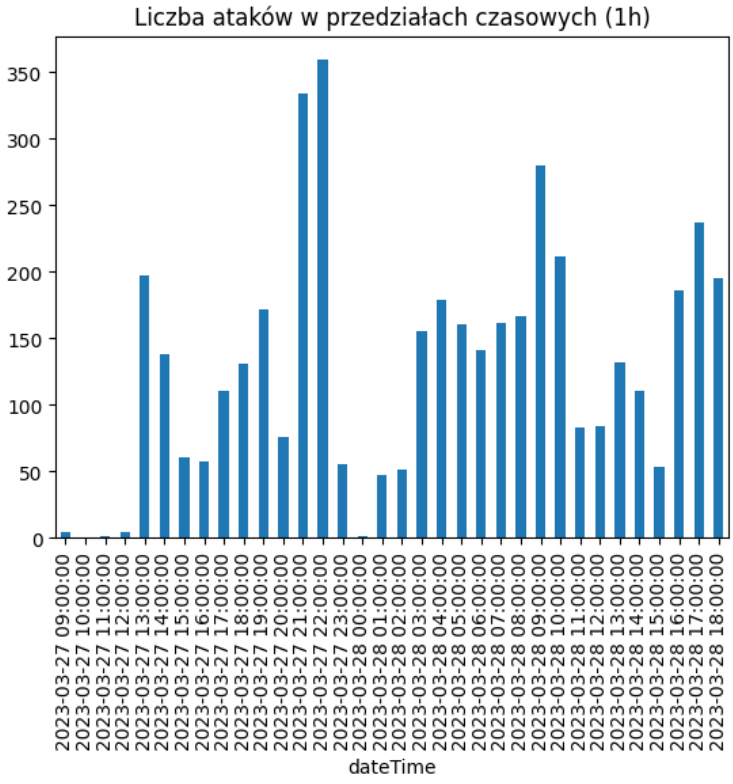
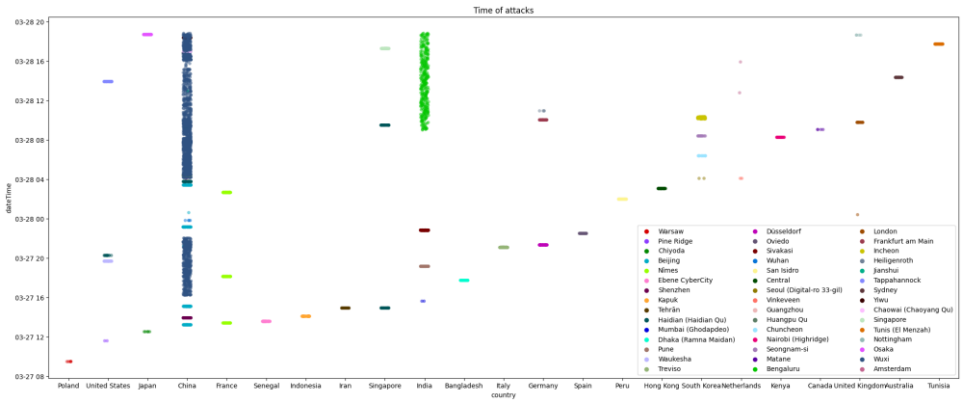
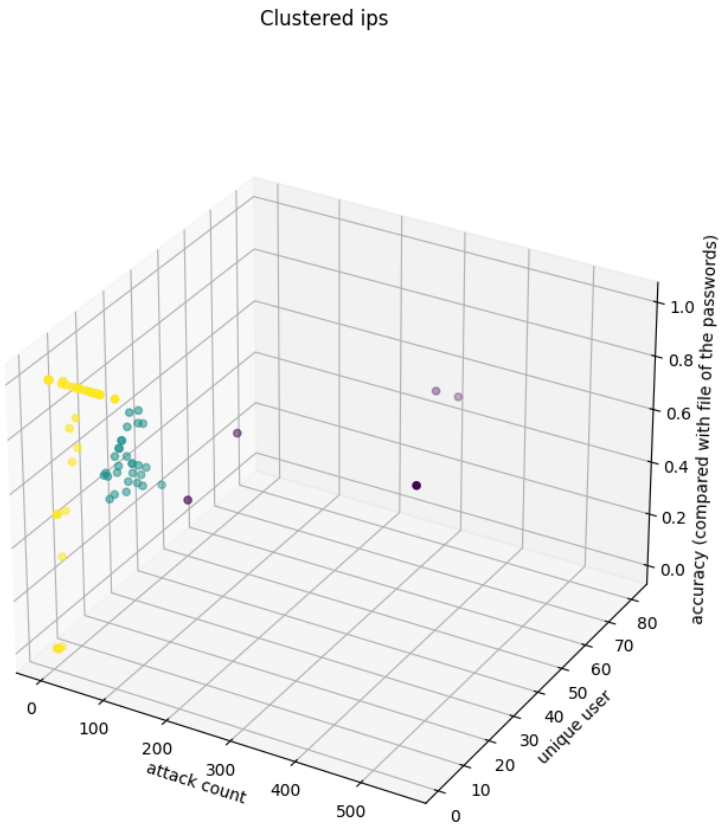
Analiza ataków hakerskich na maszynę wirtualną Azure.

