Adult Census Income

June 24, 2019

1 Adult Census Income

Predict whether income exceeds $50K/yr based on census data

1.1 Informations on the dataset

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)). The prediction task is to determine whether a person makes over $50K a year.

Original dataset open sourced, can be found here.

1.2 Goal

Predict **whether or not a person makes more than USD 50,000** from the information contained in the columns. Find clear insights on the profiles of the people that make more than 50,000USD / year. For example, which variables seem to be the most correlated with this phenomenon?

---

2 Dataset first insight

Libraries import

```
In [1]: import warnings
        warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [3]: from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score

        from sklearn.linear_model import LogisticRegression
```

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.model_selection import GridSearchCV
```

Loading the file

In [4]:
```python
df = pd.read_csv('./input/adult.csv')
df.head()
```

Out[4]:
```
   age workclass  fnlwgt     education  education.num marital.status  \
0   90         ?   77053       HS-grad              9        Widowed
1   82   Private  132870       HS-grad              9        Widowed
2   66         ?  186061  Some-college             10        Widowed
3   54   Private  140359       7th-8th              4       Divorced
4   41   Private  264663  Some-college             10      Separated

          occupation   relationship   race     sex  capital.gain  \
0                  ?  Not-in-family  White  Female             0
1    Exec-managerial  Not-in-family  White  Female             0
2                  ?      Unmarried  Black  Female             0
3  Machine-op-inspct      Unmarried  White  Female             0
4     Prof-specialty      Own-child  White  Female             0

   capital.loss  hours.per.week native.country income
0          4356              40  United-States  <=50K
1          4356              18  United-States  <=50K
2          4356              40  United-States  <=50K
3          3900              40  United-States  <=50K
4          3900              40  United-States  <=50K
```

Columns description

- age: continuous.
- workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- fnlwgt: continuous.
- education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- education-num: continuous.
- marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex: Female, Male.
- capital-gain: continuous.
- capital-loss: continuous.

- hours-per-week: continuous.
- native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

```
In [5]: df.shape
```

```
Out[5]: (32561, 15)
```

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age               32561 non-null int64
workclass         32561 non-null object
fnlwgt            32561 non-null int64
education         32561 non-null object
education.num     32561 non-null int64
marital.status    32561 non-null object
occupation        32561 non-null object
relationship      32561 non-null object
race              32561 non-null object
sex               32561 non-null object
capital.gain      32561 non-null int64
capital.loss      32561 non-null int64
hours.per.week    32561 non-null int64
native.country    32561 non-null object
income            32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

When it comes to numerical values, no information is missing. On the contrary for categorical features, there are '?', which indicated unknow information. Some rows are duplicated and need to be removed :

```
In [7]: df.duplicated().sum()
```

```
Out[7]: 24
```

```
In [8]: df = df.drop_duplicates()
        df.shape
```

```
Out[8]: (32537, 15)
```

```
In [9]: cat_feat = df.select_dtypes(include=['object']).columns
        cat_feat
```

```
Out[9]: Index(['workclass', 'education', 'marital.status', 'occupation',
               'relationship', 'race', 'sex', 'native.country', 'income'],
              dtype='object')
```

The number of missing value isn't relevant

```
In [10]: print('% of missing values :')
         for c in cat_feat:
             perc = len(df[df[c] == '?']) / df.shape[0] * 100
             print(c, f'{perc:.1f} %')
```

```
% of missing values :
workclass 5.6 %
education 0.0 %
marital.status 0.0 %
occupation 5.7 %
relationship 0.0 %
race 0.0 %
sex 0.0 %
native.country 1.8 %
income 0.0 %
```

Basic statistics for numerical values:

```
In [11]: df.describe()
```

```
Out[11]:                 age        fnlwgt  education.num  capital.gain  capital.loss  \
         count  32537.000000  3.253700e+04   32537.000000  32537.000000  32537.000000
         mean      38.585549  1.897808e+05      10.081815   1078.443741     87.368227
         std       13.637984  1.055565e+05       2.571633   7387.957424    403.101833
         min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
         25%       28.000000  1.178270e+05       9.000000      0.000000      0.000000
         50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000
         75%       48.000000  2.369930e+05      12.000000      0.000000      0.000000
         max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

                hours.per.week
         count    32537.000000
         mean        40.440329
         std         12.346889
         min          1.000000
         25%         40.000000
         50%         40.000000
         75%         45.000000
         max         99.000000
```

3 Exploratory Analysis

```
In [12]: # Taking a look at the target (income) without distinction of sex
         print(f"Ratio above 50k : {(df['income'] == '>50K').astype('int').sum() / df.shape[0]
```

