Census_ensemble_methods

November 19, 2022

Import necessary Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
data = pd.read_csv('adult.csv',sep=',')
```

[3]:
```
data.head()
```

[3]:
```
   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
```

Data Pre-Processing & Statistics

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[5]:
```
data.describe(include='all')
```

[5]:
```
                 age workclass        fnlwgt education  education.num  \
count   32561.000000     32561  3.256100e+04     32561   32561.000000
unique           NaN         9           NaN        16            NaN
top              NaN   Private           NaN   HS-grad            NaN
freq             NaN     22696           NaN     10501            NaN
mean       38.581647       NaN  1.897784e+05       NaN      10.080679
std        13.640433       NaN  1.055500e+05       NaN       2.572720
min        17.000000       NaN  1.228500e+04       NaN       1.000000
25%        28.000000       NaN  1.178270e+05       NaN       9.000000
50%        37.000000       NaN  1.783560e+05       NaN      10.000000
75%        48.000000       NaN  2.370510e+05       NaN      12.000000
max        90.000000       NaN  1.484705e+06       NaN      16.000000

            marital.status      occupation relationship   race    sex  \
count                32561           32561        32561  32561  32561
unique                   7              15            6      5      2
top     Married-civ-spouse  Prof-specialty      Husband  White   Male
freq                 14976            4140        13193  27816  21790
mean                   NaN             NaN          NaN    NaN    NaN
std                    NaN             NaN          NaN    NaN    NaN
min                    NaN             NaN          NaN    NaN    NaN
25%                    NaN             NaN          NaN    NaN    NaN
50%                    NaN             NaN          NaN    NaN    NaN
75%                    NaN             NaN          NaN    NaN    NaN
max                    NaN             NaN          NaN    NaN    NaN
```

	capital.gain	capital.loss	hours.per.week	native.country	income
count	32561.000000	32561.000000	32561.000000	32561	32561
unique	NaN	NaN	NaN	42	2
top	NaN	NaN	NaN	United-States	<=50K
freq	NaN	NaN	NaN	29170	24720
mean	1077.648844	87.303830	40.437456	NaN	NaN
std	7385.292085	402.960219	12.347429	NaN	NaN
min	0.000000	0.000000	1.000000	NaN	NaN
25%	0.000000	0.000000	40.000000	NaN	NaN
50%	0.000000	0.000000	40.000000	NaN	NaN
75%	0.000000	0.000000	45.000000	NaN	NaN
max	99999.000000	4356.000000	99.000000	NaN	NaN

[6]:
```
data.describe().T
```

[6]:

	count	mean	std	min	25%	\
age	32561.0	38.581647	13.640433	17.0	28.0	
fnlwgt	32561.0	189778.366512	105549.977697	12285.0	117827.0	
education.num	32561.0	10.080679	2.572720	1.0	9.0	
capital.gain	32561.0	1077.648844	7385.292085	0.0	0.0	
capital.loss	32561.0	87.303830	402.960219	0.0	0.0	
hours.per.week	32561.0	40.437456	12.347429	1.0	40.0	

	50%	75%	max
age	37.0	48.0	90.0
fnlwgt	178356.0	237051.0	1484705.0
education.num	10.0	12.0	16.0
capital.gain	0.0	0.0	99999.0
capital.loss	0.0	0.0	4356.0
hours.per.week	40.0	45.0	99.0

[7]:
```
data.isnull().sum()
```

[7]:
```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
```

```
income            0
dtype: int64
```

[8]:
```
data.isna().sum()
```

[8]:
```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

[9]:
```
# Check for '?' in dataset
round((data.isin(['?']).sum() / data.shape[0])
      * 100, 2).astype(str) + ' %'
```

[9]:
```
age                0.0 %
workclass         5.64 %
fnlwgt             0.0 %
education          0.0 %
education.num      0.0 %
marital.status     0.0 %
occupation        5.66 %
relationship       0.0 %
race               0.0 %
sex                0.0 %
capital.gain       0.0 %
capital.loss       0.0 %
hours.per.week     0.0 %
native.country    1.79 %
income             0.0 %
dtype: object
```

[10]:
```
# Checking the counts of label categories
income = data['income'].value_counts(normalize=True)
round(income * 100, 2).astype('str') + ' %'
```

