**Notes for Airbnb Project**:

We have been given 6 different csv files:

* 2 files countries.csv and age\_gender\_bkts.csv - background information
* The file sample\_submission\_NDF.csv - sample submission file containing final predictions
* The file train\_users.csv - primary dataset that we will use to train the model
* The file test\_users.csv - same format as train\_users.csv without destination country, users for which prediction needs to be made
* The file sessions.csv - supplementary data that can be used to train the model and make the final predictions

Points to Remember:

* All users in the data provided are from the USA.
* ‘other’ means there was a booking, but in a country not included in the list, while ‘NDF’ means there was not a booking.

A look at the data in train\_users.csv:

* At least 2 columns in each record have missing values.
* These missing values need to be filled or the rows excluded altogether.
* Most of the columns provided contain categorical data (11 of the 16 columns provided appear to be categorical).
* Most of the algorithms that are used in classification do not handle categorical data like this very well, and so when it comes to the data transformation step, we will need to find a way to change this data into a form that is more suited for classification.
* The timestamp\_first\_active column looks to be a full timestamp, but in the format of a number. For example, 20090609231247 looks like it should be 2009-06-09 23:12:47. This formatting will need to be corrected if we are to use the date values.

Simple codes to start off with:

Importing libraries and the data set:

Libraries Used:

1. numpy

2. matplotlib

3. pandas

Read the dataset using the function:

read\_csv()

Display the top 10 rows for example of your dataset:

df.head(10);

Basic Code:

import pandas as pd

import numpy as np

import matplotlib as plt

#Read the dataset in a dataframe using Pandas

df = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\train\_users\_2.csv")

#ERROR: SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: malformed \N character escape

#had to use double slash instead

#Print the rows

print(df.head(10))

#Summary of numerical fields

print(df.describe())

O/P:

timestamp\_first\_active age signup\_flow

count 2.134510e+05 125461.000000 213451.000000

mean 2.013085e+13 49.668335 3.267387

std 9.253717e+09 155.666612 7.637707

min 2.009032e+13 1.000000 0.000000

25% 2.012123e+13 NaN 0.000000

50% 2.013091e+13 NaN 0.000000

75% 2.014031e+13 NaN 0.000000

max 2.014063e+13 2014.000000 25.000000

Code to plot number of users by gender from training dataset:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from collections import Counter

train = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\train\_users\_2.csv")

print(train.columns)

"""A Counter is a dict subclass for counting hashable objects. It is an unordered collection where elements are stored as dictionary keys and their counts are stored as dictionary values"""

c1 = Counter(train.gender).keys()

c2 = Counter(train.gender).values()