```
Ratio above 50k : 24.09%
```

Distinction between numerical vs. text values

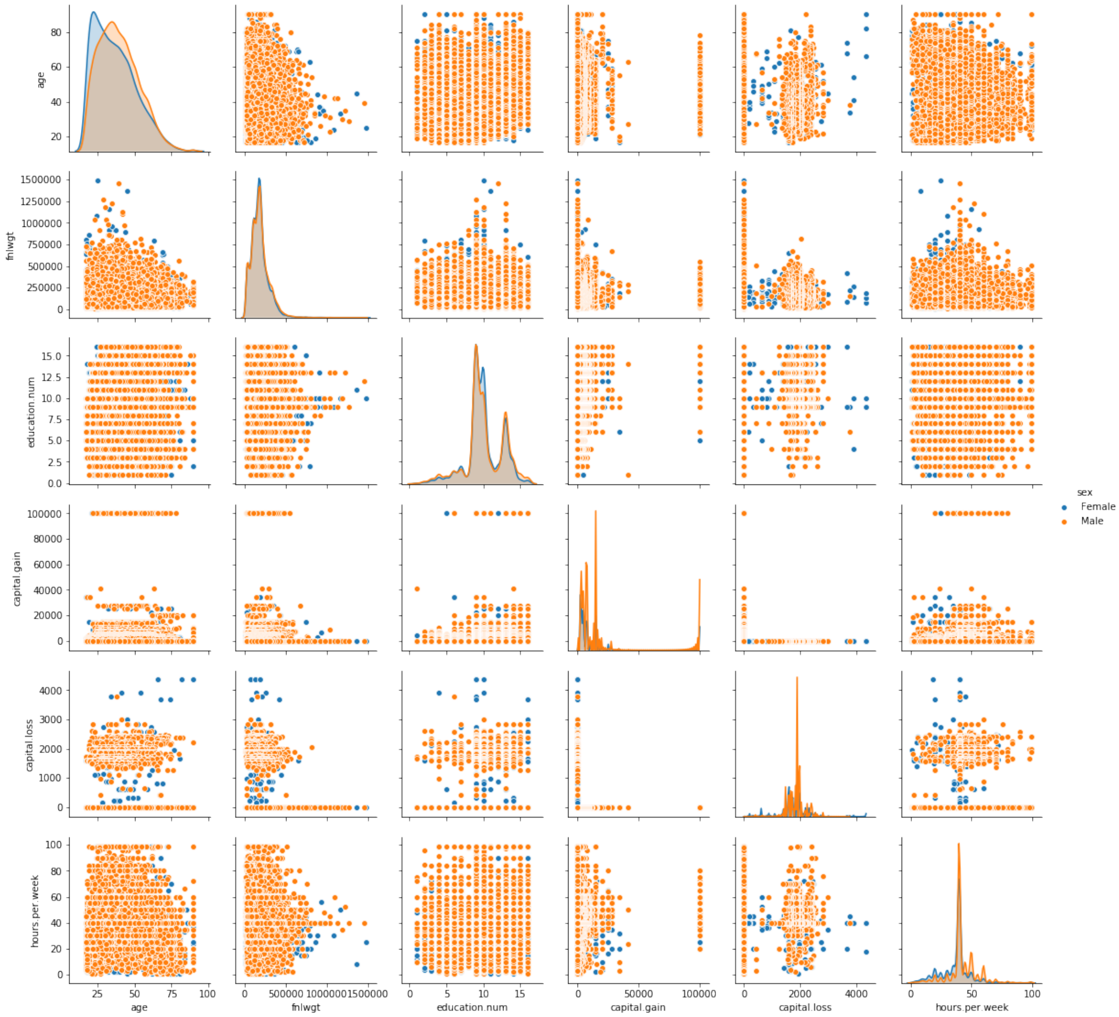
```
In [13]: num_feat = df.select_dtypes(include=['int64']).columns
         num_feat
```

```
Out[13]: Index(['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss',
                'hours.per.week'],
               dtype='object')
```

Plot pairwise relationships in a dataset.

```
In [14]: plt.figure(1, figsize=(16,10))
         sns.pairplot(data=df, hue='sex')
         plt.show()
```

```
<Figure size 1152x720 with 0 Axes>
```

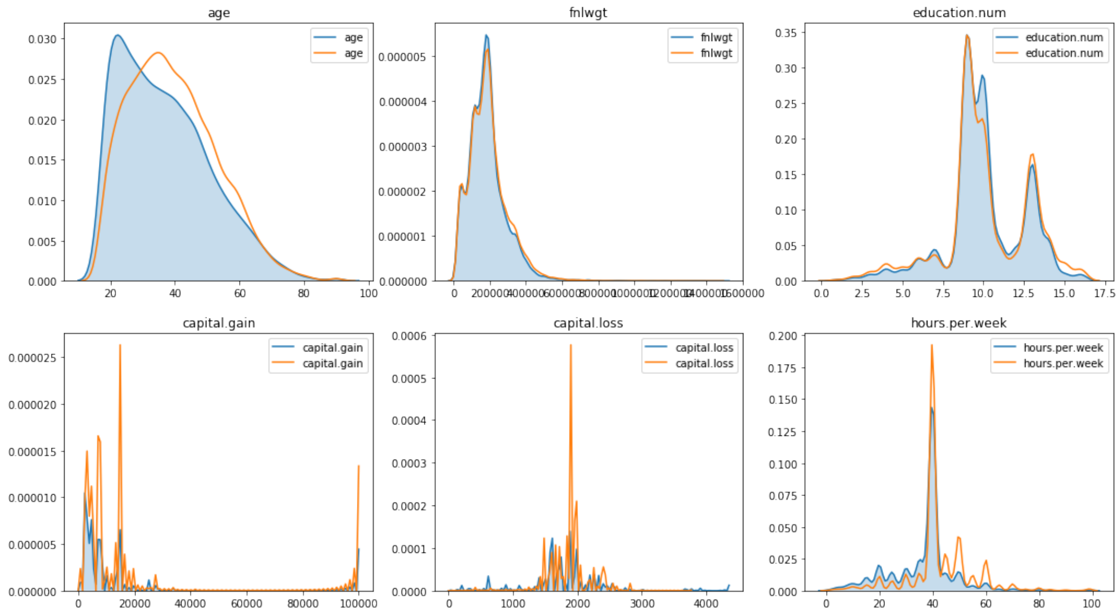


Distributions of numerical values

In [15]:
```
plt.figure(figsize=(18,10))
plt.subplot(231)

i=0
for c in num_feat:
    plt.subplot(2, 3, i+1)
    i += 1
    sns.kdeplot(df[df['sex'] == 'Female'][c], shade=True, )
    sns.kdeplot(df[df['sex'] == 'Male'][c], shade=False)
    plt.title(c)

