PMR3508 - Aprendizado de Máquina e Reconhecimento de Padrões

Testando kNN com a base adult obtida no UCI repository. Iniciando com carregamento da base e com análise básica da base e dos atributos.

Autor: Fabio G. Cozman Data: 09/08/2018

In [1]:
```
import pandas as pd
import sklearn
```

In [2]:
```
adult = pd.read_csv("/Users/imac/Desktop/HOME/Didatico/Aulas/Graduaca
o/PMR3508/2018/Datasets/Adult-UCI/adult.data.txt",
        names=[
        "Age", "Workclass", "fnlwgt", "Education", "Education-Num", "
Martial Status",
        "Occupation", "Relationship", "Race", "Sex", "Capital Gain",
"Capital Loss",
        "Hours per week", "Country", "Target"],
        sep=r'\s*,\s*',
        engine='python',
        na_values="?")
```

In [3]:
```
adult.shape
```

Out[3]: (32561, 15)

In [4]:
```
adult.head()
```

Out[4]:

	Age	Workclass	fnlwgt	Education	Education-Num	Martial Status	Occupation	Rel
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Hus
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Hus
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife

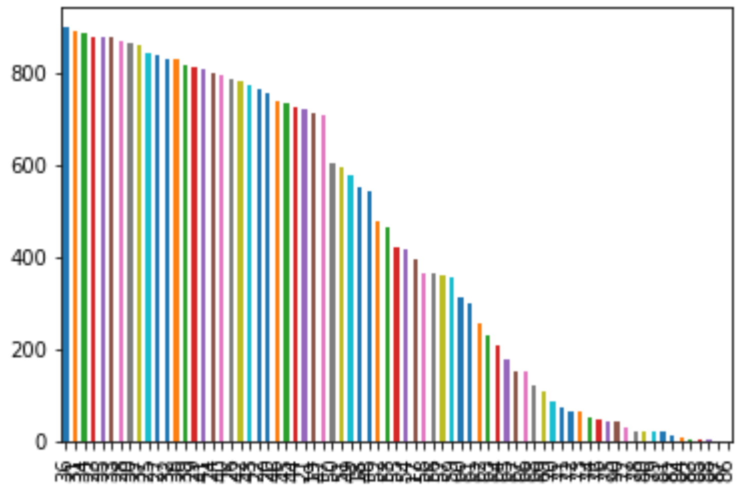
In [5]:
```
adult["Country"].value_counts()
```

Out[5]:
```
United-States                 29170
Mexico                          643
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        64
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
Greece                           29
France                           29
Ecuador                          28
Ireland                          24
Hong                             20
Trinadad&Tobago                  19
Cambodia                         19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Honduras                         13
Hungary                          13
Scotland                         12
Holand-Netherlands                1
Name: Country, dtype: int64
```

In [6]:
```
import matplotlib.pyplot as plt
```

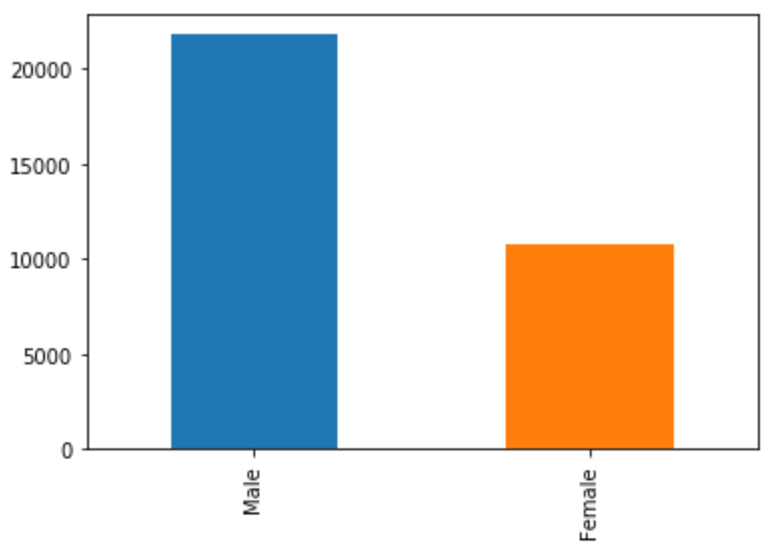
In [7]:
```
adult["Age"].value_counts().plot(kind="bar")
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0xa0b3c10f0>



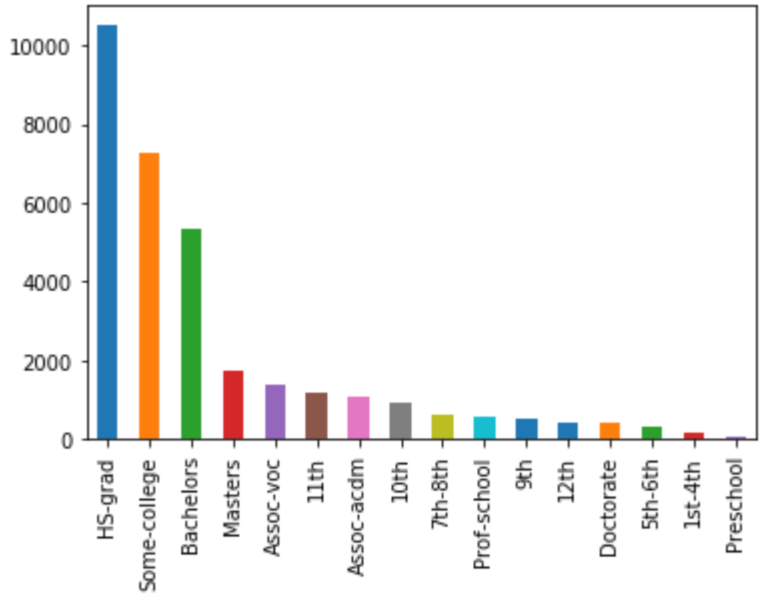
In [8]:
```
adult["Sex"].value_counts().plot(kind="bar")
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x10a1ba208>



