**Capstone Project**
Machine Learning Engineer Nanodegree
Mohammed Saleh
MohammedSaleh@ieee.org

# Finding Donors for *CharityML*

**27th April 2020**

## DEFINITION

### OVERVIEW

For some of CharityML it's important to understand the potential donors. Our goal with this implementation is to predict whether an individual makes more than $50,000. This sort of task can arise in a non-profit setting, where organizations survive on donations. Understanding an individual's income can help a non-profit better understand how large of a donation to request, or whether or not they should reach out to begin with.

### PROBLEM STATEMENT

While it can be difficult to determine an individual's general income bracket directly from public sources, we aim that we can infer this value from other publically available features.

Our problem is a binary classification problem and we will use Supervised learning models regarding solving it.

Also, what are the most effective features on 'income' that need our attention?

We aim to achieve more than 80% accuracy of predicting people' total income to help the CharityML targeting the right people.

### DATASETS AND INPUT

The dataset for this project originates from the UCI Machine Learning Repository. The dataset was donated by Ron Kohavi and Barry Becker, after being published in the article "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid". You can find the article by Ron Kohavi online. The data we investigate here consists of small changes to the original dataset, such as removing the 'fnlwgt' feature and records with missing or ill-formatted entries.

## Attribute Information:

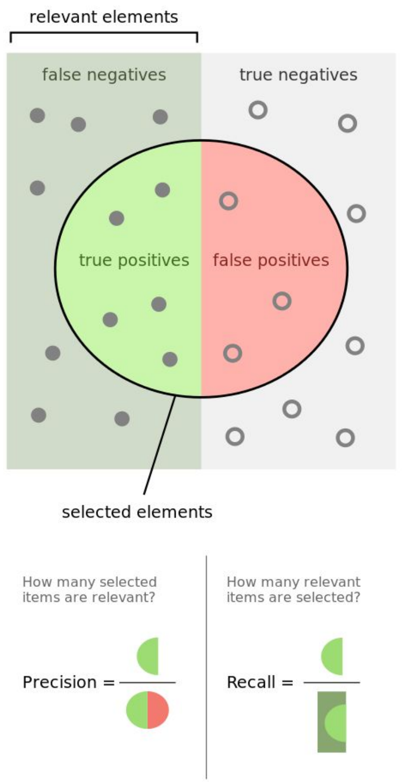
| Feature | type | Feature | type |
|---|---|---|---|
| Age | Integer | Relationship | Categorical |
| Workclass | Categorical | Sex | Categorical |
| Education | Categorical | Race | Categorical |
| Education-num | Float | Native-country | Categorical |
| Marital-status | Categorical | Capital-gain | Integer |
| occupation | Categorical | Capital-loss | Integer |
| Hours-per-week | Integer | | |

## EVALUATION METRICS

We will use the **Accuracy score** and **F-score** metrics. Since we don’t have a critical case for the False Negative or False Positive.

**Accuracy score** is the ratio of the correctly classified labels to the total number of tested data.

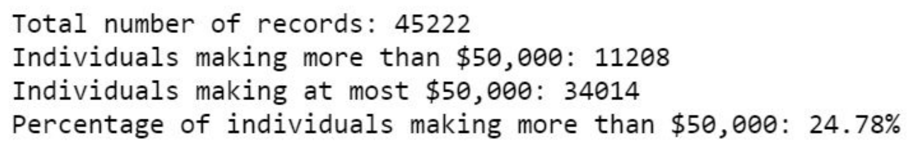
**F-Score** considers both precision and recall. It is the harmonic mean(average) of the precision and recall.



# ANALYSIS

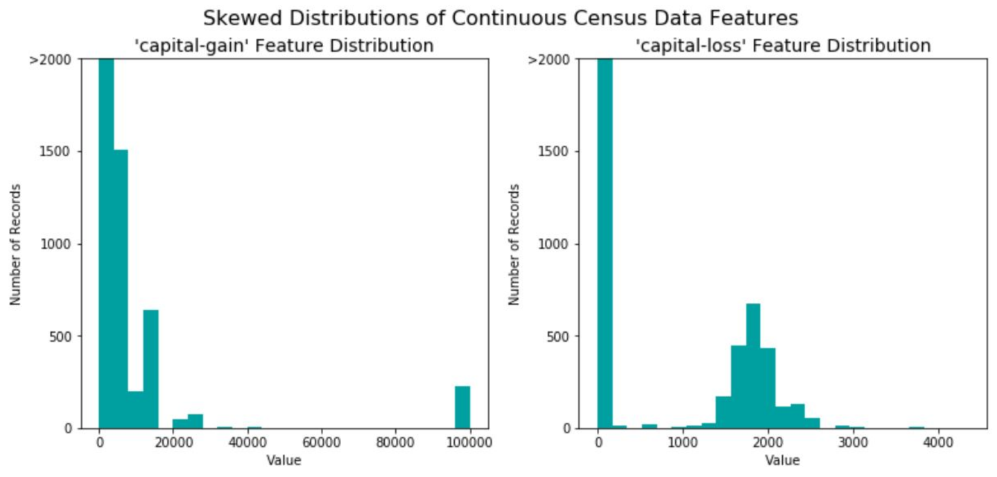
## DATA QUALITY AND EXPLORATION

1. The data are cleared from any missing values.
2. The total rows of data are 45222 with 11208 individuals whose income is greater than 50k$. So, our data is imbalanced.



```
Total number of records: 45222
Individuals making more than $50,000: 11208
Individuals making at most $50,000: 34014
Percentage of individuals making more than $50,000: 24.78%
```

3. There are skewed distributed features which are **Capital-gain** and **Capital-loss**.



4. The data values vary in different distributions.
5. As we saw above, there are many categorical features.

## ALGORITHMS AND TECHNIQUES

For this problem and after comparing between 3 models we will use the **Ensemble Method (AdaBoost)** as our main classifier:

**AdaBoost** can be used to solve a variety of real-world problems, such as predicting customer churn and classifying the types of topics customers are talking/calling about. Which is very close to our target here to detect the income class for users.

- **AdaBoost** algorithm advantages are:

    - It enhances the accuracy of weak learners.
    - It uses different classification methods.
    - The overfitting becomes harder to accord.
- **AdaBoost** algorithm disadvantages are:
    - Data imbalance leads to a decrease in classification accuracy.
    - Training is time consuming, and it is best to cut the point at each reselection of the current classifier.

Debinding on the variation of data types and the number of features and data rows it will be better to use an ensemble method like this.

## AdaBoost in Layman's Terms

We have many classification models that have many advantages and disadvantages, and the most expected disadvantage is overfitting (When the classifier memorizes the data and can't be able to generalize on the unseen data). So, what about limiting the processes of this algorithm to avoid Overfitting but use many of this model and combine these models into one model which will be hopefully, more accurate and more generalized. AdaBoost algorithm is here to rescue…

We will feed many of our models (Which we call them: "Weak learners") with data and with every iteration, we ask the model to take more attention to those misclassifications which the previous model did.

The weak learner is any machine learning algorithm that gives better accuracy than simply guessing depending on studying the data.

The default weak learner here in AdaBoost is Decision Trees which works by dividing the problem into simpler problems depending on the features and its relation to the output, for example, if we have a direct relationship between the capital gain and the total income we will give big attention to capital gain if it has extreme value. otherwise, we will divide the other features and so on.

After finishing our weak learners' training and when we get new unseen data we let them vote. And take not only the most chosen prediction, but we also give attention to the experience of the weak learner.

It looks like that you are hiring many employees "Weak learners" for a huge job and you divided the job into smaller problems and gave every problem to the specialist employee.

Now when we get a new unseen record, we ask the weak learners, what do you think the total income for this person is? and let them vote and we will take care more with the answer of this weak learner which has trained on data close to this new record that we have here.

We also will need to use somo of techniques during the pre-processing as **logarithmic transformation** for skewed data, **MinMaxScaler** for numerical data and **One-hot-encoding** for categorical data.

Finally, we used **GridSearchCV** for optimizing and model tuning.

## BENCHMARK MODEL

We didn't have any previous applied models. So, we will use an initial benchmark which predicts all of the data as 1 ("More than 50K$").

- Accuracy score: 24.78%
- F-score: 29.17%

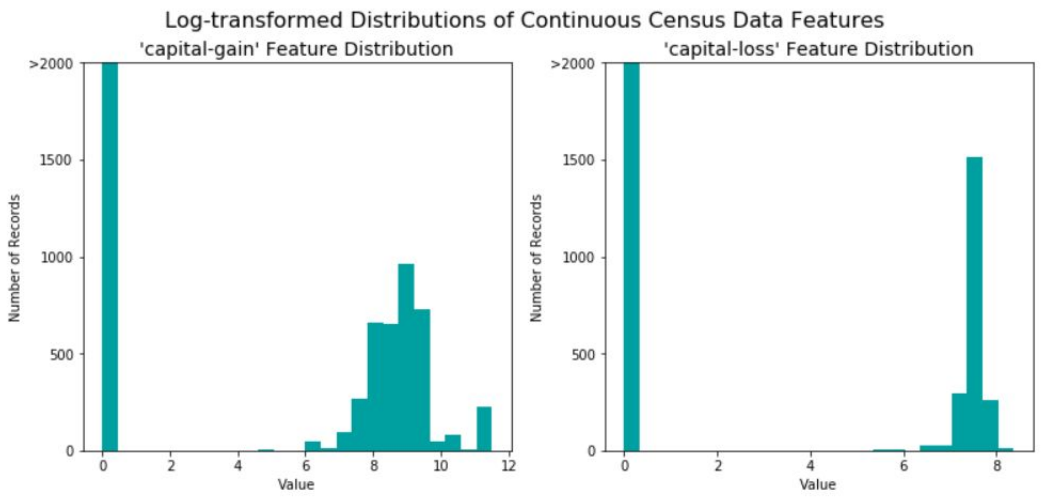
Then, we will use another benchmark which is our model scores on the test data set before optimization.

- Accuracy score: 85.76%
- F-score: 72.46%

# METHODOLOGY

## DATA PRE-PROCESSING

1. We applied a logarithmic transformation on the skewed features.

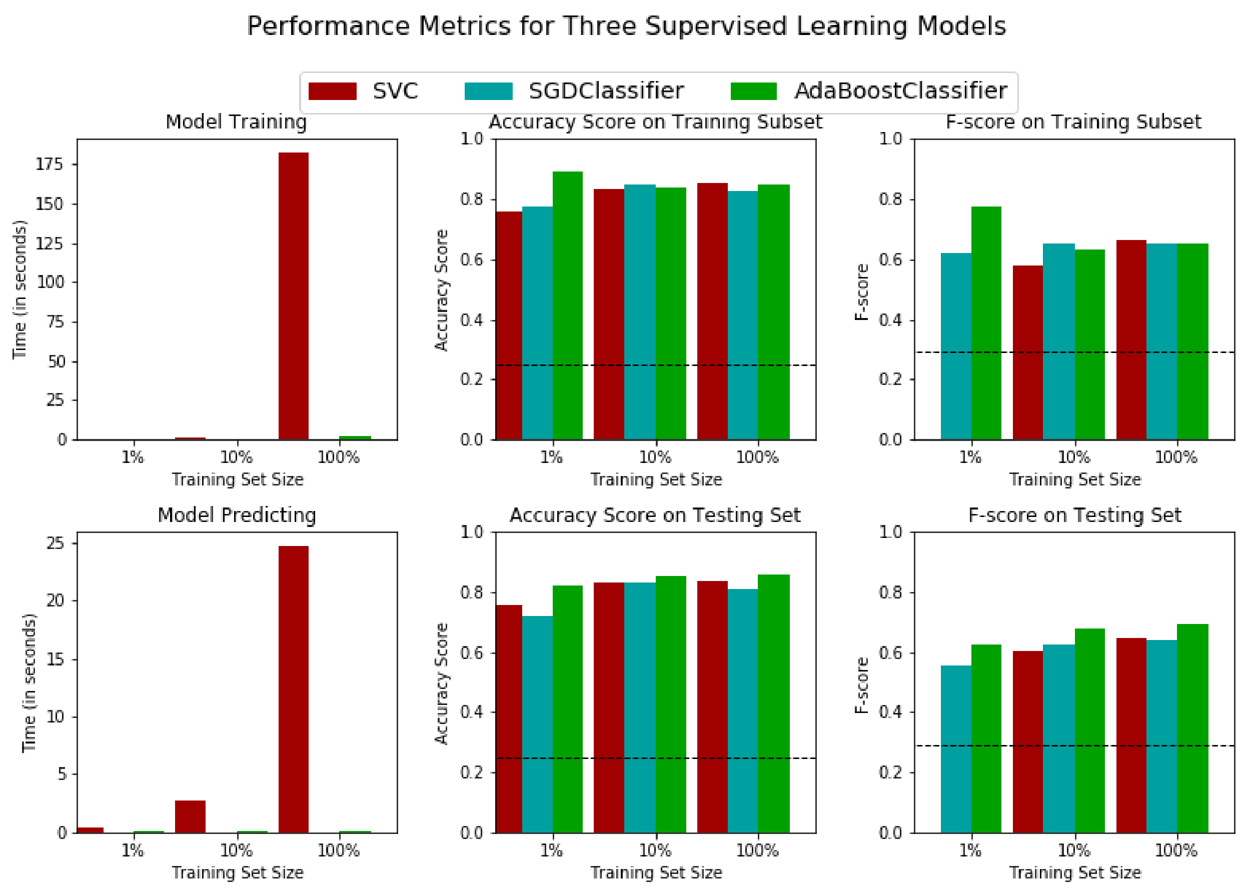


2. For numerical features we used MinMaxScaler for normalization. And now all of the numerical features are between 0 and 1.
3. We applied One-hot encoding for categorical features using DummyVariables. And we have now **103** total features after one-hot encoding.
4. Randomly, the data has been split into train and test datasets, using 20% of total data as a test set.
    a. Training set had 36177 samples.
    b. Testing set had 9045 samples.

## IMPLEMENTATION

### Initial Model Evaluation

Before choosing the **AdaBoost** algorithm we did an initial compiration between it and another two models **Stochastic Gradient Descent Classifier (SGDC)** and **Support Vector Machines (SVM)** on **1%**, **10%** and **100%** of the training dataset.



Finally, we tried features reduction and compared the results...

- Final Model trained on full data
  - Accuracy on testing data: 86.64%
  - F-score on testing data: 74.32%
- Final Model trained on reduced data
  - Accuracy on testing data: 84.26%
  - F-score on testing data: 70.44%

## REFINEMENT

Finally, we used the GridSearch algorithm to optimize the output and for the hyperparameter tuning.

The parameters we worked on are:

1. N_estimators, with values {50, 100 and 500}.
2. Learning_rate, with values {0.1, 0.5 and 1.0}.

# RESULTS

## MODEL EVALUATION AND VALIDATION

After the hyperparameters have been tuned the scores have been increased:

- Unoptimized model
  - Accuracy score on testing data: 85.76%
  - F-score on testing data: 72.46%
- Optimized Model
  - Final accuracy score on the testing data: 86.64%
  - Final F-score on the testing data: 74.32%

The optimized model has better accuracy and F-score than the unoptimized one with about 1% for accuracy and 2% for F-score.

## JUSTIFICATION

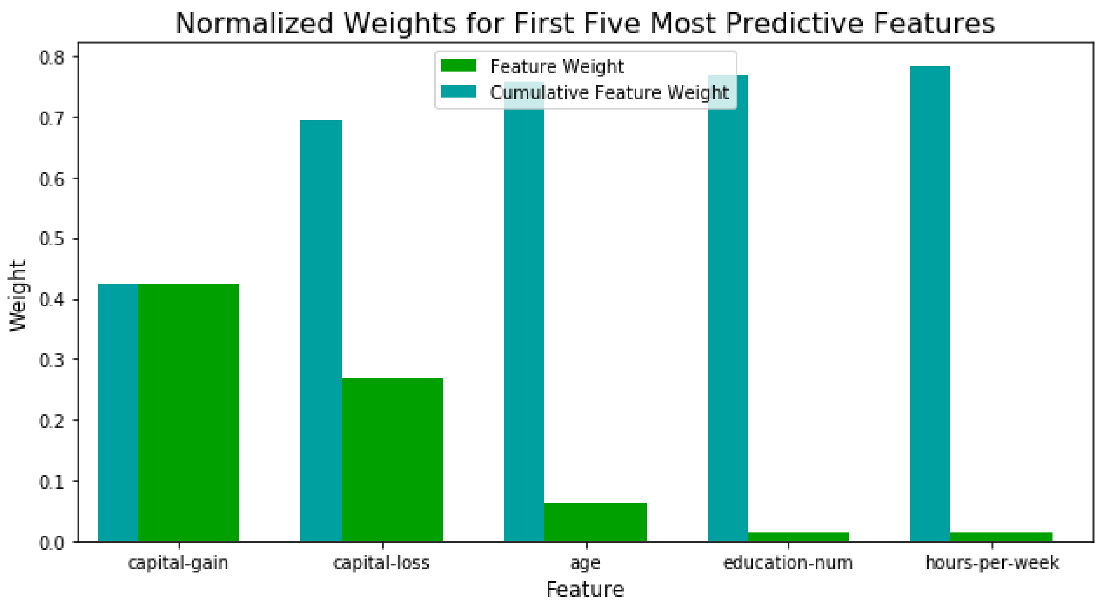
### Results Comparison

| Metric | naive predictor | Unoptimized Model | Optimized Model |
|---|---|---|---|
| Accuracy Score | 24.78% | 85.76% | 86.64% |
| F-score | 29.17% | 72.46% | 74.32% |

### Most effective features

After our investigation we noticed that the most important features which affect the total income are:

- Capital-gain
- Capital-loss
- Age
- Education level
- Hours of working per week



We ignored the features reduction because we had a significant effect on the scores without any advantage science our data isn’t large enough and doesn’t take much time.