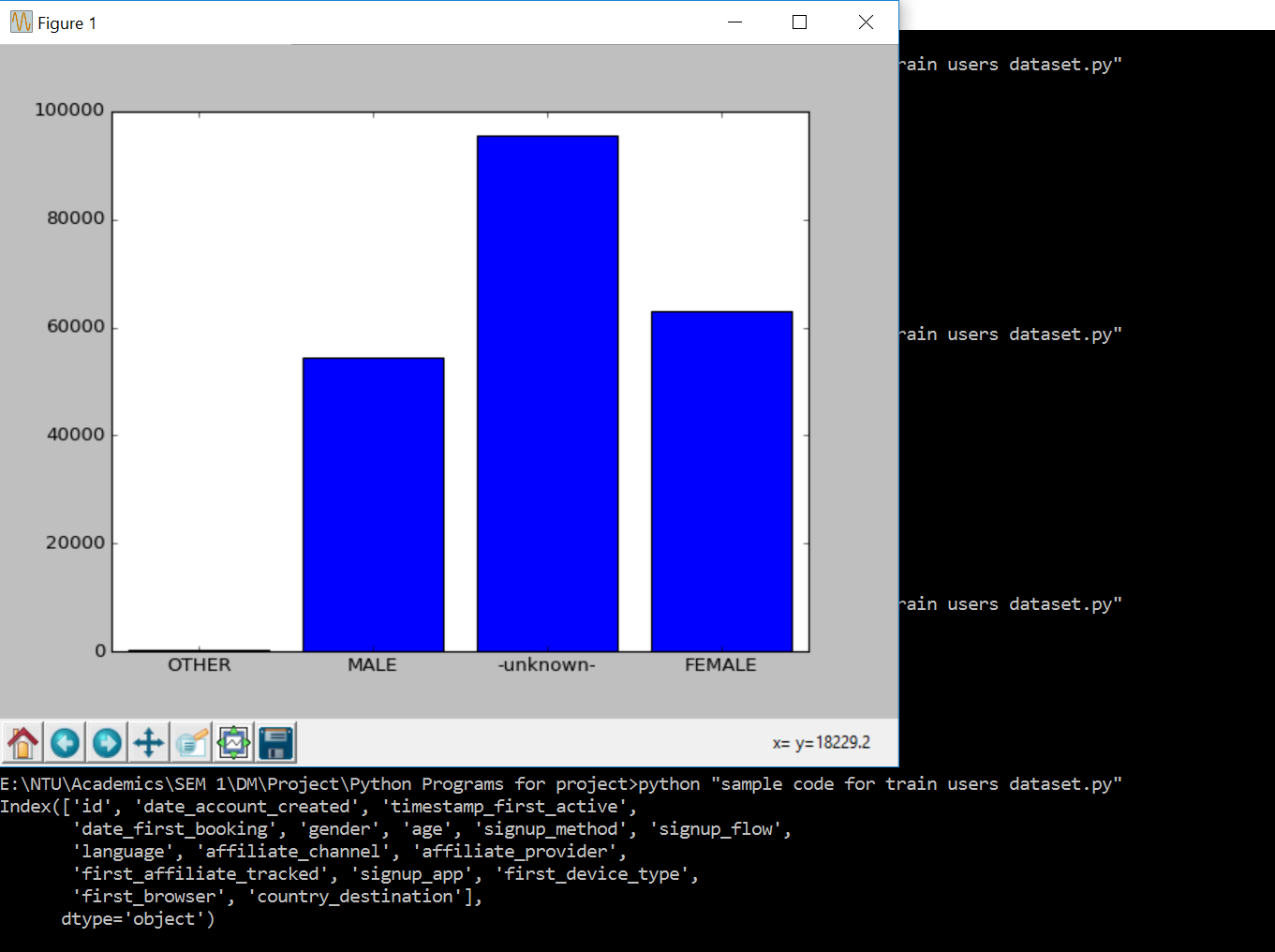
plt.bar(range(len(c1)), c2, align = 'center')

plt.xticks(range(len(c1)), c1)

plt.figure(figsize=(30, 30))

plt.show()

O/P:



Code to plot number of users by destination from training dataset:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from collections import Counter

train = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\train\_users\_2.csv")

print(train.columns)

c1 = Counter(train.country\_destination).keys()

c2 = Counter(train.country\_destination).values()

plt.bar(range(len(c1)), c2, align = 'center')

