# CSC373 Final DMP Cindy Chen

[**CSC373 Final DMP Cindy Chen**](#_bl3a98d0mk0d) **1**

[**Data Acquisition**](#_3v1zsf4nbv0x) **1**

[**Data Cleaning**](#_5e12foektw4t) **3**

[**Code Development**](#_3xz8onqyjap1) **4**

[Bagging](#_vxjt9b6gc1go) 4

[**Package Used**](#_71i5696fzg2y) **5**

[Goal 1: Association: Which features help predict income the most?](#_6w65hmvslvf0) 6

[Decision Tree](#_bkx9og41kxzo) 7

[Goal 2: Prediction: The highest prediction accuracy by training different classifier](#_jbodc2ir4p3) 10

[Random Forest](#_bt2zdtp8rrtz) 11

[XGBoost](#_oquujasbmyv2) 12

[Goal 3: Anomaly Detection](#_1v6brrh4i4rx) 14

[EDA](#_dn01zqxh2kj7) 14

[Isolation Forest](#_f7hznkuhqhuq) 16

[**Exercise Theory**](#_dh6ox6d4vxtk) **18**

[**Self-assessment**](#_5al2xquzrjtt) **18**

# 

# Data Acquisition

The dataset is obtained from the 1994 United States Census database and the initial goal is to build a model that predicts annual income based on demographic information collected from a census. This kind of predictive model is useful in various applications including credit card or mortgage approvals. The dataset is downloaded from the University of California Irvine Machine Learning Repository and is commonly titled the “Adult dataset.”

The response variable for this particular study is the binary variable “income” (divided into >50k and <=50k). We have 15 other features representing demographic information to be used for prediction. This includes information about the age of the individual, the number of years they attended school, their capital gain and loss for the year (not reflected in income), and the hours they work per week. We also have categorical features including sex, race, marital status, and native country. Here is a table describing each feature and its datatype. From my understanding, each row is not represented as one individual; instead, each row represents one type of group of people who have the same demographic features.

|  | Description | Datatype |
| --- | --- | --- |
| age |  | integer |
| workclass | Eg. Private, self-employed, state\_gov | String |
| fnlwgt | represents a weight based on the number of people represented by that data | Integer |
| education | highest education completed  Eg. High school, bachelors | String |
| education.num | Year of education | Integer |
| marital.status | Eg. Married, divorced | String |
| occupation | Eg. Tech support, caft repair | String |
| relationship | Eg. Wife, child | String |
| race | Eg. White, asian | String |
| sex | Eg. Male, female | String |
| capital.gain |  | integer |
| capital.loss |  | Integer |
| hours.per.week | Working hours | Integer |
| native.country |  | String |
| income | >50K or <=50K | String |

The dimension of the dataset is (32561, 15)

# Data Cleaning

We first check missing data. Missing data encoded as “?” in this dataset so we first convert all the strings with “?” to nan. We are missing data from only 3 features: ‘native country,’ ‘occupation,’ and ‘workclass.’ We have 3620 rows with missing data, accounting for approximately 7% of the dataset, which is only a small part of our dataset. We then remove all the missing data.

Our next step is to remove some repeated or unnecessary columns. The features ‘education level’ (categorical variable representing the highest education completed) and ‘education num’ (numerical variable representing the number of years in school) contain the same information. We don’t need to include both in the model, so we keep just the numerical version, since this preserves ordering. We also delete the column ‘relationship’ since it contains the same information as ‘marital status’.

Our response variable for this dataset is income but it is a string variable shown as either ‘>50k’ or ‘<=50k’. I decided to convert it to a binary variable represented as 0 or 1 for better modeling in the future stage.

