Income_Predictor

September 29, 2020

1 Income Classifier

This Projects involves using census data from 1994 to explore if there is a way to accuratley predict if someone makes below or above 50k a year. In total there are 14 independent variables and roughly 45,000 data points between train and test sets. This project showcases preprocessing, data explorating, modeling, and model interpretation utilizing PyCaret, seaborn, and sklearn

[96]:
```
from pycaret.classification import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import sys
```

[14]:
```
#Read in Data
data = pd.read_csv('au_train.csv')
data.head()
```

[14]:
```
   age         workclass  fnlwgt  education  education-num  \
0   39         State-gov   77516  Bachelors             13
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
4   28           Private  338409  Bachelors             13

       marital-status         occupation   relationship   race     sex  \
0       Never-married       Adm-clerical  Not-in-family  White    Male
1  Married-civ-spouse    Exec-managerial        Husband  White    Male
2            Divorced  Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse     Prof-specialty           Wife  Black  Female

   capital-gain  capital-loss  hours-per-week  native-country  class
```

```
0          2174             0             40   United-States       0
1             0             0             13   United-States       0
2             0             0             40   United-States       0
3             0             0             40   United-States       0
4             0             0             40            Cuba       0
```

[122]:
```
#look at statistics of our numerical data. Validate data makes sense in context.
data.describe()
```

[122]:
```
                age   capital-gain  capital-loss  hours-per-week        output
count  32561.000000   32561.000000  32561.000000    32561.000000  32561.000000
mean      38.581647    1077.648844     87.303830       40.437456      0.240810
std       13.640433    7385.292085    402.960219       12.347429      0.427581
min       17.000000       0.000000      0.000000        1.000000      0.000000
25%       28.000000       0.000000      0.000000       40.000000      0.000000
50%       37.000000       0.000000      0.000000       40.000000      0.000000
75%       48.000000       0.000000      0.000000       45.000000      0.000000
max       90.000000   99999.000000   4356.000000       99.000000      1.000000
```

[15]:
```
#rename class to default to avoid python class initiation
data.rename(columns = {'class' :'output'}, inplace= True)

#Drop education-num as it is the same as education/drop fnlwgt as it is not␣
↪relevant
data.drop(columns=['education-num', 'fnlwgt'],inplace = True)
```

[16]:
```
#Change Target values for visualizations to >50k and <50k
data['output'] = data['output'].replace({0:'<50k'})
data['output'] = data['output'].replace({1:'>50k'})


#Target distribution
data['output'].value_counts()
```

[16]:
```
<50k    24720
>50k     7841
Name: output, dtype: int64
```

[17]:
```
#Analyze missing data
total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum()/data.isnull().count()).
↪sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

[17]:
```
                Total   Percent
native-country    583  0.017905
```