plt.show()
```

```
/home/sunflowa/anaconda3/lib/python3.7/site-packages/matplotlib/figure.py:98: MatplotlibDeprec
Adding an axes using the same arguments as a previous axes currently reuses the earlier instanc
```

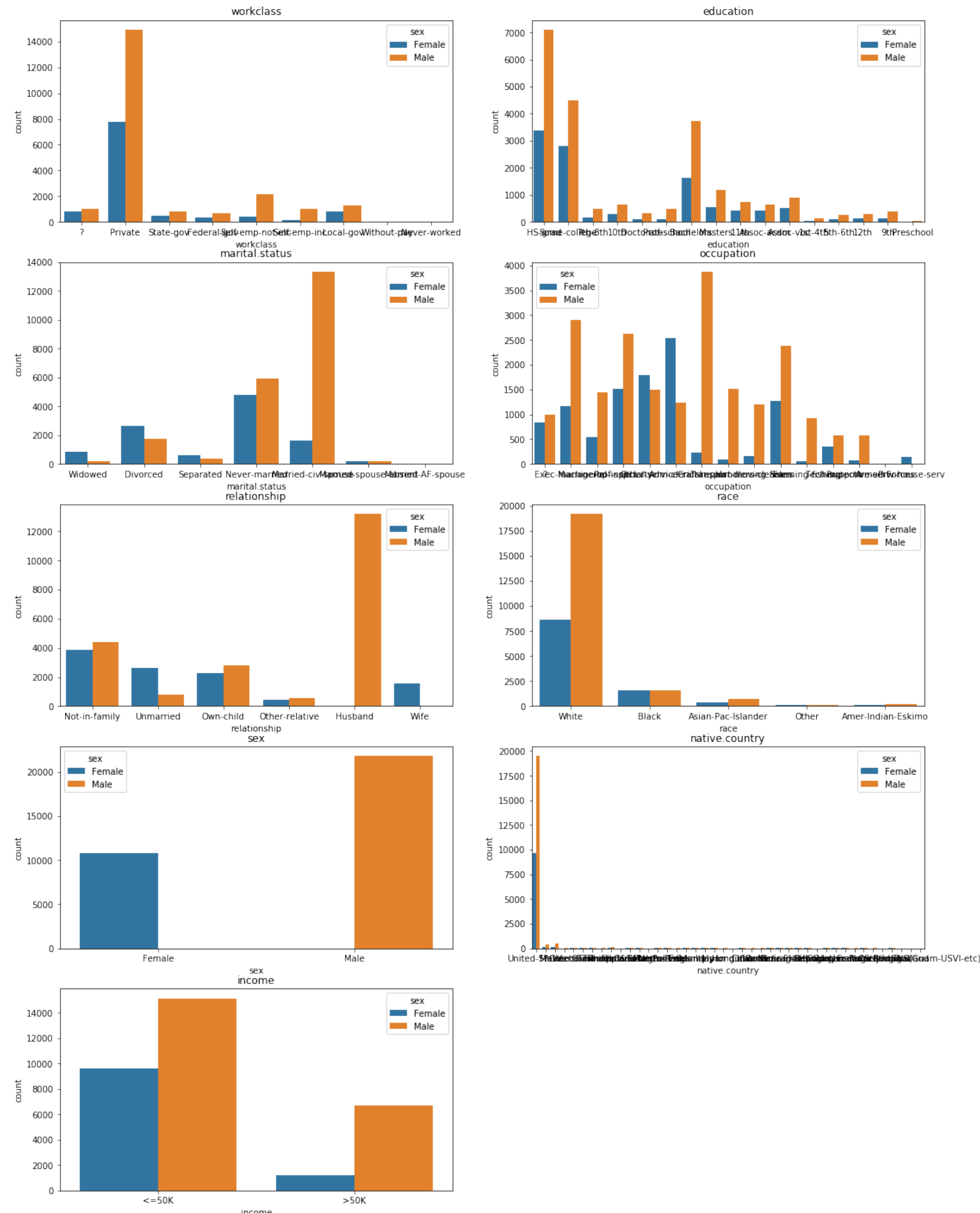
```
  "Adding an axes using the same arguments as a previous axes "
```



There are significant differences when it comes to capital gain / loss and hours per week.

In [16]:
```
plt.figure(figsize=(18,25))
plt.subplot(521)

i=0
for c in cat_feat:
    plt.subplot(5, 2, i+1)
    i += 1
    sns.countplot(x=c, data=df, hue='sex')
    plt.title(c)

plt.show()
```

```
/home/sunflowa/anaconda3/lib/python3.7/site-packages/matplotlib/figure.py:98: MatplotlibDeprec
Adding an axes using the same arguments as a previous axes currently reuses the earlier instanc
  "Adding an axes using the same arguments as a previous axes "
```



There are far more male earning >50k than female, but at the same time there are also more male earning <50k and even more males recorded in general. The counts need to be normalized.