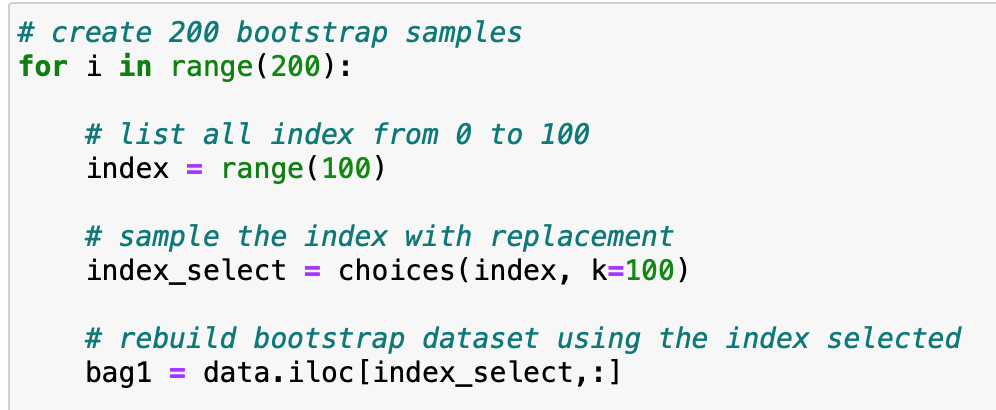
# Code Development

Link to my [jupyter notebook](https://colab.research.google.com/drive/1JrA9AR-j3hpfwBw176jQMfpvBdaxNPul)

## Bagging

We learned bagging and bootstrap samples over the last few weeks and I am very interested in this topic. Therefore, I was trying to implement bootstrap samples and a bagging classifier from scratch.

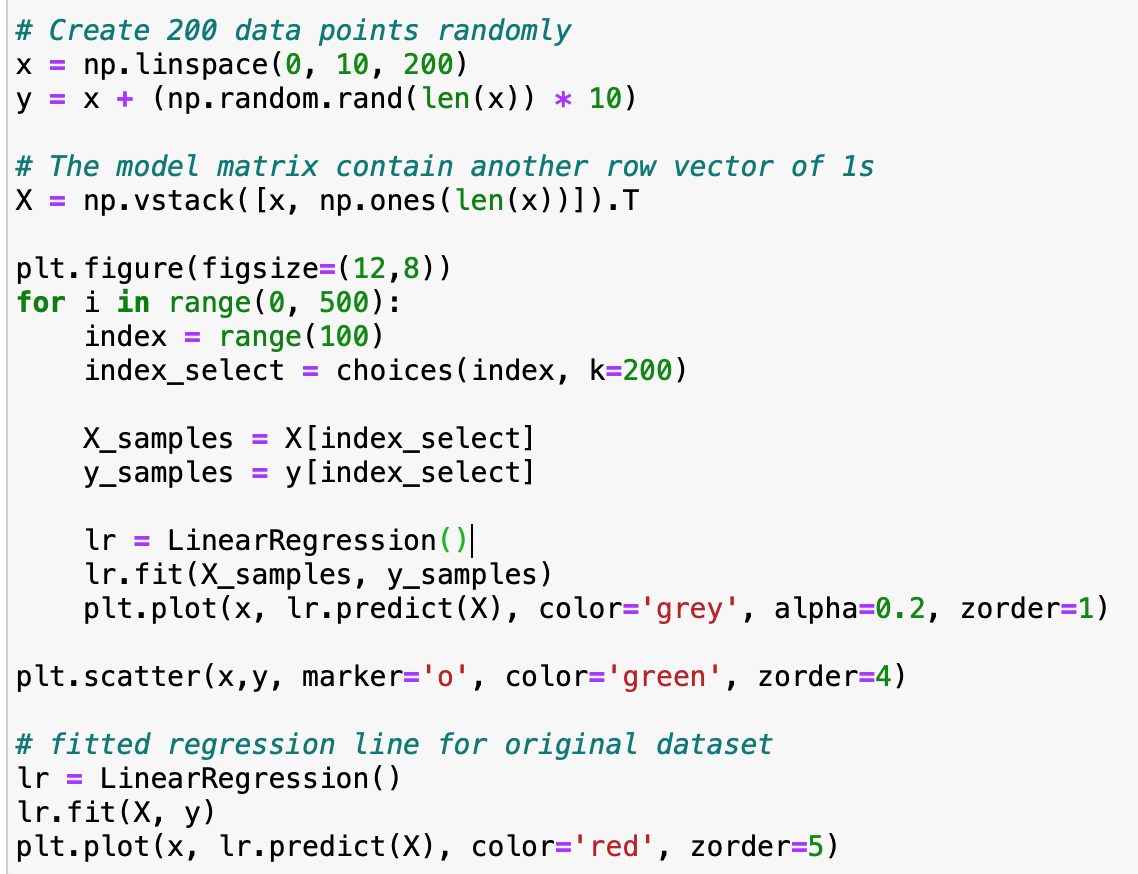
The attached screenshot is the core part of building bootstrap samples and so far it is not hard.

