Risk Flag Detection

importing libraries

In [ ]:
```
import pandas as pd
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderQueryError
from matplotlib import pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import math
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn import metrics
```

In [ ]:
```
# read and save the dataset
# df=pd.read_json('./loan_approval_dataset.json')
df=pd.read_csv('./data.csv')
```

In [ ]:
```
# set the column names
df.columns=['Id','Income', 'Age', 'Experience', 'Married/Single', 'House_Ownership',
       'Car_Ownership', 'Profession', 'CITY', 'STATE', 'CURRENT_JOB_YRS',
       'CURRENT_HOUSE_YRS', 'Risk_Flag']
```

In [ ]:
```
df.head()
```

Out[ ]:

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CL
0	1	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	
1	2	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	
2	3	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	
3	4	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu	
4	5	6915937	64	0	single	rented	no	Civil_servant	Jalgaon	Maharashtra	

In [ ]:
```
# drop id column
df.drop(labels=['Id'],axis=1,inplace=True)
```

EDA

In [ ]:
```
df.dtypes
```

Out[ ]:
```
Income               int64
Age                  int64
Experience           int64
Married/Single      object
House_Ownership     object
Car_Ownership       object
Profession          object
CITY                object
STATE               object
CURRENT_JOB_YRS      int64
CURRENT_HOUSE_YRS    int64
Risk_Flag            int64
dtype: object
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(251999, 12)
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Income              0
Age                 0
Experience          0
Married/Single      0
House_Ownership     0
Car_Ownership       0
Profession          0
CITY                0
STATE               0
CURRENT_JOB_YRS     0
CURRENT_HOUSE_YRS   0
Risk_Flag           0
dtype: int64
```

no null values are there in the data

In [ ]:
```
df['Married/Single'].unique()
```

Out[ ]:
```
array(['single', 'married'], dtype=object)
```

In [ ]:
```
df.House_Ownership.unique()
```

Out[ ]:
```
array(['rented', 'norent_noown', 'owned'], dtype=object)
```

In [ ]:
```
df.Car_Ownership.unique()
```

Out[ ]:
```
array(['no', 'yes'], dtype=object)
```

In [ ]:
```
df.Profession.unique(),len(df.Profession.unique())
```

Out[ ]:
```
(array(['Software_Developer', 'Technical_writer', 'Civil_servant',
        'Librarian', 'Economist', 'Flight_attendant', 'Architect',
        'Designer', 'Physician', 'Financial_Analyst',
        'Air_traffic_controller', 'Politician', 'Police_officer', 'Artist',
        'Surveyor', 'Design_Engineer', 'Chemical_engineer',
        'Hotel_Manager', 'Mechanical_engineer', 'Dentist', 'Comedian',
        'Biomedical_Engineer', 'Graphic_Designer',
        'Computer_hardware_engineer', 'Petroleum_Engineer', 'Secretary',
        'Computer_operator', 'Chartered_Accountant', 'Technician',
        'Microbiologist', 'Fashion_Designer', 'Aviator', 'Psychologist',
        'Magistrate', 'Lawyer', 'Firefighter', 'Engineer', 'Official',
        'Analyst', 'Geologist', 'Drafter', 'Statistician', 'Web_designer',
        'Consultant', 'Chef', 'Army_officer', 'Surgeon', 'Scientist',
        'Civil_engineer', 'Industrial_Engineer', 'Technology_specialist'],
       dtype=object),
 51)
```

In [ ]:
```
df.describe()
```

Out[ ]:

	Income	Age	Experience	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
count	2.519990e+05	251999.000000	251999.000000	251999.000000	251999.000000	251999.000000
mean	4.997131e+06	49.954178	10.084465	6.333890	11.997790	0.123000
std	2.878307e+06	17.063804	6.002585	3.647054	1.399038	0.328438
min	1.031000e+04	21.000000	0.000000	0.000000	10.000000	0.000000
25%	2.503144e+06	35.000000	5.000000	3.000000	11.000000	0.000000
50%	5.000757e+06	50.000000	10.000000	6.000000	12.000000	0.000000
75%	7.477502e+06	65.000000	15.000000	9.000000	13.000000	0.000000
max	9.999938e+06	79.000000	20.000000	14.000000	14.000000	1.000000

Encoding the categorical features

In [ ]:
```
# 1 for married 0 for unmarried
df['Married/Single']=df['Married/Single'].apply(lambda x:1 if x=="married" else 0)
```

In [ ]:
```
# ['rented=0', 'norent_noown=-1', 'owned=1']
def h_owner(x):
    if x=='owned':
        return 1
    elif x=='rented':
        return 0
    else:
        return -1
df['House_Ownership']=df['House_Ownership'].apply(h_owner)
```

In [ ]:
```
# ['yes=1', 'no=0']
df['Car_Ownership']=df['Car_Ownership'].apply(lambda x:1 if x=='yes' else 0)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag	
0	7574516	40	10	0	0	0	9	13	0	19.
1	3991815	66	4	1	0	0	4	10	0	9.
2	6256451	41	2	0	0	1	2	12	1	20.
3	5768871	47	11	0	0	0	3	14	1	10.
4	6915937	64	0	0	0	0	0	12	0	21.

5 rows × 62 columns

since dataset is very large we can not directly convert it into logitude and latitute

so we will collect all the unique addresses and convert them then we will replace in the dataframe

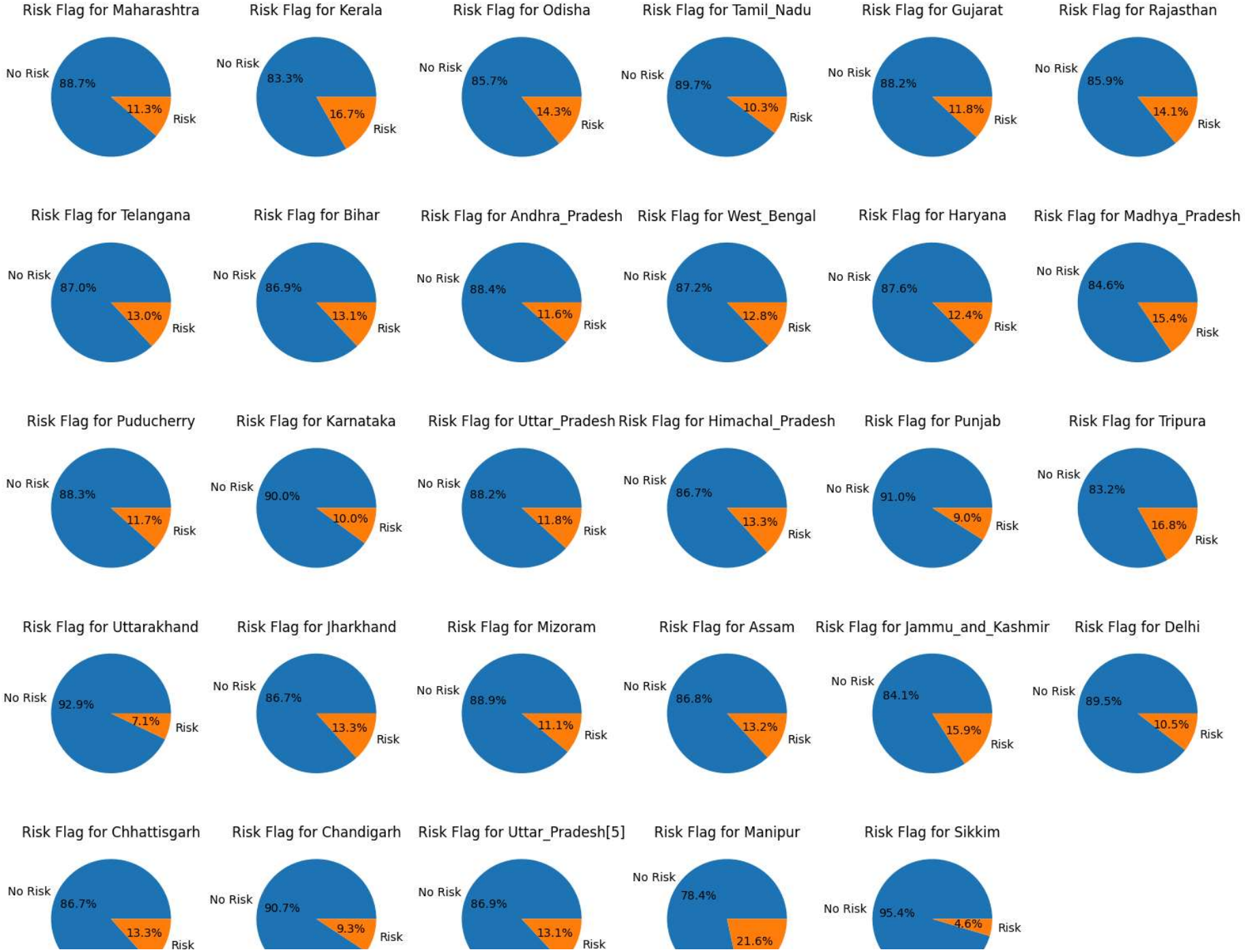
In [ ]:
```
df['STATE'].unique(),len(df['STATE'].unique())
```

Out[ ]:
```
(array(['Maharashtra', 'Kerala', 'Odisha', 'Tamil_Nadu', 'Gujarat',
        'Rajasthan', 'Telangana', 'Bihar', 'Andhra_Pradesh', 'West_Bengal',
        'Haryana', 'Madhya_Pradesh', 'Puducherry', 'Karnataka',
        'Uttar_Pradesh', 'Himachal_Pradesh', 'Punjab', 'Tripura',
        'Uttarakhand', 'Jharkhand', 'Mizoram', 'Assam',
        'Jammu_and_Kashmir', 'Delhi', 'Chhattisgarh', 'Chandigarh',
        'Uttar_Pradesh[5]', 'Manipur', 'Sikkim'], dtype=object),
 29)
```

29 states are there in the dataset

let us check which state has most risk flags

In [ ]:
```python
def risk_flag_state(df):
    unique_states = df['STATE'].unique()
    n = len(unique_states)
    # Calculate grid size: sqrt(n) x sqrt(n) or closest approximation
    grid_size = math.ceil(np.sqrt(n))

    fig, axs = plt.subplots(grid_size, grid_size, figsize=(15, 15))
    axs = axs.flatten()  # Flatten the array for easy iteration

    for i, state in enumerate(unique_states):
        df1 = df[df['STATE'] == state]
        risk_counts = df1['Risk_Flag'].value_counts(normalize=True)
        axs[i].pie(risk_counts, labels=['No Risk', 'Risk'], autopct='%1.1f%%')
        axs[i].set_title(f"Risk Flag for {state}")