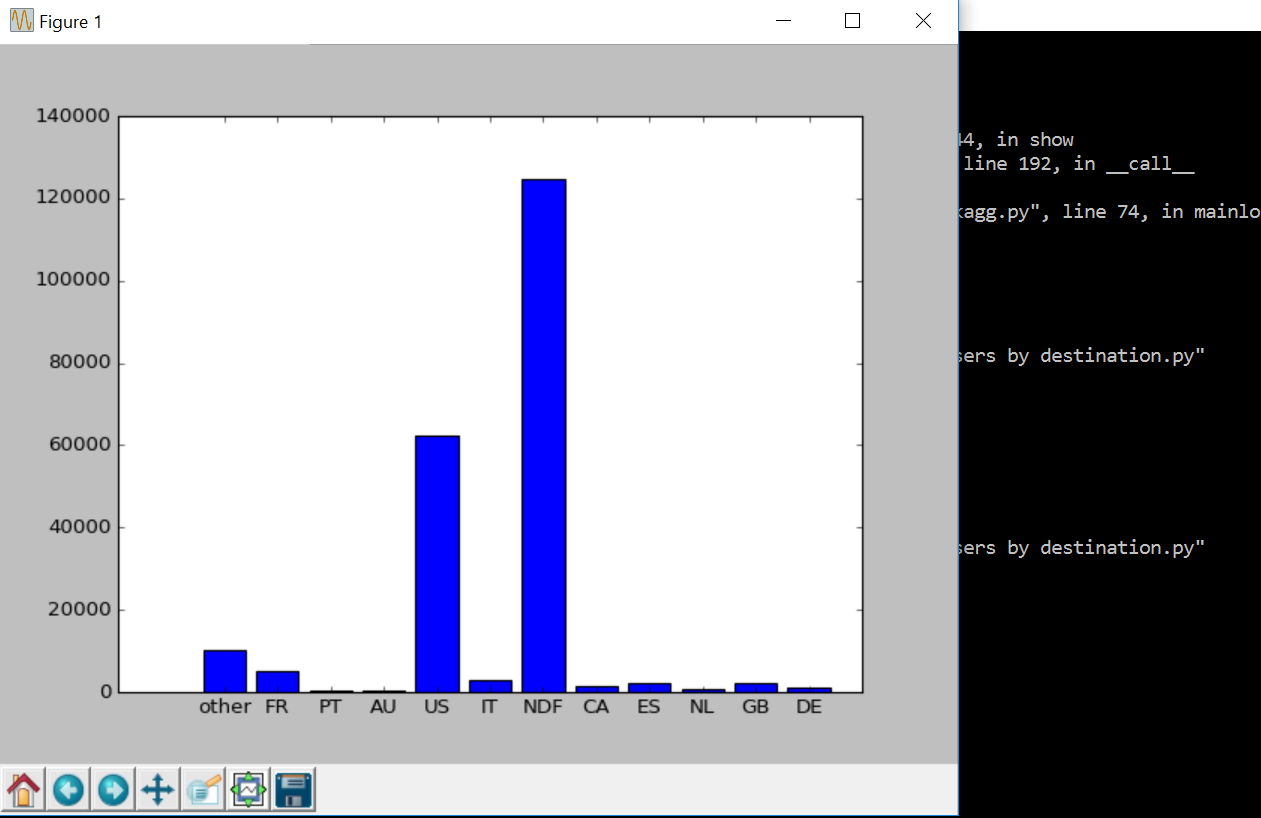
plt.xticks(range(len(c1)), c1)

#plt.yticks(range(len(c1)), c1)

plt.figure(figsize=(30, 30))

plt.show()

O/P:



Code to find out the missing values:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from collections import Counter

#To read only a particular row in a csv file

train = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\train\_users\_2.csv")

test = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\test\_users.csv")

sessions = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\sessions.csv")

print ("\*\*\*\*\*\*\*\*\*\*TRAIN USERS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print (train.apply(lambda x: sum(x.isnull()),axis=0))

print ("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TEST USERS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print (test.apply(lambda x: sum(x.isnull()),axis=0))

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SESSIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print (sessions.apply(lambda x: sum(x.isnull()),axis=0))

O/P:

\*\*\*\*\*\*\*\*\*\*TRAIN USERS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id 0

date\_account\_created 0

timestamp\_first\_active 0

date\_first\_booking 124543

gender 0

age 87990

signup\_method 0

signup\_flow 0

language 0

affiliate\_channel 0

affiliate\_provider 0

first\_affiliate\_tracked 6065

signup\_app 0

first\_device\_type 0

first\_browser 0

country\_destination 0

dtype: int64

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TEST USERS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id 0

date\_account\_created 0

timestamp\_first\_active 0

date\_first\_booking 62096

gender 0

age 28876

signup\_method 0

signup\_flow 0

language 0

affiliate\_channel 0

affiliate\_provider 0

first\_affiliate\_tracked 20

signup\_app 0

first\_device\_type 0

first\_browser 0

dtype: int64

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SESSIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

user\_id 34496

action 79626

action\_type 1126204

action\_detail 1126204

device\_type 0

secs\_elapsed 136031

dtype: int64

Code to find out account created over time for training data(year):

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from collections import Counter

train = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\train\_users\_2.csv")

train["date\_account\_created"] = pd.to\_datetime(train["date\_account\_created"])

#print(train["date\_account\_created"])

train["date\_account\_created"] = (train["date\_account\_created"].dt.year)

c1 = Counter(train.date\_account\_created).keys()

c2 = Counter(train.date\_account\_created).values()

plt.xlabel("Account created year")

plt.ylabel("Number of users")