```
In [17]: # nb of female / male
         nb_female = (df.sex == 'Female').astype('int').sum()
         nb_male = (df.sex == 'Male').astype('int').sum()
         nb_female, nb_male
```

Out[17]: (10762, 21775)

```
In [18]: # nb of people earning more or less than 50k per gender
         nb_male_above = len(df[(df.income == '>50K') & (df.sex == 'Male')])
         nb_male_below = len(df[(df.income == '<=50K') & (df.sex == 'Male')])
         nb_female_above = len(df[(df.income == '>50K') & (df.sex == 'Female')])
         nb_female_below = len(df[(df.income == '<=50K') & (df.sex == 'Female')])
         nb_male_above, nb_male_below, nb_female_above, nb_female_below
```

Out[18]: (6660, 15115, 1179, 9583)

```
In [19]: print(f'Among Males   : {nb_male_above/nb_male*100:.0f}% earn >50K // {nb_male_below/n
         print(f'Among Females : {nb_female_above/nb_female*100:.0f}% earn >50K // {nb_female_l
```

```
Among Males   : 31% earn >50K // 69% earn <=50K
Among Females : 11% earn >50K // 89% earn <=50K
```

```
In [20]: # normalization
         nb_male_above /= nb_male
         nb_male_below /= nb_male
         nb_female_above /= nb_female
         nb_female_below /= nb_female
         nb_male_above, nb_male_below, nb_female_above, nb_female_below
```

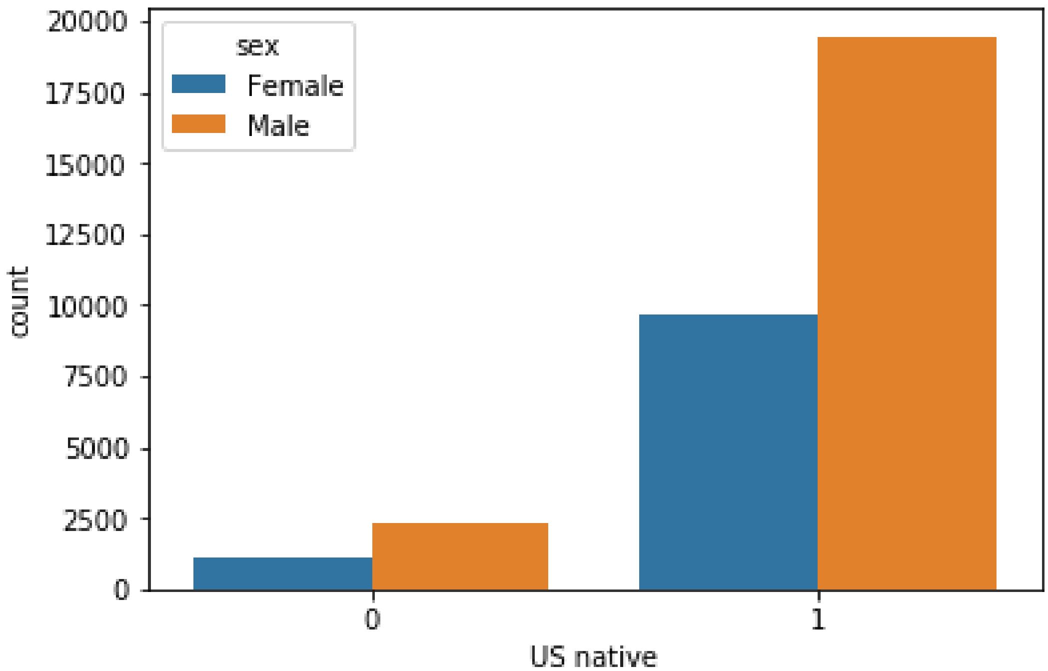
```
Out[20]: (0.3058553386911596,
          0.6941446613088404,
          0.1095521278572756,
          0.8904478721427244)
```

```
In [21]: print(f'Among people earning >50K  : {nb_male_above / (nb_male_above + nb_female_above
         print(f'Among people earning =<50K : {nb_male_below / (nb_male_below + nb_female_belo
```

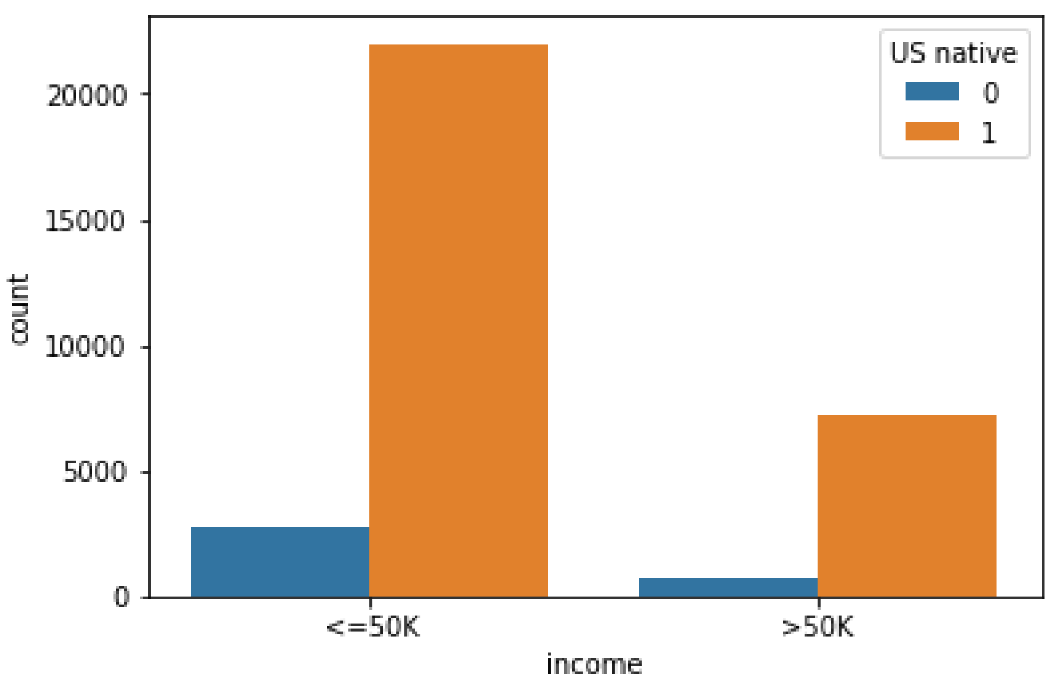
```
Among people earning >50K  : 74% are Females and 26% are Males
Among people earning =<50K : 44% are Females and 56% are Males
```

The distinction between american natives and the others should also be made