    # Hide unused subplots if any
    for j in range(i + 1, len(axs)):
        axs[j].axis('off')

    plt.tight_layout()
    plt.show()

risk_flag_state(df)
```


Risk Flag for Maharashtra
No Risk
88.7%
11.3%
Risk
Risk Flag for Kerala
No Risk
83.3%
16.7%
Risk
Risk Flag for Odisha
No Risk
85.7%
14.3%
Risk
Risk Flag for Tamil_Nadu
No Risk
89.7%
10.3%
Risk
Risk Flag for Gujarat
No Risk
88.2%
11.8%
Risk
Risk Flag for Rajasthan
No Risk
85.9%
14.1%
Risk
Risk Flag for Telangana
No Risk
87.0%
13.0%
Risk
Risk Flag for Bihar
No Risk
86.9%
13.1%
Risk
Risk Flag for Andhra_Pradesh
No Risk
88.4%
11.6%
Risk
Risk Flag for West_Bengal
No Risk
87.2%
12.8%
Risk
Risk Flag for Haryana
No Risk
87.6%
12.4%
Risk
Risk Flag for Madhya_Pradesh
No Risk
84.6%
15.4%
Risk
Risk Flag for Puducherry
No Risk
88.3%
11.7%
Risk
Risk Flag for Karnataka
No Risk
90.0%
10.0%
Risk
Risk Flag for Uttar_Pradesh
No Risk
88.2%
11.8%
Risk
Risk Flag for Himachal_Pradesh
No Risk
86.7%
13.3%
Risk
Risk Flag for Punjab
No Risk
91.0%
9.0%
Risk
Risk Flag for Tripura
No Risk
83.2%
16.8%
Risk
Risk Flag for Uttarakhand
No Risk
92.9%
7.1%
Risk
Risk Flag for Jharkhand
No Risk
86.7%
13.3%
Risk
Risk Flag for Mizoram
No Risk
88.9%
11.1%
Risk
Risk Flag for Assam
No Risk
86.8%
13.2%
Risk
Risk Flag for Jammu_and_Kashmir
No Risk
84.1%
15.9%
Risk
Risk Flag for Delhi
No Risk
89.5%
10.5%
Risk
Risk Flag for Chhattisgarh
No Risk
86.7%
13.3%
Risk
Risk Flag for Chandigarh
No Risk
90.7%
9.3%
Risk
Risk Flag for Uttar_Pradesh[5]
No Risk
86.9%
13.1%
Risk
Risk Flag for Manipur
No Risk
78.4%
21.6%
Risk Flag for Sikkim
No Risk
95.4%
4.6%
Risk

Manipur is most risky state with risk 21.6%.

Sikkim is least risky state with risk 4.6%.

In [ ]:
```
# clean the city column and state column if needed
def clean_text(x:str):
    x=x.replace('_'," ")
    txt=''
    for ch in x:
        if ch.isalpha() or ch.isspace():
            txt+=ch
    return txt
```

In [ ]:
```
df['CITY']=df['CITY'].apply(clean_text)
```

In [ ]:
```
df['STATE']=df['STATE'].apply(clean_text)
```

In [ ]:
```
df['address']=df['CITY']+","+df['STATE']+",india"
```

In [ ]:
```
df.head()
```

Out[ ]:

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURR
0	1	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	
1	2	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	
2	3	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	
3	4	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli	Tamil Nadu	
4	5	6915937	64	0	single	rented	no	Civil_servant	Jalgaon	Maharashtra	

In [ ]:
```
address=df['address'].unique()
```

In [ ]:
```
print(address)
```

```
['Parbhani,Maharashtra,india' 'Alappuzha,Kerala,india'
 'Bhubaneswar,Odisha,india' 'Tiruchirappalli,Tamil Nadu,india'
 'Jalgaon,Maharashtra,india' 'Tiruppur,Tamil Nadu,india'
 'Jamnagar,Gujarat,india' 'Kota,Rajasthan,india'
 'Karimnagar,Telangana,india' 'Hajipur,Bihar,india'
 'Adoni,Andhra Pradesh,india' 'Erode,Tamil Nadu,india'
 'Kollam,Kerala,india' 'Madurai,Tamil Nadu,india'
 'Anantapuram,Andhra Pradesh,india' 'Kamarhati,West Bengal,india'
 'Bhusawal,Maharashtra,india' 'Sirsa,Haryana,india'
 'Amaravati,Andhra Pradesh,india' 'Secunderabad,Telangana,india'
 'Ahmedabad,Gujarat,india' 'Ajmer,Rajasthan,india'
 'Ongole,Andhra Pradesh,india' 'Miryalaguda,Telangana,india'
 'Ambattur,Tamil Nadu,india' 'Indore,Madhya Pradesh,india'
 'Pondicherry,Puducherry,india' 'Shimoga,Karnataka,india'
 'Chennai,Tamil Nadu,india' 'Gulbarga,Karnataka,india'
 'Khammam,Telangana,india' 'Saharanpur,Uttar Pradesh,india'
 'Gopalpur,West Bengal,india' 'Amravati,Maharashtra,india'
 'Udupi,Karnataka,india' 'Howrah,West Bengal,india'
 'Aurangabad,Bihar,india' 'Hospet,Karnataka,india'
 'Shimla,Himachal Pradesh,india' 'Khandwa,Madhya Pradesh,india'
 'Bidhannagar,West Bengal,india' 'Bellary,Karnataka,india'
 'Danapur,Bihar,india' 'Purnia,Bihar,india' 'Bijapur,Karnataka,india'
 'Patiala,Punjab,india' 'Malda,West Bengal,india'
 'Sagar,Madhya Pradesh,india' 'Durgapur,West Bengal,india'
 'Junagadh,Gujarat,india' 'Singrauli,Madhya Pradesh,india'
 'Agartala,Tripura,india' 'Thanjavur,Tamil Nadu,india'
 'Hindupur,Andhra Pradesh,india' 'Naihati,West Bengal,india'
 'North Dumdum,West Bengal,india' 'Panchkula,Haryana,india'
 'Anantapur,Andhra Pradesh,india' 'Serampore,West Bengal,india'
 'Bathinda,Punjab,india' 'Nadiad,Gujarat,india'
 'Kanpur,Uttar Pradesh,india' 'Haridwar,Uttarakhand,india'
 'Berhampur,Odisha,india' 'Jamshedpur,Jharkhand,india'
 'Hyderabad,Telangana,india' 'Bidar,Karnataka,india'
 'Kottayam,Kerala,india' 'Solapur,Maharashtra,india'
 'Suryapet,Telangana,india' 'Aizawl,Mizoram,india'
 'Asansol,West Bengal,india' 'Deoghar,Jharkhand,india'
 'Eluru,Andhra Pradesh,india' 'Ulhasnagar,Maharashtra,india'
 'Aligarh,Uttar Pradesh,india' 'South Dumdum,West Bengal,india'
 'Berhampore,West Bengal,india' 'Gandhinagar,Gujarat,india'
 'Sonipat,Haryana,india' 'Muzaffarpur,Bihar,india'
 'Raichur,Karnataka,india' 'Rajpur Sonarpur,West Bengal,india'
```

'Ambarnath,Maharashtra,india' 'Katihar,Bihar,india'
'Kozhikode,Kerala,india' 'Vellore,Tamil Nadu,india'
'Malegaon,Maharashtra,india' 'Kochi,Kerala,india' 'Nagaon,Assam,india'
'Nagpur,Maharashtra,india' 'Srinagar,Jammu and Kashmir,india'
'Davanagere,Karnataka,india' 'Bhagalpur,Bihar,india' 'Siwan,Bihar,india'
'Meerut,Uttar Pradesh,india' 'Dindigul,Tamil Nadu,india'
'Bhatpara,West Bengal,india' 'Ghaziabad,Uttar Pradesh,india'
'Kulti,West Bengal,india' 'Chapra,Bihar,india' 'Dibrugarh,Assam,india'
'Panihati,West Bengal,india' 'Bhiwandi,Maharashtra,india'
'Morbi,Gujarat,india' 'KalyanDombivli,Maharashtra,india'
'Gorakhpur,Uttar Pradesh,india' 'Panvel,Maharashtra,india'
'Siliguri,West Bengal,india' 'Bongaigaon,Assam,india' 'Patna,Bihar,india'
'Ramgarh,Jharkhand,india' 'Ozhukarai,Puducherry,india'
'Mirzapur,Uttar Pradesh,india' 'Akola,Maharashtra,india'
'Satna,Madhya Pradesh,india' 'Motihari,Bihar,india'
'Jalna,Maharashtra,india' 'Jalandhar,Punjab,india'
'Unnao,Uttar Pradesh,india' 'Karnal,Haryana,india' 'Cuttack,Odisha,india'
'Proddatur,Andhra Pradesh,india' 'Ichalkaranji,Maharashtra,india'
'Warangal,Telangana,india' 'Jhansi,Uttar Pradesh,india'
'Bulandshahr,Uttar Pradesh,india' 'Narasaraopet,Andhra Pradesh,india'
'Chinsurah,West Bengal,india' 'Jehanabad,Bihar,india'
'Dhanbad,Jharkhand,india' 'Gudivada,Andhra Pradesh,india'
'Gandhidham,Gujarat,india' 'Raiganj,West Bengal,india'
'Kishanganj,Bihar,india' 'Varanasi,Uttar Pradesh,india'
'Belgaum,Karnataka,india' 'Tirupati,Andhra Pradesh,india'
'Tumkur,Karnataka,india' 'Coimbatore,Tamil Nadu,india'
'Kurnool,Andhra Pradesh,india' 'Gurgaon,Haryana,india'
'Muzaffarnagar,Uttar Pradesh,india' 'Aurangabad,Maharashtra,india'
'Bhavnagar,Gujarat,india' 'Arrah,Bihar,india' 'Munger,Bihar,india'
'Tirunelveli,Tamil Nadu,india' 'Mumbai,Maharashtra,india'
'Mango,Jharkhand,india' 'Nashik,Maharashtra,india'
'Kadapa,Andhra Pradesh,india' 'Amritsar,Punjab,india'
'Khora Ghaziabad,Uttar Pradesh,india' 'Ambala,Haryana,india'
'Agra,Uttar Pradesh,india' 'Ratlam,Madhya Pradesh,india'
'Surendranagar Dudhrej,Gujarat,india' 'Delhi city,Delhi,india'
'Bhopal,Madhya Pradesh,india' 'Hapur,Uttar Pradesh,india'
'Rohtak,Haryana,india' 'Durg,Chhattisgarh,india'
'Korba,Chhattisgarh,india' 'Bangalore,Karnataka,india'
'Shivpuri,Madhya Pradesh,india' 'Thrissur,Kerala,india'
'Vijayanagaram,Andhra Pradesh,india' 'Farrukhabad,Uttar Pradesh,india'
'Nangloi Jat,Delhi,india' 'Madanapalle,Andhra Pradesh,india'

'Thoothukudi,Tamil Nadu,india' 'Nagercoil,Tamil Nadu,india'
'Gaya,Bihar,india' 'Chandigarh city,Chandigarh,india'
'Jammu,Jammu and Kashmir,india' 'Kakinada,Andhra Pradesh,india'
'Dewas,Madhya Pradesh,india' 'Bhalswa Jahangir Pur,Delhi,india'
'Baranagar,West Bengal,india' 'Firozabad,Uttar Pradesh,india'
'Phusro,Jharkhand,india' 'Allahabad,Uttar Pradesh,india'
'Guna,Madhya Pradesh,india' 'Thane,Maharashtra,india'
'Etawah,Uttar Pradesh,india' 'VasaiVirar,Maharashtra,india'
'Pallavaram,Tamil Nadu,india' 'Morena,Madhya Pradesh,india'
'Ballia,Uttar Pradesh,india' 'Surat,Gujarat,india'
'Burhanpur,Madhya Pradesh,india' 'Phagwara,Punjab,india'
'Mau,Uttar Pradesh,india' 'Mangalore,Karnataka,india'
'Alwar,Rajasthan,india' 'Mahbubnagar,Telangana,india'
'Maheshtala,West Bengal,india' 'Hazaribagh,Jharkhand,india'
'Bihar Sharif,Bihar,india' 'Faridabad,Haryana,india'
'Lucknow,Uttar Pradesh,india' 'Tenali,Andhra Pradesh,india'
'Barasat,West Bengal,india' 'Amroha,Uttar Pradesh,india'
'Giridih,Jharkhand,india' 'Begusarai,Bihar,india'
'Medininagar,Jharkhand,india' 'Rajahmundry,Andhra Pradesh,india'
'Saharsa,Bihar,india' 'New Delhi,Delhi,india' 'Bhilai,Chhattisgarh,india'
'Moradabad,Uttar Pradesh,india' 'Machilipatnam,Andhra Pradesh,india'
'MiraBhayandar,Maharashtra,india' 'Pali,Rajasthan,india'
'Navi Mumbai,Maharashtra,india' 'Mehsana,Gujarat,india'
'Imphal,Manipur,india' 'Kolkata,West Bengal,india'
'Sambalpur,Odisha,india' 'Ujjain,Madhya Pradesh,india'
'Madhyamgram,West Bengal,india' 'Jabalpur,Madhya Pradesh,india'
'Jamalpur,Bihar,india' 'Ludhiana,Punjab,india'
'Bareilly,Uttar Pradesh,india' 'Gangtok,Sikkim,india'
'Anand,Gujarat,india' 'Dehradun,Uttarakhand,india'
'Pune,Maharashtra,india' 'Satara,Maharashtra,india'
'Srikakulam,Andhra Pradesh,india' 'Raipur,Chhattisgarh,india'
'Jodhpur,Rajasthan,india' 'Darbhanga,Bihar,india'
'Nizamabad,Telangana,india' 'Nandyal,Andhra Pradesh,india'
'Dehri,Bihar,india' 'Jorhat,Assam,india' 'Ranchi,Jharkhand,india'
'Kumbakonam,Tamil Nadu,india' 'Guntakal,Andhra Pradesh,india'
'Haldia,West Bengal,india' 'Loni,Uttar Pradesh,india'
'PimpriChinchwad,Maharashtra,india' 'Rajkot,Gujarat,india'
'Nanded,Maharashtra,india' 'Noida,Uttar Pradesh,india'
'Kirari Suleman Nagar,Delhi,india' 'Jaunpur,Uttar Pradesh,india'
'Bilaspur,Chhattisgarh,india' 'Sambhal,Uttar Pradesh,india'
'Dhule,Maharashtra,india' 'Rourkela,Odisha,india'