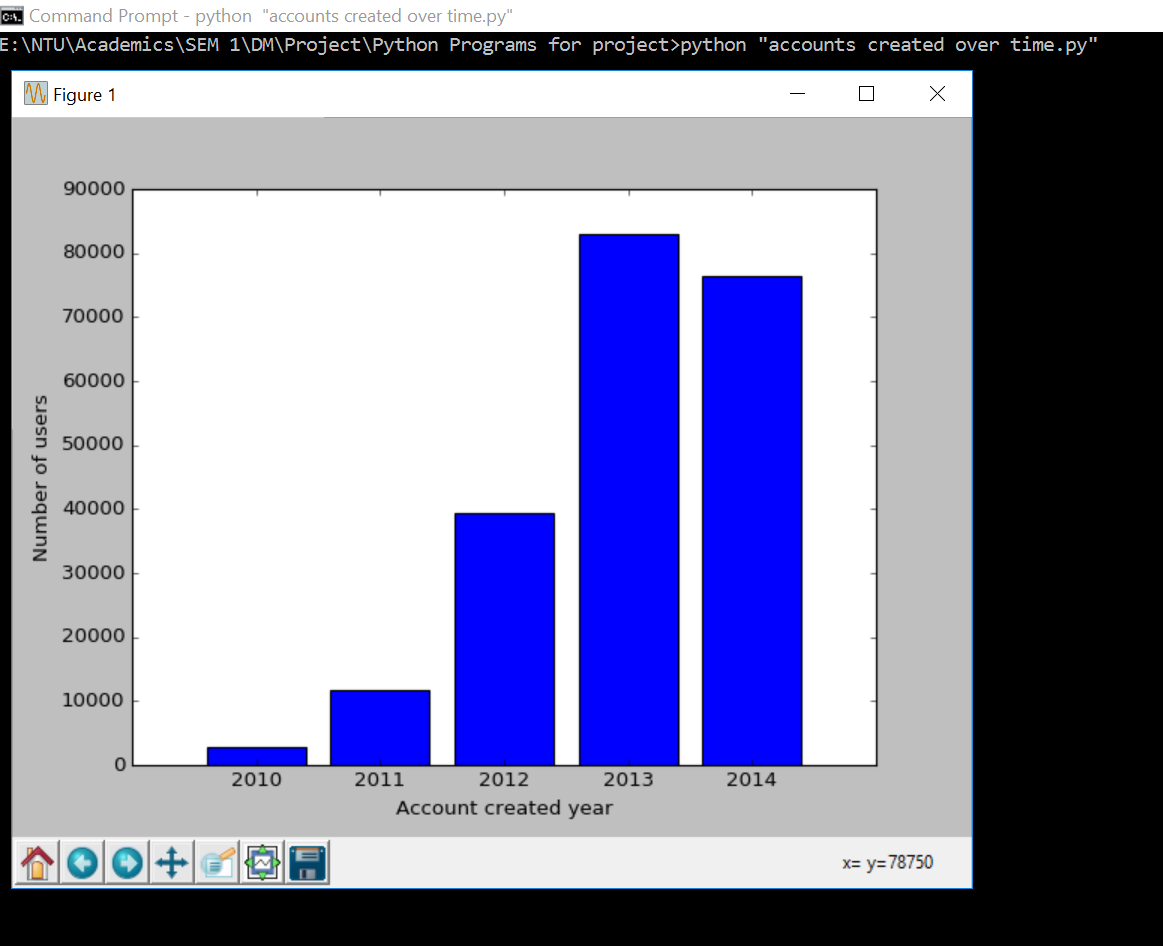
plt.bar(range(len(c1)), c2, align = 'center')

plt.xticks(range(len(c1)), c1)

plt.figure(figsize=(30, 30))

plt.show()

O/P:



CLEANING DATA:

Code for cleaning datasets – train and test: (Filling in missing values/Fixing formats)

import pandas as pd

import numpy as np

import csv

# Read Data

train = "E:\\NTU\\Academics\\SEM 1\\DM\Project\\train\_users\_2.csv"

df\_train = pd.read\_csv(train, header=0, index\_col=None)

test = "E:\\NTU\\Academics\\SEM 1\\DM\Project\\test\_users.csv"

df\_test = pd.read\_csv(test, header=0, index\_col=None)

# Combine into one dataset

#df\_both = pd.concat((df\_train, df\_test), axis=0, ignore\_index=True)

# Change Dates to consistent format

print("Fixing timestamps...")