Najpierw zbadaliśmy jak ataki na naszą maszynę wirtualną rozkładają się w czasie. System był uruchomiony przez około 32 godziny. Na początku nieudanych prób logowań było niewiele (pierwsze, później odnotowane jako z Polski, były naszymi własnymi), natomiast już po czterech godzinach zaczęło ich przybywać. Przerwa nastąpiła pomiędzy 0:00 a 1:00 czasu GMT (u nas 2 i 3 w nocy, w Chinach w mieście Wuxi 8 i 9 rano). Stworzyliśmy również stripplot, który opisuje liczbę ataków w danym przedziale czasu z poszczególnych państw oraz miast. Większość ataków była trwała względnie krótko, z nielicznymi wyjątkami się nie powtarzały. Api z którego korzystaliśmy (ipgeolocation.abstractapi) nie jest idealne (miasto Nîmes jest francuskie, a api zwracało, że niemieckie). Pomiędzy 8, a 12 godziną w Wuxi, ataki z tego miasta ustały, aby wrócić później ze zdwojoną siłą. Zmasowane próby logowania pochodziły również z Indii, a dokładniej z Bengaluru. Poza tym aktywni byli jeszcze “mieszkańcy” miasta Nîmes oraz Korei Południowej.



Następnie dokonaliśmy klasteryzacji ip, z których pochodziły ataki. Cechami, które uwzględniliśmy były: liczba ataków ogółem, liczba użytych unikalnych loginów oraz to ile procent z użytych loginów znajduje się w pliku “10-million-password-list-top-100000.txt”. Algorytm, którego użyliśmy to Bayesowska Mieszanka Gaussowska (ang. Bayesian Gaussian mixture) dla liczby klastrów 3. W ten sposób uzyskaliśmy 3 kategorie: nieszkodliwi, szkodliwi oraz groźni. Użyliśmy tego algorytmu spodziewając się, że klastry mogą być diagonalne i lepiej się sprawdzi niż kmeans. Poniższy wykres prezentuje klastry w przestrzenii trówymiarowej. Żółty kolor oznacza użytkowników nieszkodliwych, morski szkodliwych, a granatowy groźnych.



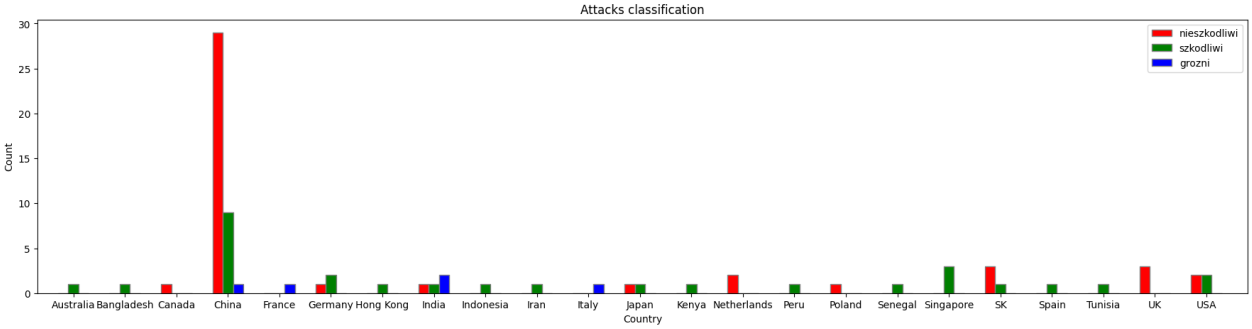

Dla różnych typów ataków zbadaliśmy także podstawowe statystyki opisowe.

Nieszkodliwi średnio atakowali 32 razy (max 105, min 1), używali średnio 2 unikalnych loginów, nie więcej niż 7, a ich celność średnia wynosiła 76% (min 0%)

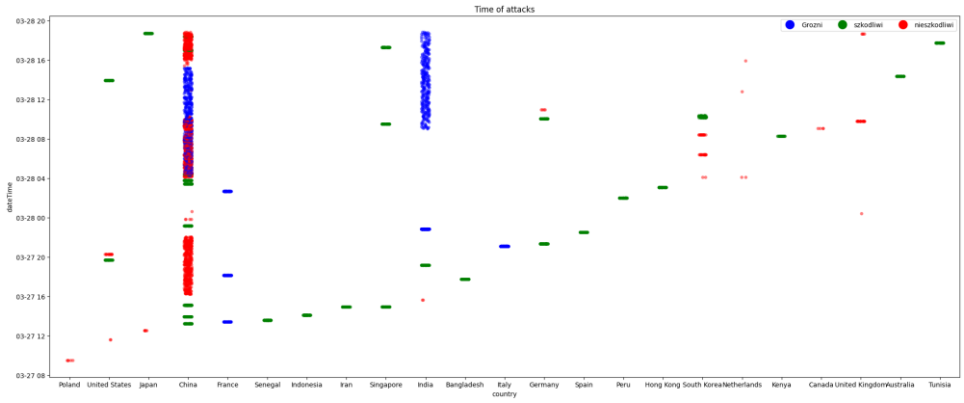
Szkodliwi co prawda mieli mniejszą celność - 60%, ale minimalna aż (42%), oddawali średnio 49 ataków, a ilość unikalnych loginów wynosiła pomiędzy 12 a 28.

Groźni oddawali ponad 148 ataków (cały czas mówimy tu o 32 analizowanych godzinach), średnio używali 40 różnych unikalnych loginów oraz minimalna celność wynosiła 50%, średnio 66%.

Stworzyliśmy także wykres, który obrazuje liczbę ip danej kategorii w poszczególnych państwach.



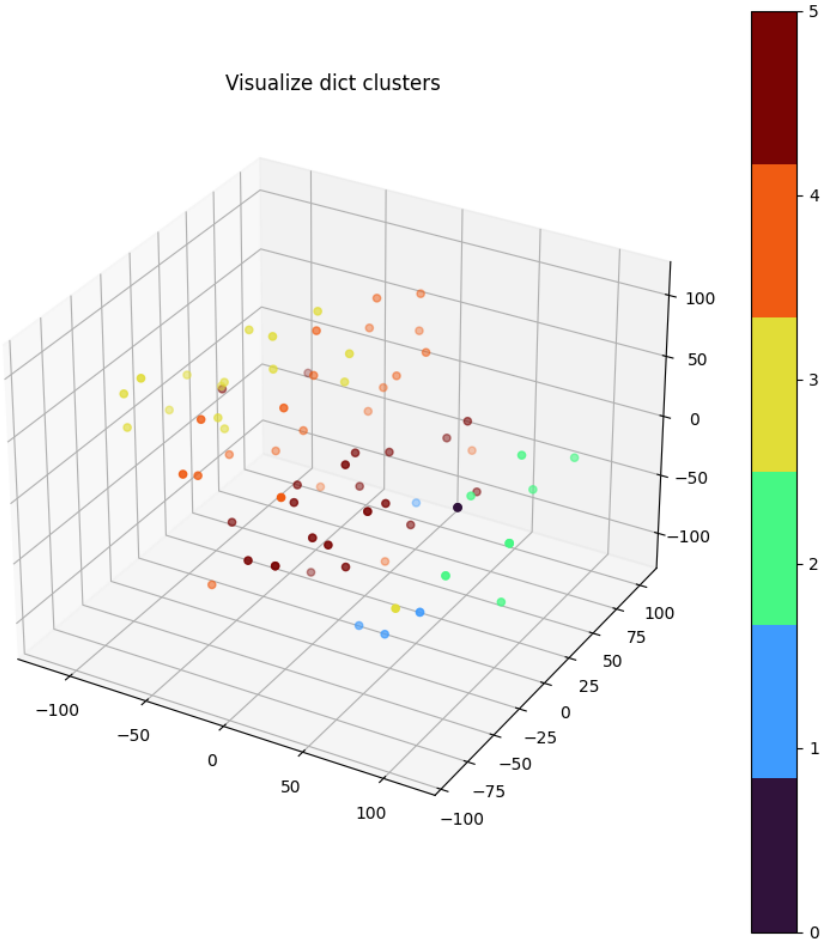

Jak widać najwięcej unikalnych ip pochodzi z Chin, ale dominowały tam ataki zaklasyfikowane jako “nieszkodliwe”. Najgroźniejsze ataki pochodziły z Chin, ale też Francji, Indii oraz Włoch. Co ciekawe ataki z Francji powtórzyły się kilka razy i wszystkie pochodziły z grupy “groźnej”, także zmasowany atak z indyjskiego Bengaluru był z grupy “groźnej” (625 ataków, 14.4% wszystkich prób logowań). Udział każdej kategorii we wszystkich atakach jest niemal równy. Do dyskusji pozostaje to, czy to przez użycie konkretnego algorytmu czy to zbieg okoliczności. Skłaniamy się ku drugiej opcji, ponieważ naszym zdaniem w oparciu o statystyki opisowe oraz wizualizacje podział wydaje się uzasadniony, natomiast już kwestią przypadku jest to, że adresów ip, które atakują więcej, jest mniej niż tych, które atakują mało. Poniżej jeszcze wykresy, które reprezentują udział poszczególnych grup ataków we wszystkich atakach, oraz ilość ataków danej kategorii ze względu na państwa w ujęciu czasowym.

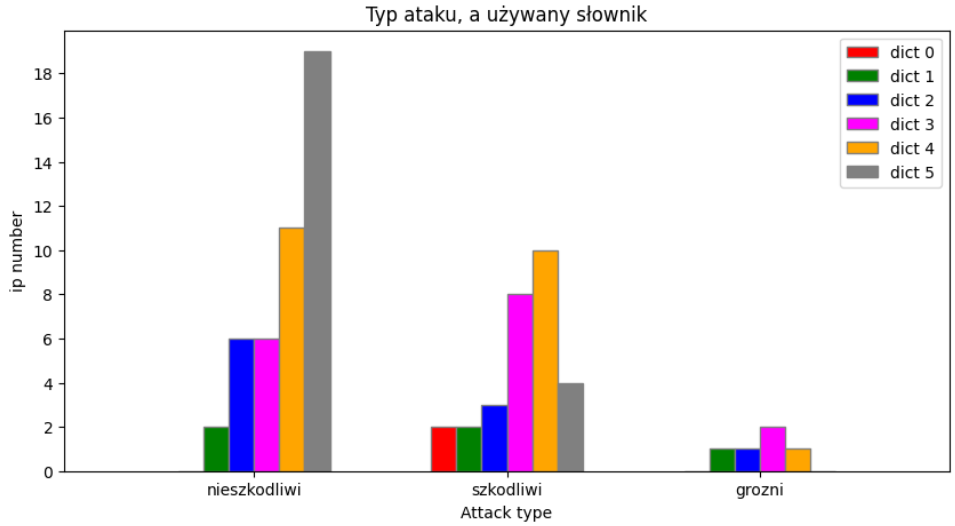

Idąc dalej dokonaliśmy analizy używanych loginów przez poszczególne ip. Wydzieliliśmy zestawy słów, które używały poszczególne adresy, obliczyliśmy odległość Levenstheina każdego zestawu od każdego (jako średnia odległość z sumy powstałej na skutek dodawania minimalnych odległości pomiędzy słowami z pierwszego zestawu, a słowami z drugiego zestawu) tworząc macierz dystansu, do której dołożyliśmy jeszcze jedną kolumnę - ilość unikalnych używanych słów. Pominęliśmy aspekt tego, czy dany “użytkownik” częściej używa jakiegoś słowa, tylko skupiliśmy się na ogólnej zbiorowości. Tutaj znowu stanęliśmy przed problemem wyboru algorytmu do klasteryzacji, tym razem zdecydowaliśmy się na klasteryzację spektralną (ang. Spectral clustering). Stwierdziliśmy, że będzie to dobry wybór dla danych z wieloma cechami, ponadto łatwiej było nam ustalić oczekiwaną liczbę klastrów, niż hyper-parametry algorytmu DBSCAN czy OPTICS. Przed przystąpieniem do klasteryzacji postanowiliśmy jeszcze znormalizować wartości odległości Levenstheina oraz zlogarytmować i znormalizować ilość unikalnych używanych słów. W ten sposób wyodrębniliśmy 6 słowników:

- słownik 0 składający się z jednego słowa - **wqmarlduiqkmgs** (co oznacza, że wszyscy jego użytkownicy używali tylko tego jednego słowa do prób logowania),
- słownik 1 składający się z następujących wyrazów: **['temp1', 'sap_adm01', 'Admin', 'admin', 'centos']**, można powiedzieć administratorskich,
- słownik 2 składający 35 unikalnych wyrazów takich jak: **['culture', 'zefeng', 'ubnt', 'as852', nan, 'q', 'jenkins', 'consul2', 'debianuser', 'uth']**,
- słownik 3 składający się 189 loginów, takich jak: **['zabbix', 'imapuser', 'hp', 'db1', 'chen', 'elastic', 'hxx', 'ubnt', 'odoo15', 'webadmin']**,
- słownik 4 składający się z 98 wyrazów, takich jak: **['ubnt', 'odoo15', 'akash', 'scan', 'upload', 'docker', 'teste', 'ali', 'github', 'john'],**
- słownik 5 składający się z dwóch słów: **['ubnt', 'root'],** przy czym słowo **ubnt** było używane tylko przez jedno ip razem ze słowem **root**, pozostałe używały słownika jednowyrazowego z wyrazem **root.**

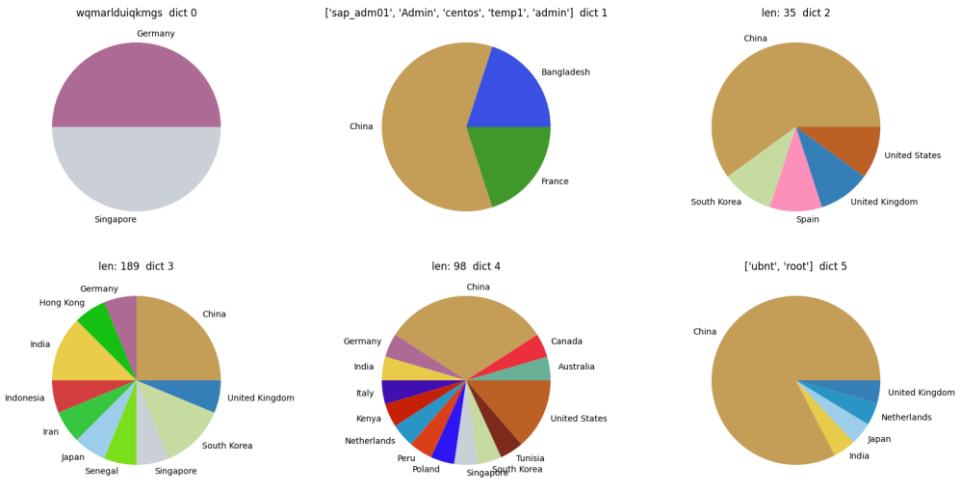
Ponadto sprawdziliśmy, że słownik 2 zawiera 27 słów (77%), których nie ma w słowniku 3 oraz 30 słów, których nie ma w słowniku 4. W słowniku 4 jest 20 słów, których nie ma w słowniku 3. Uważamy, że nie istnieje potrzeba wydzielania słownika 4, być może zarówno użytkownicy 3 jak i 4 mieli ten sam słownik, tylko wybierali słowa z różnym prawdopodobieństwem. Należałoby też rozważyć redukcję słownika 0. Dalsze wizualizacje wykonaliśmy jednak o ten pierwotny podział. Spróbowaliśmy zwizualizować ten podział w przestrzeni trójwymiarowej za pomocą probalistycznego podejścia t-SNE. Niestety spośród do tej pory trzech przytoczonych algorytmów, wykazujemy najmniejsze zrozumienie, wobec tego. Poniższy wykres ilustruje to co udało się nam osiągnąć.



Ponadto zbadaliśmy, ile adresów ip z poszczególnej kategorii używało jakiego słownika. Jak widać groźni w ogóle nie korzystali ze słowników 0 i 5, z powodu niewielkiej liczby słów. Jeden korzystał ze słownika 1 – administracyjnego, jest ze słownika drugiego oraz dwóch ze słownika 3, który zawiera najwięcej słów. Wśród szkodliwych popularnością cieszył się słownik 4 jak i 3 (być może to jeden i ten sam), zaś wśród nieszkodliwych dominował atak za pomocą **root**. Być może, wbrew przyjętemu przez założeniu, że podawane hasło jest dokładnie takie samo jak login, to właśnie ten typ ataków jest najbardziej szkodliwy.




Zamieszczamy jeszcze serię wykresów kołowych, które ilustrują związek pomiędzy użyciem danego słownika, a państwem, z którego nastąpiła próba logowania.




Na podstawie przeprowadzonej analizy wciągamy wniosek, że najniebezpieczniejsze ataki pochodziły z Chin oraz Indii, nie tylko ze względu na ilość ataków, ale także tego, że stamtąd było najwięcej groźnych “użytkowników” oraz, że korzystali ze słownika 3, który zawierał najwięcej unikatowych słów.

Na koniec chcemy jeszcze wspomnieć o dwóch rzeczach - analizując adresy ip, napotkaliśmy na blok podobnych adresów, które zaczynały się **61.177.173.XXX** lub **61.177.172.XXX**, te ataki pochodziły z Chin, być może z jednej sieci. Podczas tych ataków był używany słonik 5 – **root,** zostały zaklasyfikowane jako nieszkodliwe (być może jak już wyżej napisano - niesłusznie)

Wśród słów, których użyto do ataku znalazł się polsko brzmiący **kiosk**, co pewnie jest kompletnym przypadkiem, ponieważ to słowo funkcjonuje w wielu językach (m.in. francuskim oraz angielskim), ataki te pochodziły z Honk Kongu, gdzie jednym z języków obok chińskiego i mandaryńskiego jest angielski. Być może ofiarami takich ataków są małe budki sklepowe tudzież właśnie kioski, które niekoniecznie muszą mieć dobrze zabezpieczone urządzenia sieciowe. Ponadto niektóre słowa nawiązują do systemów operacyjny **centos, ubnt, debianuser,** generalnie do kultury informatyków **oracleuser, accesDBuser, appuser, pi, hp, docker, rasperbberypi, nvidia, mysql**, ale pojawiły się też związane z gameingiem **minecraftuser, arkserver, minecraft, csgo, teamspeak** – czy nazwy gier, serwerów do gier sieciowych czy popularnego komunikatora dla graczy.

Zamieszczamy wybrane fragmenty kodu, które dokumentują wykonaną pracę.

```
[ ]: # datetime(year, month, day, hour, minute, second)
     countries = pd.DataFrame()
     b = datetime(2023, 3, 27, 9, 0, 0)
     for i in range(0,34):
         sliceDf = df[(df.dateTime>b+timedelta(hours=i)) & (df.dateTime<b+timedelta(hours=i+1))]
         sliceDf = pd.concat([sliceDf.groupby("country").count()["user"]], keys=[f"{b+timedelta(hours=i)}"])
         countries = pd.concat([countries, sliceDf])
     countries.index = pd.MultiIndex.from_tuples(countries.index, names=('fromDateTime', 'country'))
     countries.columns = ["attack_count"]
     display(countries)
```

```
[ ]: fig, ax = plt.subplots(1,1,figsize=(25,10))
     palette = sns.color_palette(cc.glasbey, n_colors=45)
     sns.stripplot(data=df.dropna(), x="country", y="dateTime", hue="city", ax=ax, palette=palette, alpha=0.5)
     ax.legend(ncols=3)
     ax.set_title("Time of attacks")
```

```
: def get_location(ip, country, city):
      ip_address = ip
      response = requests.get("https://ipgeolocation.abstractapi.com/v1/?api_key=066c7d0f4e024bdd946b83581480e8ae"+'&
      country[ip_address] = response.get("country")
      city[ip_address] = response.get("city")
      return country, city
```

```
: fig = plt.figure(figsize=(10,10))
  ax = fig.add_subplot(projection='3d')
  ax.scatter(xs=ipDf.attackCount, ys=ipDf.usedUser, zs=ipDf.accuracy)
  ax.set_xlabel("attack count")
  ax.set_ylabel("unique user")
  ax.set_zlabel("accuracy (compared with file of the passwords)")
  ax.set_box_aspect(aspect=None, zoom=0.8)
  ax.set_title("Ips describe in 3D")
```

```
: from sklearn.mixture import BayesianGaussianMixture
  ipDf = ipDf.dropna()
  ipDf["cluster"] = BayesianGaussianMixture(n_components=3, covariance_type='diag', n_init = 100,
                              weight_concentration_prior_type= "dirichlet_distribution",
                              init_params="random_from_data", random_state=0).fit_predict(ipDf.iloc[:,1:])
  words.columns = ["set"]
```

```
|: import Levenshtein
   for index1, row1 in words.iterrows():
       values = []
       for index2, row2 in words.iterrows():
           suma = 0
           for word1 in row1.set:
               distances = []

               for word2 in row2.set:
                   distances.append(Levenshtein.distance(str(word1), str(word2)))
               suma += min(distances)

           suma /= len(row1.set)
           if suma == 0:
               suma = np.nan

           values.append(suma)
       words[index1] = values
```

```
|: words = words.fillna(0)
   words["sizeD"] = [len(x) for x in words.set]
   words["sizeD"] = [np.emath.logn(3,x) for x in words.sizeD]
   words.iloc[:,1:]=(words.iloc[:,1:]-words.iloc[:,1:].min())/(words.iloc[:,1:].max()-words.iloc[:,1:].min())
```

```
|: from sklearn.cluster import SpectralClustering
   words["cluster"] = SpectralClustering(n_clusters=6, assign_labels='cluster_qr', random_state=0).\
                       fit_predict(words.iloc[:,1:])
```

```
|: from sklearn.manifold import TSNE
   from matplotlib import cm
   from matplotlib.colors import ListedColormap, LinearSegmentedColormap

   tsne = TSNE(n_components=3, perplexity=10, init="pca", learning_rate="auto", random_state=0)
   tsnePlot = tsne.fit_transform(words.iloc[:,1:82])

   fig = plt.figure(figsize=(10,10))
   ax = fig.add_subplot(projection='3d')
   p = ax.scatter(xs=tsnePlot[:,0], ys=tsnePlot[:,1], zs=tsnePlot[:,2], c=words.cluster, cmap=cm.get_cmap('turbo', 6))
   ax.set_title("Visualize dict clusters")
   fig.colorbar(p)
```