```
output             0  0.000000
hours-per-week     0  0.000000
capital-loss       0  0.000000
capital-gain       0  0.000000
sex                0  0.000000
race               0  0.000000
relationship       0  0.000000
occupation         0  0.000000
marital-status     0  0.000000
education          0  0.000000
workclass          0  0.000000
age                0  0.000000
```

[18]:

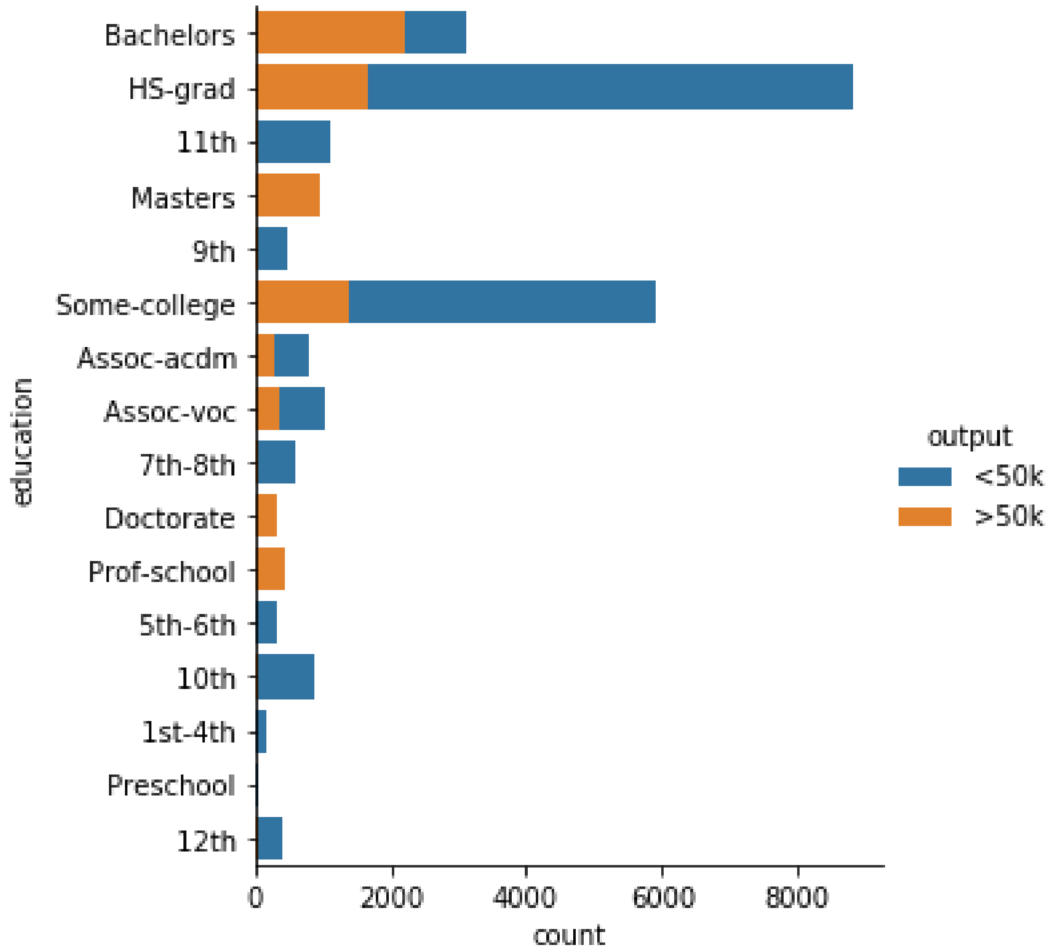
```
#Delete missing data where observations of na > 1.
#In this case we are dropping rows where native-country is na
data = data.drop((missing_data[missing_data['Total'] > 1]).index,1)