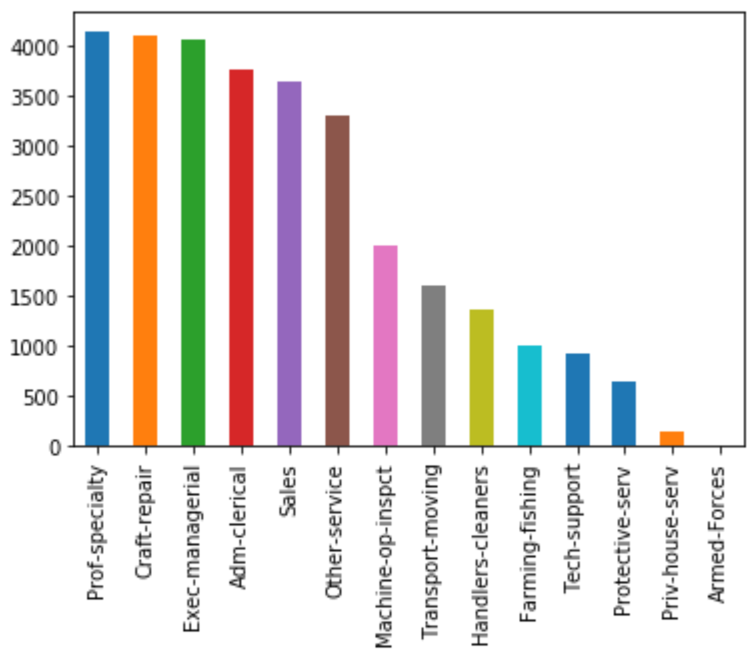
In [9]:
```
adult["Education"].value_counts().plot(kind="bar")
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0xa0b7d2a58>



In [10]:
```
adult["Occupation"].value_counts().plot(kind="bar")
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0xa0b89b0f0>



Retirando linhas com dados faltantes.

In [11]:
```
nadult = adult.dropna()
```

In [12]:
```
nadult
```

Out[12]:

	Age	Workclass	fnlwgt	Education	Education-Num	Martial Status	Occupatio
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-cleric
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-manageria
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-manageria
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-manageria
8	31	Private	45781	Masters	14	Never-married	Prof-specialty
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-manageria
10	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-manageria
11	30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty
12	23	Private	122272	Bachelors	13	Never-married	Adm-cleric
13	32	Private	205019	Assoc-acdm	12	Never-married	Sales

	Age	Workclass	fnlwgt	Education	Education-Num	Martial Status	Occupatio
15	34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving
16	25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing
17	32	Private	186824	HS-grad	9	Never-married	Machine-op inspct
18	38	Private	28887	11th	7	Married-civ-spouse	Sales
19	43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-manageria
20	40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty
21	54	Private	302146	HS-grad	9	Separated	Other-service
22	35	Federal-gov	76845	9th	5	Married-civ-spouse	Farming-fishing
23	43	Private	117037	11th	7	Married-civ-spouse	Transport-moving
24	59	Private	109015	HS-grad	9	Divorced	Tech-support
25	56	Local-gov	216851	Bachelors	13	Married-civ-spouse	Tech-support
26	19	Private	168294	HS-grad	9	Never-married	Craft-repai
28	39	Private	367260	HS-grad	9	Divorced	Exec-manageria
29	49	Private	193366	HS-grad	9	Married-civ-spouse	Craft-repai
30	23	Local-gov	190709	Assoc-acdm	12	Never-married	Protective-serv

	Age	Workclass	fnlwgt	Education	Education-Num	Martial Status	Occupatio
31	20	Private	266015	Some-college	10	Never-married	Sales
...	...	...	...	...	...	...	...
32526	32	Private	211349	10th	6	Married-civ-spouse	Transport-moving
32527	22	Private	203715	Some-college	10	Never-married	Adm-cleric
32528	31	Private	292592	HS-grad	9	Married-civ-spouse	Machine-op inspct
32529	29	Private	125976	HS-grad	9	Separated	Sales
32532	34	Private	204461	Doctorate	16	Married-civ-spouse	Prof-specialty
32533	54	Private	337992	Bachelors	13	Married-civ-spouse	Exec-manageria
32534	37	Private	179137	Some-college	10	Divorced	Adm-cleric
32535	22	Private	325033	12th	8	Never-married	Protective-serv
32536	34	Private	160216	Bachelors	13	Never-married	Exec-manageria
32537	30	Private	345898	HS-grad	9	Never-married	Craft-repai
32538	38	Private	139180	Bachelors	13	Divorced	Prof-specialty
32540	45	State-gov	252208	HS-grad	9	Separated	Adm-cleric
32543	45	Local-gov	119199	Assoc-acdm	12	Divorced	Prof-specialty
32544	31	Private	199655	Masters	14	Divorced	Other-service

	Age	Workclass	fnlwgt	Education	Education-Num	Martial Status	Occupatic
32545	39	Local-gov	111499	Assoc-acdm	12	Married-civ-spouse	Adm-cleric
32546	37	Private	198216	Assoc-acdm	12	Divorced	Tech-support
32547	43	Private	260761	HS-grad	9	Married-civ-spouse	Machine-op inspct
32548	65	Self-emp-not-inc	99359	Prof-school	15	Never-married	Prof-specialty
32549	43	State-gov	255835	Some-college	10	Divorced	Adm-cleric
32550	43	Self-emp-not-inc	27242	Some-college	10	Married-civ-spouse	Craft-repai
32551	32	Private	34066	10th	6	Married-civ-spouse	Handlers-cleaners
32552	43	Private	84661	Assoc-voc	11	Married-civ-spouse	Sales
32553	32	Private	116138	Masters	14	Never-married	Tech-support
32554	53	Private	321865	Masters	14	Married-civ-spouse	Exec-manageria
32555	22	Private	310152	Some-college	10	Never-married	Protective-serv
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support

Fazendo o mesmo processo com os dados de teste.

In [13]:
```
testAdult = pd.read_csv("/Users/imac/Desktop/HOME/Didatico/Aulas/Grad
uacao/PMR3508/2018/Datasets/Adult-UCI/adult.test.txt",
        names=[
        "Age", "Workclass", "fnlwgt", "Education", "Education-Num", "
Martial Status",
        "Occupation", "Relationship", "Race", "Sex", "Capital Gain",
"Capital Loss",
        "Hours per week", "Country", "Target"],
        sep=r'\s*,\s*',
        engine='python',
        na_values="?")
```

In [14]:
```
nTestAdult = testAdult.dropna()
```

Primeiro teste: seleção de atributos numéricos, com kNN para k=3.

In [15]:
```
Xadult = nadult[["Age","Education-Num","Capital Gain", "Capital Los
s", "Hours per week"]]
```

In [16]:
```
Yadult = nadult.Target
```

In [17]:
```
XtestAdult = nTestAdult[["Age","Education-Num","Capital Gain", "Capit
al Loss", "Hours per week"]]
```

In [18]:
```
YtestAdult = nTestAdult.Target
```

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [20]:
```
knn = KNeighborsClassifier(n_neighbors=3)
```

In [21]:
```
from sklearn.model_selection import cross_val_score
```

In [22]:
```
scores = cross_val_score(knn, Xadult, Yadult, cv=10)
```

In [23]:
```
scores
```

Out[23]:
```
array([0.79549221, 0.80841896, 0.80609877, 0.79880676, 0.80437666,
       0.81266578, 0.79608753, 0.8030504 , 0.79104478, 0.81525705])
```