[10]:
```
<=50K    75.92 %
>50K     24.08 %
Name: income, dtype: object
```

[11]:
```python
data.corr()
```

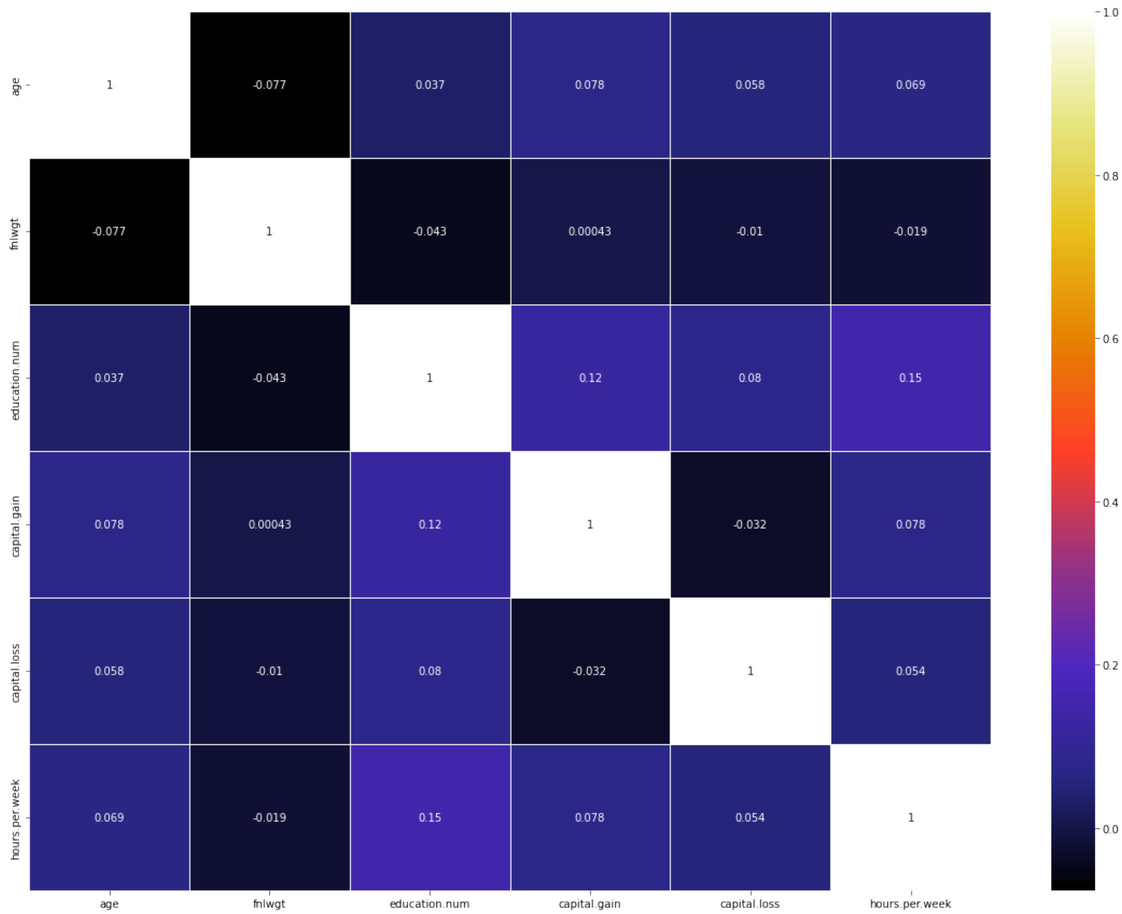
[11]:
```
                     age   fnlwgt  education.num  capital.gain  capital.loss  \
age             1.000000 -0.076646       0.036527      0.077674      0.057775
fnlwgt         -0.076646  1.000000      -0.043195      0.000432     -0.010252
education.num   0.036527 -0.043195       1.000000      0.122630      0.079923
capital.gain    0.077674  0.000432       0.122630      1.000000     -0.031615
capital.loss    0.057775 -0.010252       0.079923     -0.031615      1.000000
hours.per.week  0.068756 -0.018768       0.148123      0.078409      0.054256

                hours.per.week
age                   0.068756
fnlwgt               -0.018768
education.num         0.148123
capital.gain          0.078409
capital.loss          0.054256
hours.per.week        1.000000
```

Heatmap

[12]:
```python
plt.figure(figsize=(20,15))
sns.heatmap(data.corr(),annot= True,linewidths=1, linecolor="white", cbar=True,
            cmap = "CMRmap",xticklabels="auto", yticklabels="auto")

plt.savefig('heatmap.png')
```



[13]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[14]:
```
data['income'] = le.fit_transform(data['income'])
```

[15]:
```
columns_with_nan = ['workclass', 'occupation', 'native.country']
```

[16]:
```
for col in columns_with_nan:
    data[col].fillna(data[col].mode()[0], inplace=True)
```

[17]:
```
from sklearn.preprocessing import LabelEncoder
for col in data.columns:
    if data[col].dtypes == 'object':
        encoder = LabelEncoder()
        data[col] = encoder.fit_transform(data[col])
```

[18]:
```
corr_data = data.corr('spearman').stack().reset_index(name='corr')
corr_data.loc[corr_data['corr'] == 1, 'corr'] = 0  # Remove diagonal
# Use abs so that we can visualize the impact of negative correaltion
```

```
corr_data['abs'] = corr_data['corr'].abs()
corr_data.sort_values('abs', ascending=False).head(n=5)
```

[18]:

	level_0	level_1	corr	abs
142	sex	relationship	-0.617570	0.617570
114	relationship	sex	-0.617570	0.617570
5	age	marital.status	-0.374850	0.374850
75	marital.status	age	-0.374850	0.374850
217	income	relationship	-0.329913	0.329913

[19]:
```
X = data.drop('income', axis=1)
Y = data['income']
```

[20]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3,␣
↪random_state = 42)
```

Model Building & Model Training

[21]:
```
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.tree import DecisionTreeClassifier
```

[22]:
```
extra_model = ExtraTreeClassifier()
extra_model.fit(X,Y)
feature_imp = extra_model.feature_importances_
```

[23]:
```
for index, val in enumerate(feature_imp):
    print(index, round((val * 100), 2))
```

```
0 13.63
1 4.04
2 16.31
3 3.5
4 9.02
5 1.41
6 7.23
7 14.86
8 1.56
9 5.0
10 9.06
11 2.75
12 9.78
13 1.85
```

[24]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  int32
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  int32
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  int32
 6   occupation      32561 non-null  int32
 7   relationship    32561 non-null  int32
 8   race            32561 non-null  int32
 9   sex             32561 non-null  int32
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  int32
dtypes: int32(8), int64(6)
memory usage: 2.5 MB
```

[25]:
```
X.columns
```

[25]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education.num',
       'marital.status', 'occupation', 'relationship', 'race', 'sex',
       'capital.gain', 'capital.loss', 'hours.per.week', 'native.country'],
      dtype='object')
```

[26]:
```
X = X.drop(['workclass', 'education', 'race', 'sex',
            'capital.loss', 'native.country'], axis=1)
```

[27]:
```
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
```

[27]:
```
RandomForestClassifier()
```

[28]:
```
y_pred = rf.predict(X_test)
```

RF Accuracy

[29]:
```
accuracy_score(y_test,y_pred)
```

[29]:
```
0.8566895280990889
```

Bagging Accuracy

[31]:
```
cls = BaggingClassifier(rf, random_state=0).fit(X_train, y_train)
cls.score(X_test, y_test)
```

[31]: 0.858941549800389

Extra Tree Classifier Accuracy

[32]:
```
extra_tree = ExtraTreeClassifier(random_state=0)
cls_extra = BaggingClassifier(extra_tree, random_state=0).fit(X_train, y_train)
cls_extra.score(X_test, y_test)
```

[32]: 0.8418466577950661

[33]:
```
clf1 = ExtraTreeClassifier(random_state=0)
clf2 = RandomForestClassifier()
clf3 = DecisionTreeClassifier()
```

Voting Classifier

[34]:
```
eclf1 = VotingClassifier(estimators=[
    ('ETC', clf1), ('RF', clf2), ('DT', clf3),('Bagging',cls)], voting='hard')
```

[35]:
```
eclf1 = eclf1.fit(X, Y)
print(eclf1.predict(X))
```

```
[0 0 0 … 1 0 0]
```

[36]:
```
eclf2 = VotingClassifier(estimators=[
    ('ETC', clf1), ('RF', clf2), ('DT', clf3),('Bagging',cls)], voting='soft')
```

[37]:
```
eclf2 = eclf2.fit(X, Y)
print(eclf2.predict(X))
```

```
[0 0 0 … 1 0 0]
```

[ ]:
```

```