```
'Thiruvananthapuram,Kerala,india' 'Dharmavaram,Andhra Pradesh,india'
'Nellore,Andhra Pradesh,india' 'Visakhapatnam,Andhra Pradesh,india'
'Karawal Nagar,Delhi,india' 'Jaipur,Rajasthan,india'
'Avadi,Tamil Nadu,india' 'Bhimavaram,Andhra Pradesh,india'
'Bardhaman,West Bengal,india' 'Silchar,Assam,india' 'Buxar,Bihar,india'
'Kavali,Andhra Pradesh,india' 'Tezpur,Assam,india'
'Ramagundam,Telangana,india' 'Yamunanagar,Haryana,india'
'Sri Ganganagar,Rajasthan,india' 'Sasaram,Bihar,india'
'Sikar,Rajasthan,india' 'Bally,West Bengal,india' 'Bhiwani,Haryana,india'
'Rampur,Uttar Pradesh,india' 'Uluberia,West Bengal,india'
'SangliMiraj  Kupwad,Maharashtra,india' 'Hosur,Tamil Nadu,india'
'Bikaner,Rajasthan,india' 'Shahjahanpur,Uttar Pradesh,india'
'Sultan Pur Majra,Delhi,india' 'Vijayawada,Andhra Pradesh,india'
'Bharatpur,Rajasthan,india' 'Tadepalligudem,Andhra Pradesh,india'
'Tinsukia,Assam,india' 'Salem,Tamil Nadu,india'
'Mathura,Uttar Pradesh,india' 'Guntur,Andhra Pradesh,india'
'HubliâDharwad,Karnataka,india' 'Guwahati,Assam,india'
'Chittoor,Andhra Pradesh,india' 'Tiruvottiyur,Tamil Nadu,india'
'Vadodara,Gujarat,india' 'Ahmednagar,Maharashtra,india'
'Fatehpur,Uttar Pradesh,india' 'Bhilwara,Rajasthan,india'
'Kharagpur,West Bengal,india' 'Bettiah,Bihar,india'
'Bhind,Madhya Pradesh,india' 'Bokaro,Jharkhand,india'
'Karaikudi,Tamil Nadu,india' 'Raebareli,Uttar Pradesh,india'
'Pudukkottai,Tamil Nadu,india' 'Udaipur,Rajasthan,india'
'Mysore,Karnataka,india' 'Panipat,Haryana,india'
'Latur,Maharashtra,india' 'Tadipatri,Andhra Pradesh,india'
'Bahraich,Uttar Pradesh,india' 'Orai,Uttar Pradesh,india'
'Raurkela Industrial Township,Odisha,india' 'Rewa,Madhya Pradesh,india'
'Gwalior,Madhya Pradesh,india' 'Katni,Madhya Pradesh,india'
'Chandrapur,Maharashtra,india' 'Kolhapur,Maharashtra,india']
```

317 toal unique indian addreesses are there

In [ ]:
```
# define geolocator
geolocator = Nominatim(user_agent="geoapi-Exercises")
```

In [ ]:
```
loc=geolocator.geocode('Bahraich,Uttar Pradesh,india')
print(loc.latitude,loc.longitude)
```

```
27.75 81.75
```

In [ ]:

```python
def get_lat_lon_geopy(address):
    try:
        location = geolocator.geocode(address)
        if location:
            return location.latitude, location.longitude
        else:
            print(f"Address not found: {address}")
            return None
    except GeocoderQueryError as e:
        print(f"GeocoderQueryError for address {address}: {e}")
        return None

long_lat = {}
for add in address:
    coords = get_lat_lon_geopy(add)
    if coords:
        long_lat[add] = coords