In [24]:
```
knn.fit(Xadult,Yadult)
```

Out[24]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
```

In [25]:
```
YtestPred = knn.predict(XtestAdult)
```

In [26]:
```
YtestPred
```

Out[26]:
```
array(['<=50K', '>50K', '<=50K', ..., '>50K', '<=50K', '>50K'],
      dtype=object)
```

In [27]:
```
from sklearn.metrics import accuracy_score
```

In [28]:
```
accuracy_score(YtestAdult,YtestPred)
```

Out[28]:
```
0.7949535192563081
```

Outro teste: mesmos dados, porém kNN com k=30. Melhor resultado obtido.

In [29]:
```
knn = KNeighborsClassifier(n_neighbors=30)
```

In [30]:
```
knn.fit(Xadult,Yadult)
```

Out[30]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=30, p=2,
           weights='uniform')
```

In [31]:
```
scores = cross_val_score(knn, Xadult, Yadult, cv=10)
```

In [32]:
```
scores
```

Out[32]:
```
array([0.82830626, 0.83128936, 0.82565462, 0.83294664, 0.82990716,
       0.83687003, 0.8229443 , 0.83454907, 0.83117745, 0.8371476 ])
```

In [33]:
```
YtestPred = knn.predict(XtestAdult)
```

In [34]:
```
accuracy_score(YtestAdult,YtestPred)
```

Out[34]:
```
0.8347277556440903
```

Passando todos os dados não-numéricos para valores numéricos, e fazendo alguns testes com vários conjuntos de atributos (mantendo k=30, pois foi o valor de k que levou a melhor acurácia).

In [35]:
```
from sklearn import preprocessing
```

In [36]:
```
numAdult = nadult.apply(preprocessing.LabelEncoder().fit_transform)
```

In [37]:
```
numTestAdult = nTestAdult.apply(preprocessing.LabelEncoder().fit_tran
sform)
```

In [38]:
```
Xadult = numAdult.iloc[:,0:14]
```

In [39]:
```
Yadult = numAdult.Target
```

In [40]:
```
XtestAdult = numTestAdult.iloc[:,0:14]
```

In [41]:
```
YtestAdult = numTestAdult.Target
```

In [42]:
```
knn = KNeighborsClassifier(n_neighbors=30)
```

In [43]:
```
knn.fit(Xadult,Yadult)
```

Out[43]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=30, p=2,
           weights='uniform')
```

In [44]:
```
YtestPred = knn.predict(XtestAdult)
```

In [45]:
```
accuracy_score(YtestAdult,YtestPred)
```

Out[45]:
```
0.7837317397078353
```

In [46]:
```
Xadult = numAdult[["Age", "Workclass", "Education-Num", "Martial Stat
us",
        "Occupation", "Relationship", "Race", "Sex", "Capital Gain",
"Capital Loss",
        "Hours per week", "Country"]]
```

In [47]:
```
XtestAdult = numTestAdult[["Age", "Workclass", "Education-Num", "Mart
ial Status",
        "Occupation", "Relationship", "Race", "Sex", "Capital Gain",
"Capital Loss",
        "Hours per week", "Country"]]
```

In [48]:
```
knn = KNeighborsClassifier(n_neighbors=30)
```

In [49]:
```
knn.fit(Xadult,Yadult)
```

Out[49]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=30, p=2,
           weights='uniform')
```

In [50]:
```
YtestPred = knn.predict(XtestAdult)
```

In [51]:
```
accuracy_score(YtestAdult,YtestPred)
```

Out[51]:
```
0.8284196547144754
```

In [52]:
```
Xadult = numAdult[["Age", "Workclass", "Education-Num",
        "Occupation", "Race", "Sex", "Capital Gain", "Capital Loss",
        "Hours per week"]]
```

In [53]:
```
XtestAdult = numTestAdult[["Age", "Workclass", "Education-Num",
        "Occupation", "Race", "Sex", "Capital Gain", "Capital Loss",
        "Hours per week"]]
```

In [54]:
```
knn = KNeighborsClassifier(n_neighbors=30)
```

In [55]:
```
knn.fit(Xadult,Yadult)
```

Out[55]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=30, p=2,
           weights='uniform')
```

In [56]:
```
YtestPred = knn.predict(XtestAdult)
```

In [57]:
```
accuracy_score(YtestAdult,YtestPred)
```

Out[57]:
```
0.8223107569721115
```