CTR - With PCA

February 15, 2021

[127]:
```
import pandas as pd
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt
```

[128]:
```
cwd = os.getcwd()
```

[129]:
```
aug_test=pd.read_csv(cwd+'//archive//aug_test.csv')
aug_train=pd.read_csv(cwd+'//archive//aug_train.csv')
sample_submission=pd.read_csv(cwd+'//archive//sample_submission.csv')
```

This data taken from kaggle is HR data on data scientist, its purpose is to train a model to detect sucessfull candidates for a job position. I am goin to be Pre Processing the data performing, Cleaning, Transforming and Reduction.

At the end of this Project i will perform Principle Component Analysis to evaluate the features.

1 Cleaning

1.1 1 Missing Data - Atrributes

[130]:
```
# for each column what percentage of the data is missing ?
#aug_teat
aug_test_missing_attpct = aug_test.isna().sum()/aug_test.shape[0]
print(aug_test_missing_attpct)
print(aug_test_missing_attpct[aug_test_missing_attpct >0.5])
```

```
enrollee_id              0.000000
city                     0.000000
city_development_index   0.000000
gender                   0.238610
relevent_experience      0.000000
enrolled_university      0.014561
education_level          0.024425
major_discipline         0.146548
experience               0.002349
company_size             0.292156
company_type             0.297792
```

```
last_new_job              0.018788
training_hours            0.000000
dtype: float64
Series([], dtype: float64)
```

[131]:
```
#aug_train
aug_train_missing_attpct = aug_train.isna().sum()/aug_train.shape[0]
at_o50pct = aug_train_missing_attpct[aug_train_missing_attpct>0.5]
print(aug_train_missing_attpct)
print(at_o50pct)
```

```
enrollee_id               0.000000
city                      0.000000
city_development_index    0.000000
gender                    0.235306
relevent_experience       0.000000
enrolled_university       0.020148
education_level           0.024011
major_discipline          0.146832
experience                0.003393
company_size              0.309949
company_type              0.320493
last_new_job              0.022080
training_hours            0.000000
target                    0.000000
dtype: float64
Series([], dtype: float64)
```

[132]:
```
#sample_submission
sample_submission.isna().sum()
```

[132]:
```
enrollee_id    0
target         0
dtype: int64
```

There was enough data in each attribute to not drop any of the attributes

1.2 2 Missing Data - Instances (rows)

[133]:
```
#aug_test
augtest_missing_instpct = aug_test.isna().sum(axis =1)/aug_test.shape[1]
augtest_o35rows = augtest_missing_instpct[augtest_missing_instpct>=0.35]
aug_test.iloc[augtest_o35rows.index].head()
```

[133]:

	enrollee_id	city	city_development_index	gender	\
53	4742	city_149	0.689	NaN	
125	4417	city_103	0.920	NaN	
200	9462	city_21	0.624	NaN	
260	26325	city_21	0.624	Male	

```
289        12391   city_99                   0.915    NaN

        relevent_experience enrolled_university education_level  \
53   No relevent experience       no_enrollment             NaN
125  No relevent experience       no_enrollment             NaN
200  No relevent experience    Part time course             NaN
260  No relevent experience                 NaN             NaN
289  No relevent experience    Full time course     High School

    major_discipline experience company_size company_type last_new_job  \
53               NaN          2          NaN          NaN          NaN
125              NaN        NaN          NaN          NaN          NaN
200              NaN          1          NaN          NaN        never
260              NaN          3          NaN          NaN        never
289              NaN          3          NaN          NaN          NaN

     training_hours
53              114
125              99
200             204
260              38
289              23
```

Looking at these two rows there is not way to infer the values that are missing, in this case we're better off droping the rows.

[8]:
```
# drop the two rows.
aug_test = aug_test.drop(augtest_o35rows.index,axis=0)
#reset index
aug_test = aug_test.reset_index()
aug_test = aug_test.drop(columns='index')
```

[200]:
```
#aug_train
augtrain_missing_instpct = aug_train.isna().sum(axis=1)/aug_train.shape[1]
augtrain_40rows = augtrain_missing_instpct[augtrain_missing_instpct>=0.35]
aug_train.iloc[augtrain_40rows.index].head()
```

[200]:
```
     enrollee_id      city  city_development_index gender  \
13          5826   city_21                   0.624   Male
64          9572   city_11                   0.550    NaN
69          4830   city_90                   0.698    NaN
135        23947  city_103                   0.920    NaN
153         8241   city_16                   0.910    NaN

         relevent_experience enrolled_university education_level  \
13    No relevent experience                 NaN             NaN
64    No relevent experience    Full time course     High School
69    No relevent experience                 NaN             NaN
```

```
135   No relevent experience       no_enrollment             Phd
153  Has relevent experience       no_enrollment             NaN

    major_discipline experience company_size company_type last_new_job  \
13               NaN          2          NaN          NaN        never
64               NaN          3          NaN          NaN          NaN
69               NaN          2          NaN      Pvt Ltd        never
135             STEM        NaN          NaN          NaN          NaN
153              NaN         11          NaN          NaN            1

     training_hours  target
13               24     0.0
64               98     0.0
69              228     1.0
135              70     0.0
153               4     0.0
```

[10]:
```
#drop rows
aug_train = aug_train.drop(augtrain_40rows.index)
#reset the index
aug_train = aug_train.reset_index()
aug_train = aug_train.drop(columns='index')
```

[11]:
```
#Sample Submission
sample_submission[(sample_submission.isna().sum(axis=1)/sample_submission.
↪shape[1])>0.3].count(axis=1)
```

[11]:
```
Series([], dtype: int64)
```

1.3 3 Recheck Missing Data - Atrributes

1.3.1 aug_test

[12]:
```
#aug_test
(aug_test.isna().sum(axis=0)/aug_test.shape[0]).sort_values(ascending =False)
```