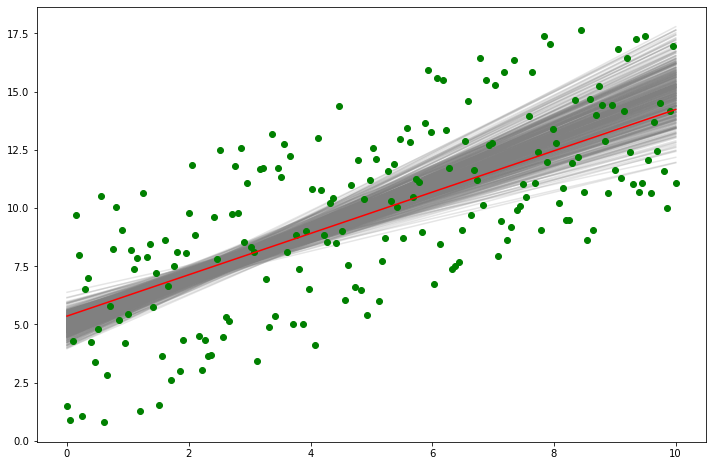


However, when I was trying to fit a decision tree for each bootstrap sample and then combine their predictions to get an ensemble result, I got stuck on this. After I got the out-of-bag prediction, I was not sure how to store the current results and then do the majority vote. Therefore, I decided to simplify it and implement bagging with linear regression.

The idea of linear regression with bagging is that we fit an individual linear model for each bootstrap sample and store the beta coefficient. This would be useful for building confidence intervals and making inferences.

Below is the screenshot of my bagging for linear regression. The data points are created with a random error added to their y values. The green dots are our original dataset and the red line is the fitted line of the original dataset. All grey lines are fitted regression lines for each bootstrap sample.





# Package Used

Link to my [jupyter notebook](https://colab.research.google.com/drive/1rQMyD4PB94wF-ER7SCocBsX7LBaFLRqI)

These are the packages I used for this DMP.

import warnings

warnings.filterwarnings('ignore')

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

# Load libraries

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model\_selection import train\_test\_split # Import train\_test\_split function

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

from sklearn.ensemble import RandomForestClassifier

from sklearn import tree

from matplotlib import pyplot

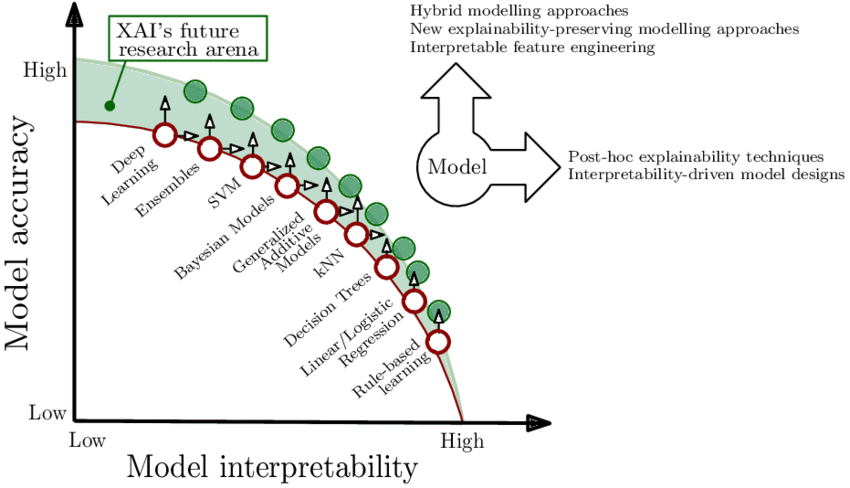
from sklearn.metrics import accuracy\_score, recall\_score

import xgboost as xgb

from sklearn.ensemble import IsolationForest

## Goal 1: Association: Which features help predict income the most?

We are interested in finding one or a small set of features that are mostly related to explaining the income level. Since relationship interpretation is our main goal, we should use models which are good at explaining. We should not try models like random forests since they are hard to interpret. Here is a graph showing the trade-off between model interpretation and model predicting ability.



The models on the right bottom are good candidates for model interpretation, so I am going to fit a model for the decision tree first.

### Decision Tree

A decision tree is a machine learning algorithm that partitions the data into subsets. The partitioning process starts with a binary split and continues until no further splits can be made. Various branches of variable length are formed.

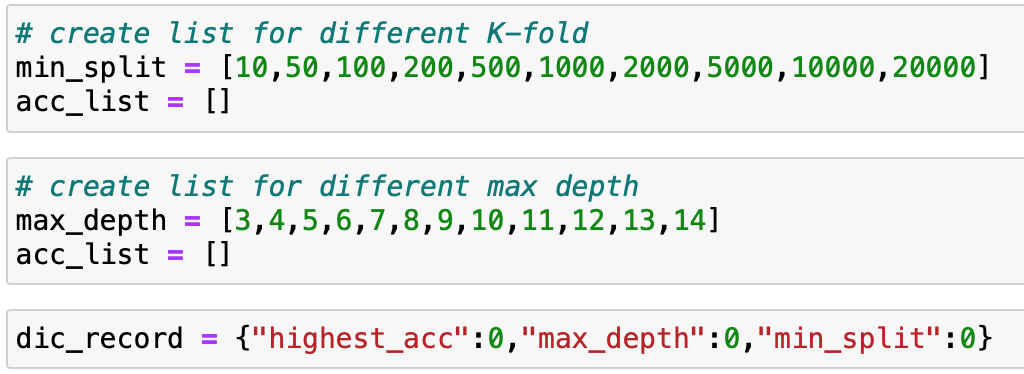
The goal of a decision tree is to encapsulate the training data in the smallest possible tree. The rationale for minimizing the tree size is the logical rule that the simplest possible explanation for a set of phenomena is preferred over other explanations. Also, small trees produce decisions faster than large trees, and they are much easier to look at and understand. There are various methods and techniques to control the depth, or prune, of the tree.[[1]](#footnote-0)

Before fitting the decision tree, I need to do one more modification to my current dataset. We will encode categorical features using **one-hot encoding**, i.e. each category will now be represented by a separate column. The decision tree classifier requires this format. Also, I split my original dataset to 30% as the testing set and 70% as the training set in order to solve the problem of overfitting. I am going to use the training set to fit the model and use the testing set to compute the accuracy.

I first fit a decision tree classifier without any parameter tuning and let’s see how it goes. Here is a visualization of the decision tree created for this dataset. The structure is messy and it is not useful for our visualization. Also, the testing accuracy of the basic classifier without tuning is around 81%. It is bad and we need to improve it!

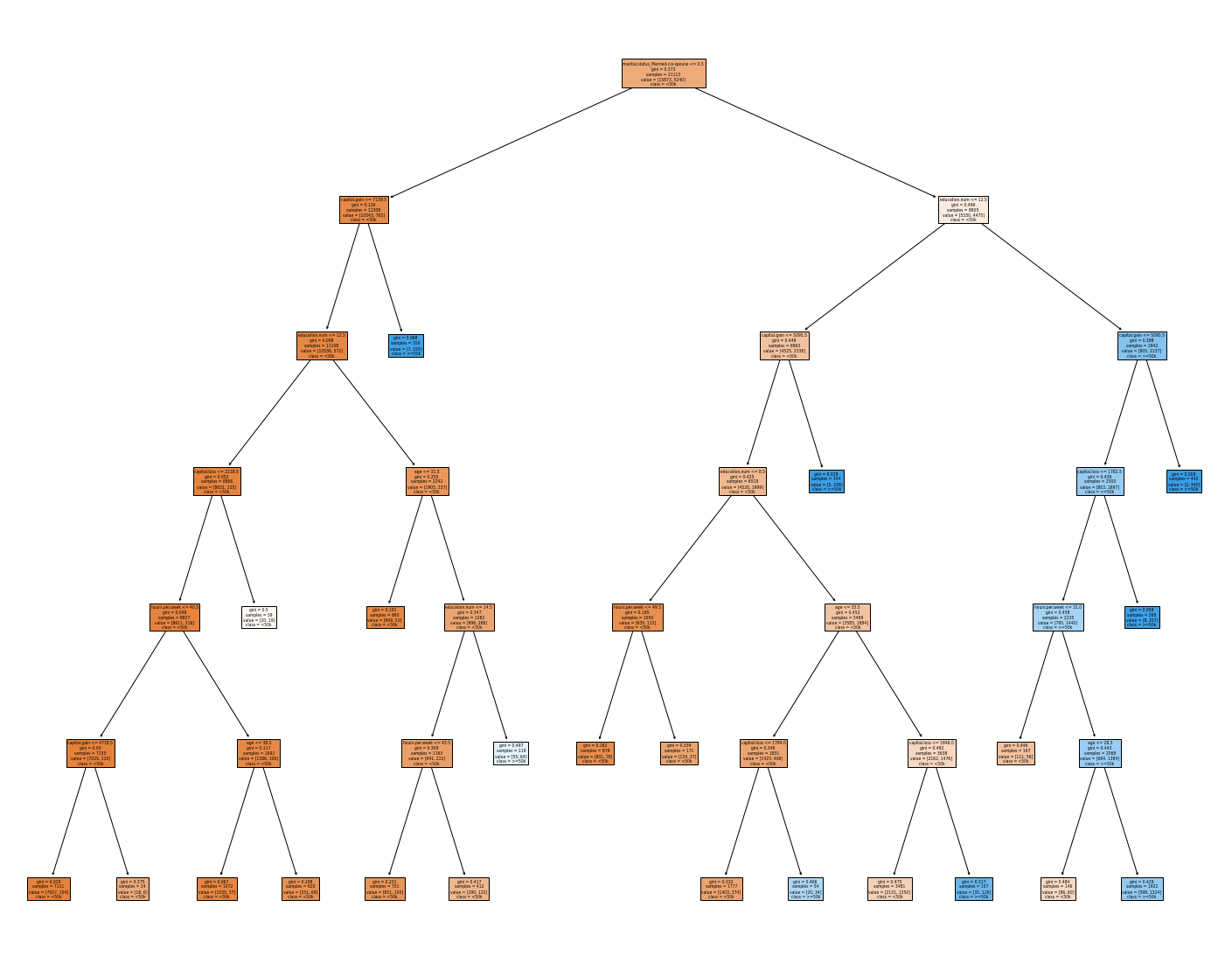


Then, I try hyperparameter tuning and change the value of max depth and min split. I make a list for each parameter and use a double for loop to fit the model looping through different parameters. I also create a dictionary to store and update the best accuracy and the most appropriate max depth and min split.



According to the result, I choose max\_depth = 6 and min\_samples\_split = 1000 to fit the model with the consideration of model interpretability. Short max depth helps in interpretation. I achieve higher accuracy compared to our last fit, which is around 85%. Here is a visualization of the fitted decision tree. This one is way better than the last one. If we zoom in on the graph a little bit, we can see the predictors we use for the first three splits are education num, capital.gain, and age.

We can get a sense of the features that are important in the model by observing the tree graph. Indeed, the variable ‘marital status’ was the most significant for reducing the Gini Index (since it forms the first splitting rule and it is the root node). The variable ‘capital gain’ is used in multiple splits, indicating that it is also important for predicting income. Specifically, a higher capital gain (in dollars) is associated with high income. Also, age and education num are frequently used for the first several splits and larger values in age and education num are associated with higher income.



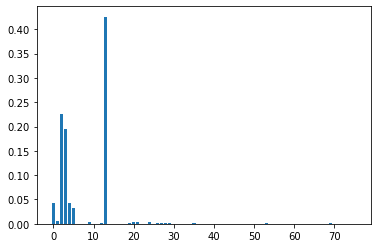
Even though the split shows the set of features which related to income the most, I want to explore the **feature importance** of each predictor used in the decision tree and get a complete understanding of which predictors influence income the most. Feature importance refers to a class of techniques for assigning scores to input features to a model that indicates the relative importance of each feature when making a prediction. The scores are useful and can be used in a range of situations in a predictive modeling problem, such as:

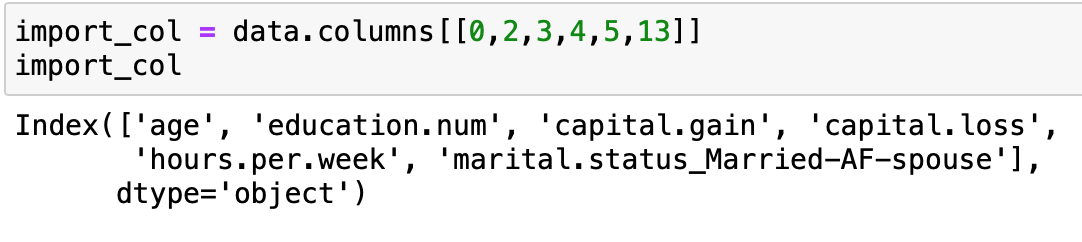
Better understanding the data.

**Better understanding a model. (It tells people which features influence income level)**

Reducing the number of input features.

We can use the CART algorithm for feature importance implemented in scikit-learn as the DecisionTreeRegressor and DecisionTreeClassifier classes. After being fit, the model provides a feature\_importances\_ property that can be accessed to retrieve the relative importance scores for each input feature.[[2]](#footnote-1) Here is a graph showing the feature importance scores of each predictor. It shows that columns 0,2,3,4,5, and 13’s importance scores are very high compared to the other columns.





I map the column back to their column name and it shows that **'age', 'education.num', 'capital.gain', 'capital.loss', 'hours.per.week', 'marital.status\_Married-AF-spouse'** explain most of the relationship of income level. So far, the relationship between our response variable income and the set of demographic features is much more clear with the help of the decision tree!

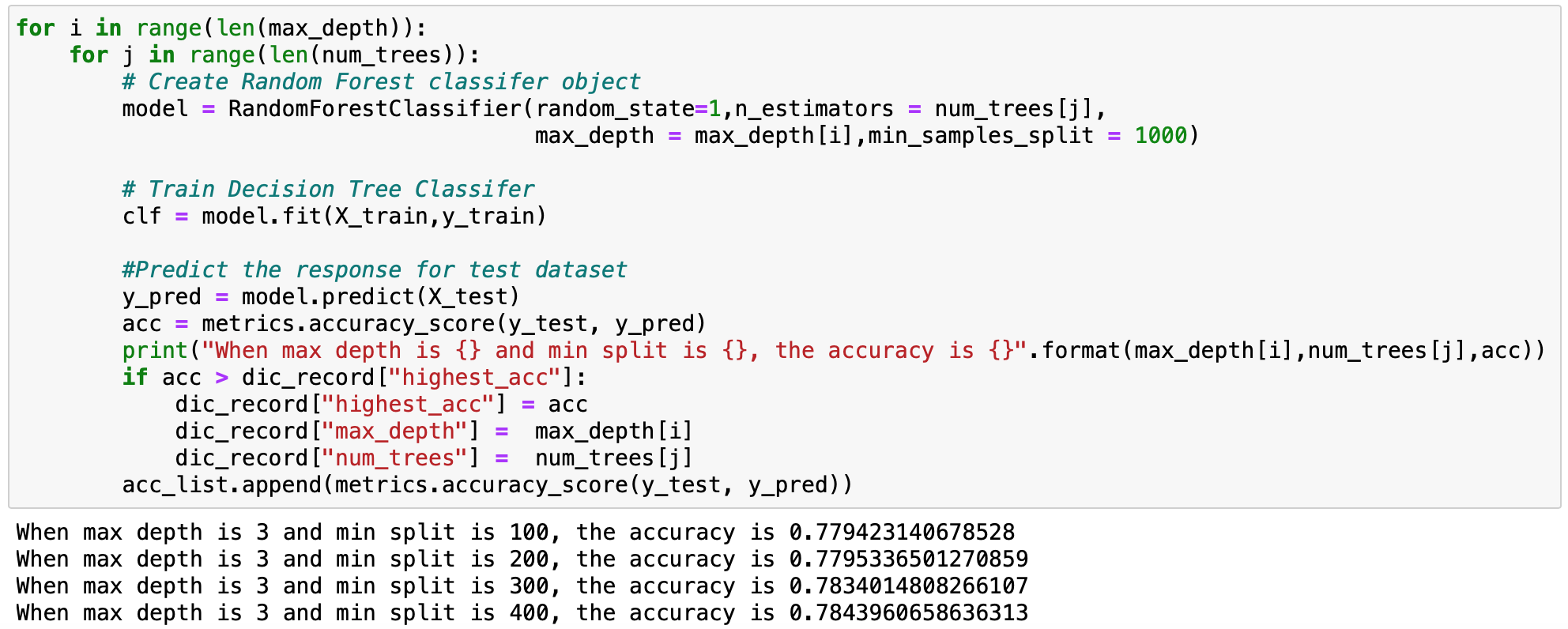
## Goal 2: Prediction: The highest prediction accuracy by training different classifier

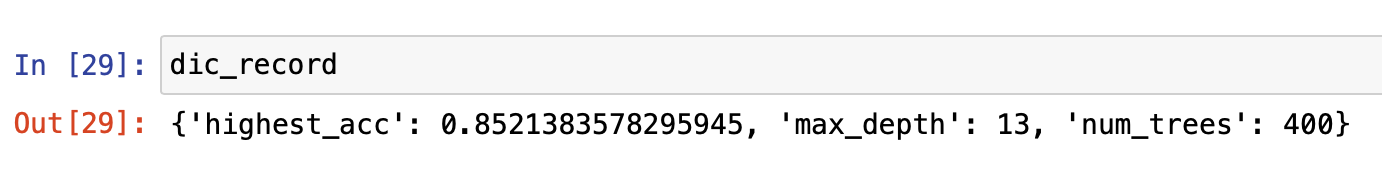
For model prediction as our goal, we want to choose those classifiers on the left top side, and the first model I want to try is the ensemble model. We learned Bagging, Boosting, and random forest in our ensemble model chapter and I would like to try all of these and compare their accuracy to see which one fits the best.

We are interested in building a predictive model for ‘income01’, which is a binary categorical feature (‘0 or ‘1) . We expect that the relationship between ‘income01’ and other demographic information is highly complex, and we are not guaranteed that it is linear. Considering this, a random forest model is worth trying since it is the model focused on highly accurate prediction instead of interpretation. I also tried several tree-related classifiers such as the bagged forest model and boosting tree for comparison.