print(long_lat)
```

```
Address not found: KalyanDombivli,Maharashtra,india
Address not found: Surendranagar Dudhrej,Gujarat,india
Address not found: Nangloi Jat,Delhi,india
Address not found: Bhalswa Jahangir Pur,Delhi,india
Address not found: VasaiVirar,Maharashtra,india
Address not found: MiraBhayandar,Maharashtra,india
Address not found: PimpriChinchwad,Maharashtra,india
Address not found: SangliMiraj  Kupwad,Maharashtra,india
Address not found: Sultan Pur Majra,Delhi,india
Address not found: HubliâDharwad,Karnataka,india
Address not found: Raurkela Industrial Township,Odisha,india
{'Parbhani,Maharashtra,india': (19.262469, 76.7718), 'Alappuzha,Kerala,india': (9.4980001, 76.333482), 'Bhubaneswar,Odisha,indi
a': (20.2602964, 85.8394521), 'Tiruchirappalli,Tamil Nadu,india': (10.804973, 78.6870296), 'Jalgaon,Maharashtra,india': (21.013
7606, 75.5627048), 'Tiruppur,Tamil Nadu,india': (11.1017815, 77.345192), 'Jamnagar,Gujarat,india': (22.4732415, 70.0552102), 'K
ota,Rajasthan,india': (25.1737019, 75.8574194), 'Karimnagar,Telangana,india': (18.4348122, 79.1328042), 'Hajipur,Bihar,india':
(25.7215853, 85.2549691837086), 'Adoni,Andhra Pradesh,india': (15.6559189, 77.26977367379274), 'Erode,Tamil Nadu,india': (11.33
06483, 77.7276519), 'Kollam,Kerala,india': (8.8879509, 76.5955013), 'Madurai,Tamil Nadu,india': (9.9261153, 78.1140983), 'Anant
apuram,Andhra Pradesh,india': (13.6830084, 78.1046457), 'Kamarhati,West Bengal,india': (22.6810246, 88.3715343), 'Bhusawal,Maha
rashtra,india': (20.9946183, 75.83962967509835), 'Sirsa,Haryana,india': (29.583333, 75.083333), 'Amaravati,Andhra Pradesh,indi
a': (16.5096679, 80.5184535), 'Secunderabad,Telangana,india': (17.4337246, 78.5006827), 'Ahmedabad,Gujarat,india': (23.0216238,
72.5797068), 'Ajmer,Rajasthan,india': (26.4691, 74.639), 'Ongole,Andhra Pradesh,india': (15.5075545, 80.06080046754784), 'Mirya
laguda,Telangana,india': (16.8710119, 79.5617353), 'Ambattur,Tamil Nadu,india': (13.1193746, 80.1507648), 'Indore,Madhya Prades
h,india': (22.7203616, 75.8681996), 'Pondicherry,Puducherry,india': (11.9340568, 79.8306447), 'Shimoga,Karnataka,india': (13.93
26093, 75.574978), 'Chennai,Tamil Nadu,india': (13.0836939, 80.270186), 'Gulbarga,Karnataka,india': (17.166667, 77.083333), 'Kh
ammam,Telangana,india': (17.2465351, 80.1500326), 'Saharanpur,Uttar Pradesh,india': (30.015538499999998, 77.60388038805745), 'G
opalpur,West Bengal,india': (22.2580984, 88.591836), 'Amravati,Maharashtra,india': (20.981485, 77.799985374555), 'Udupi,Karnata
ka,india': (13.3419169, 74.7473232), 'Howrah,West Bengal,india': (22.5736296, 88.3251045), 'Aurangabad,Bihar,india': (24.78630
6, 84.41448983395554), 'Hospet,Karnataka,india': (15.266493, 76.3872297), 'Shimla,Himachal Pradesh,india': (31.1041526, 77.1709
729), 'Khandwa,Madhya Pradesh,india': (21.8203752, 76.3555424), 'Bidhannagar,West Bengal,india': (22.590425, 88.41692), 'Bellar
y,Karnataka,india': (15.1433952, 76.9193876), 'Danapur,Bihar,india': (25.6358901, 85.0474045), 'Purnia,Bihar,india': (26.0, 87.
5), 'Bijapur,Karnataka,india': (16.8269911, 75.7175387), 'Patiala,Punjab,india': (30.3295605, 76.4127819), 'Malda,West Bengal,i
ndia': (25.0057449, 88.1398483), 'Sagar,Madhya Pradesh,india': (23.75, 78.75), 'Durgapur,West Bengal,india': (23.5350475, 87.33
80425), 'Junagadh,Gujarat,india': (21.5174104, 70.4642754), 'Singrauli,Madhya Pradesh,india': (24.1974432, 82.6661453), 'Agarta
la,Tripura,india': (23.8312377, 91.2823821), 'Thanjavur,Tamil Nadu,india': (10.7860267, 79.1381497), 'Hindupur,Andhra Pradesh,i
ndia': (13.8306127, 77.492635), 'Naihati,West Bengal,india': (22.8886213, 88.4239345), 'North Dumdum,West Bengal,india': (22.66
60837, 88.4211986), 'Panchkula,Haryana,india': (30.6975401, 76.8551066), 'Anantapur,Andhra Pradesh,india': (14.6546235, 77.5562
5984224562), 'Serampore,West Bengal,india': (22.7549951, 88.3416672), 'Bathinda,Punjab,india': (30.206791, 74.9463699), 'Nadia
d,Gujarat,india': (22.6895535, 72.8713603), 'Kanpur,Uttar Pradesh,india': (26.4609135, 80.3217588), 'Haridwar,Uttarakhand,indi
a': (29.9384473, 78.1452985), 'Berhampur,Odisha,india': (21.49308515, 86.63272982626935), 'Jamshedpur,Jharkhand,india': (22.801
5194, 86.2029579), 'Hyderabad,Telangana,india': (17.360589, 78.4740613), 'Bidar,Karnataka,india': (18.083333, 77.333333), 'Kott
ayam,Kerala,india': (9.591564, 76.5221599), 'Solapur,Maharashtra,india': (17.6699734, 75.9008118), 'Suryapet,Telangana,india':
```

(17.1405328, 79.6225105), 'Aizawl,Mizoram,india': (23.7433532, 92.7382756), 'Asansol,West Bengal,india': (23.6871297, 86.974658
7), 'Deoghar,Jharkhand,india': (24.4766423, 86.60673245386945), 'Eluru,Andhra Pradesh,india': (16.67613485, 81.17086824015968),
'Ulhasnagar,Maharashtra,india': (19.2236329, 73.1671515), 'Aligarh,Uttar Pradesh,india': (27.833333, 78.166667), 'South Dumdum,
West Bengal,india': (22.6180871, 88.412251), 'Berhampore,West Bengal,india': (24.0510297, 88.23779874587402), 'Gandhinagar,Guja
rat,india': (23.2232877, 72.6492267), 'Sonipat,Haryana,india': (28.9953758, 77.0233627), 'Muzaffarpur,Bihar,india': (26.166705
2, 85.4166314), 'Raichur,Karnataka,india': (16.083333, 77.166667), 'Rajpur Sonarpur,West Bengal,india': (22.4408748, 88.424636
8), 'Ambarnath,Maharashtra,india': (19.2015607, 73.2004771), 'Katihar,Bihar,india': (25.550475849999998, 87.59414680800401), 'K
ozhikode,Kerala,india': (11.2450558, 75.7754716), 'Vellore,Tamil Nadu,india': (12.7948109, 79.0006410968549), 'Malegaon,Maharas
htra,india': (20.58968735, 74.57175729734101), 'Kochi,Kerala,india': (9.9674277, 76.2454436), 'Nagaon,Assam,india': (26.348223
8, 92.6858215), 'Nagpur,Maharashtra,india': (21.1498134, 79.0820556), 'Srinagar,Jammu and Kashmir,india': (34.0747444, 74.82044
43), 'Davanagere,Karnataka,india': (14.4661266, 75.9206361), 'Bhagalpur,Bihar,india': (25.2494829, 86.9828131), 'Siwan,Bihar,in
dia': (26.263246600000002, 84.32254962309293), 'Meerut,Uttar Pradesh,india': (28.9826533, 77.7081013), 'Dindigul,Tamil Nadu,ind
ia': (10.3303299, 78.0673979084697), 'Bhatpara,West Bengal,india': (22.8571157, 88.3971976), 'Ghaziabad,Uttar Pradesh,india':
(28.6711527, 77.4120356), 'Kulti,West Bengal,india': (23.7302149, 86.8396711), 'Chapra,Bihar,india': (25.773344299999998, 84.78
497716754097), 'Dibrugarh,Assam,india': (27.4844597, 94.9019447), 'Panihati,West Bengal,india': (22.6950342, 88.4073585), 'Bhiw
andi,Maharashtra,india': (19.3025562, 73.0588072), 'Morbi,Gujarat,india': (22.8176662, 70.8345928), 'Gorakhpur,Uttar Pradesh,in
dia': (26.7780452, 83.41521037078049), 'Panvel,Maharashtra,india': (18.991097099999998, 73.12058456241039), 'Siliguri,West Beng
al,india': (26.7164127, 88.4309916), 'Bongaigaon,Assam,india': (26.412376950000002, 90.57149765106257), 'Patna,Bihar,india': (2
5.6093239, 85.1235252), 'Ramgarh,Jharkhand,india': (23.598775949999997, 85.53691564663127), 'Ozhukarai,Puducherry,india': (11.9
414045, 79.8064577), 'Mirzapur,Uttar Pradesh,india': (25.10160235, 82.62575622453129), 'Akola,Maharashtra,india': (20.716905949
999997, 77.10462216588823), 'Satna,Madhya Pradesh,india': (24.5, 81.0), 'Motihari,Bihar,india': (26.66951195, 84.9574106248551
4), 'Jalna,Maharashtra,india': (19.83118795, 76.0613897589651), 'Jalandhar,Punjab,india': (31.3323762, 75.576889), 'Unnao,Uttar
Pradesh,india': (26.478962199999998, 80.5228198291491), 'Karnal,Haryana,india': (29.6803266, 76.9896254), 'Cuttack,Odisha,indi
a': (20.4686, 85.8792), 'Proddatur,Andhra Pradesh,india': (14.752265900000001, 78.54855180269425), 'Ichalkaranji,Maharashtra,in
dia': (16.6959348, 74.4555755), 'Warangal,Telangana,india': (17.9820644, 79.5970954), 'Jhansi,Uttar Pradesh,india': (25.3548645
50000002, 78.42564420838272), 'Bulandshahr,Uttar Pradesh,india': (28.416667, 77.833333), 'Narasaraopet,Andhra Pradesh,india':
(16.2389242, 80.0472879), 'Chinsurah,West Bengal,india': (22.8854687, 88.3837126), 'Jehanabad,Bihar,india': (25.15295785, 85.00
650426458421), 'Dhanbad,Jharkhand,india': (23.7952809, 86.4309638), 'Gudivada,Andhra Pradesh,india': (16.4329976, 80.9937151),
'Gandhidham,Gujarat,india': (23.0718743, 70.131715), 'Raiganj,West Bengal,india': (25.680653900000003, 88.12464576019889), 'Kis
hanganj,Bihar,india': (25.6723524, 86.92184900670425), 'Varanasi,Uttar Pradesh,india': (25.3356491, 83.0076292), 'Belgaum,Karna
taka,india': (15.8572666, 74.5069343), 'Tirupati,Andhra Pradesh,india': (13.77928955, 79.83512262283737), 'Tumkur,Karnataka,ind
ia': (13.3400771, 77.1006208), 'Coimbatore,Tamil Nadu,india': (11.0018115, 76.9628425), 'Kurnool,Andhra Pradesh,india': (15.830
9251, 78.0425373), 'Gurgaon,Haryana,india': (28.4646148, 77.0299194), 'Muzaffarnagar,Uttar Pradesh,india': (29.54212515, 77.645
12491521087), 'Aurangabad,Maharashtra,india': (19.877263, 75.3390241), 'Bhavnagar,Gujarat,india': (21.7718836, 72.1416449), 'Ar
rah,Bihar,india': (25.62345725, 84.59683868653386), 'Munger,Bihar,india': (25.0, 86.25), 'Tirunelveli,Tamil Nadu,india': (8.808
2342, 77.8114843), 'Mumbai,Maharashtra,india': (18.9733536, 72.82810491917377), 'Mango,Jharkhand,india': (24.8752289, 87.734108
69629403), 'Nashik,Maharashtra,india': (20.0112475, 73.7902364), 'Kadapa,Andhra Pradesh,india': (14.4752936, 78.8216861), 'Amri
tsar,Punjab,india': (31.6343083, 74.8736788), 'Khora Ghaziabad,Uttar Pradesh,india': (28.626249, 77.3465824), 'Ambala,Haryana,i
ndia': (30.3843674, 76.770421), 'Agra,Uttar Pradesh,india': (27.1752554, 78.0098161), 'Ratlam,Madhya Pradesh,india': (23.331104
4, 75.0360846), 'Delhi city,Delhi,india': (28.690109, 77.13437649961597), 'Bhopal,Madhya Pradesh,india': (23.2584857, 77.40198
9), 'Hapur,Uttar Pradesh,india': (28.739459150000002, 77.83517038273197), 'Rohtak,Haryana,india': (28.9010899, 76.5801935), 'Du

rg,Chhattisgarh,india': (21.1896499, 81.2851077), 'Korba,Chhattisgarh,india': (22.3547468, 82.7110855), 'Bangalore,Karnataka,in
dia': (12.98815675, 77.62260003796), 'Shivpuri,Madhya Pradesh,india': (25.5, 78.0), 'Thrissur,Kerala,india': (10.5270099, 76.21
4621), 'Vijayanagaram,Andhra Pradesh,india': (18.1139259, 83.3979562), 'Farrukhabad,Uttar Pradesh,india': (27.307761499999998,
79.4535755580881), 'Madanapalle,Andhra Pradesh,india': (13.57325965, 78.47914644847344), 'Thoothukudi,Tamil Nadu,india': (8.805
2602, 78.1452745), 'Nagercoil,Tamil Nadu,india': (8.1880471, 77.4290492), 'Gaya,Bihar,india': (24.7964355, 85.0079563), 'Chandi
garh city,Chandigarh,india': (30.748224399999998, 76.78693757800895), 'Jammu,Jammu and Kashmir,india': (32.7185614, 74.858091
7), 'Kakinada,Andhra Pradesh,india': (17.1213126, 82.18422479792069), 'Dewas,Madhya Pradesh,india': (23.0, 76.166667), 'Baranag
ar,West Bengal,india': (22.6493438, 88.3685302), 'Firozabad,Uttar Pradesh,india': (27.178397150000002, 78.38936471833091), 'Phu
sro,Jharkhand,india': (23.7601823, 85.9755003), 'Allahabad,Uttar Pradesh,india': (25.4381302, 81.8338005), 'Guna,Madhya Prades
h,india': (24.5, 77.5), 'Thane,Maharashtra,india': (19.1943294, 72.9701779), 'Etawah,Uttar Pradesh,india': (26.819465450000003,
79.06011952812528), 'Pallavaram,Tamil Nadu,india': (12.989815700000001, 80.10098654184341), 'Morena,Madhya Pradesh,india': (26.
166667, 77.5), 'Ballia,Uttar Pradesh,india': (25.7557446, 84.1503303), 'Surat,Gujarat,india': (21.2094892, 72.8317058), 'Burhan
pur,Madhya Pradesh,india': (21.3118839, 76.2291992), 'Phagwara,Punjab,india': (31.2206734, 75.7696463), 'Mau,Uttar Pradesh,indi
a': (25.2356031, 81.33340219620874), 'Mangalore,Karnataka,india': (12.8698101, 74.8430082), 'Alwar,Rajasthan,india': (27.5, 76.
5), 'Mahbubnagar,Telangana,india': (16.7434538, 77.9923191), 'Maheshtala,West Bengal,india': (22.5060294, 88.2501665), 'Hazarib
agh,Jharkhand,india': (23.9675151, 85.43884568595615), 'Bihar Sharif,Bihar,india': (25.193859, 85.5208617), 'Faridabad,Haryana,
india': (28.4031478, 77.3105561), 'Lucknow,Uttar Pradesh,india': (26.8381, 80.9346001), 'Tenali,Andhra Pradesh,india': (16.2377
735, 80.6464219), 'Barasat,West Bengal,india': (22.7191072, 88.4826229), 'Amroha,Uttar Pradesh,india': (28.90662, 78.4707309),
'Giridih,Jharkhand,india': (24.1608897, 86.25113503301655), 'Begusarai,Bihar,india': (25.4139122, 86.1348804), 'Medininagar,Jha
rkhand,india': (24.02265335, 84.11191072812582), 'Rajahmundry,Andhra Pradesh,india': (17.0050454, 81.7804732), 'Saharsa,Bihar,i
ndia': (26.0, 86.75), 'New Delhi,Delhi,india': (28.6138954, 77.2090057), 'Bhilai,Chhattisgarh,india': (21.2120677, 81.3732849),
'Moradabad,Uttar Pradesh,india': (28.8334982, 78.7732864), 'Machilipatnam,Andhra Pradesh,india': (16.1817369, 81.1348181), 'Pal
i,Rajasthan,india': (25.7682376, 73.3275135), 'Navi Mumbai,Maharashtra,india': (19.0308262, 73.0198537), 'Mehsana,Gujarat,indi
a': (23.601262849999998, 72.37415155167554), 'Imphal,Manipur,india': (24.7991162, 93.9364419), 'Kolkata,West Bengal,india': (2
2.5726459, 88.3638953), 'Sambalpur,Odisha,india': (21.4706422, 83.98368580351055), 'Ujjain,Madhya Pradesh,india': (23.1885131,
75.7716562), 'Madhyamgram,West Bengal,india': (22.6947839, 88.4530183), 'Jabalpur,Madhya Pradesh,india': (23.1608938, 79.949770
2), 'Jamalpur,Bihar,india': (25.3297912, 86.45677664617543), 'Ludhiana,Punjab,india': (30.9090157, 75.851601), 'Bareilly,Uttar
Pradesh,india': (28.416667, 79.383333), 'Gangtok,Sikkim,india': (27.329046, 88.6122673), 'Anand,Gujarat,india': (22.5586555, 7
2.9627227), 'Dehradun,Uttarakhand,india': (30.3255646, 78.0436813), 'Pune,Maharashtra,india': (18.521428, 73.8544541), 'Satara,
Maharashtra,india': (17.688321, 74.0041726), 'Srikakulam,Andhra Pradesh,india': (18.32002205, 83.91607719937166), 'Raipur,Chhat
tisgarh,india': (21.2380912, 81.6336993), 'Jodhpur,Rajasthan,india': (26.2967719, 73.0351433), 'Darbhanga,Bihar,india': (26.156
999, 85.8995065), 'Nizamabad,Telangana,india': (18.6732693, 78.0978477), 'Nandyal,Andhra Pradesh,india': (15.52481495, 78.41517
103591221), 'Dehri,Bihar,india': (24.9258862, 84.17287231549471), 'Jorhat,Assam,india': (26.7577925, 94.2079645), 'Ranchi,Jhark
hand,india': (23.3700501, 85.3250387), 'Kumbakonam,Tamil Nadu,india': (10.9645549, 79.37173036154906), 'Guntakal,Andhra Prades
h,india': (15.11965065, 77.45529027323644), 'Haldia,West Bengal,india': (22.028124, 88.0632655), 'Loni,Uttar Pradesh,india': (2
8.7317295, 77.2900919), 'Rajkot,Gujarat,india': (22.3053263, 70.8028377), 'Nanded,Maharashtra,india': (19.17257215, 77.29141237
169956), 'Noida,Uttar Pradesh,india': (28.5706333, 77.3272147), 'Kirari Suleman Nagar,Delhi,india': (28.7026993, 77.047004), 'J
aunpur,Uttar Pradesh,india': (25.75, 82.75), 'Bilaspur,Chhattisgarh,india': (21.98934025, 82.10996498737376), 'Sambhal,Uttar Pr
adesh,india': (28.6187267, 78.5508909140845), 'Dhule,Maharashtra,india': (20.878922, 74.70314311020788), 'Rourkela,Odisha,indi
a': (22.253999399999998, 84.8577520139704), 'Thiruvananthapuram,Kerala,india': (8.4882267, 76.947551), 'Dharmavaram,Andhra Prad
esh,india': (14.42234715, 77.72006886674166), 'Nellore,Andhra Pradesh,india': (14.4493717, 79.9873763), 'Visakhapatnam,Andhra P