#Check to make sure there is no missing data
data.isnull().sum().max()
```

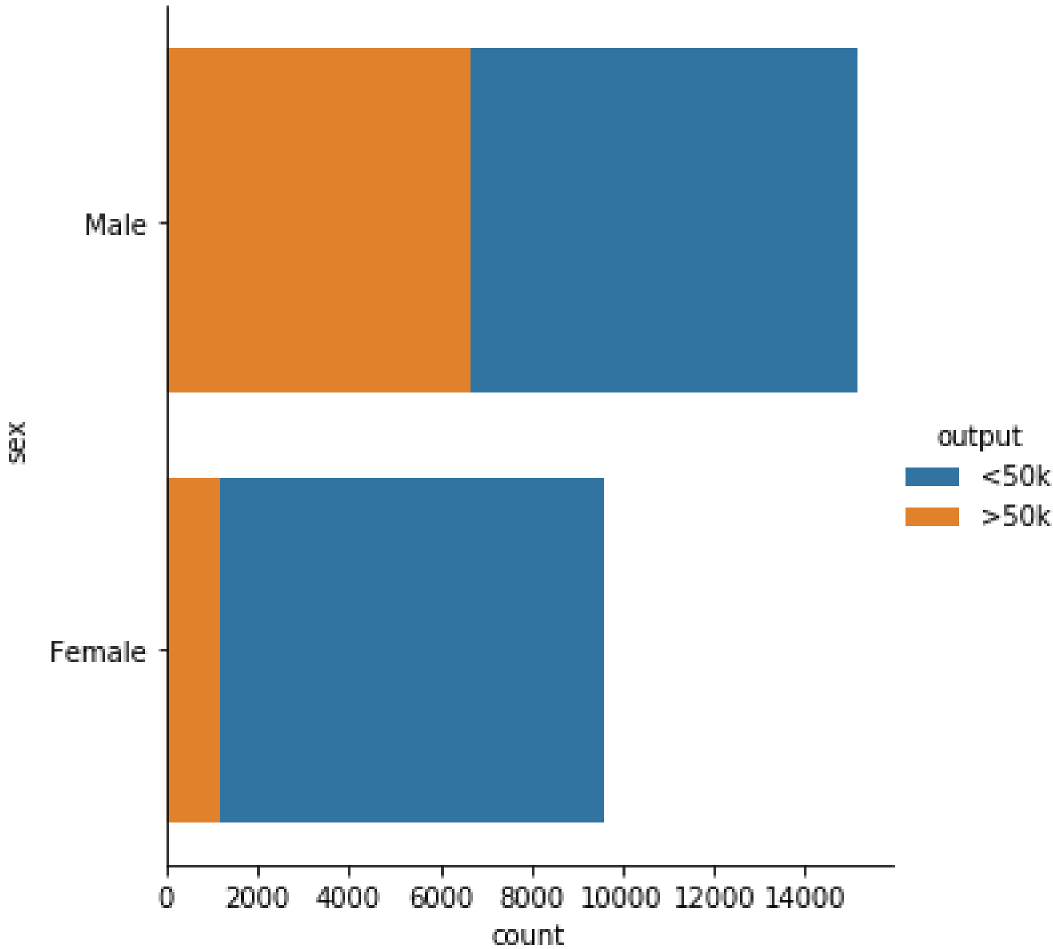
[18]: 0

[19]:

```
#Visualize Target variable by Education Type
var = 'education'
