American Income Classification Problem

the goal was to use the data taken of a sample of people living in an American city, and to predict if that person made more than 50,000 USD.

Part A)

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
import numpy as np
from sklearn.metrics import classification_report
from sklearn.ensemble import AdaBoostClassifier
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

#the below reads in the data, col_names is what I use to fix the column names (they had a *space* in th
all_data=pd.read_csv('individual_data_train.csv')
submission_data=pd.read_csv('individual_data_test.csv')
col_names=['Age','Workclass','Fnlwgt','Education','Education_Num',
           'Marital_Status','Occupation','Relationship','Race',
           'Sex','Capital_Gains','Capital_Loss','Hours_per_Week','Native_Country']


#from here I am splitting up the y value (Class), from the X values
all_y=all_data['Class']
x_setup=all_data.loc[:,all_data.columns !='Class']

#Column Names changed Here:
x_setup.columns=col_names
submission_data.columns=col_names
print(all_data.head())
```

```
##    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 Gains  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
```

the above is a quick preview of what the dataset looked like

the following were a few of the plots I looked at (using the seaborn library) to figure out if I needed to create any pseudo variables, ie indicator columns, or grouping certain numeric values together

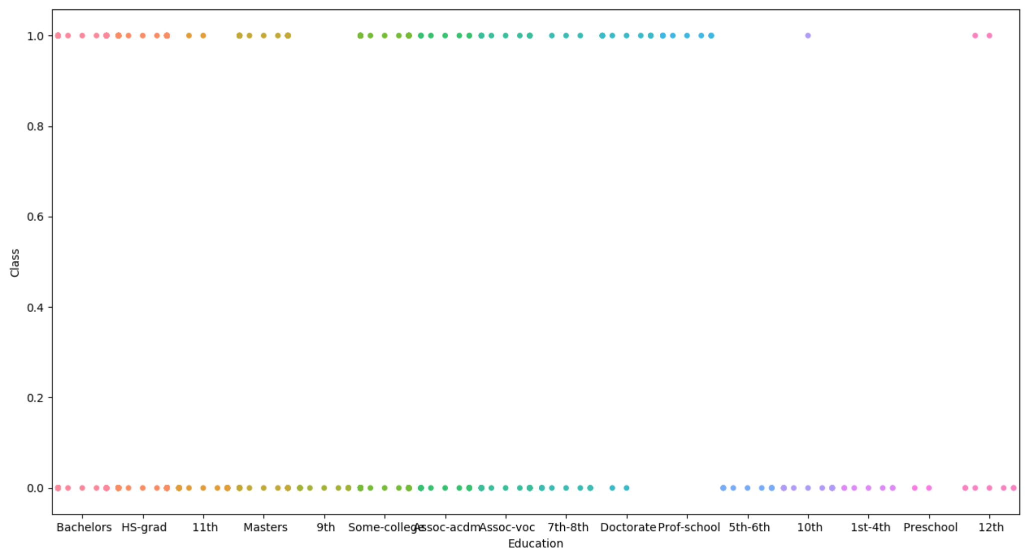

Figure 1:

from the charts above, we can see that it is possible to improve the accuracy by grouping certain numeric values together, ie instead of having the values of Age be: 40,41,42,43,44,45, can have a new column instead that says ‘Age between 40 and 45’

The following were the columns I created:

```python
#I was having difficulty getting the Natie country to be split up using pd.get_dummies, so I did the na
#manually, the ones I kept are the ones I would consider to actually have some influence on the class
x_setup['US_Indicator'] = (x_setup['Native_Country'] == ' United-States').astype(int)
submission_data['US_Indicator']=(submission_data['Native_Country'] == ' United-States').astype(int)
x_setup['Cuba_Indicator'] = (x_setup['Native_Country'] == ' Cuba').astype(int)
submission_data['Cuba_Indicator']=(submission_data['Native_Country'] == ' Cuba').astype(int)
x_setup['Jamaica_Indicator'] = (x_setup['Native_Country'] == ' Jamaica').astype(int)
submission_data['Jamaica_Indicator']=(submission_data['Native_Country'] == ' Jamaica').astype(int)
x_setup['India_Indicator'] = (x_setup['Native_Country'] == ' India').astype(int)
submission_data['India_Indicator']=(submission_data['Native_Country'] == ' India').astype(int)
x_setup['Mexico_Indicator'] = (x_setup['Native_Country'] == ' Mexico').astype(int)
submission_data['Mexico_Indicator']=(submission_data['Native_Country'] == ' Mexico').astype(int)
x_setup['South_Indicator'] = (x_setup['Native_Country'] == ' South').astype(int)
submission_data['South_Indicator']=(submission_data['Native_Country'] == ' South').astype(int)
x_setup['Puerto-Rico_Indicator'] = (x_setup['Native_Country'] == ' Puerto-Rico').astype(int)
submission_data['Puerto-Rico_Indicator']=(submission_data['Native_Country'] == ' Puerto-Rico').astype(in
x_setup['Honduras_Indicator'] = (x_setup['Native_Country'] == ' Honduras').astype(int)
submission_data['Honduras_Indicator']=(submission_data['Native_Country'] == ' Honduras').astype(int)
```

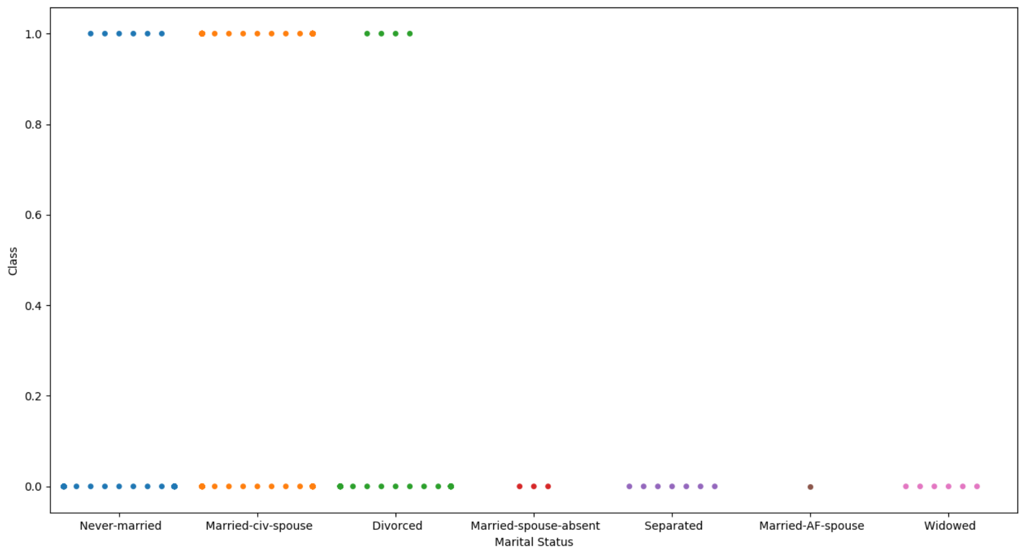

Figure 2:

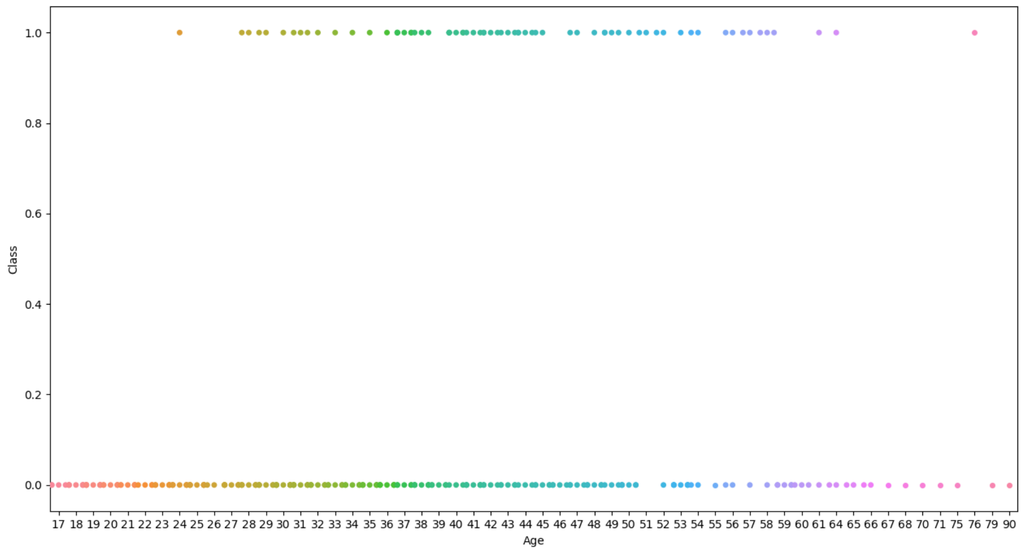

Figure 3:

```
x_setup['England_Indicator'] = (x_setup['Native_Country'] == ' England').astype(int)
submission_data['England_Indicator']=(submission_data['Native_Country'] == ' England').astype(int)
x_setup['Canada_Indicator'] = (x_setup['Native_Country'] == ' Canada').astype(int)
submission_data['Canada_Indicator']=(submission_data['Native_Country'] == ' Canada').astype(int)
x_setup['Germany_Indicator'] = (x_setup['Native_Country'] == ' Germany').astype(int)
submission_data['Germany_Indicator']=(submission_data['Native_Country'] == ' Germany').astype(int)
x_setup['Iran_Indicator'] = (x_setup['Native_Country'] == ' Iran').astype(int)
submission_data['Iran_Indicator']=(submission_data['Native_Country'] == ' Iran').astype(int)
x_setup['Philippines_Indicator'] = (x_setup['Native_Country'] == ' Philippines').astype(int)
submission_data['Philippines_Indicator']=(submission_data['Native_Country'] == ' Philippines').astype(i
x_setup['Italy_Indicator'] = (x_setup['Native_Country'] == ' Italy').astype(int)
submission_data['Italy_Indicator']=(submission_data['Native_Country'] == ' Italy').astype(int)
x_setup['Poland_Indicator'] = (x_setup['Native_Country'] == ' Poland').astype(int)
submission_data['Poland_Indicator']=(submission_data['Native_Country'] == ' Poland').astype(int)
x_setup['Columbia_Indicator'] = (x_setup['Native_Country'] == ' Columbia').astype(int)
submission_data['Columbia_Indicator']=(submission_data['Native_Country'] == ' Columbia').astype(int)
x_setup['Cambodia_Indicator'] = (x_setup['Native_Country'] == ' Cambodia').astype(int)
submission_data['Cambodia_Indicator']=(submission_data['Native_Country'] == ' Cambodia').astype(int)
x_setup['Thailand_Indicator'] = (x_setup['Native_Country'] == ' Thailand').astype(int)
submission_data['Thailand_Indicator']=(submission_data['Native_Country'] == ' Thailand').astype(int)
x_setup['Ecuador_Indicator'] = (x_setup['Native_Country'] == ' Ecuador').astype(int)
submission_data['Ecuador_Indicator']=(submission_data['Native_Country'] == ' Ecuador').astype(int)
x_setup['Laos_Indicator'] = (x_setup['Native_Country'] == ' Laos').astype(int)
submission_data['Laos_Indicator']=(submission_data['Native_Country'] == ' Laos').astype(int)
x_setup['Taiwan_Indicator'] = (x_setup['Native_Country'] == ' Taiwan').astype(int)
submission_data['Taiwan_Indicator']=(submission_data['Native_Country'] == ' Taiwan').astype(int)
x_setup['Haiti_Indicator'] = (x_setup['Native_Country'] == ' Haiti').astype(int)
submission_data['Haiti_Indicator']=(submission_data['Native_Country'] == ' Haiti').astype(int)
x_setup['Portugal_Indicator'] = (x_setup['Native_Country'] == ' Portugal').astype(int)
submission_data['Portugal_Indicator']=(submission_data['Native_Country'] == ' Portugal').astype(int)
x_setup['Dominican-Republic_Indicator'] = (x_setup['Native_Country'] == ' Dominican-Republic').astype(i
submission_data['Dominican-Republic_Indicator']=(submission_data['Native_Country'] == ' Dominican-Repub
x_setup['El-Salvador_Indicator'] = (x_setup['Native_Country'] == ' El-Salvador').astype(int)
submission_data['El-Salvador_Indicator']=(submission_data['Native_Country'] == ' El-Salvador').astype(i
x_setup['France_Indicator'] = (x_setup['Native_Country'] == ' France').astype(int)
submission_data['France_Indicator']=(submission_data['Native_Country'] == ' France').astype(int)
x_setup['Guatemala_Indicator'] = (x_setup['Native_Country'] == ' Guatemala').astype(int)
submission_data['Guatemala_Indicator']=(submission_data['Native_Country'] == ' Guatemala').astype(int)
x_setup['China_Indicator'] = (x_setup['Native_Country'] == ' China').astype(int)
submission_data['China_Indicator']=(submission_data['Native_Country'] == ' China').astype(int)
x_setup['Japan_Indicator'] = (x_setup['Native_Country'] == ' Japan').astype(int)
submission_data['Japan_Indicator']=(submission_data['Native_Country'] == ' Japan').astype(int)
x_setup['Yugoslavia_Indicator'] = (x_setup['Native_Country'] == ' Yugoslavia').astype(int)
submission_data['Yugoslavia_Indicator']=(submission_data['Native_Country'] == ' Yugoslavia').astype(int
x_setup['Peru_Indicator'] = (x_setup['Native_Country'] == ' Peru').astype(int)
submission_data['Peru_Indicator']=(submission_data['Native_Country'] == ' Peru').astype(int)
x_setup['Out_Indicator'] = (x_setup['Native_Country'] == ' Outlying-US(Guam-USVI-etc)').astype(int)
submission_data['Out_Indicator']=(submission_data['Native_Country'] == ' Outlying-US(Guam-USVI-etc)').a
x_setup['Scotland_Indicator'] = (x_setup['Native_Country'] == ' Scotland').astype(int)
submission_data['Scotland_Indicator']=(submission_data['Native_Country'] == ' Scotland').astype(int)
x_setup['Trinny_Indicator'] = (x_setup['Native_Country'] == ' Trinadad&Tobago').astype(int)
submission_data['Trinny_Indicator']=(submission_data['Native_Country'] == ' Trinadad&Tobago').astype(in
x_setup['Greece_Indicator'] = (x_setup['Native_Country'] == ' Greece').astype(int)
```

```python
submission_data['Greece_Indicator']=(submission_data['Native_Country'] == ' Greece').astype(int)
x_setup['Nicaragua_Indicator'] = (x_setup['Native_Country'] == ' Nicaragua').astype(int)
submission_data['Nicaragua_Indicator']=(submission_data['Native_Country'] == ' Nicaragua').astype(int)
x_setup['Vietnam_Indicator'] = (x_setup['Native_Country'] == ' Vietnam').astype(int)
submission_data['Vietnam_Indicator']=(submission_data['Native_Country'] == ' Vietnam').astype(int)
x_setup['Hong_Indicator'] = (x_setup['Native_Country'] == ' Hong').astype(int)
submission_data['Hong_Indicator']=(submission_data['Native_Country'] == ' Hong').astype(int)
x_setup['Ireland_Indicator'] = (x_setup['Native_Country'] == ' Ireland').astype(int)
submission_data['Ireland_Indicator']=(submission_data['Native_Country'] == ' Ireland').astype(int)
x_setup['Hungary_Indicator'] = (x_setup['Native_Country'] == ' Hungary').astype(int)
submission_data['Hungary_Indicator']=(submission_data['Native_Country'] == ' Hungary').astype(int)
x_setup['Holand_Indicator'] = (x_setup['Native_Country'] == ' Holand-Netherlands').astype(int)
submission_data['Holand_Indicator']=(submission_data['Native_Country'] == ' Holand-Netherlands').astype

#the functions below are what I used to create a new column
#that column have value 1 or 0, being an indicator variable
#or it could be the actual value from the all_data, or 0
#used to test the interaction
def age_uh (age_maker):
    if age_maker in range(20):
        return '0_to_20'
    elif age_maker in range (30):
        return '21 to 30'
    elif age_maker in range (40):
        return '31 to 40'
    elif age_maker in range (50):
        return '41 to 50'
    elif age_maker in range (60):
        return '51 to 60'
    elif age_maker in range (70):
        return '61 to 70'
    elif age_maker in range (80):
        return '70 to 80'
    else:
        return 'More_Than_80'

def cap_uh (cap_maker):
    if cap_maker in range(1000):
        return 'less_than_1k'
    elif cap_maker in range (2000):
        return '1k_to_2k'
    elif cap_maker in range (3000):
        return '2k_to_3k'
    elif cap_maker in range (4000):
        return '3k_to_4k'
    elif cap_maker in range (5000):
        return '4k_to_5k'
    elif cap_maker in range (6000):
        return '5k_to_6k'
    elif cap_maker in range (7000):
        return '6k_to_7k'
    elif cap_maker in range (8000):
        return '7k_to_8k'
    elif cap_maker in range (9000):
        return '8k_to_9k'
    elif cap_maker in range (10000):
        return '9k_to_10k'
    elif cap_maker in range (11000):
        return '10k_to_11k'
    elif cap_maker in range (12000):
        return '11k_to_12k'
    elif cap_maker in range (13000):
        return '12k_to_13k'
    elif cap_maker in range (14000):
        return '13k_to_14k'
    elif cap_maker in range (15000):
        return '14k_to_15k'
    elif cap_maker in range (16000):
        return '15k_to_16k'
    else:
        return 'More_Than_16k'

def hours_uh (hours_maker):
    if hours_maker in range(10):
        return '0_to_10'
    elif hours_maker in range (20):
        return '11 to 20'
    elif hours_maker in range (30):
        return '21 to 30'
    elif hours_maker in range (40):
        return '31 to 40'
    elif hours_maker in range (50):
        return '41 to 50'
    elif hours_maker in range (60):
        return '51 to 60'
    elif hours_maker in range (70):
        return '61 to 70'
    elif hours_maker in range (80):
        return '71 to 80'
    elif hours_maker in range (90):
        return '81 to 90'
    else:
        return 'More_Than_90'

#will have from 1 to 148
def fnl_uh (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return '0_to_10'
    elif round(fnl_maker/10000) in range (15):
        return '11 to 15'
    elif round(fnl_maker/10000) in range (20):
        return '16 to 20'
    elif round(fnl_maker/10000) in range (25):
        return '21 to 25'
    elif round(fnl_maker/10000) in range (30):
        return '26 to 30'
    elif round(fnl_maker/10000) in range (35):
        return '31 to 35'
    elif round(fnl_maker/10000) in range (40):
        return '36 to 40'
    elif round(fnl_maker/10000) in range (45):
        return '41 to 45'
    elif round(fnl_maker/10000) in range (50):
        return '46 to 50'
    elif round(fnl_maker/10000) in range (55):
        return '51 to 55'
    elif round(fnl_maker/10000) in range (60):
        return '56 to 60'
    elif round(fnl_maker/10000) in range (65):
        return '61 to 65'
    elif round(fnl_maker/10000) in range (70):
        return '66 to 70'
    elif round(fnl_maker/10000) in range (75):
        return '71 to 75'
    elif round(fnl_maker/10000) in range (80):
        return '76 to 80'
    elif round(fnl_maker/10000) in range (90):
        return '81 to 90'
    elif round(fnl_maker/10000) in range (100):
        return '91 to 100'
    elif round(fnl_maker/10000) in range (110):
        return '101 to 110'
    elif round(fnl_maker/10000) in range (120):
        return '111 to 120'
    elif round(fnl_maker/10000) in range (130):
        return '121 to 130'
    else:
        return 'More_Than_130'

##Functions to test interaction Here##
#split the hours, but this time make them numerical only (ie group all the 10s, 20s 30s, etc...
#then another column where you multiply with edu number
#Setting up Interaction columns for hours worked, ie group all the similar ones together numerically#
def hours_pre_interact_10 (hours_maker):
    if hours_maker in range(10):
        return hours_maker
    else:
        return 0
def hours_pre_interact_20 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return hours_maker
    else:
        return 0
def hours_pre_interact_30 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return 0
    elif hours_maker in range(30):
        return hours_maker
    else:
        return 0
def hours_pre_interact_40 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return 0
    elif hours_maker in range(30):
        return 0
    elif hours_maker in range(40):
        return hours_maker
    else:
        return 0
def hours_pre_interact_50 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return 0
    elif hours_maker in range(30):
        return 0
    elif hours_maker in range(40):
        return 0
    elif hours_maker in range(50):
        return hours_maker
    else:
        return 0
def hours_pre_interact_60 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return 0
    elif hours_maker in range(30):
        return 0
    elif hours_maker in range(40):
        return 0
    elif hours_maker in range(50):
        return 0
    elif hours_maker in range(60):
        return hours_maker
    else:
        return 0
def hours_pre_interact_70 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return 0
    elif hours_maker in range(30):
        return 0
    elif hours_maker in range(40):
        return 0
```

```python
    elif hours_maker in range(50):
        return 0
    elif hours_maker in range(60):
        return 0
    elif hours_maker in range(70):
        return hours_maker
    else:
        return 0
def hours_pre_interact_80 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return 0
    elif hours_maker in range(30):
        return 0
    elif hours_maker in range(40):
        return 0
    elif hours_maker in range(50):
        return 0
    elif hours_maker in range(60):
        return 0
    elif hours_maker in range(70):
        return 0
    elif hours_maker in range(80):
        return hours_maker
    else:
        return 0
def hours_pre_interact_90 (hours_maker):
    if hours_maker in range(10):
        return 0
    elif hours_maker in range(20):
        return 0
    elif hours_maker in range(30):
        return 0
    elif hours_maker in range(40):
        return 0
    elif hours_maker in range(50):
        return 0
    elif hours_maker in range(60):
        return 0
    elif hours_maker in range(70):
        return 0
    elif hours_maker in range(80):
        return 0
    elif hours_maker in range(90):
        return hours_maker
    else:
        return 0
def hours_pre_interact_100 (hours_maker):
    if hours_maker in range(90):
        return 0
    else:
        return hours_maker
```

```
#Setting up Interaction columns for capital gains, ie group all the similar ones together numerically#
def cap_pre_interact_1 (cap_maker):
    if cap_maker in range(1000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_2 (cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_3(cap_maker):
        if cap_maker in range(1000):
            return 0
        elif cap_maker in range(2000):
            return 0
        elif cap_maker in range(3000):
            return cap_maker
        else:
            return 0
def cap_pre_interact_4(cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return 0
    elif cap_maker in range(3000):
        return 0
    elif cap_maker in range(4000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_5(cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return 0
    elif cap_maker in range(3000):
        return 0
    elif cap_maker in range(4000):
        return 0
    elif cap_maker in range(5000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_6(cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return 0
    elif cap_maker in range(3000):
        return 0
    elif cap_maker in range(4000):
        return 0
    elif cap_maker in range(5000):
        return 0
    elif cap_maker in range(6000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_7(cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return 0
    elif cap_maker in range(3000):
        return 0
    elif cap_maker in range(4000):
        return 0
    elif cap_maker in range(5000):
        return 0
    elif cap_maker in range(6000):
        return 0
    elif cap_maker in range(7000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_8(cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return 0
    elif cap_maker in range(3000):
        return 0
    elif cap_maker in range(4000):
        return 0
    elif cap_maker in range(5000):
        return 0
    elif cap_maker in range(6000):
        return 0
    elif cap_maker in range(7000):
        return 0
    elif cap_maker in range(8000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_9(cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return 0
    elif cap_maker in range(3000):
        return 0
    elif cap_maker in range(4000):
        return 0
    elif cap_maker in range(5000):
        return 0
    elif cap_maker in range(6000):
        return 0
    elif cap_maker in range(7000):
        return 0
    elif cap_maker in range(8000):
        return 0
    elif cap_maker in range(9000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_10(cap_maker):
    if cap_maker in range(1000):
        return 0
    elif cap_maker in range(2000):
        return 0
    elif cap_maker in range(3000):
        return 0
    elif cap_maker in range(4000):
        return 0
    elif cap_maker in range(5000):
        return 0
    elif cap_maker in range(6000):
        return 0
    elif cap_maker in range(7000):
        return 0
    elif cap_maker in range(8000):
        return 0
    elif cap_maker in range(9000):
        return 0
    elif cap_maker in range(10000):
        return cap_maker
    else:
        return 0
def cap_pre_interact_11(cap_maker):
    if cap_maker in range(10000):
        return 0
    else:
        return cap_maker

#Setting up Interaction column for Age, ie group all the similar ones together numerically#
def age_pre_interact_2 (age_maker):
    if age_maker in range(20):
        return age_maker
    else:
        return 0
def age_pre_interact_3 (age_maker):
    if age_maker in range(20):
        return 0
    elif age_maker in range(30):
        return age_maker
    else:
        return 0
def age_pre_interact_4 (age_maker):
    if age_maker in range(20):
        return 0
    elif age_maker in range(30):
        return 0
    elif age_maker in range(40):
        return age_maker
    else:
        return 0
def age_pre_interact_5 (age_maker):
    if age_maker in range(20):
        return 0
    elif age_maker in range(30):
        return 0
    elif age_maker in range(40):
        return 0
    elif age_maker in range(50):
        return age_maker
    else:
        return 0
def age_pre_interact_6 (age_maker):
    if age_maker in range(20):
        return 0
    elif age_maker in range(30):
        return 0
    elif age_maker in range(40):
        return 0
    elif age_maker in range(50):
        return 0
    elif age_maker in range(60):
        return age_maker
    else:
        return 0
def age_pre_interact_7 (age_maker):
    if age_maker in range(20):
        return 0
    elif age_maker in range(30):
        return 0
    elif age_maker in range(40):
        return 0
    elif age_maker in range(50):
        return 0
    elif age_maker in range(60):
        return 0
    elif age_maker in range(70):
        return age_maker
    else:
        return 0
def age_pre_interact_8 (age_maker):
    if age_maker in range(20):
        return 0
    elif age_maker in range(30):
        return 0
    elif age_maker in range(40):
        return 0
    elif age_maker in range(50):
        return 0
    elif age_maker in range(60):
        return 0
    elif age_maker in range(70):
        return 0
    elif age_maker in range(80):
        return age_maker
    else:
        return 0
def age_pre_interact_9 (age_maker):
    if age_maker in range(90):
        return 0
    else:
        return age_maker

#Setting up Interaction column for Fnl, ie group all the similar ones together numerically#
def fnl_pre_interact_1 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_2 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_3 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_4 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_5 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_6 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_7 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_8 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_9 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_10 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return 0
    elif round(fnl_maker/10000) in range(55):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_11 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return 0
    elif round(fnl_maker/10000) in range(55):
        return 0
    elif round(fnl_maker/10000) in range(60):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_12 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return 0
    elif round(fnl_maker/10000) in range(55):
        return 0
    elif round(fnl_maker/10000) in range(60):
        return 0
    elif round(fnl_maker/10000) in range(65):
        return fnl_maker
    else:
        return 0
```

```
def fnl_pre_interact_13 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return 0
    elif round(fnl_maker/10000) in range(55):
        return 0
    elif round(fnl_maker/10000) in range(60):
        return 0
    elif round(fnl_maker/10000) in range(65):
        return 0
    elif round(fnl_maker/10000) in range(70):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_14 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return 0
    elif round(fnl_maker/10000) in range(55):
        return 0
    elif round(fnl_maker/10000) in range(60):
        return 0
    elif round(fnl_maker/10000) in range(65):
        return 0
    elif round(fnl_maker/10000) in range(70):
        return 0
    elif round(fnl_maker/10000) in range(75):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_15 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return 0
    elif round(fnl_maker/10000) in range(55):
        return 0
    elif round(fnl_maker/10000) in range(60):
        return 0
    elif round(fnl_maker/10000) in range(65):
        return 0
    elif round(fnl_maker/10000) in range(70):
        return 0
    elif round(fnl_maker/10000) in range(75):
        return 0
    elif round(fnl_maker/10000) in range(80):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_16 (fnl_maker):
    if round(fnl_maker/10000) in range(10):
        return 0
    elif round(fnl_maker/10000) in range(15):
        return 0
    elif round(fnl_maker/10000) in range(20):
        return 0
    elif round(fnl_maker/10000) in range(25):
        return 0
    elif round(fnl_maker/10000) in range(30):
        return 0
    elif round(fnl_maker/10000) in range(35):
        return 0
    elif round(fnl_maker/10000) in range(40):
        return 0
    elif round(fnl_maker/10000) in range(45):
        return 0
    elif round(fnl_maker/10000) in range(50):
        return 0
    elif round(fnl_maker/10000) in range(55):
        return 0
    elif round(fnl_maker/10000) in range(60):
        return 0
    elif round(fnl_maker/10000) in range(65):
        return 0
    elif round(fnl_maker/10000) in range(70):
        return 0
    elif round(fnl_maker/10000) in range(75):
        return 0
    elif round(fnl_maker/10000) in range(80):
        return 0
    elif round(fnl_maker/10000) in range(90):
        return fnl_maker
    else:
        return 0
def fnl_pre_interact_17 (fnl_maker):
    if round(fnl_maker/10000) in range(90):
        return 0
    else:
        return fnl_maker





####Below Inds groups all the similar ones together####
x_setup['Age_Indicator'] = x_setup['Age'].apply(age_uh)
submission_data['Age_Indicator']=submission_data['Age'].apply(age_uh)
x_setup['Cap_Indicator'] = x_setup['Capital_Gains'].apply(cap_uh)
submission_data['Cap_Indicator']=submission_data['Capital_Gains'].apply(cap_uh)
x_setup['Hours_Indicator'] = x_setup['Hours_per_Week'].apply(hours_uh)
submission_data['Hours_Indicator']=submission_data['Hours_per_Week'].apply(hours_uh)
x_setup['Fnl_Indicator'] = x_setup['Fnlwgt'].apply(fnl_uh)
submission_data['Fnl_Indicator']=submission_data['Fnlwgt'].apply(fnl_uh)
#Hours Intercation Here#
x_setup['Hrs_pre_interact_10']=x_setup['Hours_per_Week'].apply(hours_pre_interact_10)
submission_data['Hrs_pre_interact_10']=submission_data['Hours_per_Week'].apply(hours_pre_interact_10)
x_setup['Hrs_pre_interact_20']=x_setup['Hours_per_Week'].apply(hours_pre_interact_20)
submission_data['Hrs_pre_interact_20']=submission_data['Hours_per_Week'].apply(hours_pre_interact_20)
x_setup['Hrs_pre_interact_30']=x_setup['Hours_per_Week'].apply(hours_pre_interact_30)
submission_data['Hrs_pre_interact_30']=submission_data['Hours_per_Week'].apply(hours_pre_interact_30)
x_setup['Hrs_pre_interact_40']=x_setup['Hours_per_Week'].apply(hours_pre_interact_40)
submission_data['Hrs_pre_interact_40']=submission_data['Hours_per_Week'].apply(hours_pre_interact_40)
x_setup['Hrs_pre_interact_50']=x_setup['Hours_per_Week'].apply(hours_pre_interact_50)
submission_data['Hrs_pre_interact_50']=submission_data['Hours_per_Week'].apply(hours_pre_interact_50)
x_setup['Hrs_pre_interact_60']=x_setup['Hours_per_Week'].apply(hours_pre_interact_60)
```

```
submission_data['Hrs_pre_interact_60']=submission_data['Hours_per_Week'].apply(hours_pre_interact_60)
x_setup['Hrs_pre_interact_70']=x_setup['Hours_per_Week'].apply(hours_pre_interact_70)
submission_data['Hrs_pre_interact_70']=submission_data['Hours_per_Week'].apply(hours_pre_interact_70)
x_setup['Hrs_pre_interact_80']=x_setup['Hours_per_Week'].apply(hours_pre_interact_80)
submission_data['Hrs_pre_interact_80']=submission_data['Hours_per_Week'].apply(hours_pre_interact_80)
x_setup['Hrs_pre_interact_90']=x_setup['Hours_per_Week'].apply(hours_pre_interact_90)
submission_data['Hrs_pre_interact_90']=submission_data['Hours_per_Week'].apply(hours_pre_interact_90)
x_setup['Hrs_pre_interact_100']=x_setup['Hours_per_Week'].apply(hours_pre_interact_100)
submission_data['Hrs_pre_interact_100']=submission_data['Hours_per_Week'].apply(hours_pre_interact_100)
#Cap gains Interaction Here#
x_setup['Cap_pre_interact_1']=x_setup['Capital_Gains'].apply(cap_pre_interact_1)
submission_data['Cap_pre_interact_1']=submission_data['Capital_Gains'].apply(cap_pre_interact_1)
x_setup['Cap_pre_interact_2']=x_setup['Capital_Gains'].apply(cap_pre_interact_2)
submission_data['Cap_pre_interact_2']=submission_data['Capital_Gains'].apply(cap_pre_interact_2)
x_setup['Cap_pre_interact_3']=x_setup['Capital_Gains'].apply(cap_pre_interact_3)
submission_data['Cap_pre_interact_3']=submission_data['Capital_Gains'].apply(cap_pre_interact_3)
x_setup['Cap_pre_interact_4']=x_setup['Capital_Gains'].apply(cap_pre_interact_4)
submission_data['Cap_pre_interact_4']=submission_data['Capital_Gains'].apply(cap_pre_interact_4)
x_setup['Cap_pre_interact_5']=x_setup['Capital_Gains'].apply(cap_pre_interact_5)
submission_data['Cap_pre_interact_5']=submission_data['Capital_Gains'].apply(cap_pre_interact_5)
x_setup['Cap_pre_interact_6']=x_setup['Capital_Gains'].apply(cap_pre_interact_6)
submission_data['Cap_pre_interact_6']=submission_data['Capital_Gains'].apply(cap_pre_interact_6)
x_setup['Cap_pre_interact_7']=x_setup['Capital_Gains'].apply(cap_pre_interact_7)
submission_data['Cap_pre_interact_7']=submission_data['Capital_Gains'].apply(cap_pre_interact_7)
x_setup['Cap_pre_interact_8']=x_setup['Capital_Gains'].apply(cap_pre_interact_8)
submission_data['Cap_pre_interact_8']=submission_data['Capital_Gains'].apply(cap_pre_interact_8)
x_setup['Cap_pre_interact_9']=x_setup['Capital_Gains'].apply(cap_pre_interact_9)
submission_data['Cap_pre_interact_9']=submission_data['Capital_Gains'].apply(cap_pre_interact_9)
x_setup['Cap_pre_interact_10']=x_setup['Capital_Gains'].apply(cap_pre_interact_10)
submission_data['Cap_pre_interact_10']=submission_data['Capital_Gains'].apply(cap_pre_interact_10)
x_setup['Cap_pre_interact_11']=x_setup['Capital_Gains'].apply(cap_pre_interact_11)
submission_data['Cap_pre_interact_11']=submission_data['Capital_Gains'].apply(cap_pre_interact_11)
#Age Interaction Here#
x_setup['Age_pre_interact_2']=x_setup['Age'].apply(age_pre_interact_2)
submission_data['Age_pre_interact_2']=submission_data['Age'].apply(age_pre_interact_2)
x_setup['Age_pre_interact_3']=x_setup['Age'].apply(age_pre_interact_3)
submission_data['Age_pre_interact_3']=submission_data['Age'].apply(age_pre_interact_3)
x_setup['Age_pre_interact_4']=x_setup['Age'].apply(age_pre_interact_4)
submission_data['Age_pre_interact_4']=submission_data['Age'].apply(age_pre_interact_4)
x_setup['Age_pre_interact_5']=x_setup['Age'].apply(age_pre_interact_5)
submission_data['Age_pre_interact_5']=submission_data['Age'].apply(age_pre_interact_5)
x_setup['Age_pre_interact_6']=x_setup['Age'].apply(age_pre_interact_6)
submission_data['Age_pre_interact_6']=submission_data['Age'].apply(age_pre_interact_6)
x_setup['Age_pre_interact_7']=x_setup['Age'].apply(age_pre_interact_7)
submission_data['Age_pre_interact_7']=submission_data['Age'].apply(age_pre_interact_7)
x_setup['Age_pre_interact_8']=x_setup['Age'].apply(age_pre_interact_8)
submission_data['Age_pre_interact_8']=submission_data['Age'].apply(age_pre_interact_8)
x_setup['Age_pre_interact_9']=x_setup['Age'].apply(age_pre_interact_9)
submission_data['Age_pre_interact_9']=submission_data['Age'].apply(age_pre_interact_9)
#Fnlwgt Here#
#Fnlwgt was horrible, so I got rid of it

#Testing EduNum with hours worked Here:
```

```
#bad fit



#3rd run if somewhat decentish results above: split hrs and educataion up completely
#bad results


##With Indicator Variables as column names (get rid of native country, then remove the last few cols)
x_setup=x_setup.loc[:,x_setup.columns !='Native_Country']
x_setup2=x_setup.loc[:,x_setup.columns !='Fnlwgt']
all_x=pd.get_dummies(x_setup2)

submission_data=submission_data.loc[:,submission_data.columns !='Native_Country']
all_submit=submission_data.loc[:,submission_data.columns !='Fnlwgt']
submission_x=pd.get_dummies(all_submit)
#Clean the indicator Variables up
#1st run just keep generic Edu_nums
#x_setup['hrs_10_edu']=x_setup['Hrs_pre_interact_10']*x_setup['Education_Num']
#Bad

#2nd run if decentish results => have Edu as indicator, and have hrs as same as above, ie. instead of e
#edu1=hrs, edu2=hrs (grouped), etc.
#the above would only work if it's somewhat linear, #2nd run is meant to check if it's piecewise
#Have:Education_ 10th/11th/12th/1st-4th/5th-6th/7th-8th/9th/Assoc-acdm/Assoc-voc/Bachelors/Doctorate/Hs
#/Masters/Preschool/Prof-school/Some-college
#do Hrs* Education category to get new cols (stratified)
#2nd Run: Mediocre
#3rd Run: horrible
#for_loop_prep_edu=['Education_ 10th','Education_ 11th','Education_ 12th','Education_ 1st-4th','Educati
#                'Education_ 7th-8th','Education_ 9th','Education_ Assoc-acdm','Education_ Assoc-voc','E
#                'Education_ Doctorate','Education_ HS-grad','Education_ Masters','Education_ Preschool',
#for_loop_prep_hrs=['Hrs_pre_interact_10','Hrs_pre_interact_20','Hrs_pre_interact_30','Hrs_pre_interact
#                 'Hrs_pre_interact_60','Hrs_pre_interact_70','Hrs_pre_interact_80','Hrs_pre_interact_90

#for a in range(len(for_loop_prep)):
#    for b in range(len(for_loop_prep2)):
#        all_x[for_loop_prep[a]+for_loop_prep2[b]]=all_x[for_loop_prep[a]]*all_x[for_loop_prep2[b]]
#bad as well
##HOURS AND AGE: BAD= overfitting##

#Now testing occupation and hrs and gender#
#bad
#for_loop_prep_occ=['Occupation_ ?', 'Occupation_ Adm-clerical','Occupation_ Armed-Forces','Occupation_
#                'Occupation_ Craft-repair','Occupation_ Exec-managerial','Occupation_ Farming-fishing',
#                'Occupation_ Handlers-cleaners','Occupation_ Machine-op-inspct','Occupation_ Other-serv
#                'Occupation_ Priv-house-serv','Occupation_ Prof-specialty','Occupation_ Protective-serv
#                'Occupation_ Tech-support','Occupation_ Transport-moving']
#for a in range(len(for_loop_prep_occ)):
#        all_x[for_loop_prep_occ[a]+'Female']=all_x[for_loop_prep_occ[a]]*all_x['Sex_ Female']
#        submission_x[for_loop_prep_occ[a] + 'Female'] = submission_x[for_loop_prep_occ[a]] * submissio
```

Setting up the model and the learning set here:

```python
##Setting up Test and Training data here:
X_train, X_test, y_train, y_test = train_test_split(all_x, all_y)


#the chunk below is what I used to test different learning rates, best success with default
##The Actual Model##
#Experimenting with the Learning Rate Here:#
#mlp = MLPClassifier(hidden_layer_sizes=(30,25,20,15,10))
#the_model=mlp.fit(X_train,y_train)
#best result at learning rate around 1.1,1.2
#abc=[1.05,1.06,1.07,1.08,1.09,1.1,1.11,1.12,1.13,1.14,1.15,1.16,1.17,1.18,1.19,1.20]
#for i in range(len(abc)):
#    ada = AdaBoostClassifier(n_estimators=360, learning_rate=abc[i])
#    ##The Accuracy:
#    # predictions = mlp.predict(X_test)
#    ada_fit = ada.fit(X_train, y_train)
#    ada_predict = ada.predict(X_test)
#    # print(classification_report(y_test,predictions))
#    print('Currently doing i='+str(abc[i]))
#    print(classification_report(y_test, ada_predict, digits=4))

#Adaboost better than a simple MLP Neural Network

#learning rate = 1.06 or 1.2
#both worse than default
#n_estimators best with max number of columns

##########The chunk below can be un commented to actually run, takes a few hours to run fully.#########
#ada = AdaBoostClassifier(SVC(probability=True, kernel='linear',max_iter=50000), n_estimators=360)
#The Accuracy:
# predictions = mlp.predict(X_test)
#ada_fit = ada.fit(X_train, y_train)
#ada_predict = ada.predict(X_test)
# print(classification_report(y_test,predictions))
#print(classification_report(y_test, ada_predict, digits=4))
##The Accuracy:
#End Learning Rate Here:#
```