```
radesh,india': (17.7231276, 83.3012842), 'Karawal Nagar,Delhi,india': (28.7311406, 77.2747898), 'Jaipur,Rajasthan,india': (26.9
154576, 75.8189817), 'Avadi,Tamil Nadu,india': (13.1194809, 80.1026704), 'Bhimavaram,Andhra Pradesh,india': (16.5427689, 81.527
344), 'Bardhaman,West Bengal,india': (23.2495714, 87.8681751), 'Silchar,Assam,india': (24.817861100000002, 92.75622075294373),
'Buxar,Bihar,india': (25.5716195, 83.9729568), 'Kavali,Andhra Pradesh,india': (14.91098925, 80.01534335527676), 'Tezpur,Assam,i
ndia': (26.6229928, 92.7976082), 'Ramagundam,Telangana,india': (18.7615156, 79.4787848), 'Yamunanagar,Haryana,india': (30.12313
49, 77.286329), 'Sri Ganganagar,Rajasthan,india': (29.9270556, 73.8761519), 'Sasaram,Bihar,india': (24.9001003, 84.018211068457
5), 'Sikar,Rajasthan,india': (27.583333, 75.166667), 'Bally,West Bengal,india': (22.6469581, 88.343612), 'Bhiwani,Haryana,indi
a': (28.7931703, 76.1391283), 'Rampur,Uttar Pradesh,india': (28.8014001, 79.0284337), 'Uluberia,West Bengal,india': (22.470031
8, 88.1011108), 'Hosur,Tamil Nadu,india': (12.7328844, 77.8309478), 'Bikaner,Rajasthan,india': (28.0159286, 73.3171367), 'Shahj
ahanpur,Uttar Pradesh,india': (27.8282891, 79.82714577144085), 'Vijayawada,Andhra Pradesh,india': (16.5087573, 80.6185089), 'Bh
aratpur,Rajasthan,india': (27.2154447, 77.4986962), 'Tadepalligudem,Andhra Pradesh,india': (16.8102437, 81.526592), 'Tinsukia,A
ssam,india': (27.52016575, 95.5306347312879), 'Salem,Tamil Nadu,india': (11.6551982, 78.1581771), 'Mathura,Uttar Pradesh,indi
a': (27.633333, 77.583333), 'Guntur,Andhra Pradesh,india': (16.2915189, 80.4541588), 'Guwahati,Assam,india': (26.1805978, 91.75
3943), 'Chittoor,Andhra Pradesh,india': (13.1601048, 79.15555061803596), 'Tiruvottiyur,Tamil Nadu,india': (13.1563873, 80.30052
79), 'Vadodara,Gujarat,india': (22.2973142, 73.1942567), 'Ahmednagar,Maharashtra,india': (19.092952, 74.7493451), 'Fatehpur,Utt
ar Pradesh,india': (27.17463745, 81.10574256080653), 'Bhilwara,Rajasthan,india': (25.5, 74.75), 'Kharagpur,West Bengal,india':
(22.34309, 87.3012875), 'Bettiah,Bihar,india': (26.791072749999998, 84.56010678772793), 'Bhind,Madhya Pradesh,india': (26.5, 7
8.75), 'Bokaro,Jharkhand,india': (23.6544338, 86.1456444), 'Karaikudi,Tamil Nadu,india': (10.0728444, 78.7795194), 'Raebareli,U
ttar Pradesh,india': (26.253467, 81.17537278201644), 'Pudukkottai,Tamil Nadu,india': (10.5, 78.833333), 'Udaipur,Rajasthan,indi
a': (24.578721, 73.6862571), 'Mysore,Karnataka,india': (12.3051828, 76.6553609), 'Panipat,Haryana,india': (29.3912753, 76.97716
75), 'Latur,Maharashtra,india': (18.426698000000002, 76.48706130861112), 'Tadipatri,Andhra Pradesh,india': (14.9069559, 78.0097
071), 'Bahraich,Uttar Pradesh,india': (27.75, 81.75), 'Orai,Uttar Pradesh,india': (25.9359555, 79.42432768008604), 'Rewa,Madhya
Pradesh,india': (24.75, 81.5), 'Gwalior,Madhya Pradesh,india': (26.2037247, 78.1573628), 'Katni,Madhya Pradesh,india': (23.7477
258, 80.40158570175886), 'Chandrapur,Maharashtra,india': (20.0, 80.0), 'Kolhapur,Maharashtra,india': (16.7028412, 74.2405329)}
```

these addresses can not be converted because of spelling mistakes or some other reasons. So we have to correct these mistakes manually and retry.

In [ ]:
```
address2={'KalyanDombivli,Maharashtra,india':'Kalyan Dombivli,Maharashtra,india',
          'Surendranagar Dudhrej,Gujarat,india':'Surendranagar Dudhrej,Gujarat,india',
          'Nangloi Jat,Delhi,india':'Nangloi Jat,Delhi,india',
         "Bhalswa Jahangir Pur,Delhi,india":'Bhalswa Jahangir Pur,Delhi,india',
          ' VasaiVirar,Maharashtra,india':' Vasai Virar,Maharashtra,india' ,
          "MiraBhayandar,Maharashtra,india":'Mira Bhayandar,Maharashtra,india',
          "PimpriChinchwad,Maharashtra,india":'Pimpri Chinchwad,Maharashtra,india',
         "SangliMiraj  Kupwad,Maharashtra,india":'Sangli Miraj  Kupwad,Maharashtra,india',
         "Sultan Pur Majra,Delhi,india":'Sultan Pur Majra,Delhi,india',
        "HubliâDharwad,Karnataka,india":'Hubli Dharwad,Karnataka,india',
        "Raurkela Industrial Township,Odisha,india":'Raurkela,Odisha,india',}
```

In [ ]:
```
for add1,add2 in address2.items():
    coords = get_lat_lon_geopy(add2)
    if coords:
        long_lat[add1] = coords
```

```
Address not found: Surendranagar Dudhrej,Gujarat,india
Address not found: Nangloi Jat,Delhi,india
Address not found: Bhalswa Jahangir Pur,Delhi,india
Address not found: Sultan Pur Majra,Delhi,india
```

In [ ]:
```
# we can use google search for these addresses
long_lat['Surendranagar Dudhrej,Gujarat,india']=(22.7271,71.6485)
long_lat['Nangloi Jat,Delhi,india']=(28.6849,77.0689)
long_lat['Bhalswa Jahangir Pur,Delhi,india']=(25.257428,85.751975)
long_lat["Sultan Pur Majra,Delhi,india"]=(29.029486,76.925703)
long_lat['Raurkela Industrial Township,Odisha,india']=(22.2494,84.8828)
long_lat['VasaiVirar,Maharashtra,india']=(20.502147,75.664579)
```

In [ ]:
```
# store the lat and long in the dataframe
df['lat']=df['address'].map(lambda x:long_lat[x][0] if x in long_lat else None)
```

In [ ]:
```
df['lon']=df['address'].map(lambda x:long_lat[x][1] if x in long_lat else None)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURR
0	1	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	
1	2	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	
2	3	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	
3	4	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli	Tamil Nadu	
4	5	6915937	64	0	single	rented	no	Civil_servant	Jalgaon	Maharashtra	

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Id                 0
Income             0
Age                0
Experience         0
Married/Single     0
House_Ownership    0
Car_Ownership      0
Profession         0
CITY               0
STATE              0
CURRENT_JOB_YRS    0
CURRENT_HOUSE_YRS  0
Risk_Flag          0
address            0
lat                0
lon                0
dtype: int64
```

In [ ]:
```
df['Profession'].unique(),len(df['Profession'].unique())
```

Out[ ]:
```
(array(['Software_Developer', 'Technical_writer', 'Civil_servant',
        'Librarian', 'Economist', 'Flight_attendant', 'Architect',
        'Designer', 'Physician', 'Financial_Analyst',
        'Air_traffic_controller', 'Politician', 'Police_officer', 'Artist',
        'Surveyor', 'Design_Engineer', 'Chemical_engineer',
        'Hotel_Manager', 'Mechanical_engineer', 'Dentist', 'Comedian',
        'Biomedical_Engineer', 'Graphic_Designer',
        'Computer_hardware_engineer', 'Petroleum_Engineer', 'Secretary',
        'Computer_operator', 'Chartered_Accountant', 'Technician',
        'Microbiologist', 'Fashion_Designer', 'Aviator', 'Psychologist',
        'Magistrate', 'Lawyer', 'Firefighter', 'Engineer', 'Official',
        'Analyst', 'Geologist', 'Drafter', 'Statistician', 'Web_designer',
        'Consultant', 'Chef', 'Army_officer', 'Surgeon', 'Scientist',
        'Civil_engineer', 'Industrial_Engineer', 'Technology_specialist'],
       dtype=object),
 51)
```

51 unique professions are there

In [ ]:
```
# one hot encoding for profession
df=pd.get_dummies(df,columns=['Profession'])
```

In [ ]:
```
df.head()
```

Out[ ]:

	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag	
0	7574516	40	10	single	rented	no	9	13	0	19.
1	3991815	66	4	married	rented	no	4	10	0	9.
2	6256451	41	2	single	rented	yes	2	12	1	20.
3	5768871	47	11	single	rented	no	3	14	1	10.
4	6915937	64	0	single	rented	no	0	12	0	21.

5 rows × 62 columns

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['Id', 'Income', 'Age', 'Experience', 'Married/Single',
       'House_Ownership', 'Car_Ownership', 'CITY', 'STATE', 'CURRENT_JOB_YRS',
       'CURRENT_HOUSE_YRS', 'Risk_Flag', 'address', 'lat', 'lon',
       'Profession_Air_traffic_controller', 'Profession_Analyst',
       'Profession_Architect', 'Profession_Army_officer', 'Profession_Artist',
       'Profession_Aviator', 'Profession_Biomedical_Engineer',
       'Profession_Chartered_Accountant', 'Profession_Chef',
       'Profession_Chemical_engineer', 'Profession_Civil_engineer',
       'Profession_Civil_servant', 'Profession_Comedian',
       'Profession_Computer_hardware_engineer', 'Profession_Computer_operator',
       'Profession_Consultant', 'Profession_Dentist',
       'Profession_Design_Engineer', 'Profession_Designer',
       'Profession_Drafter', 'Profession_Economist', 'Profession_Engineer',
       'Profession_Fashion_Designer', 'Profession_Financial_Analyst',
       'Profession_Firefighter', 'Profession_Flight_attendant',
       'Profession_Geologist', 'Profession_Graphic_Designer',
       'Profession_Hotel_Manager', 'Profession_Industrial_Engineer',
       'Profession_Lawyer', 'Profession_Librarian', 'Profession_Magistrate',
       'Profession_Mechanical_engineer', 'Profession_Microbiologist',
       'Profession_Official', 'Profession_Petroleum_Engineer',
       'Profession_Physician', 'Profession_Police_officer',
       'Profession_Politician', 'Profession_Psychologist',
       'Profession_Scientist', 'Profession_Secretary',
       'Profession_Software_Developer', 'Profession_Statistician',
       'Profession_Surgeon', 'Profession_Surveyor',
       'Profession_Technical_writer', 'Profession_Technician',
       'Profession_Technology_specialist', 'Profession_Web_designer'],
      dtype='object')
```