[12]:
```
company_type             0.289449
company_size             0.284696
gender                   0.231464
major_discipline         0.136882
education_level          0.016160
last_new_job             0.013783
enrolled_university      0.009981
experience               0.001901
training_hours           0.000000
relevent_experience      0.000000
city_development_index   0.000000
city                     0.000000
```

```
enrollee_id                0.000000
dtype: float64
```

Enrolled_University

[13]:
```
aug_test['education_level'].unique()
```

[13]:
```
array(['Graduate', 'High School', 'Masters', nan, 'Phd', 'Primary School'],
      dtype=object)
```

[14]:
```
#if they're education level is highschool or less then ill assume they are not enrolled at university.
mask = ((aug_test['education_level']!='Masters')&
        (aug_test['education_level']!='Phd')&
        (aug_test['education_level']!='Graduate')&
        aug_test['education_level'].notnull())
```

[15]:
```
not_enrolled = aug_test[(aug_test['enrolled_university'].isna())& mask]
not_enrolled
```

[15]:
```
     enrollee_id      city  city_development_index gender  \
77          4334  city_103                   0.920   Male
997        25726   city_24                   0.698   Male

         relevent_experience enrolled_university education_level  \
77   Has relevent experience                 NaN     High School
997  Has relevent experience                 NaN  Primary School

    major_discipline experience company_size company_type last_new_job  \
77               NaN          9          NaN          NaN            2
997              NaN          3          NaN          NaN        never

     training_hours
77               19
997             150
```

[16]:
```
#changing enrollment values. to 'no_enrollment'
aug_test.iloc[not_enrolled.index,5]= 'no_enrollment'
```

Major discipline

[17]:
```
augtest_md = aug_test['major_discipline'].unique()
augtest_md
```

[17]:
```
array(['STEM', nan, 'Other', 'Business Degree', 'Arts', 'Humanities',
       'No Major'], dtype=object)
```

[18]:
```
# where not enrolled at university and education under graduate level. then major discipline should be 'No_Major'
```

```
aug_test.
 ↪loc[(aug_test['enrolled_university']=='no_enrollment')&(mask),'major_discipline']␣
 ↪= 'No_Major'
```

1.3.2 aug_train

[19]:
```
(aug_train.isna().sum()/aug_train.shape[0]).sort_values(ascending=False)
```

[19]:
```
company_type              0.311720
company_size              0.301195
gender                    0.227417
major_discipline          0.136027
last_new_job              0.017717
education_level           0.015020
enrolled_university       0.014333
experience                0.002433
target                    0.000000
training_hours            0.000000
relevent_experience       0.000000
city_development_index    0.000000
city                      0.000000
enrollee_id               0.000000
dtype: float64
```

major_discipline

[20]:
```
augtrain_md = aug_train['major_discipline'].unique()
augtrain_md
```

[20]:
```
array(['STEM', 'Business Degree', nan, 'Arts', 'Humanities', 'No Major',
       'Other'], dtype=object)
```

[21]:
```
aug_train.loc[(aug_train['major_discipline'].isna())&
          (aug_train['enrolled_university']=='no_enrollment')&
          (((aug_train['education_level']!='Masters')&
        (aug_train['education_level']!='Phd')&
        (aug_train['education_level']!='Graduate')&
        aug_train['education_level'].notnull())),'major_discipline']='No Major'
```

1.4 4 Outliers

[22]:
```
# aug_test
```

[23]:
```
aug_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2104 entries, 0 to 2103
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
```

```
---  ------                   --------------  -----
 0   enrollee_id              2104 non-null   int64
 1   city                     2104 non-null   object
 2   city_development_index   2104 non-null   float64
 3   gender                   1617 non-null   object
 4   relevent_experience      2104 non-null   object
 5   enrolled_university      2085 non-null   object
 6   education_level          2070 non-null   object
 7   major_discipline         1950 non-null   object
 8   experience               2100 non-null   object
 9   company_size             1505 non-null   object
 10  company_type             1495 non-null   object
 11  last_new_job             2075 non-null   object
 12  training_hours           2104 non-null   int64
dtypes: float64(1), int64(2), object(10)
memory usage: 213.8+ KB
```

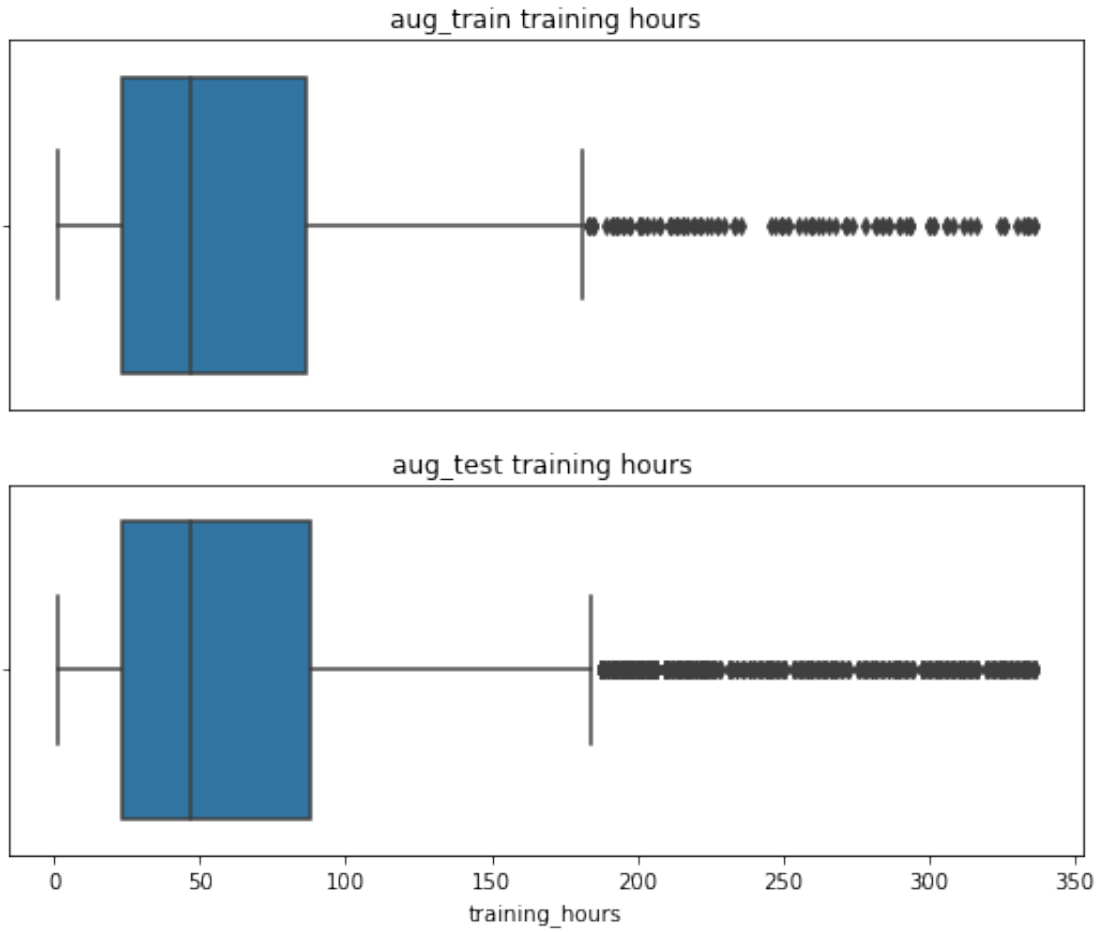
[199]:
```
aug_test.select_dtypes(include=['int64','float64']).head()
#enrollee_id - unique identifier.
#city_development_index - Developement index of the city (scaled) - normalised
 ↪0 to 1
#training_hours - training hours completed
```

