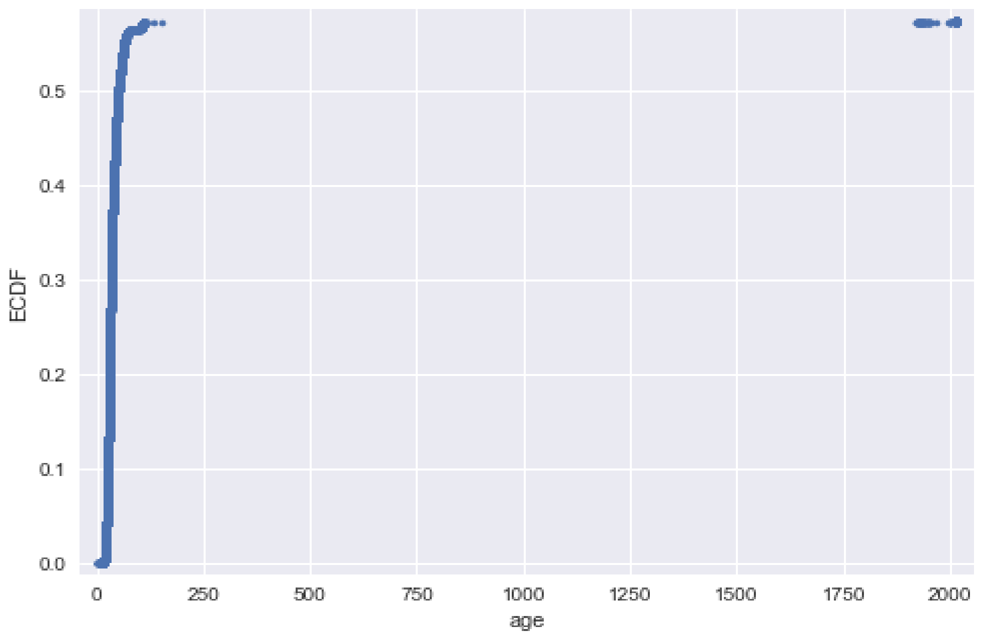


Fig 19 : ECDF for age

The obtained value of the arrays are:

```
(array([  1.,   1.,   1., ...,  nan,  nan,  nan]),
 array([  3.62914494e-06,   7.25828987e-06,   1.08874348e-05, ...,
          9.99992742e-01,   9.99996371e-01,   1.00000000e+00]))
```

And ECDF for the "signup_flow" with the percentile distribution is as follows:

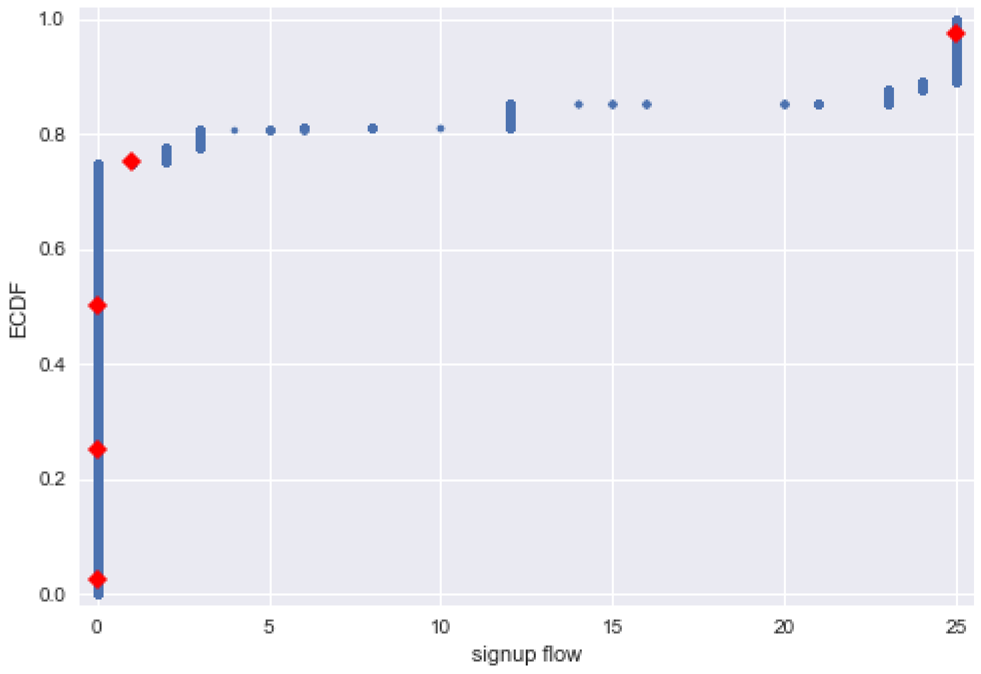


Fig 20: ECDF for signup flow and percentile distribution

The two ECDF for the "signup_flow" and "timestamp_first_active" is done side by side in order to get a proper comparison between the data.