In [ ]:
```
# drop the address column
df.drop(labels=['CITY','STATE','address'],axis=1,inplace=True)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
0	1	7574516	40	10	single	rented	no	9	13	0
1	2	3991815	66	4	married	rented	no	4	10	0
2	3	6256451	41	2	single	rented	yes	2	12	1
3	4	5768871	47	11	single	rented	no	3	14	1
4	5	6915937	64	0	single	rented	no	0	12	0

5 rows × 63 columns

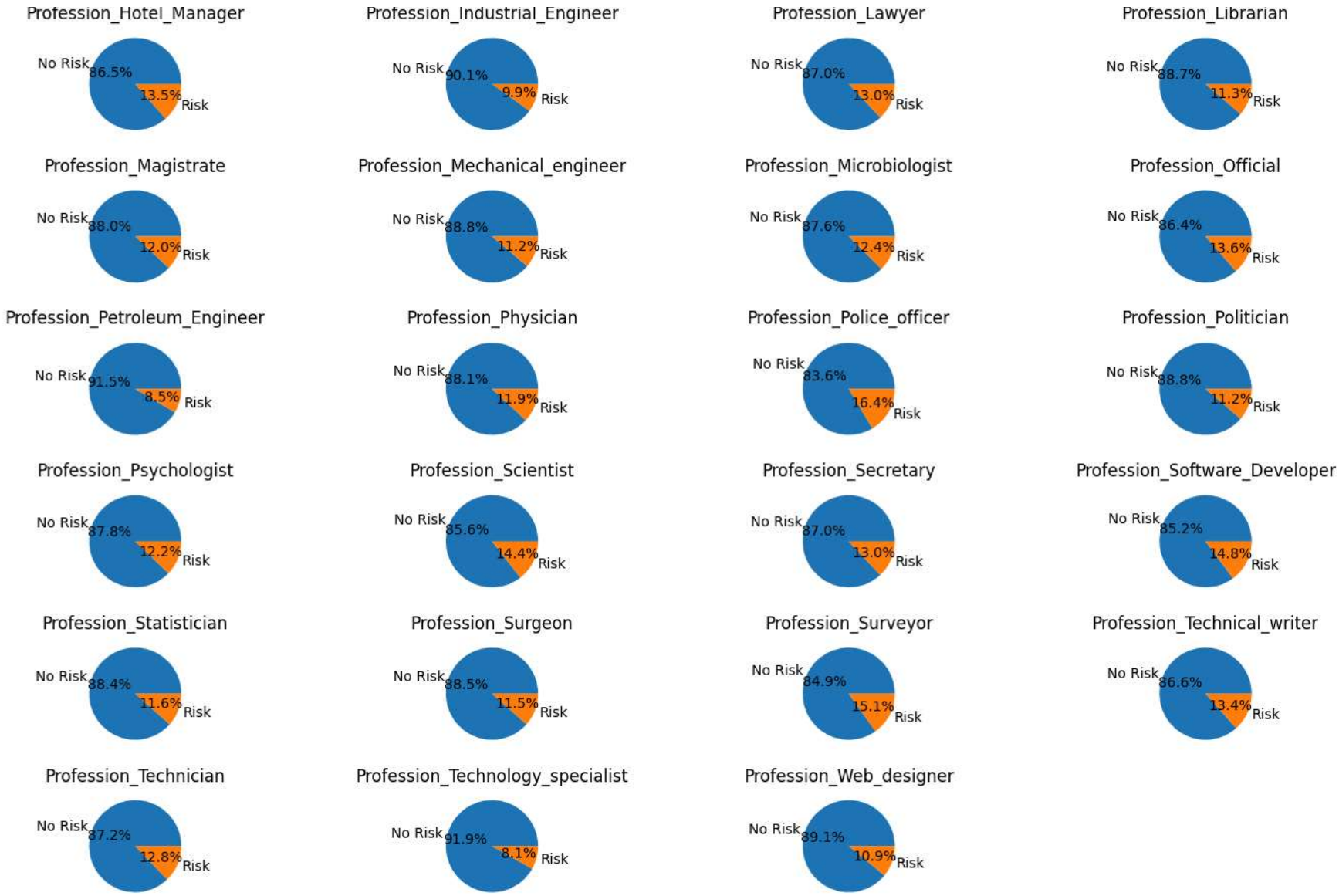
In [ ]:

```
professions=['Profession_Air_traffic_controller', 'Profession_Analyst',
       'Profession_Architect', 'Profession_Army_officer', 'Profession_Artist',
       'Profession_Aviator', 'Profession_Biomedical_Engineer',
       'Profession_Chartered_Accountant', 'Profession_Chef',
       'Profession_Chemical_engineer', 'Profession_Civil_engineer',
       'Profession_Civil_servant', 'Profession_Comedian',
       'Profession_Computer_hardware_engineer', 'Profession_Computer_operator',
       'Profession_Consultant', 'Profession_Dentist',
       'Profession_Design_Engineer', 'Profession_Designer',
       'Profession_Drafter', 'Profession_Economist', 'Profession_Engineer',
       'Profession_Fashion_Designer', 'Profession_Financial_Analyst',
       'Profession_Firefighter', 'Profession_Flight_attendant',
       'Profession_Geologist', 'Profession_Graphic_Designer',
       'Profession_Hotel_Manager', 'Profession_Industrial_Engineer',
       'Profession_Lawyer', 'Profession_Librarian', 'Profession_Magistrate',
       'Profession_Mechanical_engineer', 'Profession_Microbiologist',
       'Profession_Official', 'Profession_Petroleum_Engineer',
       'Profession_Physician', 'Profession_Police_officer',
       'Profession_Politician', 'Profession_Psychologist',
       'Profession_Scientist', 'Profession_Secretary',
       'Profession_Software_Developer', 'Profession_Statistician',
       'Profession_Surgeon', 'Profession_Surveyor',
       'Profession_Technical_writer', 'Profession_Technician',
       'Profession_Technology_specialist', 'Profession_Web_designer']
```

In [ ]:

```
def risk_flag_profession(professions):
    n = len(professions)
    row=math.ceil(n/4)
    col=4

    fig, axs = plt.subplots(row,col, figsize=(15, 20))
    axs = axs.flatten()

    for i, profession in enumerate(professions):
        df1 = df[df[profession] == 1]
        risk_counts = df1['Risk_Flag'].value_counts(normalize=True)
        axs[i].pie(risk_counts, labels=['No Risk', 'Risk'], autopct='%1.1f%%')
        axs[i].set_title(f"{profession}")

    for j in range(i + 1, len(axs)):
        axs[j].axis('off')

    plt.tight_layout()
    plt.show()

risk_flag_profession(professions)
```


Profession_Air_traffic_controller
No Risk 86.5%
13.5% Risk
Profession_Analyst
No Risk 87.9%
12.1% Risk
Profession_Architect
No Risk 86.9%
13.1% Risk
Profession_Army_officer
No Risk 84.8%
15.2% Risk
Profession_Artist
No Risk 87.7%
12.3% Risk
Profession_Aviator
No Risk 86.5%
13.5% Risk
Profession_Biomedical_Engineer
No Risk 87.2%
12.8% Risk
Profession_Chartered_Accountant
No Risk 84.6%
15.4% Risk
Profession_Chef
No Risk 87.9%
12.1% Risk
Profession_Chemical_engineer
No Risk 88.8%
11.2% Risk
Profession_Civil_engineer
No Risk 86.4%
13.6% Risk
Profession_Civil_servant
No Risk 88.4%
11.6% Risk
Profession_Comedian
No Risk 88.0%
12.0% Risk
Profession_Computer_hardware_engineer
No Risk 87.2%
12.8% Risk
Profession_Computer_operator
No Risk 87.6%
12.4% Risk
Profession_Consultant
No Risk 87.5%
12.5% Risk
Profession_Dentist
No Risk 89.0%
11.0% Risk
Profession_Design_Engineer
No Risk 89.3%
10.7% Risk
Profession_Designer
No Risk 89.1%
10.9% Risk
Profession_Drafter
No Risk 88.7%
11.3% Risk
Profession_Economist
No Risk 90.1%
9.9% Risk
Profession_Engineer
No Risk 88.2%
11.8% Risk
Profession_Fashion_Designer
No Risk 88.5%
11.5% Risk
Profession_Financial_Analyst
No Risk 89.7%
10.3% Risk
Profession_Firefighter
No Risk 86.4%
13.6% Risk
Profession_Flight_attendant
No Risk 87.6%
12.4% Risk
Profession_Geologist
No Risk 85.6%
14.4% Risk
Profession_Graphic_Designer
No Risk 88.5%
11.5% Risk

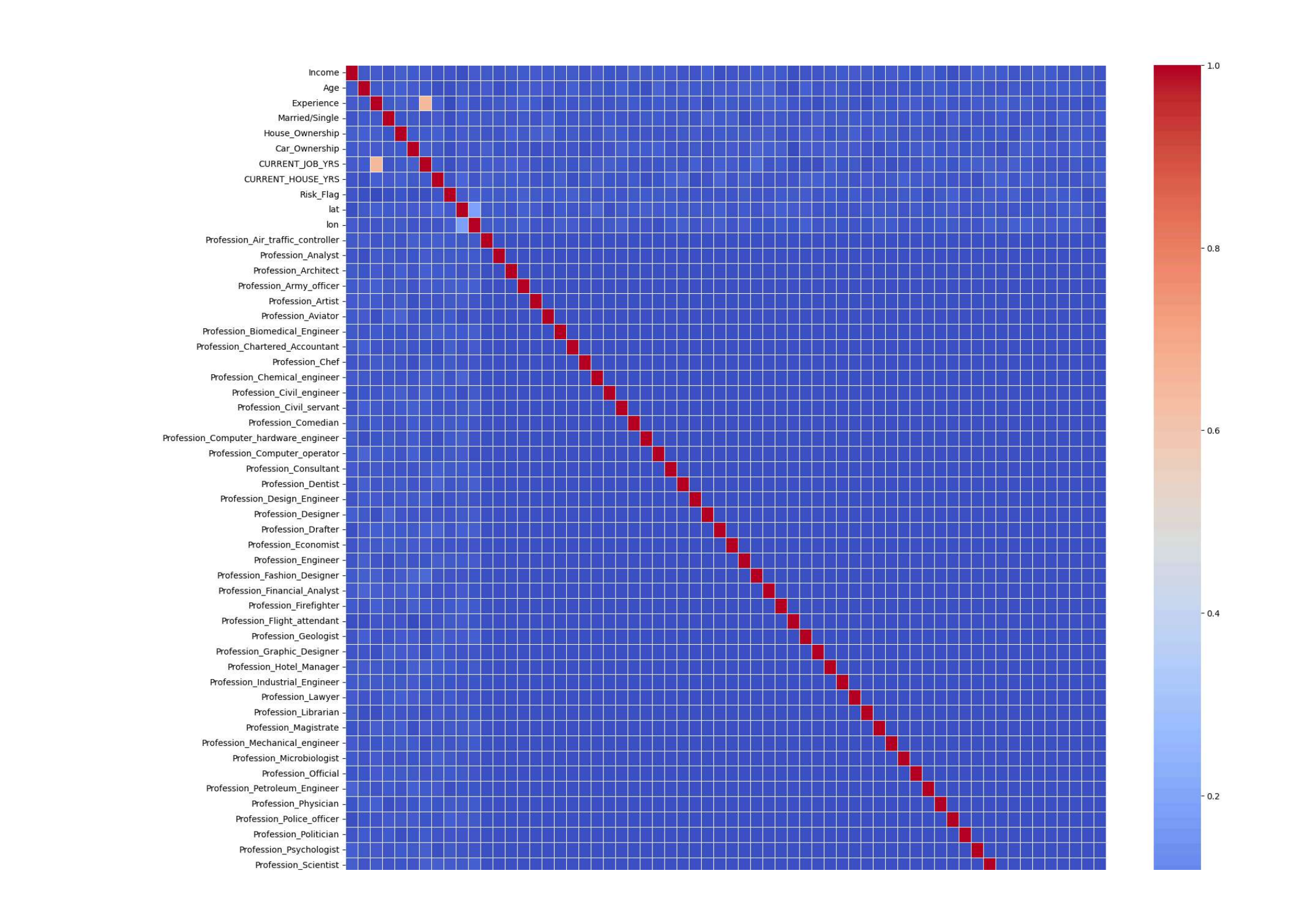


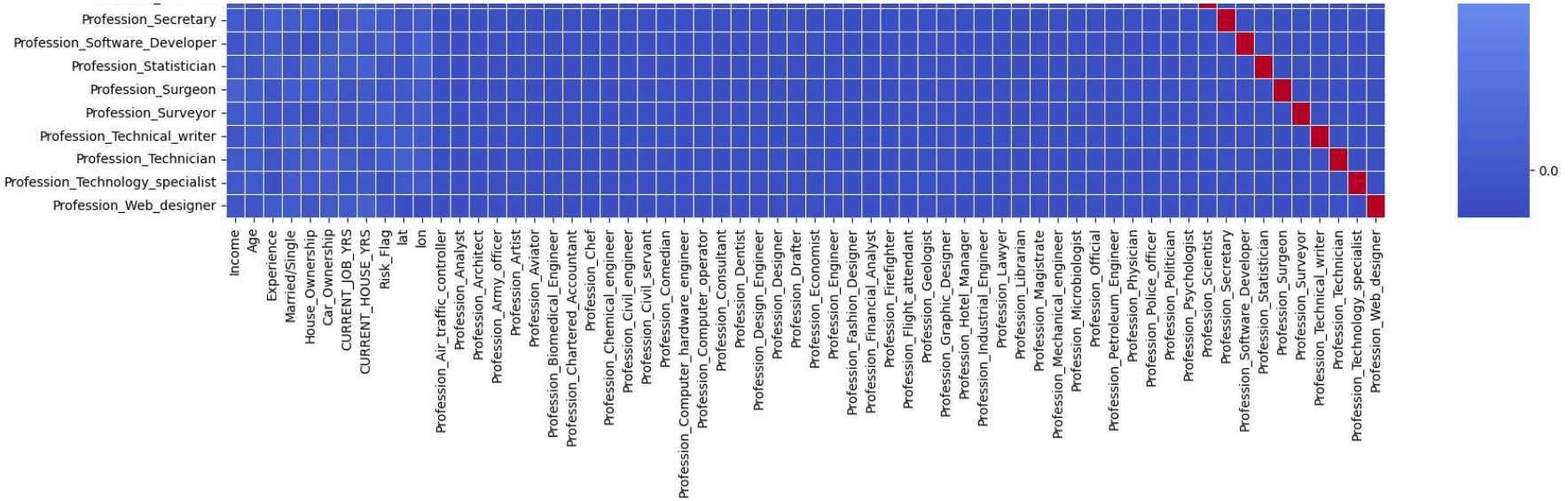
Police officer is most risky with 16.4% of risk flag.

Technology Specialist is least risky with 8.1% of risk flag.

In [ ]:

```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),annot=False,cmap='coolwarm',linewidths=0.5)
plt.show()
```


Income
Age
Experience
Married/Single
House_Ownership
Car_Ownership
CURRENT_JOB_YRS
CURRENT_HOUSE_YRS
Risk_Flag
lat
lon
Profession_Air_traffic_controller
Profession_Analyst
Profession_Architect
Profession_Army_officer
Profession_Artist
Profession_Aviator
Profession_Biomedical_Engineer
Profession_Chartered_Accountant
Profession_Chef
Profession_Chemical_engineer
Profession_Civil_engineer
Profession_Civil_servant
Profession_Comedian
Profession_Computer_hardware_engineer
Profession_Computer_operator
Profession_Consultant
Profession_Dentist
Profession_Design_Engineer
Profession_Designer
Profession_Drafter
Profession_Economist
Profession_Engineer
Profession_Fashion_Designer
Profession_Financial_Analyst
Profession_Firefighter
Profession_Flight_attendant
Profession_Geologist
Profession_Graphic_Designer
Profession_Hotel_Manager
Profession_Industrial_Engineer
Profession_Lawyer
Profession_Librarian
Profession_Magistrate
Profession_Mechanical_engineer
Profession_Microbiologist
Profession_Official
Profession_Petroleum_Engineer
Profession_Physician
Profession_Police_officer
Profession_Politician
Profession_Psychologist
Profession_Scientist
1.0
0.8
0.6
0.4
0.2



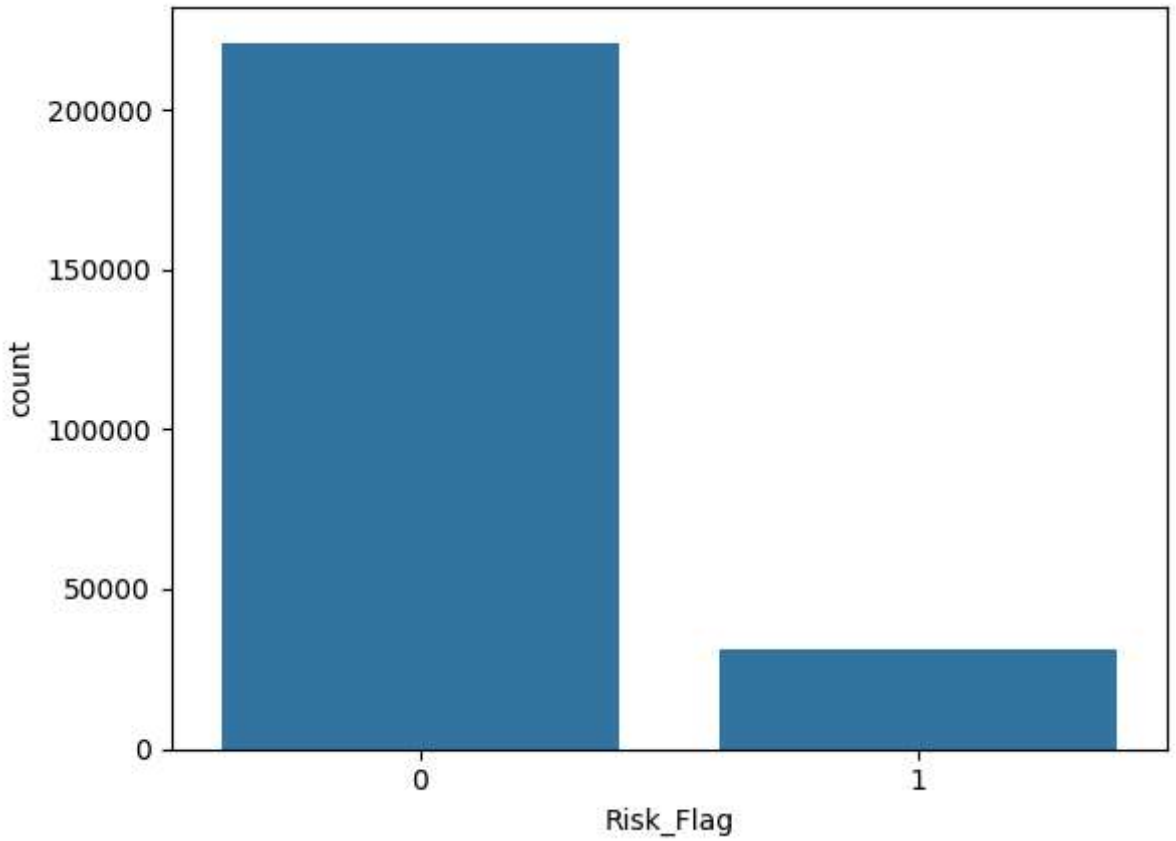
In [ ]:
```
df.to_csv('./cleaned_data.csv',index=False)
```

In [ ]:
```
counts=df['Risk_Flag'].value_counts()
print(counts)
```

