iCEL_workshop_hands_on_session

April 2, 2024

[1]:
```
import pandas as pd
from collections import defaultdict
import numpy as np
import copy
import re
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
################ DOWNLOAD #################
#### http://tinyurl.com/bitsandscripts ####
###########################################
wordlist = '../rawData/English_word_list_10k.txt'
```

[3]:
```
df = pd.read_csv(wordlist, header=None, names=['Word'])
word_list = [list(w) for w in list(df.Word)]
```

[4]:
```
df
```

[4]:

	Word
0	THE
1	OF
2	AND
3	TO
4	IN
…	…
9995	PROJECTING
9996	MODERNIZATION
9997	STOCKHOLM
9998	ORNAMENT
9999	OVERT

[10000 rows x 1 columns]

[ ]:
```

```

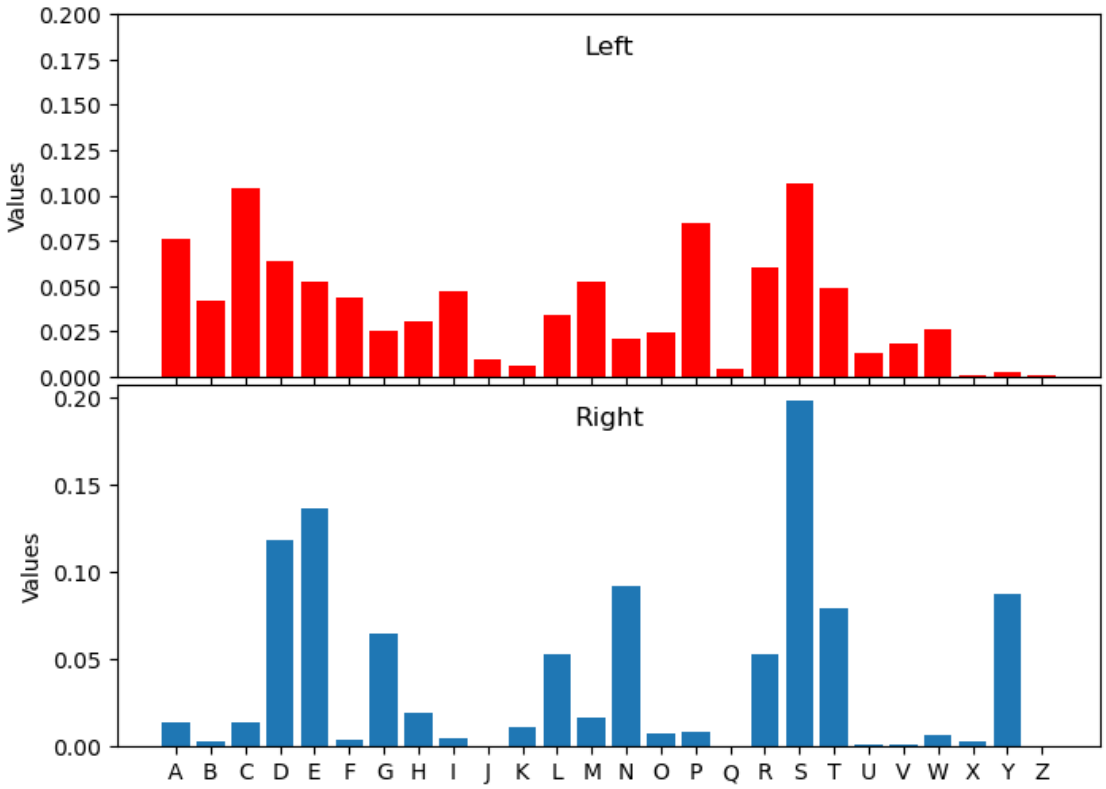
[ ]:
```

```

[5]:
```
left = defaultdict(int)
right = defaultdict(int)

for word in word_list:
    left[word[0]] +=1
    right[word[-1]] +=1
```

[6]:
```
left = {k: v for k, v in sorted(left.items(), key=lambda item: item[0],␣
 ↪reverse=False)}
right = {k: v for k, v in sorted(right.items(), key=lambda item: item[0],␣
 ↪reverse=False)}

alphabet = list(left.keys())
pleft = list(left.values()) / np.sum(list(left.values()))
pright = list(right.values()) / np.sum( list(right.values()) )
```