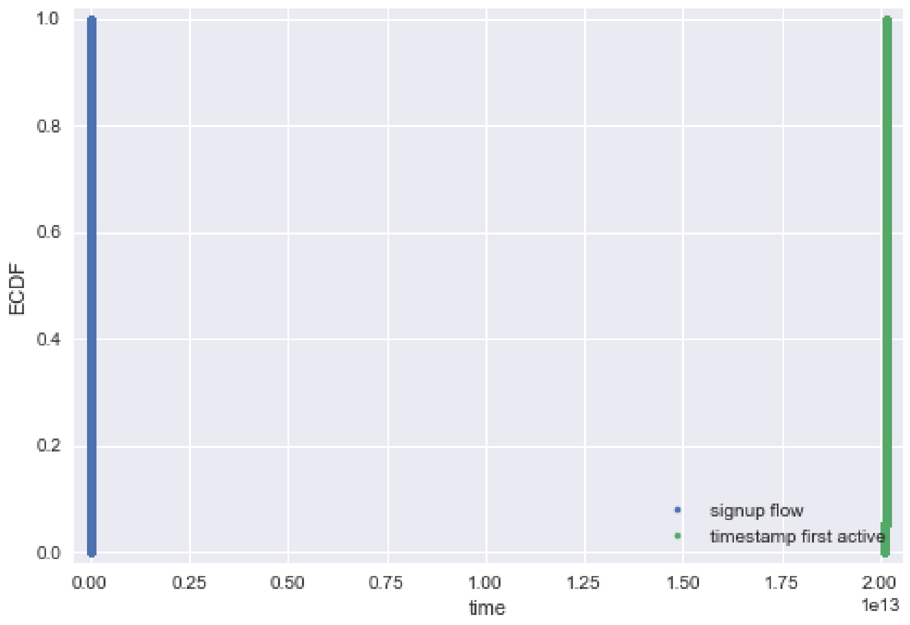


Fig 21: ECDF of signup flow and timestamp first active

After doing much of CDF and test statistics, it was required to go back to basic statistical measurements to get an idea of how the data is being distributed. Therefore, plain histogram was plotted for the "signup_flow" and "timestamp_first_active". Below are the two plots:

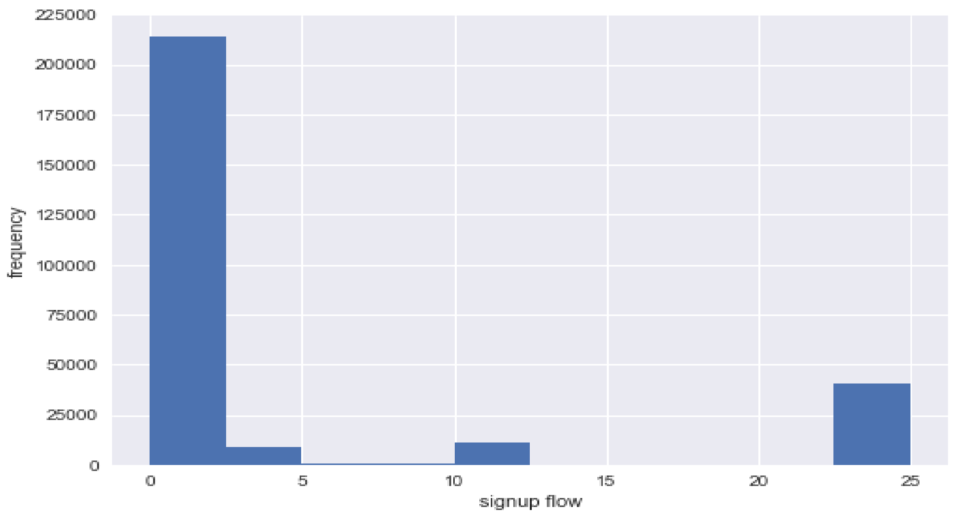


Fig 22: Histogram plot for the signup flow against frequencies

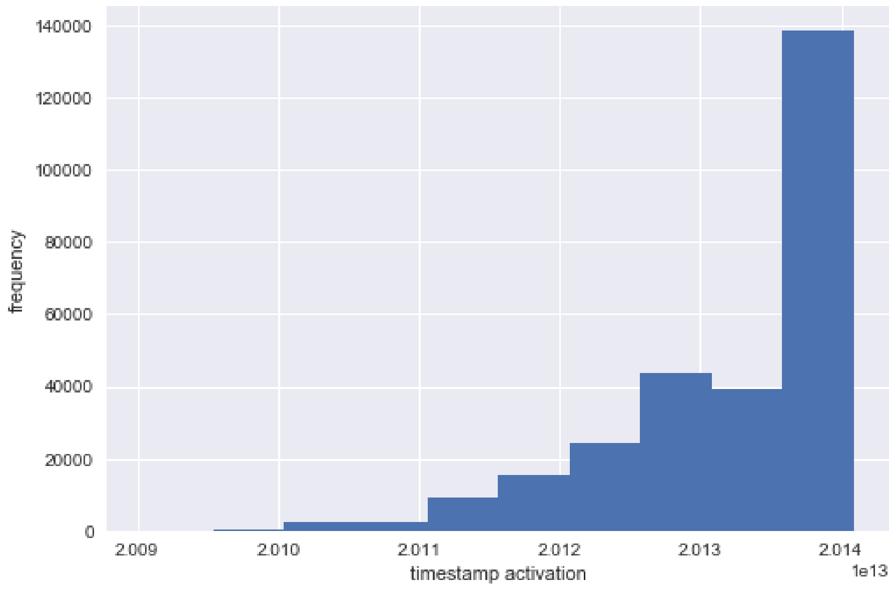
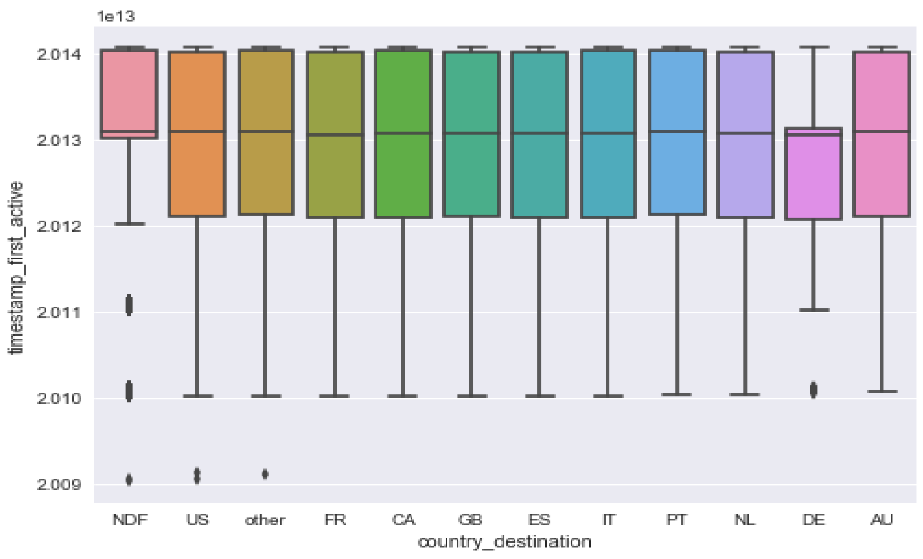


Fig 23: Timestamp Activation Histogram Distribution

The "timestamp_activation" histogram is heavily skewed to the left and solidifies the assumption we made in the Exploratory Data Analysis section that the Users' activity increased in an steady rate but it changed heavily around 2014.

Finally, a box plot was made against individual "country_destination" against "timestamp_first_active".



<u>Fig 24: "timestamp_first_active" vs "country_destination"</u>

The above box plot reveals the fact that the data from 2010,2011 and 2012 was stored and there wasn't much "lost" data. However, there are a lot of lost data from 2013 and onwards.

A scatter plot was made with the "signup_flow" and "activation_timestamp". This is what the plot looks like:

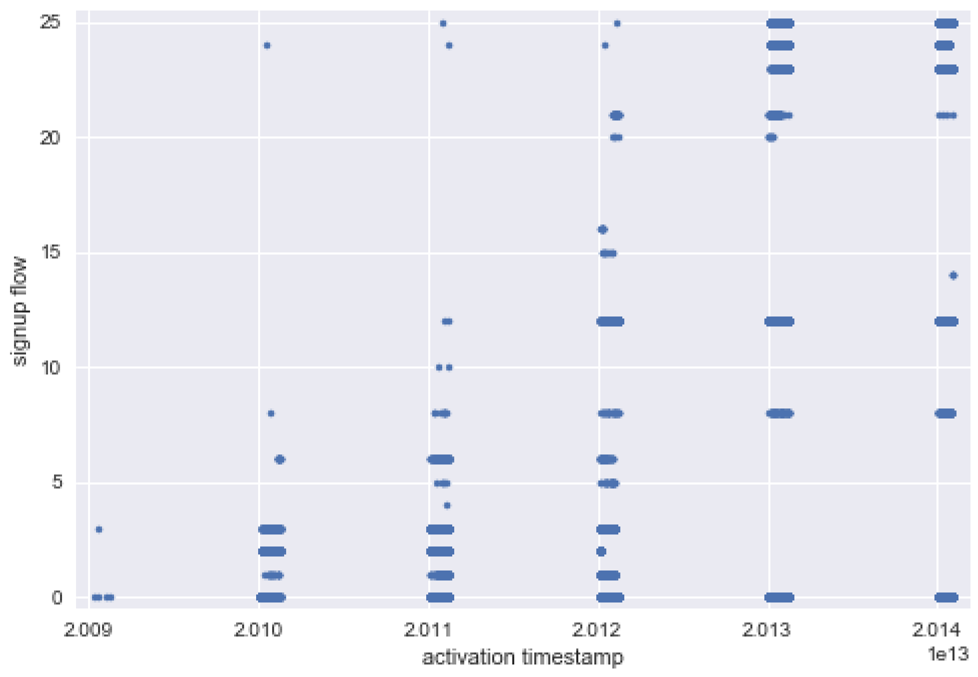


Fig 25: "Signup flow" vs the "activation timestamp"

A probability value was computed for the users to choose their next vacation destination to be in US by using the following formula:

```
u=users

u=u[u.country_destination=='US']

len(u)

#Probability of staying at U.S. can be computed by:

p=len(u)/len(users)
```

The p was calculated to be `0.2263715446003767.`
A Binomial Distribution plot is then made with the above test statistics :

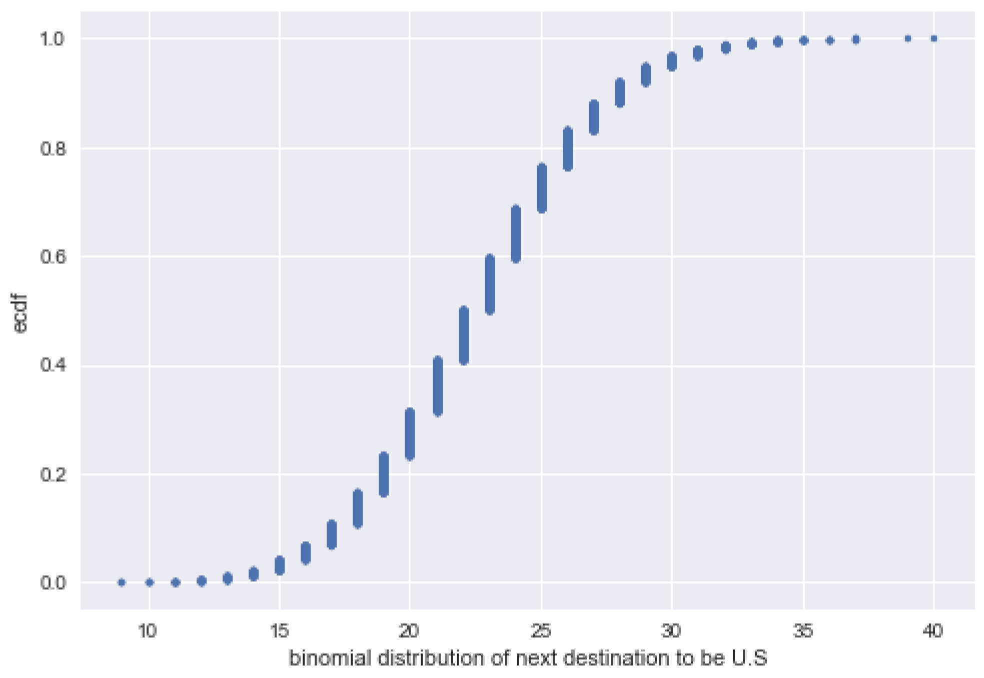


Fig 26 : Binomial Distribution of the next destination to be in US

There is about 90% chance of that less than 30 users would choose U.S. to be their next vacation destination. The value of the CDF at x = 30 is about 0.95, so the probability that x < 10 is 0.95. Thus, the probability that x > 10 is 0.05.

Thus, a histogram was plotted with adjusted bin size. This is how the Distribution looks like:

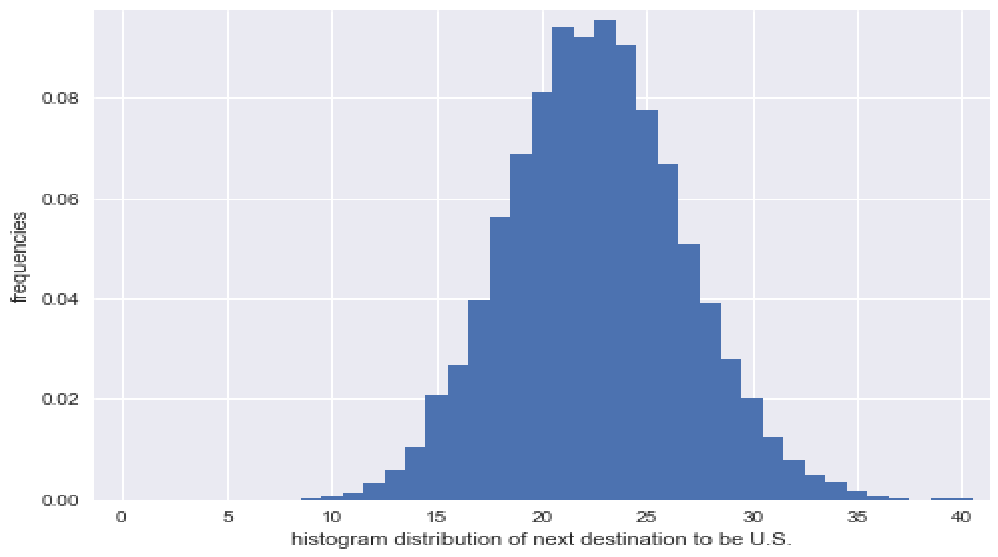


Fig 27: Histogram distribution of next destination to be US

The above histogram satisfies the Central Limit Theorem and converges to a Normal Distribution Function when taken a large sample size.

Pearson Correlation Coefficient was also computed between the "timestamp_first_active" and "signup_flow". The Pearson Correlation Coefficient between signup flow and first activity is 0.23215.

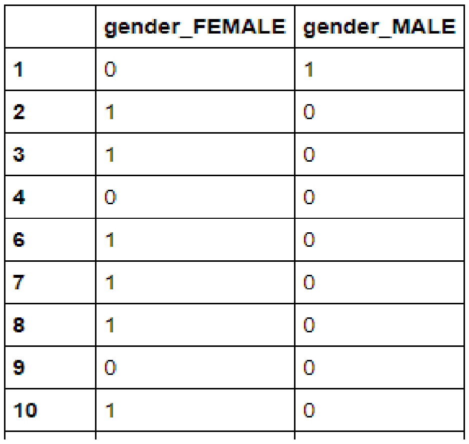
6.0 Machine Learning

The task for this dataset is to find where the next vacation destination will be for the new users of Airbnb. Therefore, we need to choose a right classifier that can do this task in efficient manner. But, Machine Learning classifiers don't perform very well with Categorical Data. Therefore, it is required to change our target data from categorical to numerical data type. In the next sections, a step by step approach of how it is being done is discussed thoroughly.

6.1 Feature Selection, Dummy Variables and One Hot Encoding

All the necessary modules has been imported at the beginning of the Ipython Notebook. Therefore, it is not necessary to import any new modules. Pandas module has been imported as `pd` throughout the work flow. Pandas module has a built-in function called `get_dummies` that converts categorical values into numerical values. It could also be done by using `One Hot Encoding`. The following line of code does this:

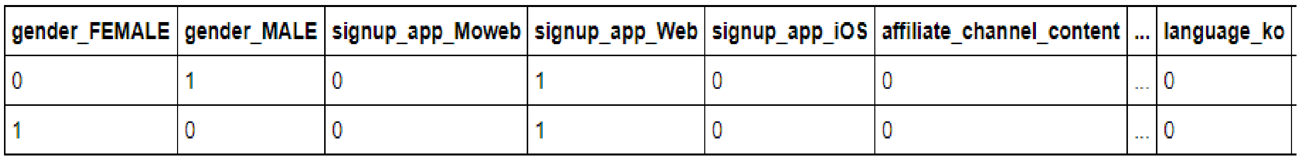
```
gender_dummies=pd.get_dummies(df.gender,prefix='gender').iloc[:,0:2]

gender_dummies
```



	gender_FEMALE	gender_MALE
1	0	1
2	1	0
3	1	0
4	0	0
6	1	0
7	1	0
8	1	0
9	0	0
10	1	0

Fig 28 : Dummy variable creation

In the above table, it can be seen that the gender_MALE is 1 and the gender_Female is 0 for the row 1.What it means is that the users in row 1 is "MALE". The gender type "other" is being dropped because if there are three categorical values then we need (3-1) =2 possible values to work with. Once it is done, the real "Gender" attributes from the original DataFrame is being dropped and the new numerical gender columns are concatenated with the original DataFrame. Other feature columns such as "signup_app","affiliate_channel","affiliate_provider","first_affiliate_tracked","first_browser","first_devi ce_type,"signup_method" and "language" is also done with similar methodology. The drop_first is set to True in order to drop the original columns and concatenate the new feature columns. This is how the modified DataFrame looks like:



gender_FEMALE	gender_MALE	signup_app_Moweb	signup_app_Web	signup_app_iOS	affiliate_channel_content	...	language_ko
0	1	0	1	0	0	...	0
1	0	0	1	0	0	...	0

Since it is not possible to include all the feature columns, therefore the keys of the DataFrame is printed as follows:

```
Index(['age', 'country_destination', 'signup_flow',
'timestamp_first_active',
       'gender_FEMALE', 'gender_MALE', 'signup_app_Moweb',
'signup_app_Web',
       'signup_app_iOS', 'affiliate_channel_content',
       ...
       'language_ko', 'language_nl', 'language_no', 'language_pl',
       'language_pt', 'language_ru', 'language_sv', 'language_th',
       'language_tr', 'language_zh'],
      dtype='object', length=116)
```

Then the DataFrame is split into two variables; X and y. X contains all my feature variables and y contains all the target variables; in our case, the "destination" countries. Now we are ready to train and test our Dataset.

6.2 Size of Train and Test Sets

After much Data preprocessing pipelines, the datas are further divided into two categories: Train and Test Sets. For this, the `train_test_split` is being imported from `sklearn.cross_validation.` Once it is done,

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,rand
om_state=42)
```
is used to split the X and y into train and test sets. The `test_size = 0.30` implies that the size of the test set is30% of the original data. The `random_state=42` is where the splitting happens on the original Data.

6.3 KNN or K-Nearest Classifier

For Supervised Classification problems, K-NN is one of the most effective and yet simple Algorithm to implement and easy to deploy. KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

KNN algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time. The principle of how KNN works the theoretical aspects of it are not discussed in this report.

How to choose the factor K?

In the KNN Algorithm, K stands for the number of neighbors that works as classifiers. With increasing number of K, the classification boundary becomes smoother.With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access on different K-value. The error rate at K=1 is always zero for the training sample. This is because the closest point to any training data point is itself. Hence the prediction is always accurate with K=1. If validation error curve would have been similar, our choice of K would have been 1. But, it is other way round and may cause overfitting. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increase with increasing K. To get the optimal value of K, you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K. This value of K should be used for all predictions. The challenge is to find the sweet spot that would be optimal for both the training and testing accuracy.

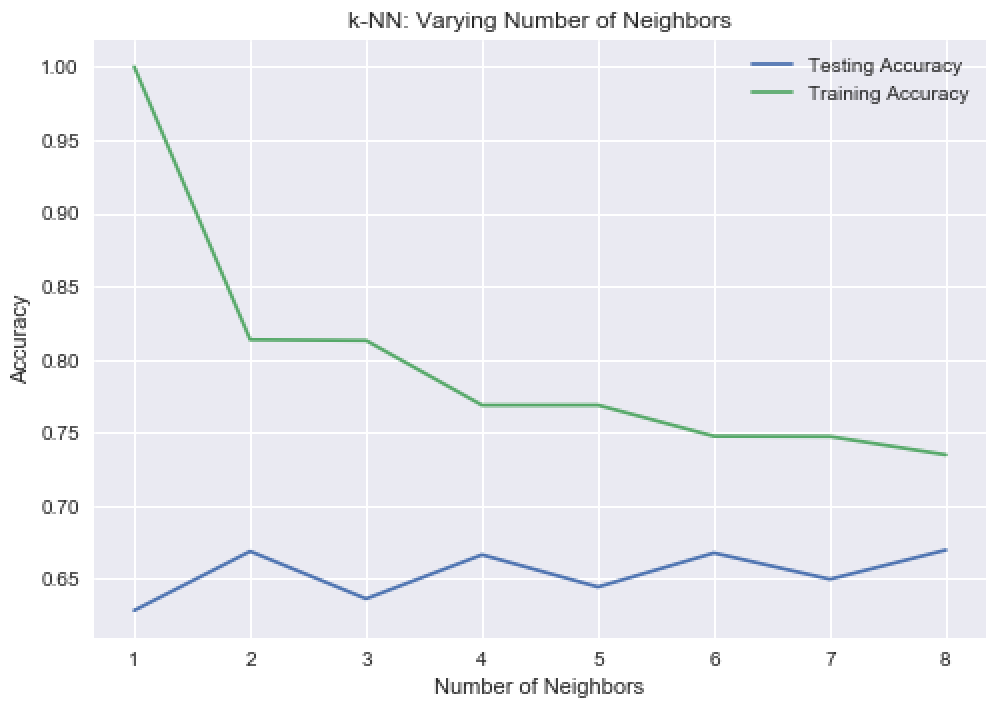


Fig 29: K-NN varying Number of Neighbors

With the k=6 value, the predictions and accuracy are as follows:

```
Prediction for X :[0 0 1 ..., 0 0 0]
Prediction: [0 0 0 ..., 0 0 0]
0.66754952855175331
```

6.4 Measuring Model Performance

The Confusion Matrix and the classification report is as follows:

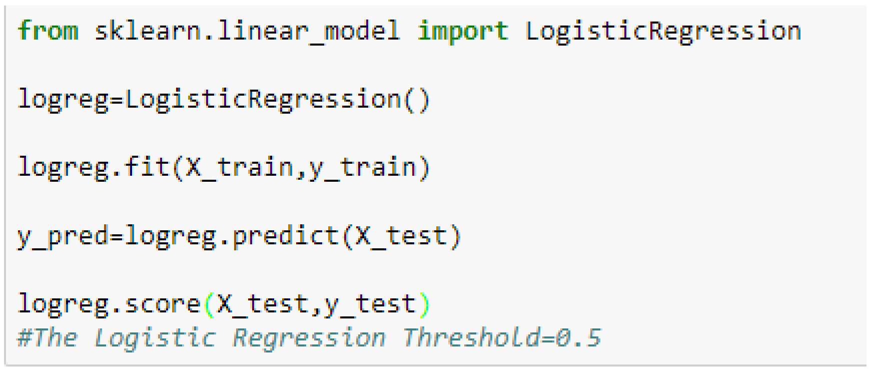
```
[[29018  3741]
 [11949  2487]]
             precision    recall  f1-score   support

          0       0.71      0.89      0.79     32759
          1       0.40      0.17      0.24     14436

avg / total       0.61      0.67      0.62     47195
```

Fig 30: Confusion Matrix and Classification Report for k=6

The support gives the number of samples of the true response that lie in that class. The precision, recall, and f1-score columns, then, gave the respective metrics for that particular class.

6.4 Logistic Regression

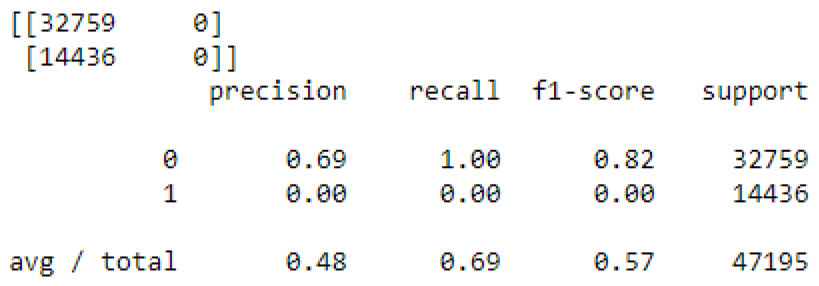
The other Algorithm that was implemented in order for the binary classification problems. The snaps of the Algorithm is given below:



```
from sklearn.linear_model import LogisticRegression

logreg=LogisticRegression()

logreg.fit(X_train,y_train)

y_pred=logreg.predict(X_test)

logreg.score(X_test,y_test)
#The Logistic Regression Threshold=0.5
```

The obtained score was `0.69412013984532261` which shows a better performance than the KNN algorithm.

The confusion matrix and the classification report is as follows:



```
[[32759     0]
 [14436     0]]
             precision    recall  f1-score   support

          0       0.69      1.00      0.82     32759
          1       0.00      0.00      0.00     14436

avg / total       0.48      0.69      0.57     47195
```

Fig 31: Confusion Matrix and Classification Report for Logistic Regression

While comparing the Confusion Matrix and the Classification Report of both the presented model, the confusion matrix obtained for the KNN classifier shows that out of 32,759 data points, it could correctly classify 29018 as class 0 or "Outside U.S." and 3741 was missed. Whereas in case of Logistic Regression, it could label all the 32759 data as class 0 successfully and there wasn't any false negative. Again, out of 14436 data that belongs to class 1 KNN labeled 11949 successfully and 2487 was misclassified. Again, Logistic Regression labeled all the 14,436 data as class 1 successfully.

Coming to Recall/Precision. They are some of the mostly used measures in evaluating how good the model works. Now the model had 32759 data points 0 class. Out of them KNN classifier was able to get 29018 elements correctly. For the KNN Algorithm, the recall for class 0 is 29018/32759 = 0.89 and the Logistic Regression gives a recall value of 32759/32759 = 1. The interpretation of Confusion Matrix is as follows:

On the upper left side, is the number of data points that is correctly labeled or **True Positive**. On the upper right side are the classes that are predicted as class 1 but belongs to class 0 in actual and those are categorize as **False Negative**. On the bottom left corner, are the **False Positive** i.e. those which are predicted as class 0 but in actual belongs to class 1and on the bottom right corner are the ones that are **True Negative** or classes that belongs to class 1 and is labeled correctly as Class 1 label. Then the accuracy can be calculated as :

$$accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}$$

Therefore, with the above formula the accuracy for KNN is 31505/47195 = 0.67 and for Logistic Regression it is 0.69.

The formula for precision is :

$$precision = \frac{t_p}{t_p + f_p}$$

The precision is also called the positive predicted value.

The calculation of recall is as follows:

$$recall = \frac{t_p}{t_p + f_n}$$

which is also known as the sensitivity value.

The F1 score is as follows:

$$\text{F1 score :} \quad 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

To put it into perspective, if the classifier has high precision then it has low false positive rate and high recall means most of the positive values had been identified successfully.

A graph was plotted to have a better visual for the True Positive and False Positive Rate known as **ROC Curve**.

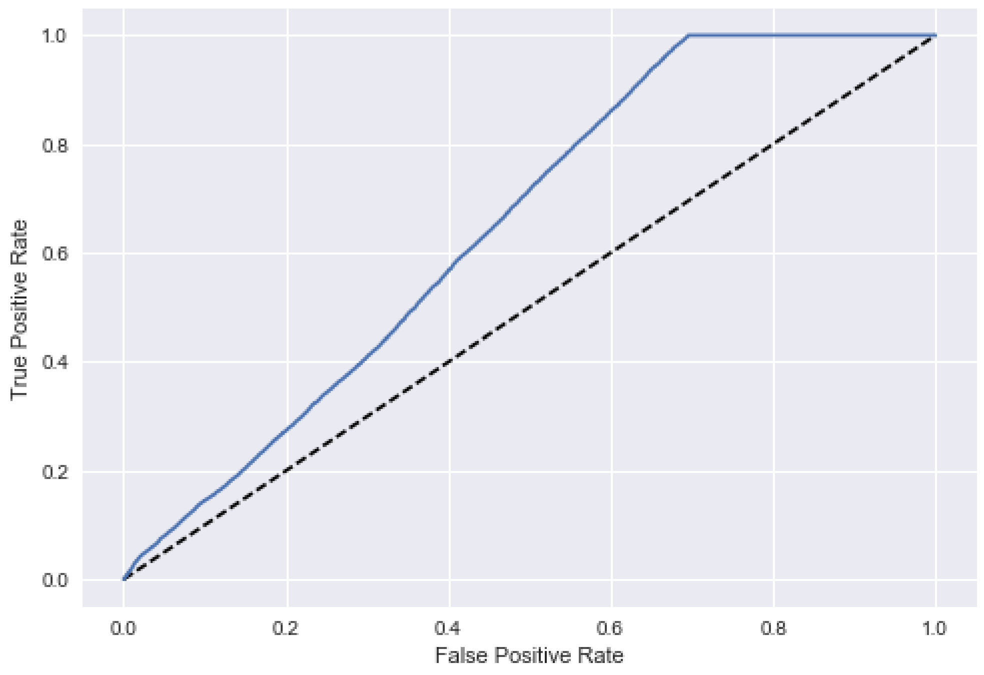


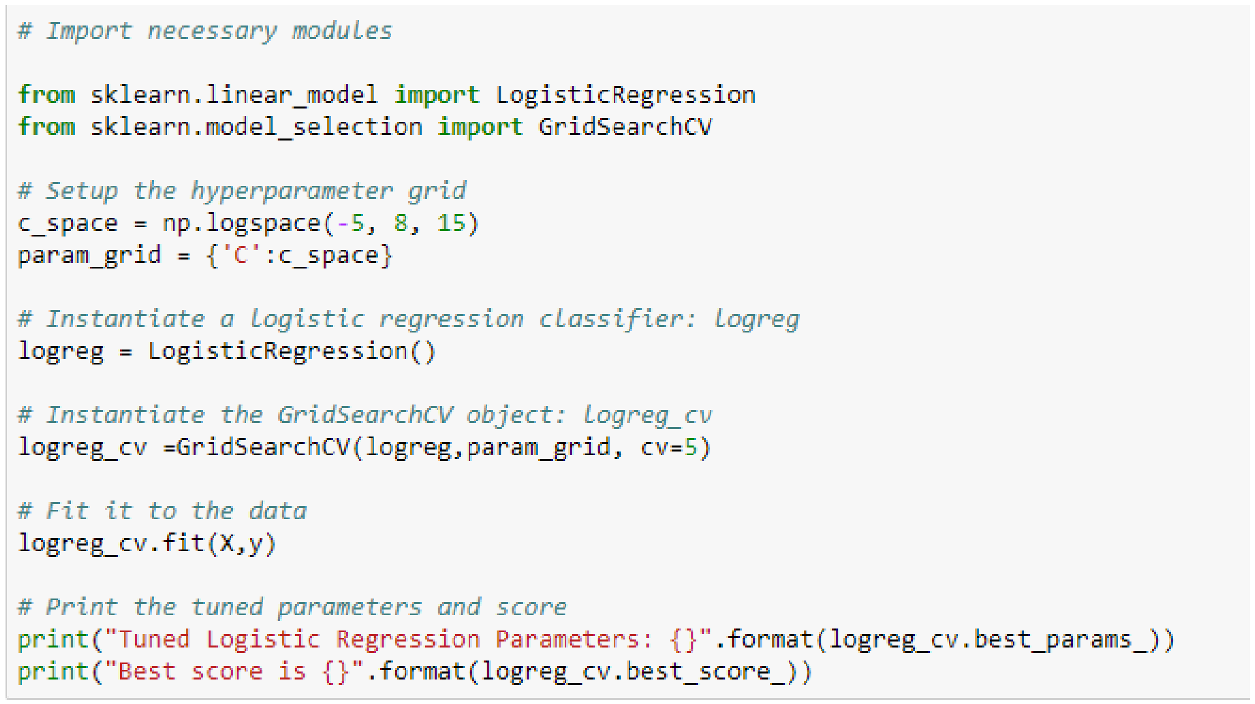
Fig 32: ROC Curve for the True Positive vs False Positive Rate

AUC scores were also computed with 5-fold Cross-Validation and the score is `AUC:`