```
In [22]: df['US native'] = (df['native.country'] == 'United-States').astype('int')
         plt.figure(figsize=(6,4))
         sns.countplot(x='US native', data=df, hue='sex')
         plt.show()
```



In [23]:
```
plt.figure(figsize=(6,4))
sns.countplot(x='income', data=df, hue='US native')
plt.show()
```



```
In [24]: # nb of people earning more or less than 50k per origin
         nb_native_above = len(df[(df.income == '>50K') & (df['US native'] == 1)])
         nb_native_below = len(df[(df.income == '<=50K') & (df['US native'] == 1)])
         nb_foreign_above = len(df[(df.income == '>50K') & (df['US native'] == 0)])
         nb_foreign_below = len(df[(df.income == '<=50K') & (df['US native'] == 0)])
         nb_native_above, nb_native_below, nb_foreign_above, nb_foreign_below
```

```
Out[24]: (7169, 21984, 670, 2714)
```

```
In [25]: nb_native = (df['US native'] == 1).astype('int').sum()
         nb_foreign = df.shape[0] - nb_native
         nb_native, nb_foreign
```

```
Out[25]: (29153, 3384)
```

```
In [26]: print(f'Among natives    : {nb_native_above/nb_native*100:.0f}% earn >50K // {nb_nati
         print(f'Among foreigners : {nb_foreign_above/nb_foreign*100:.0f}% earn >50K // {nb_fo
```

```
Among natives    : 25% earn >50K // 75% earn <=50K
Among foreigners : 20% earn >50K // 80% earn <=50K
```

```
In [27]: # normalization
         nb_native_above /= nb_native
         nb_native_below /= nb_native
         nb_foreign_above /= nb_foreign
         nb_foreign_below /= nb_foreign
         nb_native_above, nb_native_below, nb_foreign_above, nb_foreign_below
```

```
Out[27]: (0.24590951188556923,
          0.7540904881144308,
          0.1979905437352246,
          0.8020094562647754)
```

```
In [28]: print(f'Among people earning >50K  : {nb_native_above / (nb_native_above + nb_foreign_
         print(f'Among people earning =<50K : {nb_native_below / (nb_native_below + nb_foreign_
```

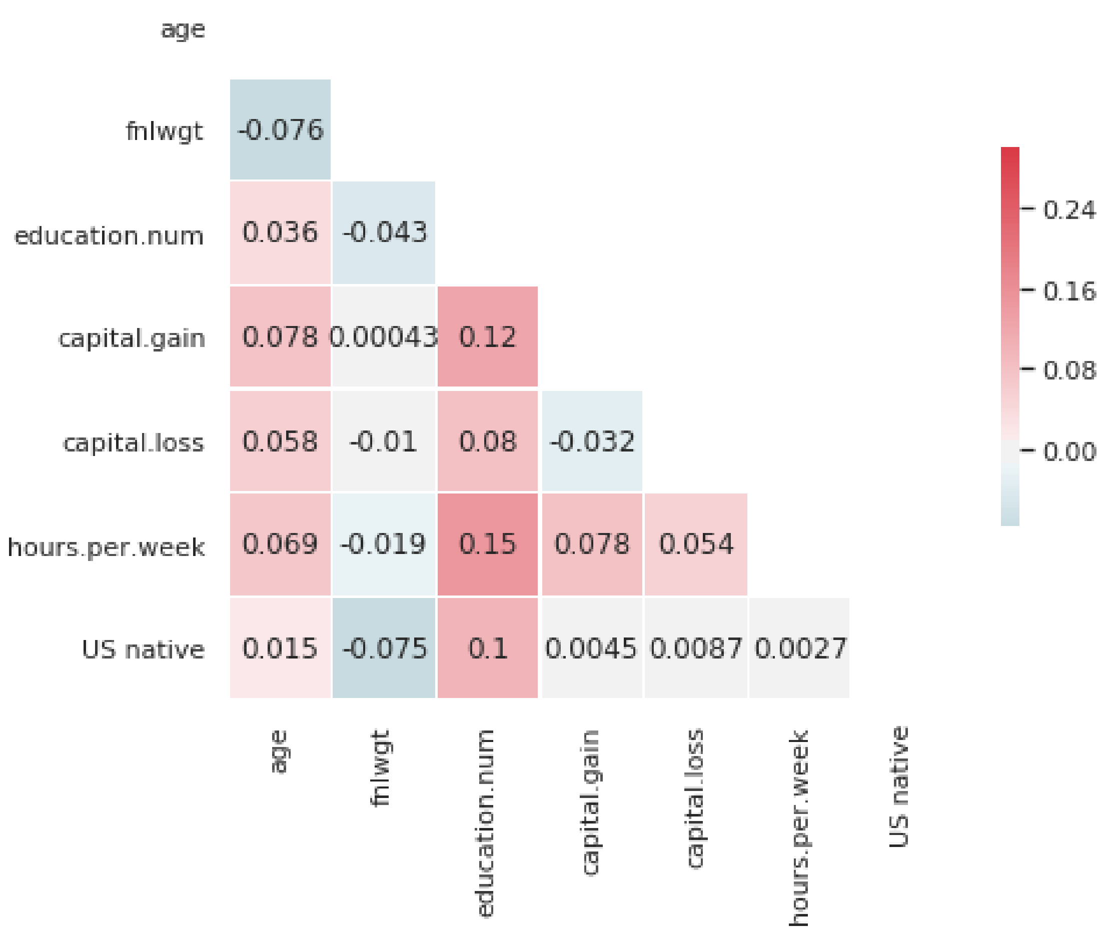
```
Among people earning >50K  : 55% are natives and 45% are foreigners
Among people earning =<50K : 48% are natives and 52% are foreigners
```

```
In [29]: num_feat = df.select_dtypes(include=['float', 'int']).columns
         num_feat
```

```
Out[29]: Index(['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss',
                'hours.per.week', 'US native'],
               dtype='object')
```

In [30]:
```
sns.set(style="white")

# Compute the correlation matrix
corr = df[num_feat].corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(7, 6))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, annot=True, cbar_kws={"shrink": .5})
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe274a74710>
```



4 Preparing data

```
In [31]: df['income'] = pd.get_dummies(df['income'], prefix='income', drop_first=True)

In [32]: y = df.income
         df = df.drop(columns=['income'])

In [33]: print(f'Ratio above 50k:  {y.sum()/len(y)*100:.2f}%')

Ratio above 50k:  24.09%


In [34]: #cat_columns = ['workclass', 'education', 'marital-status', 'occupation', 'relationsh

In [35]: #df_clean['sex'] = df_clean['sex'].str.replace('Female', '0').str.replace('Male', '1'

In [36]: df.head()

Out[36]:    age workclass  fnlwgt     education  education.num marital.status  \
         0   90         ?   77053       HS-grad              9        Widowed
         1   82   Private  132870       HS-grad              9        Widowed
         2   66         ?  186061  Some-college             10        Widowed
         3   54   Private  140359       7th-8th              4       Divorced
         4   41   Private  264663  Some-college             10      Separated

                   occupation   relationship   race     sex  capital.gain  \
         0                  ?  Not-in-family  White  Female             0
         1    Exec-managerial  Not-in-family  White  Female             0
         2                  ?      Unmarried  Black  Female             0
         3  Machine-op-inspct      Unmarried  White  Female             0
         4     Prof-specialty      Own-child  White  Female             0

            capital.loss  hours.per.week native.country  US native
         0          4356              40  United-States          1
         1          4356              18  United-States          1
         2          4356              40  United-States          1
         3          3900              40  United-States          1
         4          3900              40  United-States          1

In [37]: cols = list(df.columns)
         cols
```

```
Out[37]: ['age',
          'workclass',
          'fnlwgt',
          'education',
          'education.num',
          'marital.status',
          'occupation',
          'relationship',
          'race',
          'sex',
          'capital.gain',
          'capital.loss',
          'hours.per.week',
          'native.country',
          'US native']
```

```
In [38]: selected_feat = cols.copy()
         selected_feat.remove('US native')
         selected_feat
```

```
Out[38]: ['age',
          'workclass',
          'fnlwgt',
          'education',
          'education.num',
          'marital.status',
          'occupation',
          'relationship',
          'race',
          'sex',
          'capital.gain',
          'capital.loss',
          'hours.per.week',
          'native.country']
```

```
In [42]: df_final = df[selected_feat]
```

```
In [43]: cat_feat = df_final.select_dtypes(include=['object']).columns
         X = pd.get_dummies(df_final[cat_feat], drop_first=True)
```

```
In [44]: #X = pd.concat([df_final[continuous_columns], df_dummies], axis=1)
```

```
In [45]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

---

5 Model training and predictions

Choose carefully your features, you can even create new ones, and focus on training Decision Trees and Random Forests.

Improve your models by finding the optimal hyperparameters that minimize your models' error.

```
In [46]: def print_score(model, name):
             model.fit(X_train, y_train)
             print('Accuracy score of the', name, f': on train = {model.score(X_train, y_train
```

5.1 Baseline LogisticRegression

```
In [47]: print_score(LogisticRegression(), 'LogisticReg')
```

```
Accuracy score of the LogisticReg : on train = 83.61%, on test = 81.78%
```

5.2 Decision Tree

```
In [48]: print_score(DecisionTreeClassifier(), 'DecisionTreeClf')
```

```
Accuracy score of the DecisionTreeClf : on train = 87.14%, on test = 80.19%
```

5.3 Random Forest

```
In [49]: rf = RandomForestClassifier().fit(X_train, y_train)
         print(f'Accuracy score of the RandomForrest: on train = {rf.score(X_train, y_train)*1
```

```
Accuracy score of the RandomForrest: on train = 86.83%, on test = 80.64%
```

5.4 ExtraTreesClassifier

```
In [50]: # fit an Extra Tree model to the data
         print_score(DecisionTreeClassifier(), 'ExtraTreesClf')
```

```
Accuracy score of the ExtraTreesClf : on train = 87.14%, on test = 80.26%
```

5.5 Tuned model

```
In [53]: rfc = RandomForestClassifier()
         param_grid = {
             'n_estimators': [50, 100, 150, 200, 250],
             'max_features': [1, 2, 3, 4, 5],
             'max_depth' : [4, 6, 8]
         }
```

```
In [54]: rfc_cv = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=5)
         rfc_cv.fit(X_train, y_train)
```

```
Out[54]: GridSearchCV(cv=5, error_score='raise-deprecating',
                estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion=
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'n_estimators': [50, 100, 150, 200, 250], 'max_features': [1, 2, 3
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring=None, verbose=0)
```

```
In [55]: rfc_cv.best_params_
```

```
Out[55]: {'max_depth': 8, 'max_features': 5, 'n_estimators': 250}
```

```
In [57]: rfc_best = RandomForestClassifier(max_depth=8, max_features=5, n_estimators=250).fit(X
         print(f'Accuracy score of the RandomForrest: on train = {rfc_best.score(X_train, y_tra
```

```
Accuracy score of the RandomForrest: on train = 80.54%, on test = 79.78%
```

6 Profiling

Let's find clear insights on the profiles of the people that make more than USD 50K a year. Which features seem to be the most correlated with this phenomenon.

6.1 Based on the rf model

```
In [58]: # indexes of columns which are the most important
         np.argsort(rf.feature_importances_)[-16:]
```

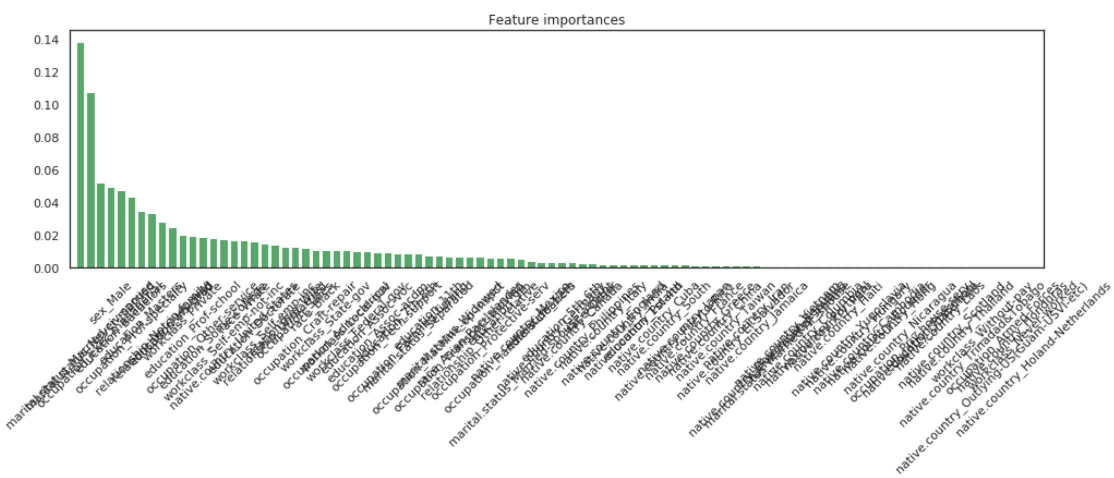
```
Out[58]: array([91,  5, 22, 36, 21,  3, 18, 45, 43, 19, 38, 16, 52, 32, 26, 24])
```

```
In [59]: # most important features
         [list(X.columns)[i] for i in np.argsort(rf.feature_importances_)[-16:]][::-1]
```

```
Out[59]: ['marital.status_Married-civ-spouse',
          'marital.status_Never-married',
          'occupation_Exec-managerial',
          'sex_Male',
          'education_Bachelors',
          'occupation_Prof-specialty',
          'education_Masters',
          'relationship_Not-in-family',
          'relationship_Own-child',
          'education_HS-grad',
          'workclass_Private',
```

```
'education_Prof-school',
 'occupation_Other-service',
 'education_Some-college',
 'workclass_Self-emp-not-inc',
 'native.country_United-States']
```

```
In [60]: # Feature importances
         features = X.columns
         importances = rf.feature_importances_
         indices = np.argsort(importances)[::-1]
         num_features = len(importances)

         # Plot the feature importances of the tree
         plt.figure(figsize=(16, 4))
         plt.title("Feature importances")
         plt.bar(range(num_features), importances[indices], color="g", align="center")
         plt.xticks(range(num_features), [features[i] for i in indices], rotation='45')
         plt.xlim([-1, num_features])
         plt.show()

         # Print values
         for i in indices:
             print ("{0} - {1:.3f}".format(features[i], importances[i]))
```



