import pandas as pd

import numpy as np

import time

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import f\_oneway

* Import a 311 NYC service request.

hr1=pd.read\_csv("311\_Service\_Requests\_from\_2010\_to\_Present.csv")

hr1.head()

hr1.describe()

hr1.info()

hr1.dtypes

* \*first we see what % of missing data in dataset
* and ploting a values in bar chart

hr1.isnull().sum()/len(hr1)\*100

* to visualize number of null values in dataset by ploting bar chart we can see which column has what num of null values

hr1.isnull().sum().plot(kind='bar', figsize=(10,5),title = 'missing values')

* as visible in bar graph many columns has max missing values that contant null
* second task is to remove not columns having maximum null values.

hr1.keys()

un\_useble= ['Agency Name','Incident Address','Street Name','Cross Street 1','Cross Street 2','Intersection Street 1',

'Intersection Street 2','Address Type','Park Facility Name','Park Borough','School Name',

'School Number','School Region','School Code','School Phone Number','School Address','School City',

'School State','School Zip','School Not Found','School or Citywide Complaint','Vehicle Type',

'Taxi Company Borough','Taxi Pick Up Location','Bridge Highway Name','Bridge Highway Direction',

'Road Ramp','Bridge Highway Segment','Garage Lot Name','Ferry Direction','Ferry Terminal Name','Landmark',

'X Coordinate (State Plane)','Y Coordinate (State Plane)','Due Date','Resolution Action Updated Date','Community Board','Facility Type']

* + serching a values in status column and visualize what num of tipe values related to which cattegiry by bar chart

hr1['Status'].value\_counts().plot(kind='bar',alpha=0.6,figsize=(6,10))

plt.show()

hr1.drop(un\_useble, inplace=True, axis=1)

hr1=hr1[(hr1['Latitude'].notnull())&(hr1['Longitude'].notnull()) & (hr1['Descriptor'].notnull())]

hr1 = hr1[hr1['Status']=='Closed']

hr1.drop(['Status'],inplace=True, axis=1)

hr1.info()

* + second task changing data type from object to date and time by using date and time module

hr1["Created Date"]=pd.to\_datetime(hr1['Created Date'])

hr1["Closed Date"]=pd.to\_datetime(hr1['Closed Date'])

hr1.info()

* + add new calumn "Request\_closing\_time" for colepsed time between created date anf closed date

hr1['Request\_closing\_time']=hr1["Closed Date"]-hr1["Created Date"]

hr1['Request\_closing\_time']

hr1.info()

hr1.columns

* + then again see the % of null values remain in data set

hr1.isnull().sum()/len(hr1)\*100

* + complain distribution across borough
  + visualizing in pie chart

hr1['Borough'].value\_counts()

colors = ['#639ace','#ca6b39','#7f67ca','#5ba85f','#c360aa','#a7993f','#cc566a']

hr1['Borough'].value\_counts().plot(kind='pie',autopct='%1.1f%%',

explode = (0.15, 0, 0, 0,0), startangle=45, shadow=False, colors = colors,

figsize = (8,6))

#plt.legend(title='BOROUGH', loc='upper right', bbox\_to\_anchor=(1.5,1))

plt.axis('equal')

plt.title('# complaints distribution across Boroughs (2015)\n')

plt.tight\_layout()

plt.show()

hr1['Request\_closing\_time'].sort\_values()

* + to calculating avarage time we have to cal aprox time for each values of time
  + and view values of request closing time in hour (aprox)

hr1['Request\_Closing\_Hours'] = hr1['Request\_closing\_time'].astype('timedelta64[h]')+1

hr1[['Request\_closing\_time','Request\_Closing\_Hours']].head()

* + grouping complaint type and borough based on Request Closing Hour

grouped\_data=hr1.groupby(['Complaint Type','Borough'])[['Request\_Closing\_Hours']].mean().unstack()

grouped\_data.head()

#visualizing top 5 complaints in each borough using subplots

col\_number = 2

row\_number = 3

fig, axes = plt.subplots(row\_number,col\_number, figsize=(12,8))

for i, (label,col) in enumerate(grouped\_data.iteritems()):

ax = axes[int(i/col\_number), i%col\_number]

col = col.sort\_values(ascending=True)[:15]

col.plot(kind='barh', ax=ax)

ax.set\_title(label)

plt.tight\_layout()

(hr1['Complaint Type'].value\_counts()).head(25).plot(kind='bar',

figsize=(10,6),title = 'Most Frequent Complaints in Brooklyn')

* + #doing ANOVA test to check whether the average response time across complaint types is similar or not
  + h0= average response time across complaint types is similar if p>0.05
  + ha=average response time across complaint types is not similar so for proove that we have cheake p value p<0.05

data = {}

for complaint in hr1['Complaint Type'].unique():

data[complaint]=np.log(hr1[hr1['Complaint Type']==complaint]['Request\_Closing\_Hours'])

data[complaint].head()

data.keys()

# import f\_oneway from scipy.stats library

stat, p = f\_oneway(data['Noise - Street/Sidewalk'],data['Blocked Driveway'],data['Illegal Parking'],data['Derelict Vehicle'],

data['Noise - Commercial'])

print('Statistics=%.3f, p=%.3f' % (stat, p))

if p > 0.05:

print('the average response time across complaint types is similar hence "fail to reject H0"')

else:

print('the average response time across complaint types is not similar hence "reject H0"')

* + checking correlation between location and complaint types
  + to performe corelation test we have all the value in numerical formate so first task is to change cattegorical values to numerical by using getdummies ()
  + then perform a test between location and comp. type

corr\_test\_data = hr1[['Complaint Type','Borough','Longitude','Latitude','City']]

corr\_test\_data['Complaint Type']=pd.get\_dummies(corr\_test\_data['Complaint Type'])

corr\_test\_data['Borough']=pd.get\_dummies(corr\_test\_data['Borough'])

corr\_test\_data['City']=pd.get\_dummies(corr\_test\_data['City'])

corr\_test\_data.corr()

* + - view the correlation by using heatmap using seaborn lib

import seaborn as sns

ax = sns.heatmap(corr\_test\_data.corr())