```
0.6477268802819208
AUC scores computed using 5-fold cross-validation: [ 0.66745863
0.85891883  0.99051627  1.          1.        ]
```

6.5 Hyper Parameter Tuning

The following snippets of codes tunes the hyper parameters that has been used in the above mentioned classifiers. The code is given below:



```
# Import necessary modules

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Setup the hyperparameter grid
c_space = np.logspace(-5, 8, 15)
param_grid = {'C':c_space}

# Instantiate a logistic regression classifier: logreg
logreg = LogisticRegression()

# Instantiate the GridSearchCV object: logreg_cv
logreg_cv =GridSearchCV(logreg,param_grid, cv=5)

# Fit it to the data
logreg_cv.fit(X,y)

# Print the tuned parameters and score
print("Tuned Logistic Regression Parameters: {}".format(logreg_cv.best_params_))
print("Best score is {}".format(logreg_cv.best_score_))
```

The obtained
```
Tuned Logistic Regression Parameters: {'C':
1.0000000000000001e-05}
Best score is 0.6931125448940024
Tuned Decision Tree Parameters: {'criterion': 'entropy', 'max_depth':
3, 'max_features': 7, 'min_samples_leaf': 4}
Best score is 0.6931443282585894
```

7.0 Conclusion and Future Work

Both the model shows a good performance with 0.67% and 0.69% accuracy rate. Most new users preferred vacation destination is inside U.S. Logistic Regression is a better classifier with high accuracy rate. However, this accuracy can be further improved by reducing the number of features and dimensional reduction. I plan to implement SVM and Random Forrest and other crossbreed model in order to get a better performance for the classification of "Airbnb New Users' next Vacation Destination".