

## Features Exploration

```
1. User Language
2. Profile Creation Timestamp
3. Is Profile View Size Customized?
4. Username
5. Profile Verification Status

In [103].
# Imports and set up
### FOR MAC OSX USERS ###
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
#####

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import KNNImputer
from imblearn.under_sampling import RandomUnderSampler
from sklearn.datasets import make_classification

from utils.missing_values_filler import MissingValuesFiller

/usr/local/Caskroom/miniconda/base/envs/dsp/lib/python3.5/site-packages/matplotlib/_init.py:1405: UserWarning:
  This call to matplotlib.use() has no effect because the backend has already
  been chosen: matplotlib.use() must be called "before" pylab, matplotlib.pyplot,
  or matplotlib.backends is imported for the first time.

warnings.warn(_use_error_msg)

In [104].
# Load train dataset
raw_df = pd.read_csv('src_data/train.csv')
```

## User Language

```
In [105].
# Histogram
raw_df['User Language'].value_counts().plot(kind='bar')
plt.xlabel('User Language')
plt.ylabel('Count')
plt.title('Initial Histogram')
plt.show()

print("Not english count: {}".format(len(raw_df[raw_df['User Language'] != 'en'])))

# Barplot
fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='User Language', y='Num of Profile Likes', data=raw_df, estimator=np.mean).set_title('NOL mean')
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='User Language', y='Num of Profile Likes', data=raw_df, estimator=np.median).set_title('NOL me')
plt.show()

# Separate 'en' and non-'en' records
lang_df = raw_df.copy(deep=True)
lang_df['User Language'] = raw_df['User Language'].apply(lambda val: 'en' if val == 'en' else 'none-en')

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='User Language', y='Num of Profile Likes', data=lang_df, estimator=np.mean).set_title('NOL mea')
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='User Language', y='Num of Profile Likes', data=lang_df, estimator=np.median).set_title('NOL m')
plt.show()

Initial Histogram
Count
5000
4000
3000
2000
1000
0
g t h i a d e s s i z e t e x t u s e r l a n g u a g e
User Language

Not english count: 2179

NOL mean
Num of Profile Likes
35000
30000
25000
20000
15000
10000
5000
0
en pt fr it es ja de zh-cn r i sr f k u ru en-gb hu id cs pl ch sk sv ef th zh-TW fi da sr uk
User Language

NOL median
Num of Profile Likes
20000
17500
15000
12500
10000
7500
5000
2500
0
en pt fr it es ja de zh-cn r i sr f k u ru en-gb hu id cs pl ch sk sv ef th zh-TW fi da sr uk
User Language

NOL mean
Num of Profile Likes
6000
5000
4000
3000
2000
1000
0
en none-en
User Language

NOL median
Num of Profile Likes
1600
1400
1200
1000
800
600
400
200
0
en none-en
User Language
```