```
Risk_Flag
0    221003
1     30996
Name: count, dtype: int64
```

In [ ]:
```
sns.countplot(x='Risk_Flag',data=df)
```

Out[ ]: <Axes: xlabel='Risk_Flag', ylabel='count'>



Clearly data is imbalanced.

In [ ]:
```
df.head()
```

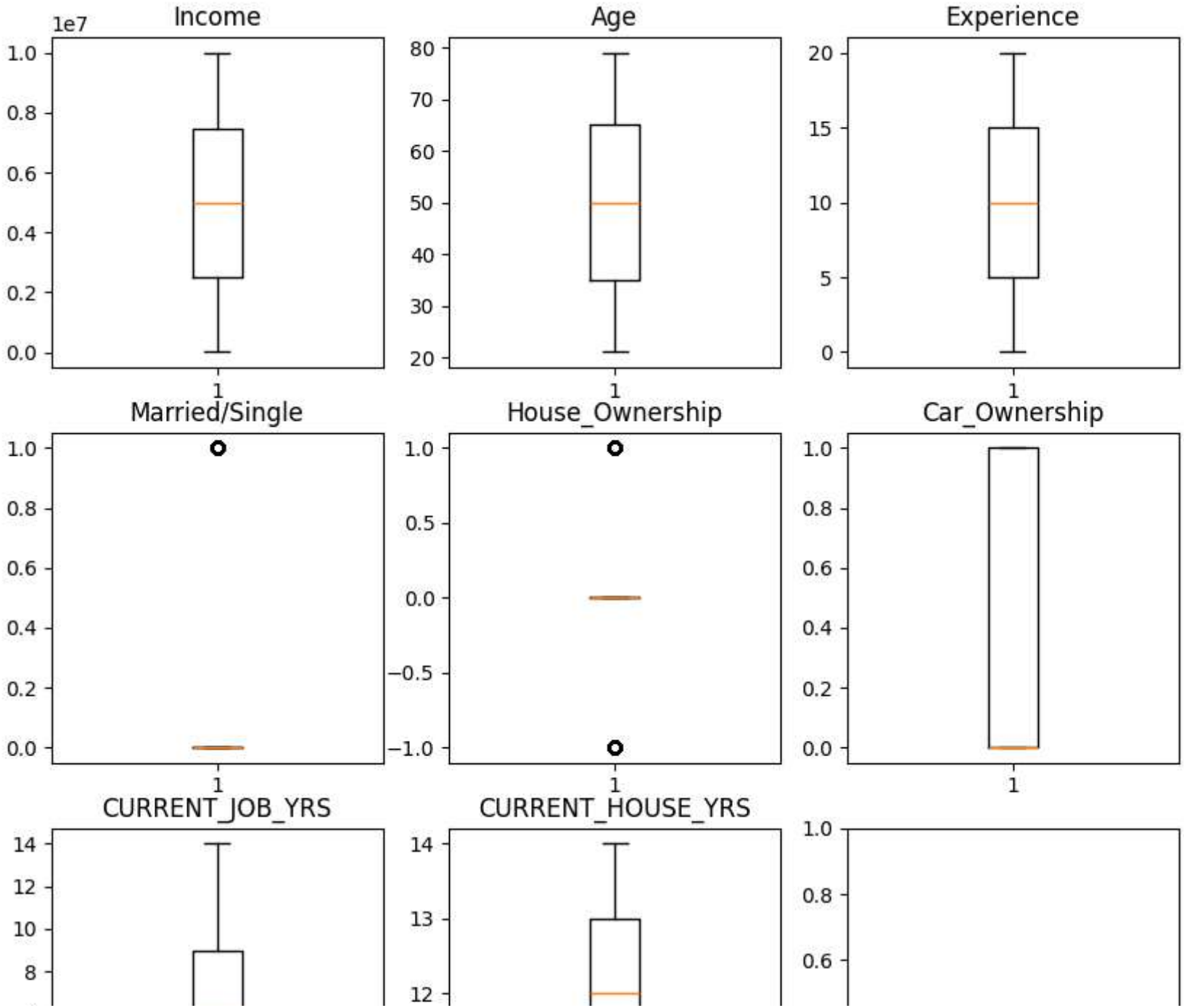
Out[ ]:

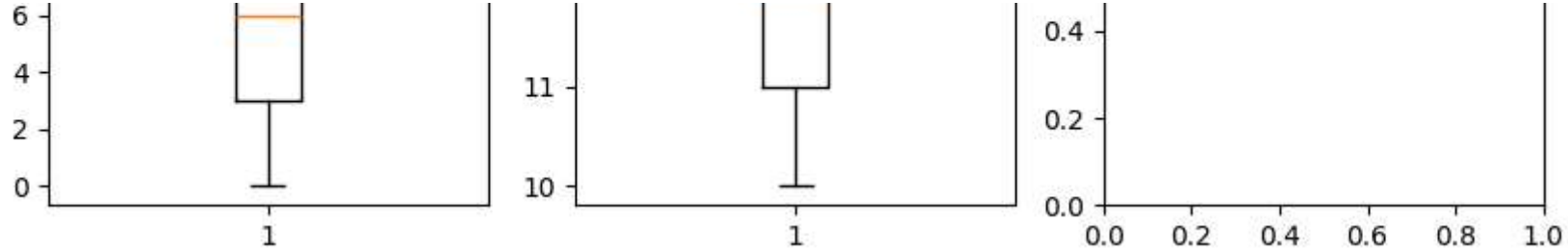
	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag	
0	7574516	40	10	0	0	0	9	13	0	19.
1	3991815	66	4	1	0	0	4	10	0	9.
2	6256451	41	2	0	0	1	2	12	1	20.
3	5768871	47	11	0	0	0	3	14	1	10.
4	6915937	64	0	0	0	0	0	12	0	21.

5 rows × 62 columns

In [ ]:
```
# outlier detection
def plot_box():
    fig, axs = plt.subplots(3,3, figsize=(10,10))
    axs = axs.flatten()
    cols=['Income', 'Age', 'Experience', 'Married/Single', 'House_Ownership',
       'Car_Ownership', 'CURRENT_JOB_YRS',
       'CURRENT_HOUSE_YRS']
    for i, col in enumerate(cols):
        axs[i].boxplot(df[col])
        axs[i].set_title(f"{col}")
    plt.show()
```

In [ ]:
```
plot_box()
```


Income
1e7
1.0
0.8
0.6
0.4
0.2
0.0
1
Age
80
70
60
50
40
30
20
1
Experience
20
15
10
5
0
1
Married/Single
1.0
0.8
0.6
0.4
0.2
0.0
1
House_Ownership
1.0
0.5
0.0
−0.5
−1.0
1
Car_Ownership
1.0
0.8
0.6
0.4
0.2
0.0
1
CURRENT_JOB_YRS
14
12
10
8
CURRENT_HOUSE_YRS
14
13
12
1.0
0.8
0.6



We do not have any outliers in the data.

Since our dataset is imbalanced so we have to do resample our data.

In [ ]:
```
df1=df[df['Risk_Flag']==1]
df0=df[df['Risk_Flag']==0]
df1.shape,df0.shape
```

Out[ ]:
```
((30996, 62), (221003, 62))
```

In [ ]:
```
# we will consider 30000 of each class
df1=df1.sample(30000)
df0=df0.sample(30000)
new_df=pd.concat([df1,df0],axis=0)
```

In [ ]:
```
new_df.shape
```

Out[ ]:
```
(60000, 62)
```

In [ ]:
```
new_df.head()
```

Out[ ]:

	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Fla
13457	4613553	27	13	0	1	0	13	10	
227630	2092479	78	12	0	0	0	6	10	
109801	5832406	24	4	0	0	1	4	12	
35238	2755203	64	15	0	0	0	11	12	
180067	4514928	28	20	0	0	1	5	11	

5 rows × 62 columns

In [ ]:
```
new_df.to_csv('./balanced_data.csv',index=False)
```

Since dimensionality of our data is quite high so we will use PCA to reduce the dimensionality.

In [ ]:
```
target=new_df['Risk_Flag']
new_df.drop(labels=['Risk_Flag'],axis=1,inplace=True)
```

In [ ]:
```
# perform PCA
pca = PCA(n_components=4)
principalComponents = pca.fit_transform(new_df)
```

In [ ]:
```
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2','principal
```

Fitting and selection of the best model

In [ ]:
```
x_train,x_test,y_train,y_test=train_test_split(principalDf,target,test_size=0.2,random_state=42)
```

Logistic regression

In [ ]:
```
logistic_model=LogisticRegression()
```

In [ ]:
```
logistic_model.fit(x_train,y_train)
```

Out[ ]:
```
LogisticRegression()
```

In [ ]:
```
y_pred=logistic_model.predict(x_test)
```

In [ ]:
```
metrics.accuracy_score(y_test,y_pred)
```

Out[ ]:
```
0.5184166666666666
```

it is slightly better than a random guess.

now let's try this with original data(without pca)

In [ ]:
```
logistic_model2=LogisticRegression(penalty='l1',solver='liblinear')
```

In [ ]:
```
logistic_model2.fit(x_train,y_train)
y_pred=logistic_model2.predict(x_test)
metrics.accuracy_score(y_test,y_pred)
```

Out[ ]:
```
0.51375
```

In [ ]:
```
# let us try to fit it directly without pca
x_train1,x_test1,y_train1,y_test1=train_test_split(new_df,target,test_size=0.2,random_state=45)
```

In [ ]:
```
logistic_model3=LogisticRegression(max_iter=1000)
```

In [ ]:
```
logistic_model3.fit(x_train1,y_train1)
y_pred=logistic_model3.predict(x_test1)
```

In [ ]:
```
print(metrics.classification_report(y_test1,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.52      0.51      0.51      5966
           1       0.53      0.54      0.54      6034

    accuracy                           0.53     12000
   macro avg       0.52      0.52      0.52     12000
weighted avg       0.52      0.53      0.52     12000
```

it is also same.

K-Neighbors

In [ ]:
```
kn_model1=KNeighborsClassifier()
kn_model2=KNeighborsClassifier()
```

In [ ]:
```
kn_model1.fit(x_train,y_train)
y_pred=kn_model1.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.85      0.84      6091
           1       0.84      0.82      0.83      5909

    accuracy                           0.84     12000
   macro avg       0.84      0.84      0.84     12000
weighted avg       0.84      0.84      0.84     12000
```

In [ ]:
```
kn_model2.fit(x_train1,y_train1)
y_pred=kn_model2.predict(x_test1)
print(metrics.classification_report(y_test1,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.84      0.84      5966
           1       0.84      0.83      0.84      6034

    accuracy                           0.84     12000
   macro avg       0.84      0.84      0.84     12000
weighted avg       0.84      0.84      0.84     12000
```

K-neighbors has higher accuracy in both type of data.

Support Vector Classifier

In [ ]:
```
svc1=SVC()
svc2=SVC()
```

In [ ]:
```
svc1.fit(x_train,y_train)
y_pred=svc1.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.51      0.65      0.57      6091
           1       0.50      0.37      0.42      5909

    accuracy                           0.51     12000
   macro avg       0.51      0.51      0.50     12000
weighted avg       0.51      0.51      0.50     12000
```

In [ ]:
```
svc2.fit(x_train1,y_train1)
y_pred=svc2.predict(x_test1)
print(metrics.classification_report(y_test1,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.50      0.78      0.61      5966
           1       0.52      0.24      0.33      6034

    accuracy                           0.51     12000
   macro avg       0.51      0.51      0.47     12000
weighted avg       0.51      0.51      0.47     12000
```

In [ ]:
```
svc3=SVC(kernel='poly')
svc3.fit(x_train,y_train)
y_pred=svc3.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.52      0.38      0.44      6091
           1       0.50      0.63      0.56      5909

    accuracy                           0.50     12000
   macro avg       0.51      0.51      0.50     12000
weighted avg       0.51      0.50      0.50     12000
```

support vector classifier's performance is poor.

Decision Tree Classifier

In [ ]:
```
tree_model=DecisionTreeClassifier()
```

In [ ]:
```
params={
    'max_depth':[2,3,4,5,6,7,8,9,10],
    'criterion':['gini','entropy']
}
```

In [ ]:
```
clf=GridSearchCV(tree_model,param_grid=params,cv=5)
```

In [ ]:
```
clf.fit(x_train,y_train)
```

Out[ ]:

▸ GridSearchCV

▸ best_estimator_: DecisionTreeClassifier

▸ DecisionTreeClassifier

In [ ]:
```
clf.best_params_
```

Out[ ]:
```
{'criterion': 'gini', 'max_depth': 10}
```

In [ ]:
```
tree_model=DecisionTreeClassifier(max_depth=10,criterion='gini')
tree_model.fit(x_train,y_train)
y_pred=tree_model.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.59      0.78      0.67      6091
           1       0.66      0.44      0.53      5909

    accuracy                           0.61     12000
   macro avg       0.62      0.61      0.60     12000
weighted avg       0.62      0.61      0.60     12000
```

Decision Tree has a slighly better accuracy than other models.

Adaboost Classifier

In [ ]:
```
adaboot_model=AdaBoostClassifier(estimator=tree_model,n_estimators=50,learning_rate=1.0)
adaboot_model.fit(x_train,y_train)
y_pred=adaboot_model.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
d:\Programs\Python\PW\projects\NLP\env\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R a
lgorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
```

```
              precision    recall  f1-score   support

           0       0.86      0.86      0.86      6091
           1       0.86      0.86      0.86      5909

    accuracy                           0.86     12000
   macro avg       0.86      0.86      0.86     12000
weighted avg       0.86      0.86      0.86     12000
```

It's accuracy is better than others.

In [ ]:
```
randomforest_model=RandomForestClassifier()
randomforest_model.fit(x_train,y_train)
y_pred=randomforest_model.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      6091
           1       0.87      0.83      0.85      5909

    accuracy                           0.85     12000
   macro avg       0.85      0.85      0.85     12000
weighted avg       0.85      0.85      0.85     12000
```

In [ ]:
```
randomforest_model=RandomForestClassifier(criterion='entropy',n_estimators=100)
randomforest_model.fit(x_train,y_train)
y_pred=randomforest_model.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      6091
           1       0.87      0.83      0.85      5909

    accuracy                           0.85     12000
   macro avg       0.85      0.85      0.85     12000
weighted avg       0.85      0.85      0.85     12000
```

In [ ]:
```
gradient_boost_model=GradientBoostingClassifier()
gradient_boost_model.fit(x_train,y_train)
y_pred=gradient_boost_model.predict(x_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.61      0.67      0.64      6091
           1       0.62      0.56      0.59      5909

    accuracy                           0.61     12000
   macro avg       0.62      0.61      0.61     12000
weighted avg       0.62      0.61      0.61     12000
```

Conclusion

Overall we have adaboost as best model. Although we can further try deep learning based models for better accuracy. Adaboost model has accuracy 86% and precision and recall both are 86%. Here we are predicting the positiveness of the test example whether it is risky or not. So positive class is more important for us and give preference to the precision over recall. Adaboost model has the best precion.