ax = sns.catplot(y = var, data=data, kind="count",hue='output',dodge=False)
```



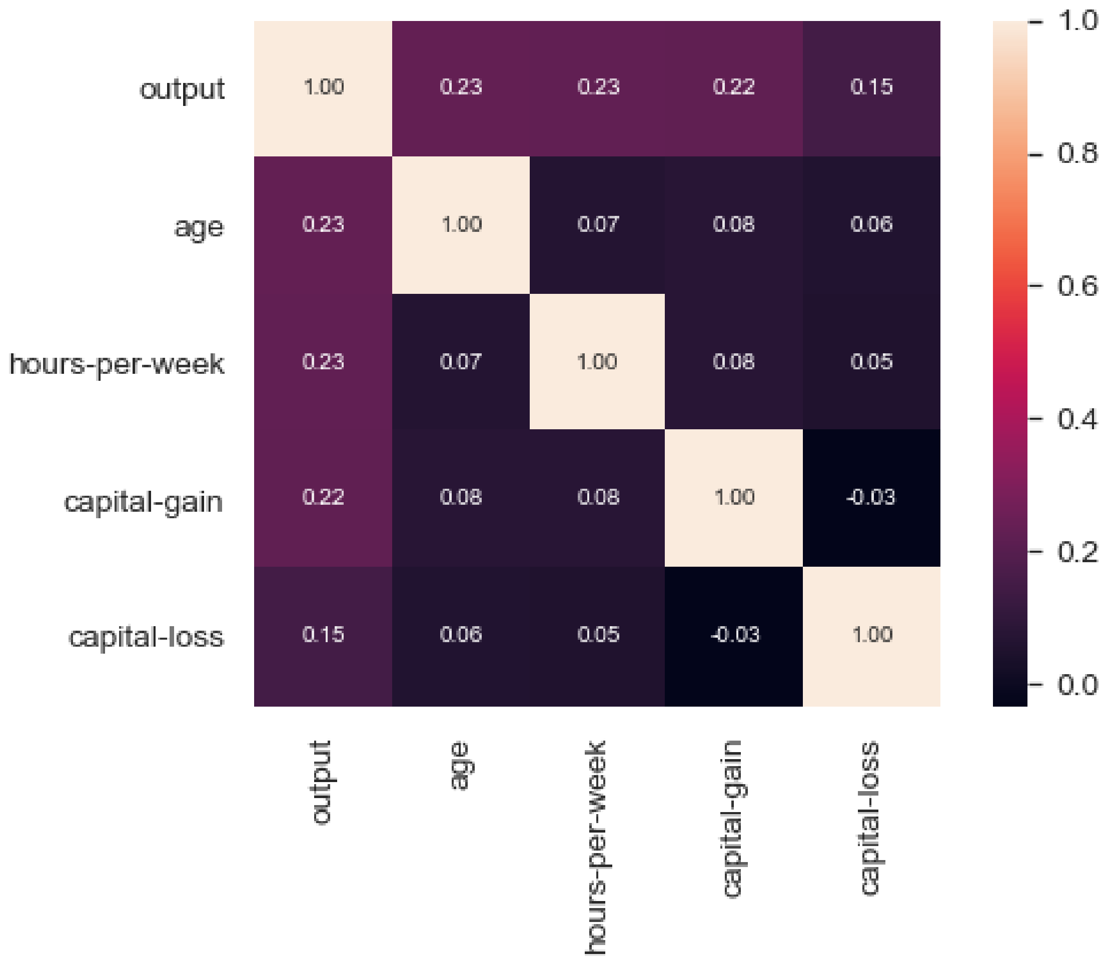
[20]:
```
#Visualize Target variable by Sex
var = 'sex'
ax = sns.catplot(y = var, data=data, kind="count",hue='output',dodge=False)
```



[95]:

```
#Correlation matrix table

k = 10 #number of variables for heatmap
corrmat = data.corr()
cols = corrmat.nlargest(k, 'output')['output'].index
cm = np.corrcoef(data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',␣
 ↪annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```



[21]:
```
#Change target back to 0 and 1 before modeling

data['output'] = data['output'].replace({'<50k':0})
data['output'] = data['output'].replace({'>50k':1})
```

[22]:
```
#Setup Pycaret

ctl = setup(data=data, target = 'output',use_gpu = True, normalize=True )
```

```
Setup Succesfully Completed!

<pandas.io.formats.style.Styler at 0x1e6c7010d48>
```

[23]:
```
#Run 14 Classification models on 10 fold cross validation


compare_models(fold=10)
```

<pandas.io.formats.style.Styler at 0x1e6c5c6b208>

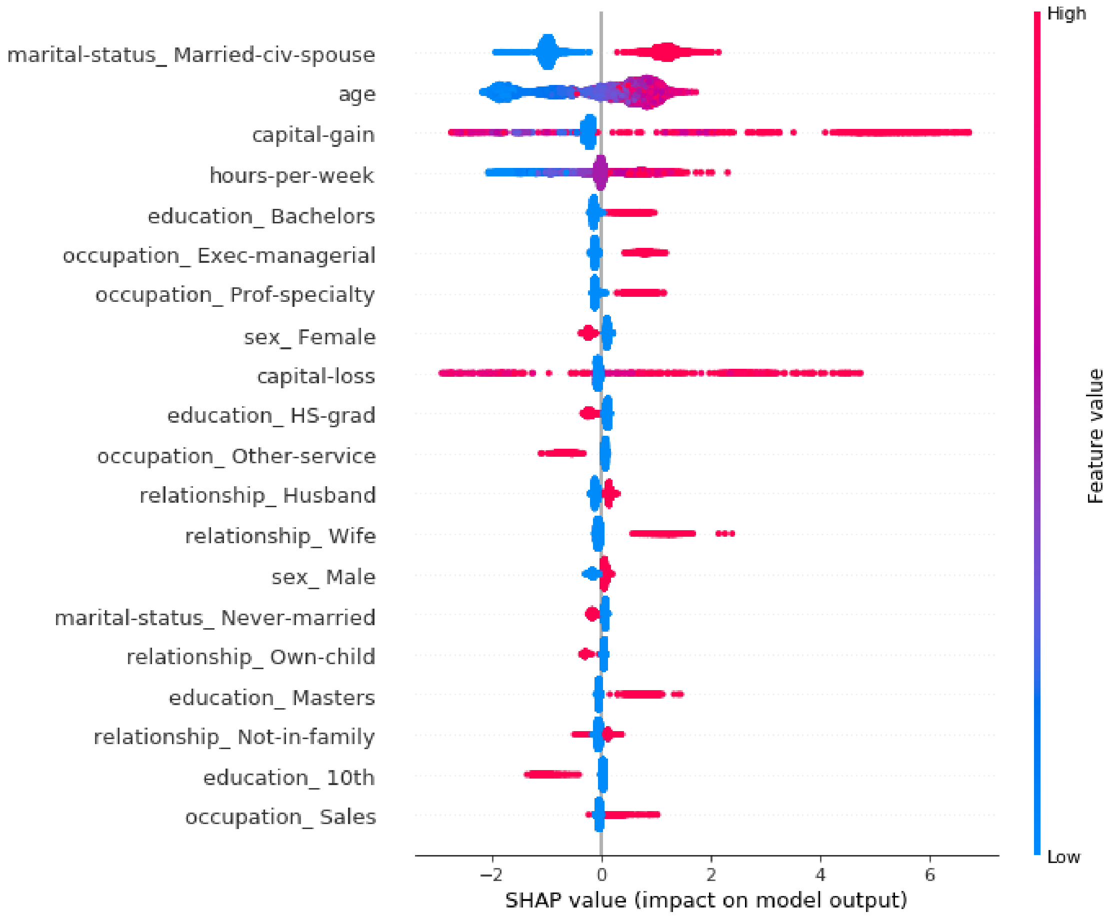
[23]: <catboost.core.CatBoostClassifier at 0x1e6c410d208>

[24]:
```
#We decide to go with the Catboost Classifier as it maximizes AUC and F1.
CBR = create_model('catboost')
```

<pandas.io.formats.style.Styler at 0x1e6c5da4a88>

[32]:
```
##SHAP graph helps up understand feature importance of the CATmodel
interpret_model(CBR)
```

```
INFO:logs:Initializing interpret_model()
INFO:logs:interpret_model(estimator=<catboost.core.CatBoostClassifier object at
0x000001E6C7006148>, plot=summary, feature=None, observation=None)
INFO:logs:Checking exceptions
INFO:logs:Importing libraries
INFO:logs:plot type: summary
INFO:logs:model type detected: type 2
INFO:logs:Creating TreeExplainer
INFO:logs:Compiling shap values
```




```
INFO:logs:Visual Rendered Successfully
INFO:logs:interpret_model() succesfully
completed…
```

[119]:
```
#Load in test Data and preprocess for prediction accuracy

test = pd.read_csv('au_test.csv')
test.drop(columns=['education-num', 'fnlwgt'],inplace = True)
test.dropna(how='all',inplace=True)

pred = predict_model(CBR, data=test, verbose=True)
```

[119]: 0.871

[120]:
```
#The y_pred was in string format. y_pred is converted to string to produce␣
↪accuracy score
pred['class'].value_counts()
pred['Label'].value_counts()
```

```
y_test = pred['class'].to_numpy()
y_pred =  pred['Label'].to_numpy()
y_pred = [int(i) for i in y_pred]


round(accuracy_score(y_test,y_pred),3)
```

[120]: 0.871

[121]:
```
#Confusion Matrix
confusion_matrix(y_test,y_pred)
```

[121]:
```
array([[11684,   751],
       [ 1347,  2499]], dtype=int64)
```

[ ]:
```
#Save model as a pkl file for later use.
save_model(CBR,'CBR')
```

Overall, the PyCaret packages combined with some of the sklearn packages works very effectively. We were able to complete an end to end machine learning project in a fraction of the time and code. Pycaret also has packages to deploy model a production setting. For this project we didn't tune the hyperparemeters, but PyCaret has the ability to tune in 1 line of code.