[ ]:
```

```

[7]:
```
fig, axs = plt.subplots(2, 1, figsize=(8, 6), sharex=True)

axs[0].bar( alphabet, pleft, color='red' )
axs[0].set_title('Left', y=0.85)
axs[0].set_ylabel('Values')
axs[0].set_ylim([0, 0.2])

# Plotting the second bar plot
axs[1].bar(alphabet, pright )
axs[1].set_title('Right', y=0.85)
axs[1].set_ylabel('Values')
axs[0].set_ylim([0, 0.2])

plt.subplots_adjust(hspace=0.02)
```



[8]:
```
Hl = -np.sum(pleft*np.log2(pleft))
Hr = -np.sum(pright*np.log2(pright))
```

[9]:
```
edelH = (Hl-Hr)/( (Hl+Hr)/2)
```

[10]:
```
edelH
```

[10]: 0.15975699541876504

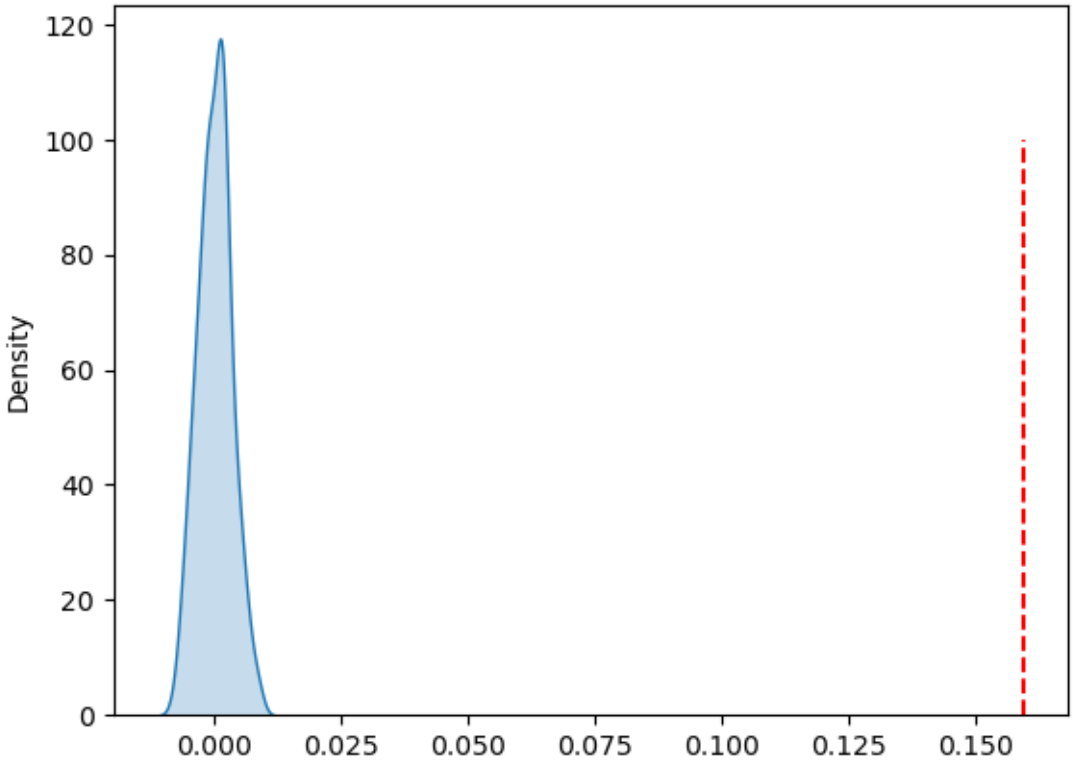
Randomization

[11]:
```
word_list_rand = copy.deepcopy(word_list)
num_trials=200

randH = []
for _ in range(0, num_trials):
    templ  =defaultdict(int)
    tempr  =defaultdict(int)
    for rword in word_list_rand:
        np.random.shuffle(rword)
        templ[rword[0]] +=1
        tempr[rword[-1]] +=1
```

```
    templ = {k: v for k, v in sorted(templ.items())}
    tempr = {k: v for k, v in sorted(tempr.items())}

    ptl = list(templ.values()) / np.sum(list(templ.values()))
    ptr = list(tempr.values()) / np.sum(list(tempr.values()))

    htl = -np.sum( ptl*np.log2(ptl) )
    htr = -np.sum( ptr*np.log2(ptr) )
    rdelH = (htl-htr) / ((htl+htr)/2)
    randH.append(rdelH)
```