[199]:

	enrollee_id	city_development_index	training_hours
0	32403	0.827	21
1	9858	0.920	98
2	31806	0.624	15
3	27385	0.827	39
4	27724	0.920	72

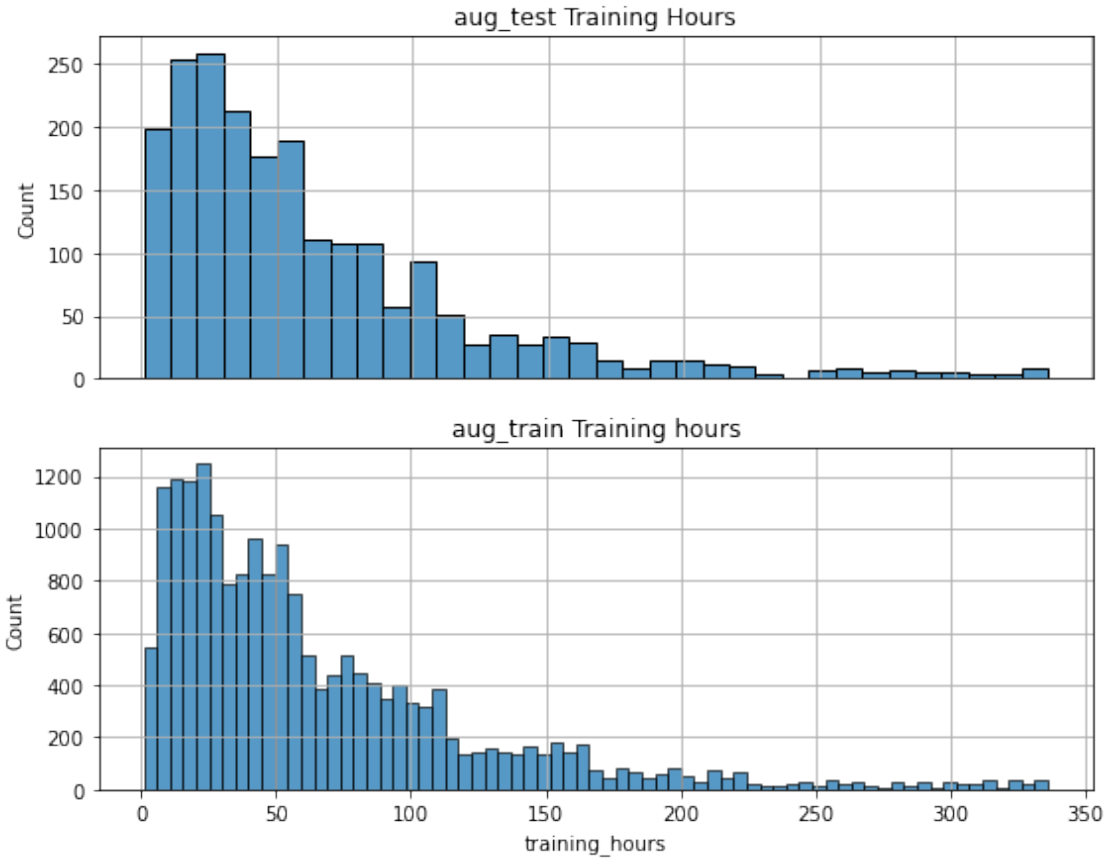
Training hours

[25]:
```
fig, (ax1,ax2) = plt.subplots(2,figsize=(9,7))

sns.boxplot(x= aug_test['training_hours'],ax=ax1)
sns.boxplot(x= aug_train['training_hours'],ax=ax2)
ax1.set_xlabel('')
ax1.tick_params(axis='x',labelbottom=False,bottom =False)
ax1.set_title('aug_train training hours')
ax2.set_title('aug_test training hours ')

plt.show()
```



[26]:
```
fig, (ax3,ax4) = plt.subplots(2,figsize=(9,7))

ax3.grid()
ax4.grid()

sns.histplot(aug_test['training_hours'],ax =ax3)
sns.histplot(aug_train['training_hours'],ax =ax4)

ax3.set_xlabel('')
ax3.tick_params(axis='x',labelbottom=False,bottom=False)

ax3.set_title('aug_test Training Hours')
ax4.set_title('aug_train Training hours')
```