## Profile Category

```
In [106].
# Clean
profile_df = raw_df.copy(deep=True)
profile_df['Profile Category'] = profile_df['Profile Category'].replace(r'''\s+$', 'unknown', regex=True)

# Histogram
profile_df['Profile Category'].value_counts().plot(kind='bar')
plt.xlabel('Profile Category')
plt.ylabel('Count')
plt.title('Initial Histogram')
plt.show()

# Barplots
fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Profile Category', y='Num of Profile Likes', data=profile_df, estimator=np.mean).set_title('N')
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Profile Category', y='Num of Profile Likes', data=profile_df, estimator=np.median).set_title('N')
plt.show()

# Missing values filler
mwf = MissingValuesFiller()

# K = 1
profile_df = raw_df.copy(deep=True)
profile_df['Profile Category'] = profile_df['Profile Category'].replace(r'''\s+$', 'unknown', regex=True)
profile_df = mwf.fill_missing_values(profile_df, 'Profile Category', 'unknown', 'Num of Status Updates', 1)

profile_df['Profile Category'].value_counts().plot(kind='bar')
plt.xlabel('Profile Category')
plt.ylabel('Count')
plt.title('Post missing values filled (k=1)')
plt.show()

# K = 2
profile_df = raw_df.copy(deep=True)
profile_df['Profile Category'] = profile_df['Profile Category'].replace(r'''\s+$', 'unknown', regex=True)
profile_df = mwf.fill_missing_values(profile_df, 'Profile Category', 'unknown', 'Num of Status Updates', 2)

profile_df['Profile Category'].value_counts().plot(kind='bar')
plt.xlabel('Profile Category')
plt.ylabel('Count')
plt.title('Post missing values filled (k=2)')
plt.show()

# K = 5
profile_df = raw_df.copy(deep=True)
profile_df['Profile Category'] = profile_df['Profile Category'].replace(r'''\s+$', 'unknown', regex=True)
profile_df = mwf.fill_missing_values(profile_df, 'Profile Category', 'unknown', 'Num of Status Updates', 5)

profile_df['Profile Category'].value_counts().plot(kind='bar')
plt.xlabel('Profile Category')
plt.ylabel('Count')
plt.title('Post missing values filled (k=5)')
plt.show()

# K = 20
profile_df = raw_df.copy(deep=True)
profile_df['Profile Category'] = profile_df['Profile Category'].replace(r'''\s+$', 'unknown', regex=True)
profile_df = mwf.fill_missing_values(profile_df, 'Profile Category', 'unknown', 'Num of Status Updates', 20)

profile_df['Profile Category'].value_counts().plot(kind='bar')
plt.xlabel('Profile Category')
plt.ylabel('Count')
plt.title('Post missing values filled (k=20)')
plt.show()

Initial Histogram
Count
3500
3000
2500
2000
1500
1000
500
0
unknown business celebrity government
Profile Category

NOL mean
Num of Profile Likes
10000
8000
6000
4000
2000
0
business unknown celebrity government
Profile Category

NOL median
Num of Profile Likes
3000
2500
2000
1500
1000
500
0
business unknown celebrity government
Profile Category

Post missing values filled (k=1)
Count
3500
3000
2500
2000
1500
1000
500
0
business government celebrity
Profile Category

Post missing values filled (k=2)
Count
3000
2500
2000
1500
1000
500
0
business celebrity government
Profile Category

Post missing values filled (k=5)
Count
4000
3500
3000
2500
2000
1500
1000
500
0
celebrity business government
Profile Category

Post missing values filled (k=20)
Count
4000
3500
3000
2500
2000
1500
1000
500
0
celebrity business government
Profile Category
```

## Profile Verification Status

```
In [107].
# Load a copy
profile_df = raw_df.copy(deep=True)

# Histogram
profile_df['Profile Verification Status'].value_counts().plot(kind='bar')
plt.xlabel('Profile Verification Status')
plt.ylabel('Count')
plt.title('Initial Histogram')
plt.show()

# Barplot
fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Profile Verification Status', y='Num of Profile Likes', data=profile_df, estimator=np.median).set_title('NOL median')
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Profile Verification Status', y='Num of Profile Likes', data=profile_df, estimator=np.mean).set_title('NOL mean')
plt.show()

# Clean pending
print("Pending" is the exact same as 'Not verified' when we want to predict the 'Profile Num of Likes' ...)
profile_df['Profile Verification Status'] = profile_df['Profile Verification Status'].replace('Pending', 'Not verified')

# Histogram
profile_df['Profile Verification Status'].value_counts().plot(kind='bar')
plt.xlabel('Profile Verification Status')
plt.ylabel('Count')
plt.title('Initial Histogram (merged pending)')
plt.show()

# Barplot
fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Profile Verification Status', y='Num of Profile Likes', data=profile_df, estimator=np.median).set_title('NOL median')
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Profile Verification Status', y='Num of Profile Likes', data=profile_df, estimator=np.mean).set_title('NOL mean')
plt.show()

Initial Histogram
Count
5000
4000
3000
2000
1000
0
Verified Not verified Pending
Profile Verification Status

NOL median (before merge)
Num of Profile Likes
3500
3000
2500
2000
1500
1000
500
0
Verified Not verified Pending
Profile Verification Status

NOL mean (before merge)
Num of Profile Likes
7000
6000
5000
4000
3000
2000
1000
0
Verified Not verified Pending
Profile Verification Status

'Pending' is the exact same as 'Not verified' when we want to predict the 'Profile Num of Likes' ...

Initial Histogram (merged pending)
Count
5000
4000
3000
2000
1000
0
Verified Not verified
Profile Verification Status

NOL median (after merge)
Num of Profile Likes
2000
1500
1000
500
0
Verified Not verified
Profile Verification Status

NOL mean (after merge)
Num of Profile Likes
7000
6000
5000
4000
3000
2000
1000
0
Verified Not verified
Profile Verification Status
```