[12]:
```
h = sns.kdeplot(randH, fill=True)
h.plot([edelH, edelH], [0, 100], '--r')
```

[12]: [<matplotlib.lines.Line2D at 0x74b3b1966fd0>]



[13]:
```
[np.mean(randH), np.std(randH)]
```

[13]: [0.00011079976204593096, 0.0032659104286897264]

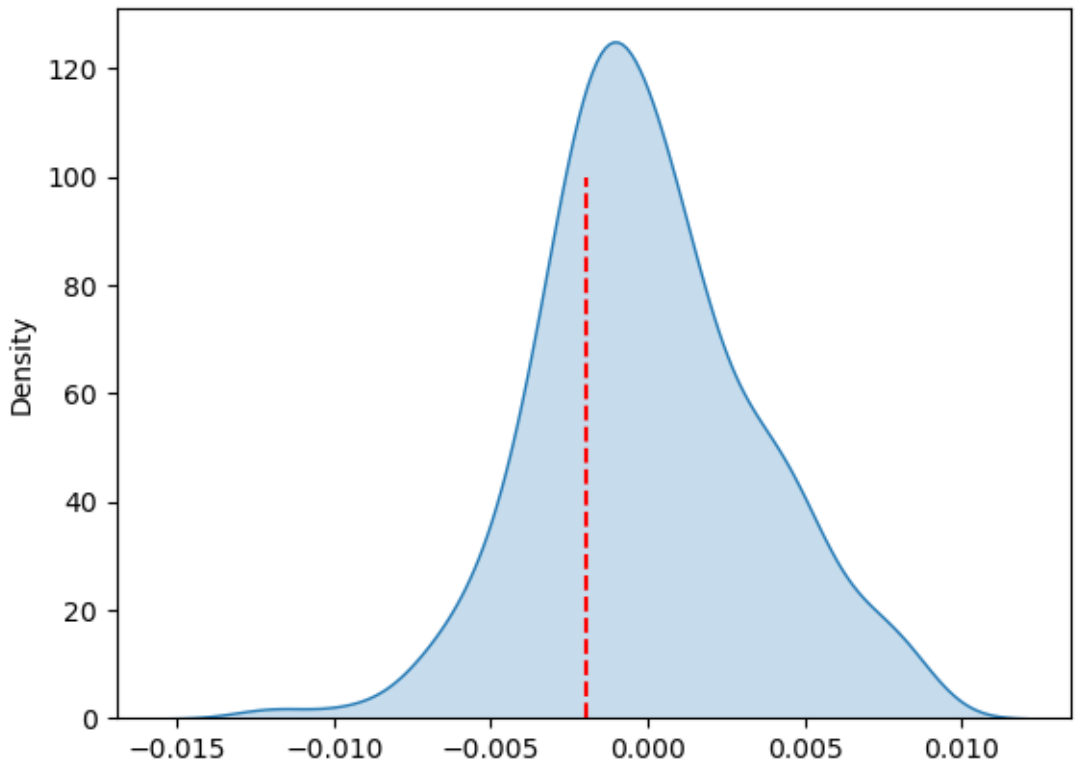
[ ]:

Bootstrap

[14]:
```
rindx_list = np.random.randint(0, len(word_list), len(word_list))

word_list_boot = [word_list[rindx] for rindx in rindx_list]
num_trials=200

bootH = []
for _ in range(0, num_trials):
    templ  =defaultdict(int)
    tempr  =defaultdict(int)
    for bword in word_list_boot:
        np.random.shuffle(bword)
        templ[bword[0]] +=1
        tempr[bword[-1]] +=1
    templ = {k: v for k, v in sorted(templ.items())}
    tempr = {k: v for k, v in sorted(tempr.items())}

    ptl = list(templ.values()) / np.sum(list(templ.values()))
    ptr = list(tempr.values()) / np.sum(list(tempr.values()))

    htl = -np.sum( ptl*np.log2(ptl) )
    htr = -np.sum( ptr*np.log2(ptr) )
    bdelH = (htl-htr) / ((htl+htr)/2)
    bootH.append(bdelH)
```

[15]:
```
h = sns.kdeplot(bootH, fill=True)
h.plot([bdelH, bdelH], [0, 100], '--r')
```