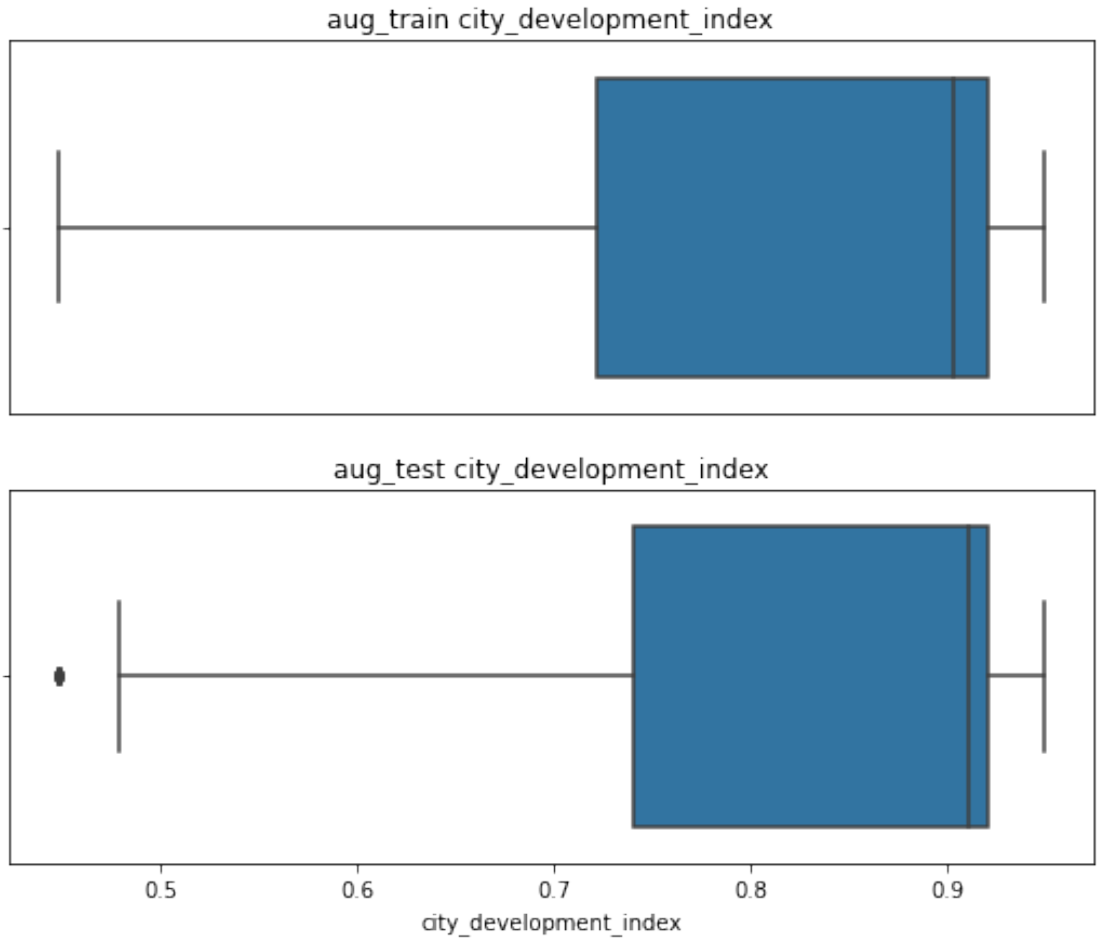
[26]:
```
Text(0.5, 1.0, 'aug_train Training hours')
```



Right skewed unimodal distriubtion.