## Is Profile View Customized

```
In [108].
# Load a copy
profile_df = raw_df.copy(deep=True)

# Histogram
profile_df['Is Profile View Size Customized?'].value_counts().plot(kind='bar')
plt.xlabel('Is Profile View Size Customized?')
plt.ylabel('Count')
plt.title('Initial Histogram')
plt.show()

# Barplot
fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Is Profile View Size Customized?', y='Num of Profile Likes', data=profile_df, estimator=np.mean).set_title('NOL mean')
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.barplot(x='Is Profile View Size Customized?', y='Num of Profile Likes', data=profile_df, estimator=np.median).set_title('NOL median')
plt.show()

Initial Histogram
Count
7000
6000
5000
4000
3000
2000
1000
0
False True
Is Profile View Size Customized?

NOL median
Num of Profile Likes
1600
1400
1200
1000
800
600
400
200
0
False True
Is Profile View Size Customized?

NOL mean
Num of Profile Likes
6000
5000
4000
3000
2000
1000
0
False True
Is Profile View Size Customized?
```

## Profile Creation Timestamp

```
In [109].
# Load a copy
field_name = 'Profile Creation Timestamp'
profile_df = raw_df.copy(deep=True)

# Validate it all in UTC
for i, v in profile_df[field_name].items():
    if not v < "2000-01-01":
        print(v)

# Validate 24h based
if not v < "24":
    print(v)

# Build datetime col
profile_df['PCT_dt'] = pd.to_datetime(profile_df[field_name])

# Calc min and max dates
min_date = min(profile_df['PCT_dt'])
max_date = max(profile_df['PCT_dt'])

# Convert to int column based on years
profile_df['PCT_total_years'] = (profile_df['PCT_dt'] - min_date).astype('timedelta64[Y]')

# Convert to int column based on six-months
profile_df['PCT_total_six_months'] = (profile_df['PCT_dt'] - min_date).astype('timedelta64[M]') / 6

# Convert to int column based on days
profile_df['PCT_total_days'] = (profile_df['PCT_dt'] - min_date).astype('timedelta64[D]')

# Convert to int column based on hours
profile_df['PCT_total_hours'] = (profile_df['PCT_dt'] - min_date).astype('timedelta64[h]')

# Convert to int column based on seconds
profile_df['PCT_total_seconds'] = (profile_df['PCT_dt'] - min_date).astype('timedelta64[s]')

# Verify
# Following is the same between the max & min dates
# 3958 days, 10 hours, 35 minutes, and 46 seconds
# 3,995,441.5 days
# 95,890,596 hours
# 5,753,435.77 minutes
# 345,206,146 seconds
# print(min_date)
# print(max_date)
# print(min(profile_df['PCT_total_hours']))
# print(max(profile_df['PCT_total_hours']))
# print(max(profile_df['PCT_total_hours']) - max_date.astype('timedelta64[h]'))

# Scatterplot
fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.scatterplot(data=profile_df, x='PCT_total_years', y='Num of Profile Likes').set_title("Scatterplot (year)")
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.scatterplot(data=profile_df, x='PCT_total_six_months', y='Num of Profile Likes').set_title("Scatterplot (mon)")
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.scatterplot(data=profile_df, x='PCT_total_months', y='Num of Profile Likes').set_title("Scatterplot (mon)")
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.scatterplot(data=profile_df, x='PCT_total_days', y='Num of Profile Likes').set_title("Scatterplot (days)")
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.scatterplot(data=profile_df, x='PCT_total_hours', y='Num of Profile Likes').set_title("Scatterplot (hours)")
plt.show()
```



```
fig, ax = plt.subplots()
fig.set_size_inches(11, 8.5)
sns.scatterplot(data=profile_df, x="FCT_total_seconds", y="Num of Profile Likes").set_title("Scatterplot (see
plt.show()
```