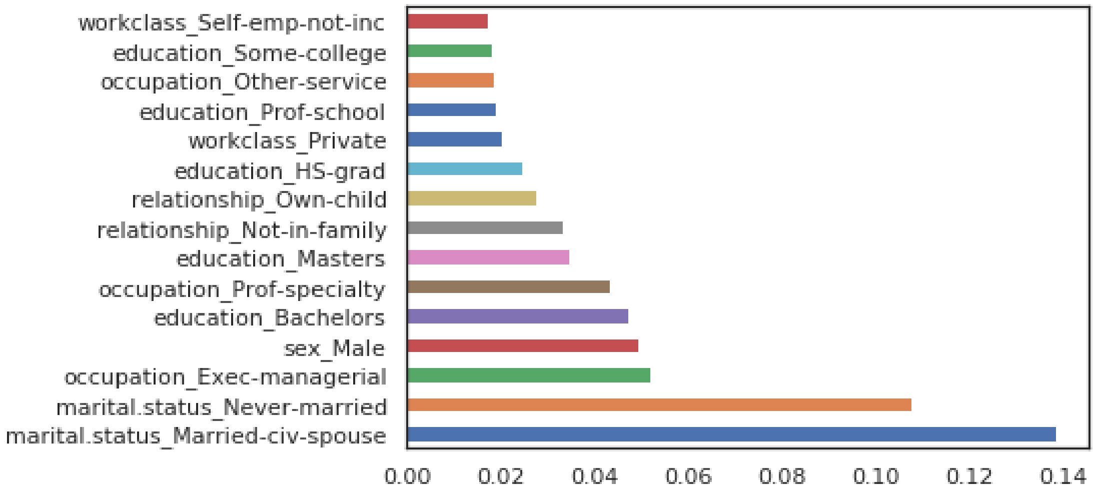

```
marital.status_Married-civ-spouse - 0.139
marital.status_Never-married - 0.108
occupation_Exec-managerial - 0.052
sex_Male - 0.050
education_Bachelors - 0.047
occupation_Prof-specialty - 0.043
education_Masters - 0.034
```

```
relationship_Not-in-family - 0.033
relationship_Own-child - 0.028
education_HS-grad - 0.025
workclass_Private - 0.020
education_Prof-school - 0.019
occupation_Other-service - 0.019
education_Some-college - 0.018
workclass_Self-emp-not-inc - 0.017
native.country_United-States - 0.017
race_White - 0.017
education_Doctorate - 0.016
workclass_Self-emp-inc - 0.015
relationship_Unmarried - 0.014
relationship_Wife - 0.013
occupation_Sales - 0.013
occupation_Craft-repair - 0.012
race_Black - 0.011
workclass_State-gov - 0.011
occupation_Adm-clerical - 0.011
workclass_Local-gov - 0.011
workclass_Federal-gov - 0.010
education_Assoc-voc - 0.010
education_Assoc-acdm - 0.010
occupation_Tech-support - 0.010
education_7th-8th - 0.009
occupation_Farming-fishing - 0.009
marital.status_Separated - 0.008
education_11th - 0.007
occupation_Machine-op-inspct - 0.007
marital.status_Widowed - 0.007
occupation_Transport-moving - 0.007
race_Asian-Pac-Islander - 0.007
relationship_Other-relative - 0.007
occupation_Protective-serv - 0.006
education_9th - 0.006
occupation_Handlers-cleaners - 0.006
native.country_Mexico - 0.005
marital.status_Married-spouse-absent - 0.004
education_12th - 0.004
native.country_Germany - 0.003
education_5th-6th - 0.003
native.country_Canada - 0.003
native.country_Philippines - 0.003
race_Other - 0.003
native.country_Italy - 0.002
native.country_England - 0.002
native.country_India - 0.002
native.country_Poland - 0.002
```

```
education_1st-4th - 0.002
native.country_Cuba - 0.002
native.country_South - 0.002
native.country_Puerto-Rico - 0.002
native.country_Japan - 0.002
native.country_France - 0.001
native.country_Greece - 0.001
native.country_China - 0.001
native.country_Taiwan - 0.001
native.country_El-Salvador - 0.001
native.country_Iran - 0.001
native.country_Jamaica - 0.001
native.country_Dominican-Republic - 0.001
marital.status_Married-AF-spouse - 0.001
native.country_Vietnam - 0.001
native.country_Ireland - 0.001
native.country_Columbia - 0.001
native.country_Portugal - 0.001
native.country_Peru - 0.001
native.country_Haiti - 0.001
native.country_Yugoslavia - 0.001
native.country_Hungary - 0.001
native.country_Cambodia - 0.000
native.country_Ecuador - 0.000
native.country_Hong - 0.000
native.country_Nicaragua - 0.000
occupation_Priv-house-serv - 0.000
native.country_Guatemala - 0.000
education_Preschool - 0.000
native.country_Laos - 0.000
native.country_Scotland - 0.000
native.country_Thailand - 0.000
native.country_Trinadad&Tobago - 0.000
workclass_Without-pay - 0.000
native.country_Outlying-US(Guam-USVI-etc) - 0.000
occupation_Armed-Forces - 0.000
native.country_Honduras - 0.000
workclass_Never-worked - 0.000
native.country_Holand-Netherlands - 0.000
```

```
In [61]: (pd.Series(rf.feature_importances_, index=X_train.columns)
            .nlargest(15)
            .plot(kind='barh'))
```

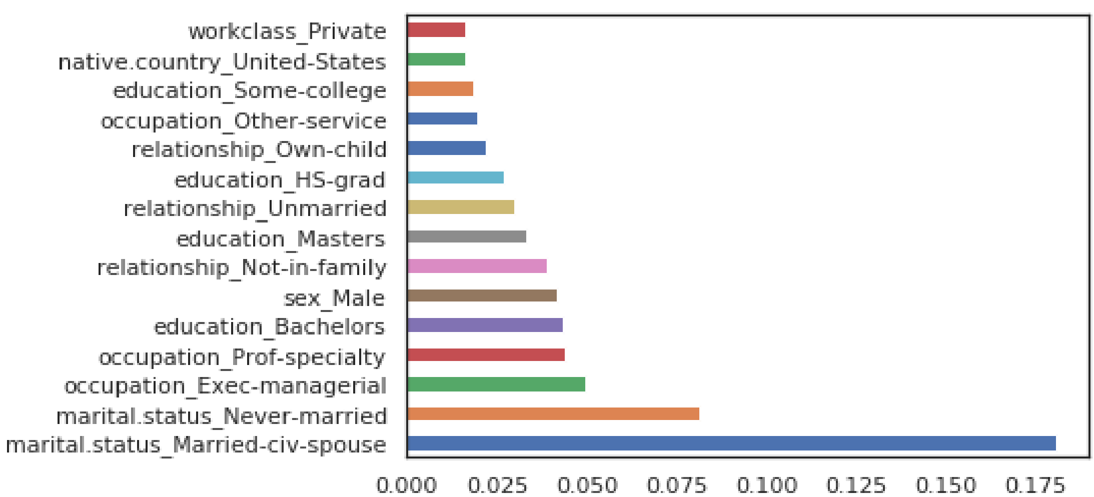
```
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe2780ecef0>
```



6.2 Based on the ExtraTree model

In [62]:
```
extree = ExtraTreesClassifier().fit(X_train, y_train)
(pd.Series(extree.feature_importances_, index=X_train.columns)
   .nlargest(15)
   .plot(kind='barh'))
```

Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe277f0f7f0>



The same features come first.