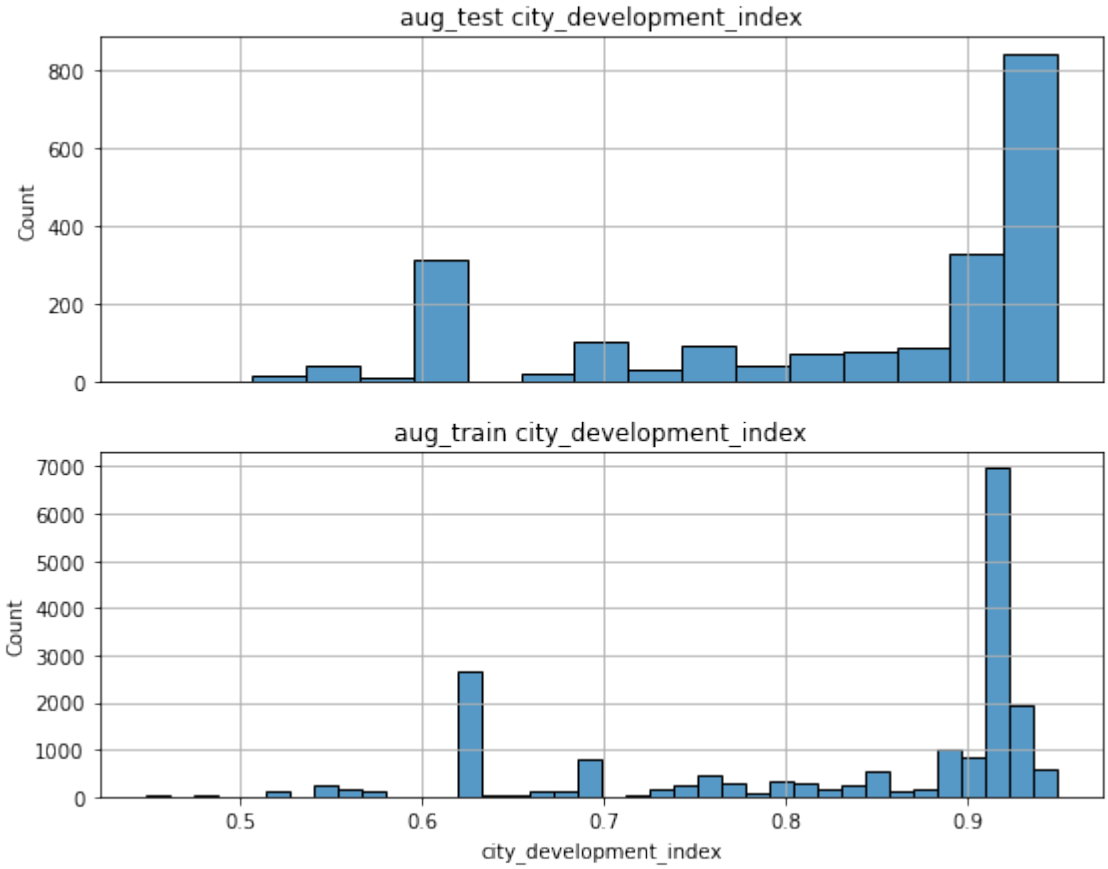
City development index

[27]:
```
fig, (ax1,ax2) = plt.subplots(2,figsize=(9,7))

sns.boxplot(x= aug_test['city_development_index'],ax=ax1)
sns.boxplot(x= aug_train['city_development_index'],ax=ax2)
ax1.set_xlabel('')
ax1.tick_params(axis='x',labelbottom=False,bottom =False)
ax1.set_title('aug_train city_development_index')
ax2.set_title('aug_test city_development_index')

plt.show()
```



[28]:
```
fig, (ax3,ax4) = plt.subplots(2,figsize=(9,7))

ax3.grid()
ax4.grid()

sns.histplot(aug_test['city_development_index'],ax =ax3)
sns.histplot(aug_train['city_development_index'],ax =ax4)

ax3.set_xlabel('')
ax3.tick_params(axis='x',labelbottom=False,bottom=False)

ax3.set_title('aug_test city_development_index')
ax4.set_title('aug_train city_development_index')
```

[28]:
```
Text(0.5, 1.0, 'aug_train city_development_index')
```



2 Transforming

[29]:
```
train_data = aug_train.copy()
codified = train_data.copy()
aug_train.head(3)
```

[29]:
```
   enrollee_id      city  city_development_index gender  \
0         8949  city_103                   0.920   Male
1        29725   city_40                   0.776   Male
2        11561   city_21                   0.624    NaN

       relevent_experience enrolled_university education_level  \
0  Has relevent experience       no_enrollment        Graduate
1   No relevent experience       no_enrollment        Graduate
2   No relevent experience    Full time course        Graduate

  major_discipline experience company_size company_type last_new_job  \
0             STEM        >20          NaN          NaN            1
1             STEM         15        50-99      Pvt Ltd           >4
```

```
2              STEM          5          NaN          NaN        never

   training_hours  target
0              36     1.0
1              47     0.0
2              83     0.0
```

2.1 Nominal Data

-name or identify something. no order or rank

enrolled_university

```
      no_enrollment = 0
    Part time course = 1
    Full time course = 2
```

major_discipline

```
    No Major = 0
    Arts = 1
    Humanities = 2
    Buisness Degree = 3
    STEM = 4
    Other = 5
```

Company Type

```
   Pvt Ltd = 0
   Funded Startup = 1
   Early Stage Startup = 2
   Other = 3
   Public sector = 4
   NGO = 5
```

City

```
  city is alread nominal i will just remove the string preffix
```

Gender

```
male = 1
female = 0
Other = 2
```

[30]:
```python
def codification (series,dict_):
    #values must be passes as a dict, keys as old values and values as new␣
 ↪values.
    sname = series.name
```

```
    for key in dict_.keys():
        series[series==key] = dict_[key]
    return

def get_keys(series):
    keys = series.dropna().unique()
    return keys
```

[31]:
```
#enrolled_university
enrol_values = [0,2,1]
enrol_keys = get_keys(train_data['enrolled_university'])
enrol_cfn = dict(zip(enrol_keys,enrol_values))

codification(codified['enrolled_university'],enrol_cfn)

#major_discipline
mj_values = [4,3,0,1,2,5]
mj_keys = get_keys(train_data['major_discipline'])
mj_cfn = dict(zip(mj_keys,mj_values))

codification(codified['major_discipline'],mj_cfn)

#company_type
ct_values = [0,1,2,3,4,5]
ct_keys = get_keys(train_data['company_type'])
ct_cfn = dict(zip(ct_keys,ct_values))

codification(codified['company_type'],ct_cfn)

#city
codified['city'] = pd.to_numeric(codified['city'].str.strip().str.
 ↪replace('city_',''))

#gender
codified.loc[codified['gender']=='Male','gender']=1
codified.loc[codified['gender']=='Female','gender']=0
codified.loc[codified['gender']=='Other','gender']=2
```

```
C:\Users\windows\miniconda3\lib\site-packages\ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

[198]:
```
codified[['company_type','major_discipline','enrolled_university','gender','city']].
↪head()
```

[198]:

	company_type	major_discipline	enrolled_university	gender	city
0	NaN	4	0	1	103
1	0	4	0	1	40
2	NaN	4	2	NaN	21
3	0	3	NaN	NaN	115
4	1	4	0	1	162

2.2 Cardinal

- counting numbers, indicate quantity.

company_size

<10 = 0
10/49 = 1
50-99 = 2
100-500 = 3
500-999 = 4
1000-4999 = 5
5000=9999 = 6
10000+ = 7

[33]:
```
#company_size
cs_values = [2,0,7,6,5,1,3,4]
cs_keys = get_keys(train_data['company_size'])
cs_cfn = dict(zip(cs_keys,cs_values))

codification(codified['company_size'],cs_cfn)

#training hours
codified['training_hours'].unique()
```

```
C:\Users\windows\miniconda3\lib\site-packages\ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

[33]:
```
array([ 36,  47,  83,  52,   8,  24,  18,  46, 123,  32, 108,  23,  26,
       106,   7, 132,  68,  50,  48,  65,  13,  22, 148,  72,  40, 141,
        82, 145, 206, 152,  42,  14, 112,  87,  20,  21,  92, 102,  43,
        45,  19,  90,  25,  15, 142,  28,  29,  12,  17,  35,   4, 136,
        27,  74,  86,  75, 332, 140, 182, 172,  33,  34, 150, 160,   3,
```

```
  2, 210, 101,  59, 260, 131, 109,  70,  51,  60, 164, 290, 133,
 76, 156, 120, 100,  39,  55,  49,   6, 125, 326, 198,  11,  41,
114, 246,  81,  31,  84, 105,  38, 178, 104, 202,  88, 218,  62,
 10,  80,  77,  37, 162, 190,  30,  16,   5,  54,  44, 110, 262,
107, 134, 103,  96,  57, 240,  94, 113,  56,  64, 320,   9, 129,
 58, 126, 166,  95,  97, 204, 116, 161, 146,  53, 143, 124, 214,
288, 306,  98, 322,  67,  61, 130, 220,  78, 314, 226, 280,  91,
234, 163, 151,  85, 256, 168, 144,  66, 128,  73, 122, 154,  63,
292, 188,  71, 135, 138, 184,  89, 157, 118, 111, 192, 127, 216,
139, 196,  99, 167, 276, 121,  69, 155, 316, 242, 304, 284, 278,
310, 222, 212, 250, 180, 258, 330, 158, 149, 165,  79, 194, 176,
174, 312, 200, 328, 300, 153, 232, 336, 308, 228, 147, 298, 224,
254, 248, 236, 170, 264, 119, 117, 302, 334, 324,   1, 238, 266,
282, 268, 244, 272, 294, 270, 286], dtype=int64)
```

[197]:
```
codified[['company_size','training_hours']].head()
```

[197]:

	company_size	training_hours
0	NaN	36
1	2	47
2	NaN	83
3	NaN	52
4	2	8

2.3 Ordinal

- indicate the rank or order of something.

last_new_job

```
'<1' = 1
'>4'= 5
'never' = 0
'4' = 4
'3' = 3
'2' = 2
```

Education level

```
Graduate' = 3
'Masters' = 4
'High School'= 2
'Phd'= 5
'Primary School' = 1
```

[35]:
```
#experience
codified.loc[codified['experience']=='>20','experience']=21
codified.loc[codified['experience']=='<1','experience']=0
codified['experience'] = pd.to_numeric(codified['experience'])
```

```
#last new job
codified.loc[codified['last_new_job']=='>4','last_new_job']=5
codified.loc[codified['last_new_job']=='never','last_new_job']=0
codified['last_new_job'] = pd.to_numeric(codified['last_new_job'])


#education level
el_values = [3,4,2,5,1]
el_keys = get_keys(codified['education_level'])
el_cfn = dict(zip(el_keys,el_values))

codification(codified['education_level'],el_cfn)
```

```
C:\Users\windows\miniconda3\lib\site-packages\ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

[196]:
```
codified[['last_new_job','experience','education_level']].head()
```

[196]:

	last_new_job	experience	education_level
0	1.0	21.0	3
1	5.0	15.0	3
2	0.0	5.0	3
3	0.0	0.0	3
4	4.0	21.0	4

2.4 Binary

relevent_experience

No relevent experience = 0
Has relevent experience = 1

[37]:
```
#relevent_experience
codified.loc[codified['relevent_experience']=='No relevent␣
 ↪experience','relevent_experience'] = 0
codified.loc[codified['relevent_experience']=='Has relevent␣
 ↪experience','relevent_experience'] = 1

#target
codified['target'].unique()
```

[37]:
```
array([1., 0.])
```

[195]:
```
codified[['relevent_experience','target']].head()
```

[195]:

	relevent_experience	target
0	1	1.0
1	0	0.0
2	0	0.0
3	0	1.0
4	1	0.0

[39]:
```
#codified.to_csv(cwd+'\\Cleaned_data\\clean.csv')
```

2.5 Scaling Data

[40]:
```
scaled = codified.copy()
```

[41]:
```
#run only once
#training hours
th_min  = scaled['training_hours'].min()
th_max =  scaled['training_hours'].max()

scaled['training_hours'] = scaled['training_hours'] - th_min
scaled['training_hours'] = scaled['training_hours'] /(th_max-th_min)

#experience
exp_min  = scaled['experience'].min()
exp_max =  scaled['experience'].max()

scaled['experience'] = scaled['experience'] - exp_min
scaled['experience'] = scaled['experience'] /(exp_max-exp_min)
```

[42]:
```
#correlation of our numerical collumns
correlation = scaled.corr()
correlation>0.5
```

[42]:

	enrollee_id	city	city_development_index	experience	last_new_job	training_hours	target
enrollee_id	True	False	False	False	False	False	False
city	False	True	False	False	False	False	False
city_development_index	False	False	True	False	False	False	False
experience	False	False	False	False	False	False	False
last_new_job	False	False	False	False	False	False	False
training_hours	False	False	False	False	False	False	False
target	False	False	False	False	False	False	False

```
experience                      True          False           False   False
last_new_job                   False           True           False   False
training_hours                 False          False            True   False
target                         False          False           False    True
```

The fact that there are no correlations is a good sign, it means there appear to be no redudant attributes for the ml. it seems that we have reduced the data as much as it needs, hopefull it will now be usefull to use in a model.

[194]:
```
scaled.head()
```

[194]:
```
       city  city_development_index gender relevent_experience  \
0  0.569832                   0.920      1                    1
1  0.217877                   0.776      1                    0
2  0.111732                   0.624    NaN                    0
3  0.636872                   0.789    NaN                    0
4  0.899441                   0.767      1                    1

  enrolled_university education_level major_discipline  experience  \
0                   0               3                4    1.000000
1                   0               3                4    0.714286
2                   2               3                4    0.238095
3                 NaN               3                3    0.000000
4                   0               4                4    1.000000

  company_size company_type  last_new_job  training_hours  target
0          NaN          NaN           1.0        0.104478     1.0
1            2            0           5.0        0.137313     0.0
2          NaN          NaN           0.0        0.244776     0.0
3          NaN            0           0.0        0.152239     1.0
4            2            1           4.0        0.020896     0.0
```

after looking at the columns, ive realised that city will need to be scales too, this makes sense and im not sure why i missed it to start with.

unline enrollee_id the city attribute should have some infulence on our predictions.

[44]:
```
#scale the city attribute.
max_c = scaled['city'].max()
min_c = scaled['city'].min()
scaled['city'] = scaled['city'] - scaled['city'].min()
scaled['city'] = scaled['city']/(max_c-min_c)
```

[45]:
```
scaled.drop(columns = 'enrollee_id',inplace=True)
```

[46]:
```
scaled2 = scaled.copy()
```

[47]:
```
#scale columns that i forgot to scale initaly
```

```
columns = ['gender',
           'enrolled_university',
           'education_level',
           'major_discipline',
           'company_size',
           'last_new_job',
           'company_type',
          'city_development_index']

for col in columns:
    scaled2[col] =(scaled2[col] - scaled2[col].min())/(scaled2[col].
 ↪max()-scaled2[col].min())
```

[49]:
```
#if not os.path.isdir(cwd+'\\training_data'):
   # os.mkdir(cwd+'\\training_data')
#scaled2.to_csv(cwd+'\\training_data\\train_ready.csv')
#os.startfile(cwd+'\\training_data')
```

3 PCA reduction of components

principle component analysis, which of our columns affect the data the most ?

[92]:
```
from sklearn import preprocessing
from sklearn.decomposition import PCA
#dir(preprocessing)
```

[93]:
```
data = scaled2.copy()
st_no_nulls = data.dropna(axis = 0)
st_no_nulls = st_no_nulls.reset_index()
st_no_nulls = st_no_nulls.drop(columns='index')
```

[94]:
```
#scale the data
scaler = preprocessing.MinMaxScaler(feature_range=(-1,1))
col_names = data.columns
scaled_data = scaler.fit_transform(data)
standardized = pd.DataFrame(data = scaled_data,columns = col_names)

# Create a Covariance Matrix
pca = PCA(n_components = st_no_nulls.shape[1])

pca_data = pca.fit(st_no_nulls)
components = pca.transform(st_no_nulls)

projected = pca.inverse_transform(components)
```

[95]:
```
#dir(scaler)
#dir(preprocessing)
```

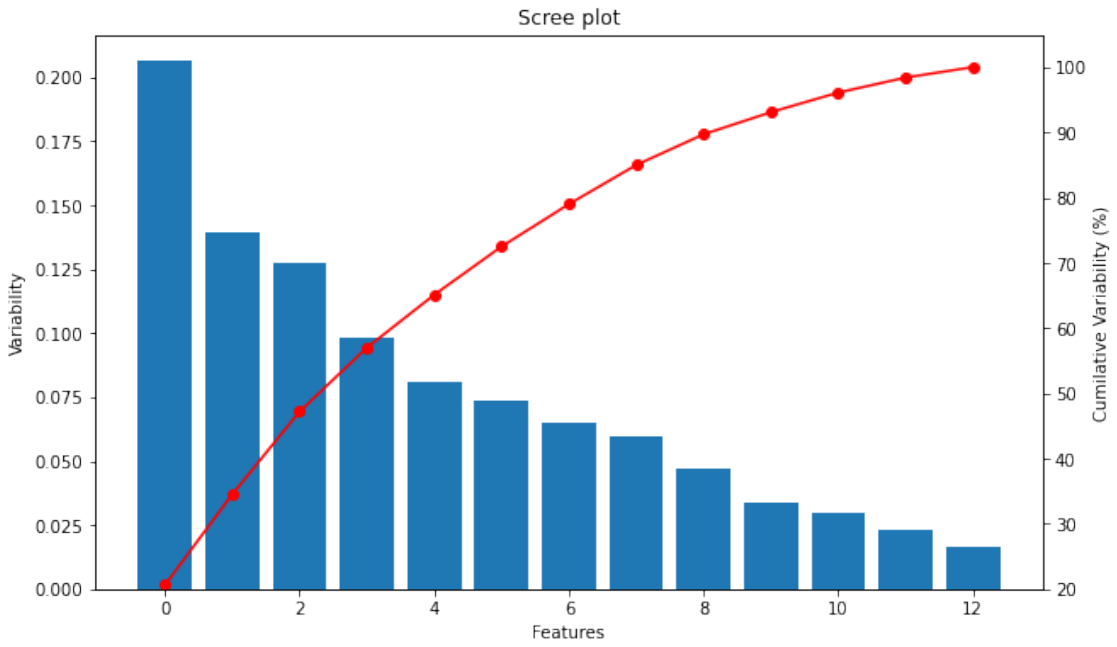
```
#dir(pca_data)
```

3.1 Variability of Features

[155]:
```
#calculates just the variace of our covariance matrix
variance = pca_data.explained_variance_ratio_

#calculates the cumilative sum of our co cariance matrix
var=np.cumsum(np.round(pca_data.explained_variance_ratio_, decimals=3))
```

[155]: 13

[186]:
```
fig, ax = plt.subplots(figsize = (10,6))

ax.bar(x=range(0,len(variance)),height = variance)
ax2 = ax.twinx()
ax2.plot((var*100),c='r',marker='o')
ax2.set_ylim(20,105)

ax.set_xlabel('Features')
ax.set_ylabel('Variability')
ax2.set_ylabel('Cumilative Variability (%)')
plt.title('Scree plot')
plt.show()
```



Based on this we should keep all our features. here we see column 1 accounts for 20.6% of the total

vairance, att2 14.5, att3 12.8 ….

https://etav.github.io/python/scikit_pca.html

3.2 PCA Plots

[137]:
```
pca = ['pca_'+ str(i) for i in range(0,components.shape[1])]
comp_data =pd.DataFrame(components,columns =pca)

print((st_no_nulls.join(comp_data).isnull()).sum().sum())
pca_joined = st_no_nulls.join(comp_data)
pca_joined.head()
```

0

[137]:

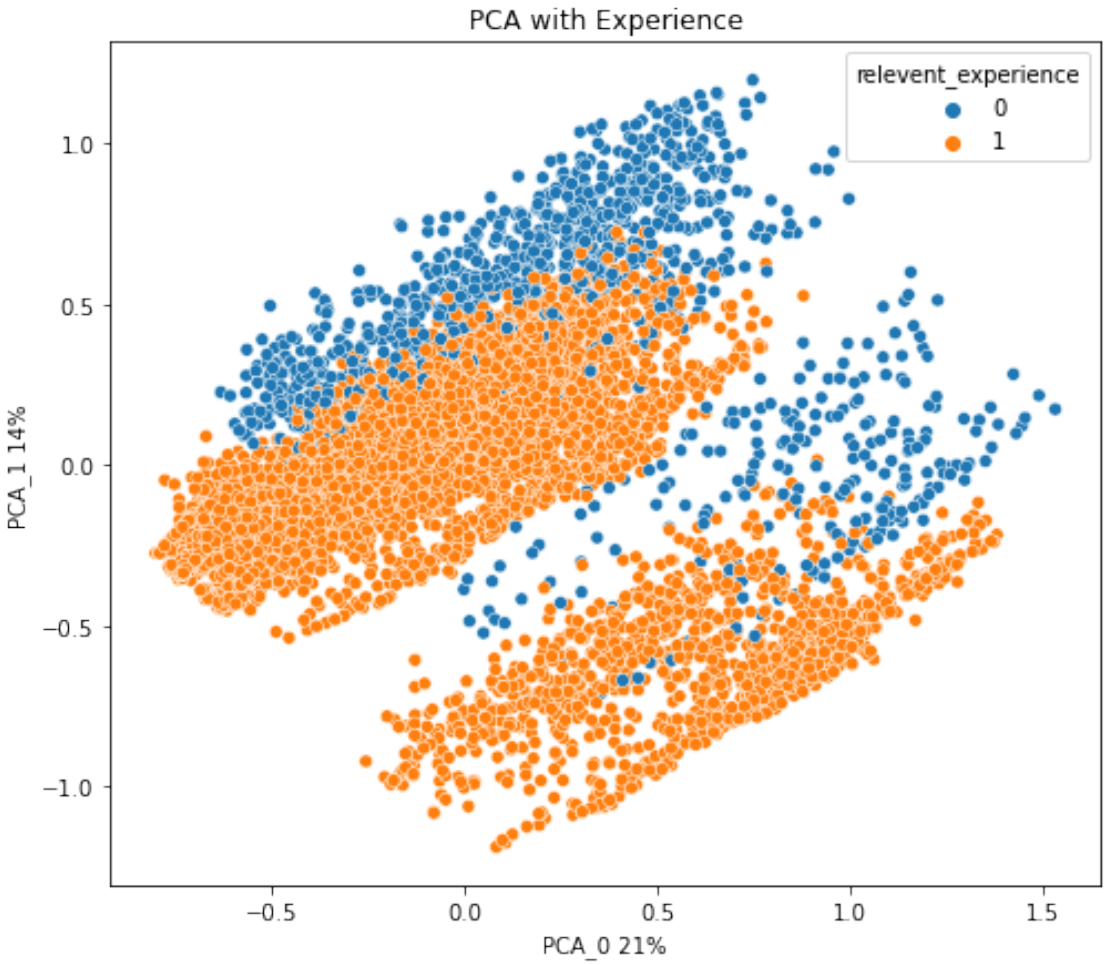
	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	…	pca_3	pca_4	pca_5	pca_6	pca_7	pca_8	pca_9	pca_10	pca_11	pca_12
0	0.217877	0.654691	0.5	0	0	0.5	0.8	0.714286	0.285714	0	…	0.416457	-0.112860	-0.659420	0.272216	-0.179267	-0.133250	-0.047061	0.079739	0.039777	-0.110541
1	0.899441	0.636727	0.5	1	0	0.75	0.8	1.000000	0.285714	0.2	…	0.316969	0.053877	0.180530	-0.153994	-0.061426	-0.215342	-0.154749	0.392579	-0.038506	0.121430
2	0.888268	0.942116	0.5	1	0	0.25	0	0.238095	0.285714	0.2	…	0.104678	-0.312985	0.046741	-0.325692	0.543482	0.525898	-0.112327	0.181928	0.017938	0.031432
3	0.251397	0.626747	0.5	1	0	0.5	0.8	0.619048	0	0	…	0.663912	0.032545	-0.141164	-0.040526	0.081326	-0.064340	-0.144806	-0.091706	0.040688	-0.067905
4	0.569832	0.942116	0.5	1	0	0.5	0.8	0.333333	0.285714	0	…	0.136769	-0.265658	0.123726	-0.478750	0.061134	-0.045445	-0.068228	-0.251149	0.087832	-0.062668

[5 rows x 26 columns]

[225]:
```
fig = plt.figure(figsize = (8,7))
sns.scatterplot(x=pca_joined.pca_0,y= pca_joined.pca_1,hue = pca_joined.
↪relevent_experience)
```

```
plt.xlabel('PCA_0 {}%'.format(round(var[0]*100)))
plt.ylabel('PCA_1 {}%'.format(round((var[1]-var[0])*100)))
plt.title('PCA with Experience')
plt.plot()
```

[225]: []



[236]:
```
plt.figure(figsize=(12,7))
ax = plt.axes(projection='3d')
xdata = pca_joined.pca_0
ydata = pca_joined.pca_1
zdata = pca_joined.pca_2
ax.scatter3D(xdata,ydata,zdata,c=pca_joined.relevent_experience)
ax.set_xlabel('PCA_0 {}%'.format(round(var[0]*100)))
ax.set_ylabel('PCA_1 {}%'.format(round((var[1]-var[0])*100)))
ax.set_zlabel('PCA_2 {}%'.format(round((var[2]-var[1]-var[0])*100)))
plt.title('PCA with Experience')
```

[236]: Text(0.5, 0.92, 'PCA with Experience')