Untitled3

September 24, 2019

[1]:

```
import itertools
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from matplotlib.ticker import NullFormatter
from sklearn import preprocessing
%matplotlib inline
```

[2]:

```
!wget -O teleCust1000t.csv https://s3-api.us-geo.objectstorage.softlayer.net/
 ↪cf-courses-data/CognitiveClass/ML0101ENv3/labs/teleCust1000t.csv
```

```
--2019-09-24 07:17:38--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-data/CognitiveClass/ML0101ENv3/labs/teleCust1000t.csv
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-
geo.objectstorage.softlayer.net)… 67.228.254.193
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-
geo.objectstorage.softlayer.net)|67.228.254.193|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 37048 (36K) [text/csv]
Saving to: ‘teleCust1000t.csv’

teleCust1000t.csv   100%[===================>]  36.18K  --.-KB/s    in 0.02s

2019-09-24 07:17:39 (1.68 MB/s) - ‘teleCust1000t.csv’ saved [37048/37048]
```

[3]:

```
df = pd.read_csv('teleCust1000t.csv')
df.head()
```

[3]:

```
   region  tenure  age  marital  address  income  ed  employ  retire  gender  \
0       2      13   44        1        9    64.0   4       5     0.0       0
1       3      11   33        1        7   136.0   5       5     0.0       0
2       3      68   52        1       24   116.0   1      29     0.0       1
3       2      33   33        0       12    33.0   2       0     0.0       1
4       2      23   30        1        9    30.0   1       2     0.0       0

   reside  custcat
```

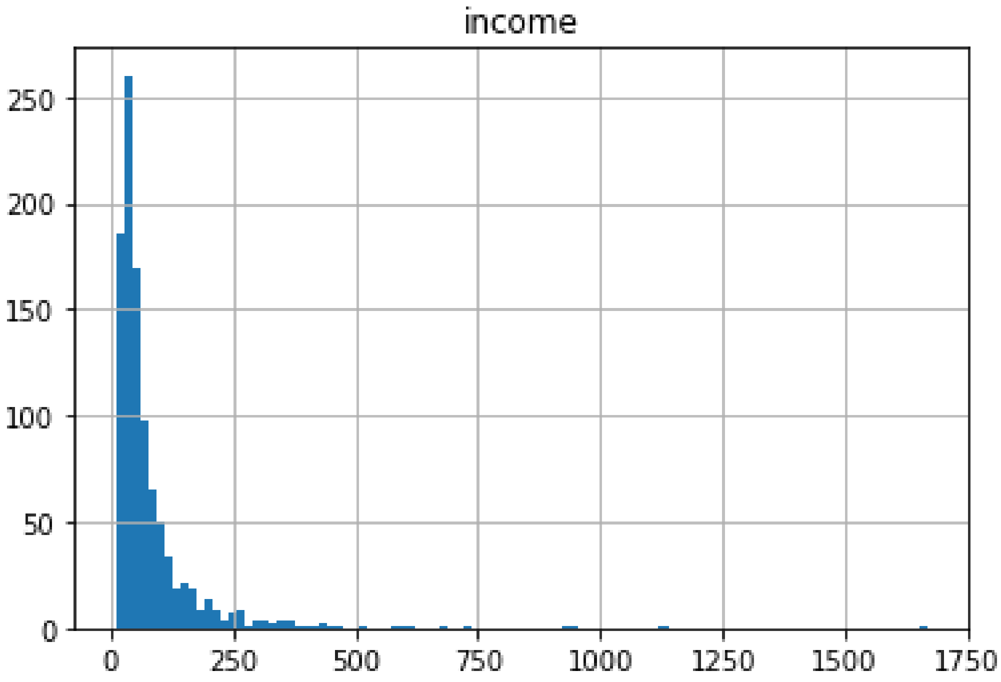
```
0        2          1
1        6          4
2        2          3
3        1          1
4        4          3
```

[4]:
```
df['custcat'].value_counts()
```

[4]:
```
3    281
1    266
4    236
2    217
Name: custcat, dtype: int64
```

[5]:
```
df.hist(column='income', bins=100)
```

[5]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f436279d0b8>]],
      dtype=object)
```



[6]:
```
df.columns
```

[6]:
```
Index(['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside', 'custcat'],
      dtype='object')
```

```
[7]: X=df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
            'employ', 'retire', 'gender', 'reside']].values
```

```
[8]: X[0:10]
```

```
[8]: array([[  2.,  13.,  44.,   1.,   9.,  64.,   4.,   5.,   0.,   0.,   2.],
            [  3.,  11.,  33.,   1.,   7., 136.,   5.,   5.,   0.,   0.,   6.],
            [  3.,  68.,  52.,   1.,  24., 116.,   1.,  29.,   0.,   1.,   2.],
            [  2.,  33.,  33.,   0.,  12.,  33.,   2.,   0.,   0.,   1.,   1.],
            [  2.,  23.,  30.,   1.,   9.,  30.,   1.,   2.,   0.,   0.,   4.],
            [  2.,  41.,  39.,   0.,  17.,  78.,   2.,  16.,   0.,   1.,   1.],
            [  3.,  45.,  22.,   1.,   2.,  19.,   2.,   4.,   0.,   1.,   5.],
            [  2.,  38.,  35.,   0.,   5.,  76.,   2.,  10.,   0.,   0.,   3.],
            [  3.,  45.,  59.,   1.,   7., 166.,   4.,  31.,   0.,   0.,   5.],
            [  1.,  68.,  41.,   1.,  21.,  72.,   1.,  22.,   0.,   0.,   3.]])
```

```
[9]: X[0]
```

```
[9]: array([ 2., 13., 44.,  1.,  9., 64.,  4.,  5.,  0.,  0.,  2.])
```

```
[10]: X
```

```
[10]: array([[ 2., 13., 44., …,  0.,  0.,  2.],
             [ 3., 11., 33., …,  0.,  0.,  6.],
             [ 3., 68., 52., …,  0.,  1.,  2.],
             …,
             [ 3., 67., 59., …,  0.,  1.,  1.],
             [ 3., 70., 49., …,  0.,  1.,  1.],
             [ 3., 50., 36., …,  0.,  1.,  3.]])
```

```
[11]: y = df['custcat'].values
```

```
[12]: y[0:5]
```

```
[12]: array([1, 4, 3, 1, 3])
```

```
[13]: X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
```

```
[14]: X[0]
```

```
[14]: array([-0.02696767, -1.055125  ,  0.18450456,  1.0100505 , -0.25303431,
             -0.12650641,  1.0877526 , -0.5941226 , -0.22207644, -1.03459817,
             -0.23065004])
```

```
[15]: from sklearn.model_selection import train_test_split
```

[16]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,␣
 ↪random_state=4)
```

[17]:
```
X_train.shape
```

[17]:
```
(800, 11)
```

[ ]:
```

```

[18]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[19]:
```
k = 4
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

[19]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=4, p=2,
           weights='uniform')
```

[20]:
```
yhat= neigh.predict(X_test)
```

[ ]:
```

```

[21]:
```
k=6
neigh = KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
```

[22]:
```
neigh
```

[22]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=6, p=2,
           weights='uniform')
```

[23]:
```
yhat= neigh.predict(X_test)
```

[25]:
```
yhat[0:10]
```

[25]:
```
array([3, 3, 3, 4, 4, 3, 3, 4, 2, 4])
```

[ ]:
```

```