[15]:
```
[<matplotlib.lines.Line2D at 0x74b3a99644f0>]
```



[ ]:

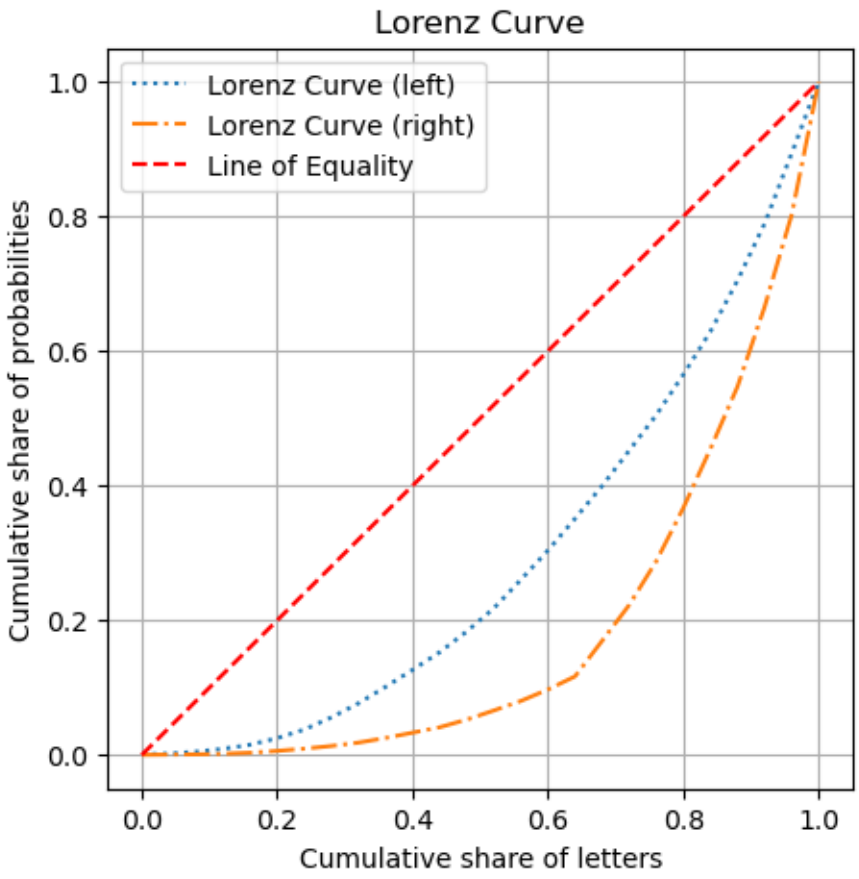
Gini index

[16]:
```
sorted_prob_l = np.sort(pleft)
cumsum_prob_l = np.cumsum(sorted_prob_l)

sorted_prob_r = np.sort(pright)
cumsum_prob_r = np.cumsum(sorted_prob_r)
```

[17]:
```
uniform_distribution = np.linspace(0, 1, len(pleft))


plt.figure(figsize=(5, 5))
plt.plot(uniform_distribution, cumsum_prob_l, ':', label='Lorenz Curve (left)')
plt.plot(uniform_distribution, cumsum_prob_r, '-.', label='Lorenz Curve␣
 ↪(right)')
plt.plot(uniform_distribution, uniform_distribution, '--r', label='Line of␣
 ↪Equality')
plt.legend()
plt.grid()
plt.title('Lorenz Curve')
```

```
plt.xlabel('Cumulative share of letters')
plt.ylabel('Cumulative share of probabilities')
```

[17]:
```
Text(0, 0.5, 'Cumulative share of probabilities')
```



[18]:
```
cumsum_population = np.cumsum(np.ones_like(uniform_distribution) /␣
 ↪len(uniform_distribution))

lorenz_area_l = np.trapz(cumsum_prob_l, cumsum_population)
lorenz_area_r = np.trapz(cumsum_prob_r, cumsum_population)

gini_index_l = 1 - 2 * lorenz_area_l
gini_index_r = 1 - 2 * lorenz_area_r
```

[19]:
```
[gini_index_l - gini_index_r]
```

[19]:
```
[-0.21384999999999987]
```

[20]:
```
(gini_index_l - gini_index_r)/ ( (gini_index_l + gini_index_r)/2)
```

[20]: -0.39540878915347727

[ ]: