UCI Adult dataset

Machine Learning

Martin Saarman

Introduction

In this report I am gonna use the *Adult data set* from UCI Machine Learning Repository[1] to classify if a person earns above or under 50,000$ a year. I will use statistical methods to examine the data to see how we should process the data. After examination of the data I will move on to the preprocessing where I construct the data to fit machine learning algorithms for the best performance. To know which algorithm to use I will spot check several algorithms and choose the best performing. Finally I will tune the algorithm I chose and finalize the model. In this report "<=50k" equals "1" and >50k equals "2".

Data understanding

Looking at the first 20 rows of the dataset we can see a few things,
"education-num" is an encoded copy of the "education" feature, the greater the number is the higher the education is. The features "capital-gain" and "capital-loss" seems to contain mostly 0 values. We have a null value represented by "?" in the feature "native-country".

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	1
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	1
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	1
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	1
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	1
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	1
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	1
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	2
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	2
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	2
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	2
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	2
23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	1
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	1
40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	?	2
34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Amer-Indian-Eskimo	Male	0	0	45	Mexico	1
25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	White	Male	0	0	35	United-States	1
32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	White	Male	0	0	40	United-States	1
38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States	1
43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	White	Female	0	0	45	United-States	2

Looking at the "education" and "education-num" features in more detail below, we can clearly see that there is equal number of instances between the two feature values. Because a grade can be considered greater or smaller we will keep the "education-num" feature and drop the "education" feature.

[1] "Adult Data Set - UCI." 1 maj. 1996, https://archive.ics.uci.edu/ml/datasets/adult. Öppnades 21 nov.. 2018.

HS-grad	10501
Some-college	7291
Bachelors	5355
Masters	1723
Assoc-voc	1382
11th	1175
Assoc-acdm	1067
10th	933
7th-8th	646
Prof-school	576
9th	514
12th	433
Doctorate	413
5th-6th	333
1st-4th	168
Preschool	51

Name: education, dtype: int64

9	10501
10	7291
13	5355
14	1723
11	1382
7	1175
12	1067
6	933
4	646
15	576
5	514
8	433
16	413
3	333
2	168
1	51

Name: education-num, dtype: int64

Taking a closer look at the two features “capital-gain” and “capital-loss” below we can see that the mean is very small for both features considering the “max” value, we can also see that the IQR[2] is equal to 0.

	capital-gain	capital-loss
count	32561.000000	32561.000000
mean	1077.648844	87.303830
std	7385.292085	402.960219
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	99999.000000	4356.000000

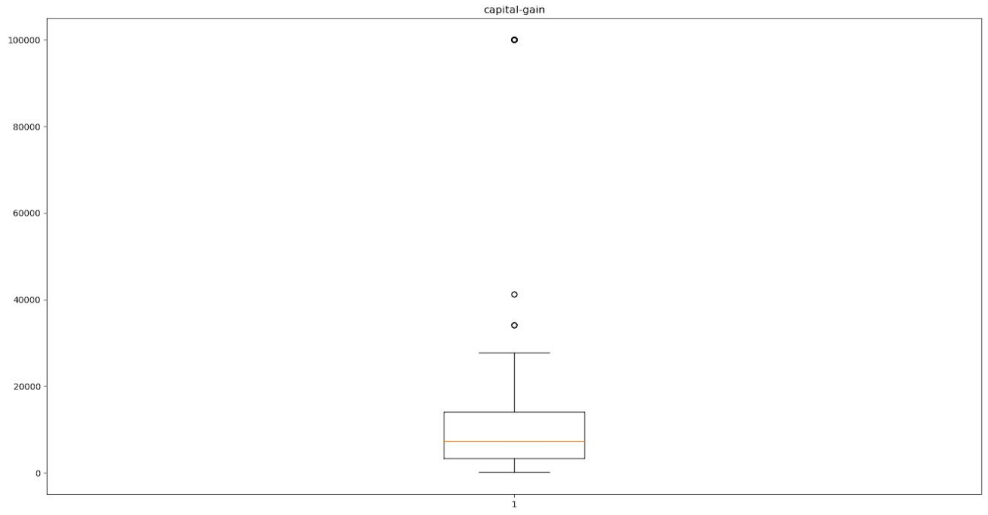
This is probably the majority of 0 values that is the cause for this, let’s take a look at this again after removing all 0 values.

After removing all 0 values it looks much better,

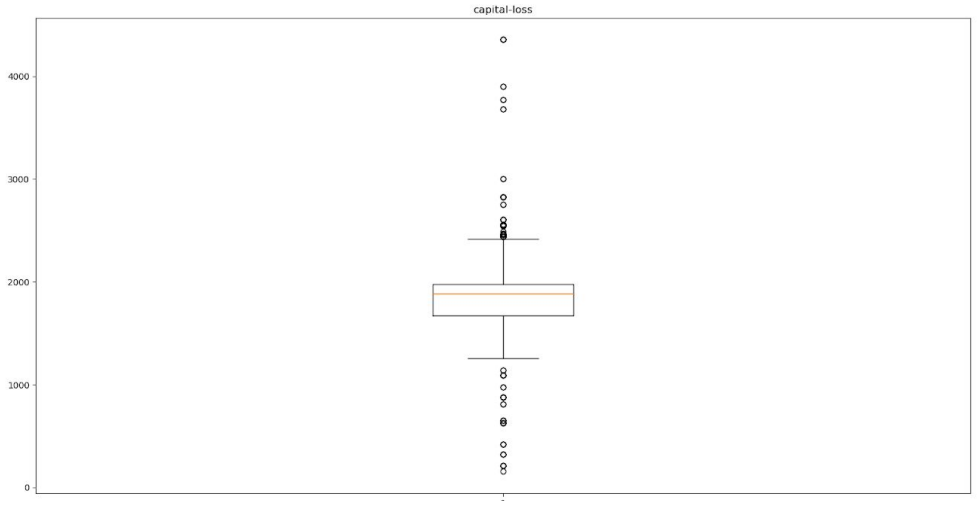
	capital-gain	capital-loss
count	2712.000000	1519.000000
mean	12938.541298	1871.428571
std	22395.413530	376.571535
min	114.000000	155.000000
25%	3411.000000	1672.000000
50%	7298.000000	1887.000000
75%	14084.000000	1977.000000
max	99999.000000	4356.000000

[2] "Interquartile range - Wikipedia." https://en.wikipedia.org/wiki/Interquartile_range. Öppnades 22 nov.. 2018.

Looking at the boxplot below we can see that the majority of values in “capital-gain” are approximately in between 3500 - 14000, with the mean at approximately 13000 and some outliers.



Looking at the boxplot below we can see that the majority of values in “capital-loss” are approximately in between 1500-2000 with the mean at approximately 1900 and a lot of outliers.



My plan is to make two new categorical features that will have the values “Low”, “Medium” and “High”, where “Low” represents < IQR, “Medium” represents IQR and “High” represents >IQR, I will then replace “capital-gain” and “capital-loss” with these new features.

```
 Private             22696
Self-emp-not-inc     2541
Local-gov            2093
?                    1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

```
 Prof-specialty       4140
Craft-repair          4099
Exec-managerial       4066
Adm-clerical          3770
Sales                 3650
Other-service         3295
Machine-op-inspct     2002
?                     1843
Transport-moving      1597
Handlers-cleaners     1370
Farming-fishing        994
Tech-support           928
Protective-serv        649
Priv-house-serv        149
Armed-Forces             9
Name: occupation, dtype: int64
```

```
 United-States          29170
Mexico                    643
?                         583
Philippines               198
Germany                   137
Canada                    121
Puerto-Rico               114
```

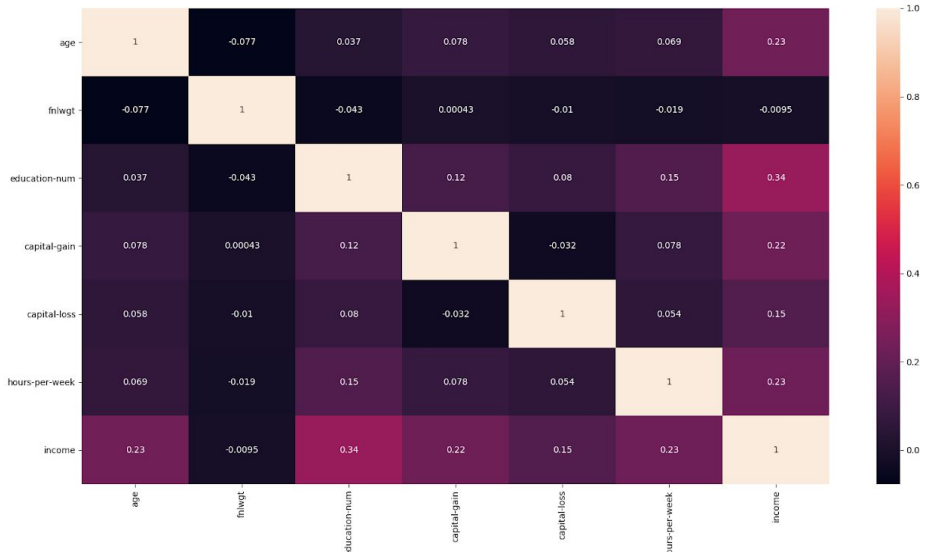
Looking closer on the categorical data I noticed that there are in total 3 features witt missing values represented by “?”.

I also noticed that the feature “native-country” has a lot more unique values then any other categorical feature.

```
Unique values in native-country 42
```

I have an idea to group countries by region to decrease dimensionality.

Moving on we can check the correlation between the numerical data. Looking at the graph below we can see that there are no strong correlation between the data features, strong correlation is generally not good and can be a sign of bad data.



I am gonna check if we have other null value in the dataset.

```
 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
```

Looking at the numbers above we can see that there are no null values, but we came across null values represented by “?” before, so let’s check that as well.

Looking at the numbers below it seems like we have already discovered all null values in the dataset.

```
There are 0 ? values in age
There are 1836 ? values in workclass
There are 0 ? values in fnlwgt
There are 0 ? values in education
There are 0 ? values in education-num
There are 0 ? values in marital-status
There are 1843 ? values in occupation
There are 0 ? values in relationship
There are 0 ? values in race
There are 0 ? values in sex
There are 0 ? values in capital-gain
There are 0 ? values in capital-loss
There are 0 ? values in hours-per-week
There are 583 ? values in native-country
There are 0 ? values in income
```

Last but not least I want to look at the class distribution, if we have one class that is over represented the model could learn to always predict one class.

```
 1    24720
2     7841
```

Looking at the numbers above we can see that class "1" is over represented, I will have to fix this in the preprocessing stage.

Preprocessing

I start of the preprocessing by dropping the "education" feature.

```
adult_data = adult_data.drop('education', axis=1)
```

We have to transform the the "capital-gain" and "capital-loss" features into categorical features

```
capital-gain capital-loss
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
      Medium          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
         Low          Low
```

As we can see above the values are now changed, where under IQR is “Low”, IQR is “Medium” and above IQR is “High”.

We have to remove the null values from “native-country”, “workclass” and “occupation”.

```
There are 0 ? values in workclass
There are 0 ? values in education
There are 0 ? values in marital-status
There are 0 ? values in occupation
There are 0 ? values in relationship
There are 0 ? values in race
There are 0 ? values in sex
There are 0 ? values in capital-gain
There are 0 ? values in capital-loss
There are 0 ? values in native-country
```

I did not want to remove necessary data so I replaced the null values with the most frequent value in each feature.

Moving on to the “native-country”, I start off by mapping all unique values to a region and then I use that mapping to create a new feature “region”.

native-country	region
United_States	Central_America
United_States	Central_America
United_States	Central_America
United_States	Central_America
Cuba	South_America
United_States	Central_America
Jamaica	South_America
United_States	Central_America
United_States	Central_America
United_States	Central_America
United_States	Central_America
India	South_Asia
United_States	Central_America
United_States	Central_America
United_States	Central_America
Mexico	South_America
United_States	Central_America
United_States	Central_America
United_States	Central_America
United_States	Central_America

Central_America	29888
South_America	1401
Europe	432
South_East_Asia	391
South_Asia	143
East_Asia	137
East_Europe	89
Africa	80

Name: region, dtype: int64

Above we can see “native-country” side-by-side with the new feature “region”, next I drop the “native-country” feature.

As mentioned in the *Data understanding* stage class “1” is over represented, to fix this I will create more data by shuffle the values in the features that belongs to class “2”.

1	24720
2	15682

Looking above we can see that it is much better distribution than before.

For the categorical data I will binarize all features except “capital-gain”, “capital-loss” and “education-num”, this is because the other features can not be sorted in a good way.

region_South_America	region_South_Asia
0	0
0	0
0	0
0	0
1	0
0	0
1	0
0	0
0	0
0	0
0	0
0	1

The picture above shows how the binarized data works, there will be a new feature for every value in the features. Above we have the two features “region_South_America” and “region_South_Asia”, when an observation is from South America the value will be 1 in that feature, while all other feature equals 0.

For the numerical data I will have 3 different datasets, one original, one standardized and one rescaled. I do this so I can test all datasets on different algorithms to see which data performs best.

I will wrap up the preprocessing with feature selection, I will use the Random Forest Classifier for feature selection.

```
0) capital-gain 0.16293973962127437
1) age  0.1605688140495507
2) hours-per-week       0.11781006771419036
3) capital-loss 0.08888491422090544
4) occupation_ Craft-repair     0.08444936241189843
5) region_East_Asia     0.05809509427232596
6) education-num        0.051092641638688284
7) occupation_ Farming-fishing  0.03212921502274337
8) race_ Black  0.015456636725475944
9) region_South_America 0.01336272674818517
10) region_South_East_Asia      0.013063342735857962
11) region_East_Europe  0.01240217484535466
12) fnlwgt      0.01205263329495357
```

Looking at the numbers above we can see that the feature “fnlwgt” is the lowest ranked non-binary feature. I will drop the “fnlwgt” to decrease dimensionality.

It seems like the features “relationship” and “marital-status” are fairly similar.

relationship	marital-status
Not-in-family	Never-married
Husband	Married-civ-spouse
Not-in-family	Divorced
Husband	Married-civ-spouse
Wife	Married-civ-spouse
Wife	Married-civ-spouse
Not-in-family	Married-spouse-absent
Husband	Married-civ-spouse
Not-in-family	Never-married
Husband	Married-civ-spouse
Husband	Married-civ-spouse
Husband	Married-civ-spouse
Own-child	Never-married
Not-in-family	Never-married

Looking at the data above we can see that there are similarities. I am gonna drop one of them because I want to reduce dimensionality to prevent overfitting.

```
martial-status score:  0.03674953243612828
relationship score:  0.030364633037603617
```

Taking the scores from the feature selection I will drop the relationship feature.

Spot checking

The algorithms I will use for spot checking is Logistic Regression, K-Nearest Neighbors, Decision Tree Classifier, Random Forest Classifier and AdaBoost Classifier.

Running my test harness on the different algorithms the Random Forest classifier outperformed every other algorithm except Decision Tree Classifier.

```
LogisticRegression with org_data.csv data accuracy: 0.7985481610521854
LogisticRegression with std_data.csv data accuracy: 0.7986718866764182
LogisticRegression with res_data.csv data accuracy: 0.7984738975011086


K-nearest neighbors with org_data.csv data accuracy: 0.7989202199729994
K-nearest neighbors with std_data.csv data accuracy: 0.8215923602382509
K-nearest neighbors with res_data.csv data accuracy: 0.8211963206342906


DecisionTreeClassifier with org_data.csv data accuracy: 0.8761699816975016
DecisionTreeClassifier with std_data.csv data accuracy: 0.8747591334857316
DecisionTreeClassifier with res_data.csv data accuracy: 0.874511590357254


RandomForestClassifier with org_data.csv data accuracy: 0.8853770204414652
RandomForestClassifier with std_data.csv data accuracy: 0.88535228634222
RandomForestClassifier with res_data.csv data accuracy: 0.8877287261020088


AdaBoostClassifier with org_data.csv data accuracy: 0.8043896104532502
AdaBoostClassifier with std_data.csv data accuracy: 0.8043896104532502
AdaBoostClassifier with res_data.csv data accuracy: 0.8043896104532502
```

I will run another test when I have tuned the algorithms to see if we get a better performing algorithm.

The best score I got from Logistic Regression by tuning was 80,2% accuracy.

```
LogisticRegression with std_data.csv data accuracy: 0.802186799414908
```

The best score I got from K-Nearest Neighbors by tuning was 82,2% accuracy.

```
K-nearest neighbors with std_data.csv data accuracy: 0.8215923602382509
```

The best score I got from Decision Tree Classifier by tuning was 88,1% accuracy.

```
DecisionTreeClassifier with org_data.csv data accuracy: 0.8812192845119702
```

The best score I got from Random Forest Classifier by tuning was 89,6% accuracy.

```
RandomForestClassifier with org_data.csv data accuracy: 0.8958224363639037
```

The best score I got from AdaBoost Classifier with Decision Tree Classifier as base estimator by tuning was 90,7% accuracy.

```
AdaBoostClassifier with org_data.csv data accuracy: 0.9069357158432994
```

Looking at the scores above we can see that Random Forest Classifier and AdaBoost Classifier was the best performing algorithms, because of the similar accuracy I will take both algorithms into model tuning.

Model tuning

Because of the big dataset I will only use the training data for tuning to speed up the process.

When I did the first model evaluation on Random Forest Classifier to set a threshold I noticed the model is overfitting.

```
Accuracy score (test data): 0.8503160827172399
Log loss (test data): 21.24999950155591
Accuracy score (train data): 0.9686811168258633
Log loss (train data): 21.11456813051257
```

Looking at the scores above we can see clearly that it performs much better on the training data than the unseen data.

After trying to fix the overfitting with hyperparameter tuning without success I decided to drop the Random Forest Classifier and continue on with a simpler algorithm.

After som experimentation with Logistic Regression and AdaBoost Classifier the best scores I could get was the following:

```
Accuracy score (test data): 0.8142076502732241
Log loss (test data): 0.3906177845909877
Accuracy score (train data): 0.8159441587068332
Log loss (train data): 0.38880027690091506
```

The overfitting seems to be gone and the Log Loss is lower, but I think we can do better so I will move back to *Data preprocessing* to see what we can do.

Data preprocessing (2)

Looking at the data again I see that age could be moved to categorical data and then be mapped.

```
[17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88
 90]
```

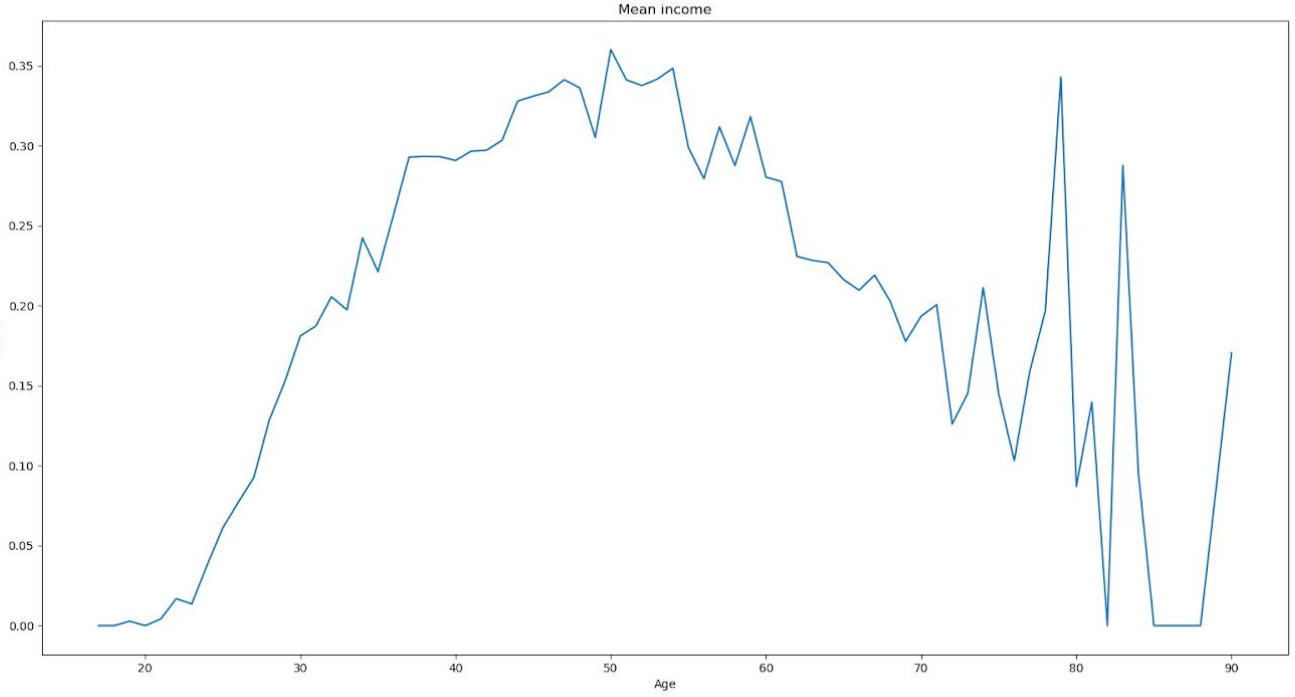
The numbers above is the unique values in the "age" feature, we can see that we have ages from 17-90.

The numbers below shows the mean age at each income class, looking at it I am gonna assume that at a certain age you earn more.

```
income
1       36.783738
2       44.249841
```

To get more insight in the "age" feature I am gonna plot the mean income to age.

Looking at the graph below we can see a pattern between income and age.

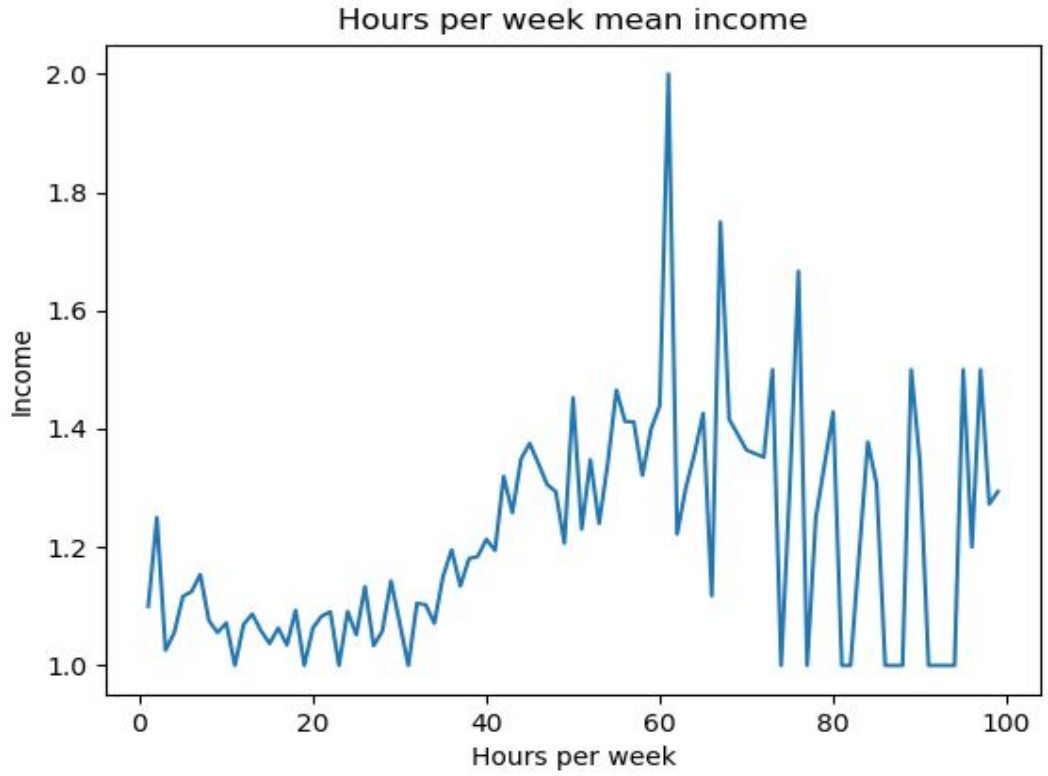


I will divide the “age” feature into 5 groups, 17-23(Start of career), 24-37(Career growth), 38-61(Stable career), 62-73(End of Career) and 74-90(Selling assets).

I also want to transform the “hours-per-week” feature into a categorical feature. There are values from 1-99 as we can see below, and I want to create subgroups with about 20 values in each, so that will be 5 groups.

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 70 72 73 74
 75 76 77 78 80 81 82 84 85 86 87 88 89 90 91 92 94 95 96 97 98 99]
```

Looking at the graph below we see that dividing the values over 5 groups (1-20, 21-40. 41-60, 61-80, 81-99) seems as a good approach.



Looking at the “race” feature below we can see that the three bottom values are uncommon.

```
White                 27816
Black                  3124
Asian-Pac-Islander     1039
Amer-Indian-Eskimo      311
Other                   271
Name: race, dtype: int64
```

I am gonna generalize these features to reduce dimensionality and reduce overfitting.

I want to reduce dimensionality even more to prevent overfitting, the idea is to do feature selection again and remove all features that has a score under 0,005.

```
30) workclass_ Self-emp-not-inc 0.005776260331652323
31) occupation_ Machine-op-inspct       0.004386495061406734
32) race_other  0.004211231401293003
33) region_East_Europe  0.003957434503140498
34) occupation_ Other-service   0.0038224427495541016
35) workclass_ Self-emp-inc     0.002955265651795349
36) region_South_East_Asia      0.00249467727253762
37) occupation_ Farming-fishing 0.0021393847495056457
38) workclass_ Without-pay      0.0017715196001833646
39) workclass_ State-gov        0.0014911359623933375
40) region_South_America        0.0014699931082000216
41) workclass_ Never-worked     0.0012929149349077945
42) occupation_ Craft-repair    0.001121602321881756
43) workclass_ Private  0.0009461767612801753
44) workclass_ Federal-gov      0.0007847155810000534
45) race_ Other 0.0003593818616494174
46) occupation_ Adm-clerical    0.00015345217144254223
47) occupation_ Prof-specialty  5.5147337539306685e-05
48) marital-status_ Married-civ-spouse  3.315931841841094e-05
```

Looking at the numbers above we can see that I will remove 31-48 leaving me with 30 features.

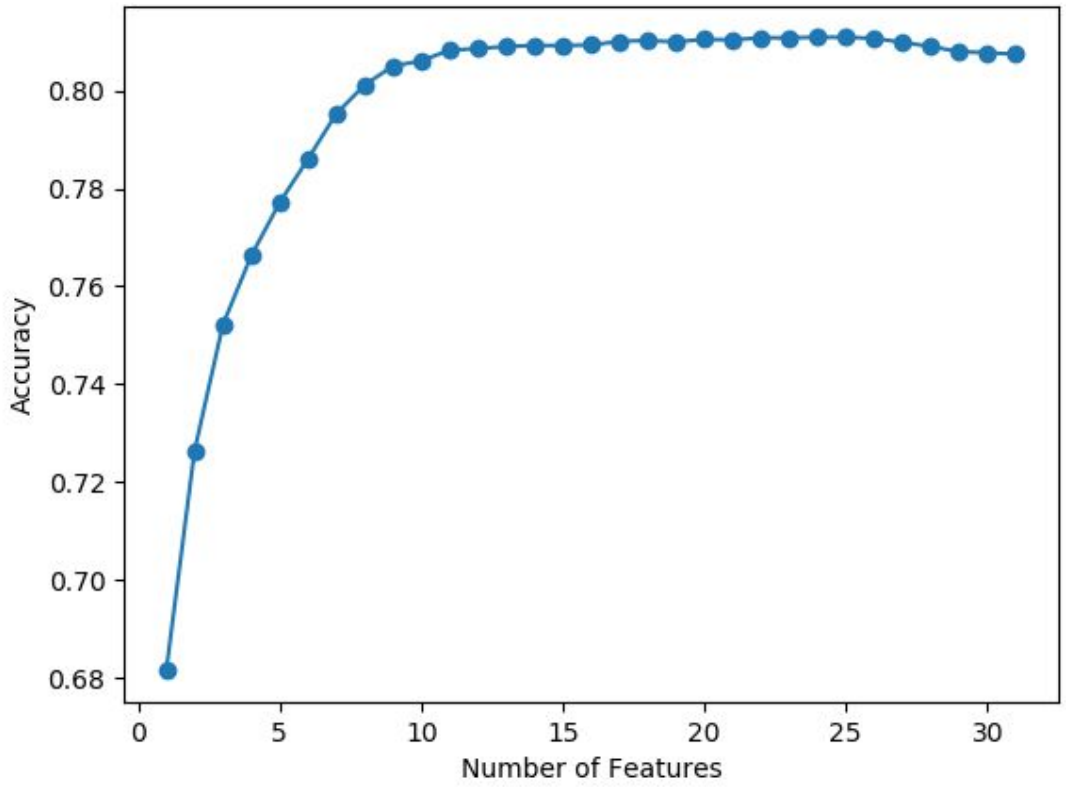
The last thing I am gonna do is to get more samples with the value “2” in the “income” feature, this is to get value “2” and “1” on the same level.

```
1    24720
2    15682
Name: income, dtype: int64
```

Looking at the numbers above we can see that I need about 9000 new samples, this time I will randomly pick already existing samples and put them into the dataset again.

```
2    24720
1    24720
Name: income, dtype: int64
```

Before moving on I am gonna do *sbs* feature selection to see if I can remove some more features.



Looking at the graph above we can see that the performance is highest with 25 features, let's look at those features.

```
Index(['age', 'capital-gain', 'capital-loss', 'hours-per-week',
       'education-num', 'workclass_ Local-gov', 'workclass_ Self-emp-not-inc',
       'marital-status_ Divorced', 'marital-status_ Married-spouse-absent',
       'marital-status_ Never-married', 'marital-status_ Separated',
       'marital-status_ Widowed', 'occupation_ Armed-Forces',
       'occupation_ Exec-managerial', 'occupation_ Handlers-cleaners',
       'occupation_ Priv-house-serv', 'occupation_ Protective-serv',
       'occupation_ Transport-moving', 'race_ Asian-Pac-Islander',
       'race_ Black', 'race_ White', 'region_Africa', 'region_Central_America',
       'region_Europe', 'region_South_Asia'],
      dtype='object')
```

I will drop all features that are not in the picture above, which leave us with 25 features.

Model tuning

After some more time with tuning I can not seem to improve accuracy as good as I want to. My plan B is to find some of the best performing algorithms that does not overfit and use ensemble tuning to improve performance.

Score for SGD:

```
Accuracy score (test data): 0.8103503696560591
Log loss (test data): 0.39630391972990064
Accuracy score (train data): 0.8077240999265246
Log loss (train data): 0.3978224370016883
```

Score for LR:

```
Accuracy score (test data): 0.8101360762884389
Log loss (test data): 0.3961970750256893
Accuracy score (train data): 0.8076322556943424
Log loss (train data): 0.39776407209941517
```

Score for DTC:

```
Accuracy score (test data): 0.7604200150005357
Log loss (test data): 0.4732238428046517
Accuracy score (train data): 0.7551432770022043
Log loss (train data): 0.470594054232587
```

Stochastic Gradient Descent, Logistic Regression and Decision Tree Classifier are the algorithms I am gonna use for ensembles tuning. Even though the Decision Tree does not get as good ratings it is the only non-linear model that does not overfit, and to get the most out of ensemble tuning we want to vary our algorithms as much as possible.

For ensembles I used the Voting Classifier from Scikit library, which uses majority vote to make predictions. Using SGD, LR and DTC as base estimators gets us to a accuracy off 81,2% with no overfitting.

```
Accuracy score [test]: 0.811792071197411
Accuracy score [train]: 0.8060022249190939
```

Summary

We started of examine the data and quickly found features that had to be processed before a machine learning algorithm would understand the pattern in the data. In the preprocessing stage I did a lot of changes and did a lot of experimentation, and I think some made the data better while other methods made it worse.

I spot checked a few algorithms, and found a few that had high variance (overfitting), I tried to fix the high variance in model tuning but it never got good enough, which made me continue on with simpler algorithms.

After model tuning each algorithm lacked of performance, which made me use ensembles tuning to get a stronger model with better performance. When finalizing the model the accuracy was just over 81% which is not as good as I wanted.

I chose this dataset to challenge myself and to see where I lacked in performance. After working with this machine learning problem I understand that I have to get a better understanding of the underlying details for the process in a machine learning problem. I will continue my pursuit for machine learning mastery with linear algebra and statistical methods while I work on smaller machine learning problems to apply and test what I learn.