df\_train['date\_account\_created'] = pd.to\_datetime(df\_train['date\_account\_created'], format='%Y-%m-%d')

df\_test['date\_account\_created'] = pd.to\_datetime(df\_test['date\_account\_created'], format='%Y-%m-%d')

df\_train['timestamp\_first\_active'] = pd.to\_datetime(df\_train['timestamp\_first\_active'], format='%Y%m%d%H%M%S')

df\_test['timestamp\_first\_active'] = pd.to\_datetime(df\_test['timestamp\_first\_active'], format='%Y%m%d%H%M%S')

df\_train['date\_account\_created'].fillna(df\_train.timestamp\_first\_active, inplace=True)

df\_test['date\_account\_created'].fillna(df\_test.timestamp\_first\_active, inplace=True)

# Remove date\_first\_booking column

df\_train.drop('date\_first\_booking', axis=1, inplace=True)

df\_test.drop('date\_first\_booking', axis=1, inplace=True)

# Remove outliers function

def remove\_outliers(df, column, min\_val, max\_val):

col\_values = df[column].values

df[column] = np.where(np.logical\_or(col\_values<=min\_val, col\_values>=max\_val), np.NaN, col\_values)

return df

# Fixing age column(Age is fixed by taking min value as 15 and max value as 90 - outliers are taken as -1)

print("Fixing age column...")

df\_train = remove\_outliers(df=df\_train, column='age', min\_val=15, max\_val=90)

df\_train['age'].fillna(-1, inplace=True)

df\_test = remove\_outliers(df=df\_test, column='age', min\_val=15, max\_val=90)

df\_test['age'].fillna(-1, inplace=True)

# Fill first\_affiliate\_tracked column

print("Filling first\_affiliate\_tracked column...")

df\_train['first\_affiliate\_tracked'].fillna(-1, inplace=True)

df\_test['first\_affiliate\_tracked'].fillna(-1, inplace=True)

#print(df\_train.head())

#print(df\_test.head())

print (df\_train.apply(lambda x: sum(x.isnull()),axis=0))

print (df\_test.apply(lambda x: sum(x.isnull()),axis=0))

#output to a csv file

df\_train.to\_csv('clean\_train\_data.csv', sep=',');

df\_test.to\_csv('clean\_test\_data.csv', sep=',');

FEATURE EXTRACTION:

Code for one hot encoding (convert all categorical values to numerical):

1st option : Using Built-in function in pandas:

import pandas as pd

train = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\Python Programs for project\\clean\_train\_data.csv")

selected\_column = ['gender', 'signup\_method', 'signup\_flow', 'language', 'affiliate\_channel', 'affiliate\_provider', 'first\_affiliate\_tracked', 'signup\_app', 'first\_device\_type', 'first\_browser']

for column in selected\_column:

one\_hot = pd.get\_dummies(train[column])

train = train.join(one\_hot, rsuffix="\_r")

train.drop(column, axis=1, inplace=True)

print(train.head())

#output to a csv file

train.to\_csv('train\_data\_after\_one\_hot\_encoding.csv', sep=',');

2nd option: One hot encoder code:

#convert categorical data into numerical form

import pandas as pd

import numpy as np

# One Hot Encoding

def one\_hot\_encoding(df, selected\_column):

#find out all possible values of each column

values = list(df[selected\_column].drop\_duplicates())

for value in values:

cat\_name = str(value).replace(" ", "\_").replace("/", "\_").replace("-","").lower()

col\_name = selected\_column + '\_' + cat\_name

df[col\_name] = 0

df.loc[(df[selected\_column] == value), col\_name] = 1

return df

train = pd.read\_csv("E:\\NTU\\Academics\\SEM 1\\DM\Project\\Python Programs for project\\clean\_train\_data.csv")

selected\_column = ['gender', 'signup\_method', 'signup\_flow', 'language', 'affiliate\_channel', 'affiliate\_provider', 'first\_affiliate\_tracked',

'signup\_app', 'first\_device\_type', 'first\_browser']

for column in selected\_column:

train = one\_hot\_encoding(df=train, selected\_column=column)

train.drop(column, axis=1)

#print(train.head())

#output to a csv file

train.to\_csv('train\_data\_after\_one\_hot\_encoding.csv', sep=',');