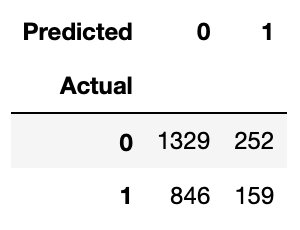
### Random Forest

I first tried the random forest classifier. Like what I did for training the decision tree model, I conducted hyperparameter tuning and finally decided to use the model with the number of trees = 400 and max depth = 13 because the testing accuracy is the highest. I use max depth (3,4,5….14) and the number of trees from 100 to 2000(100,200…2000) as candidate hyperparameters. And below is the related code and a confusion matrix. The testing accuracy is around 85.2%, which is slightly greater than the accuracy achieved by the decision tree model.





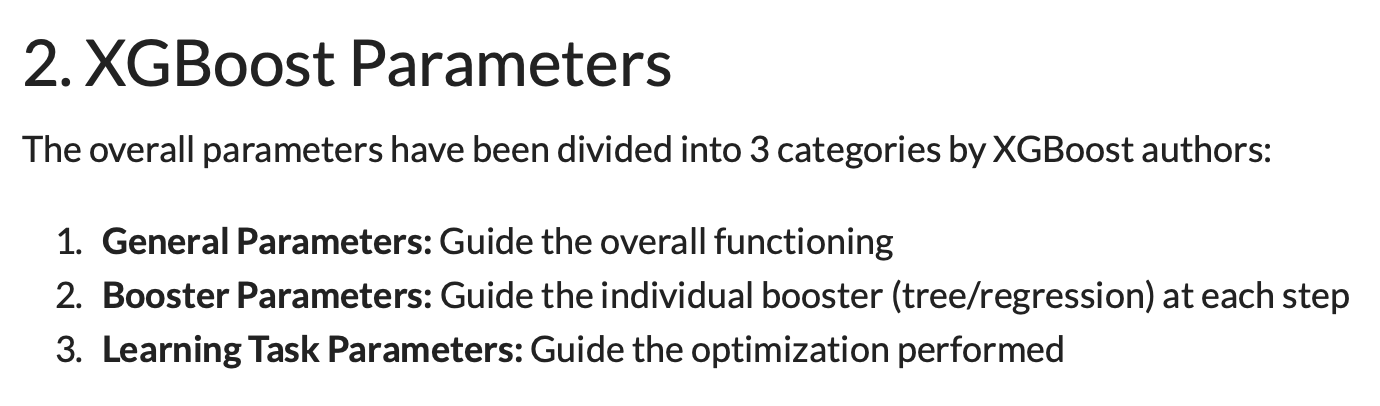
Here is the confusion matrix of our test set prediction. It shows that the model sometimes misclassifies those who are high income as low income(acutal:1;predicted:0).

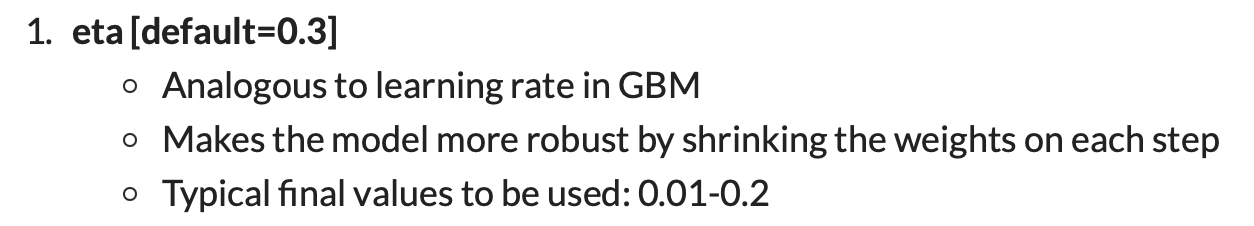


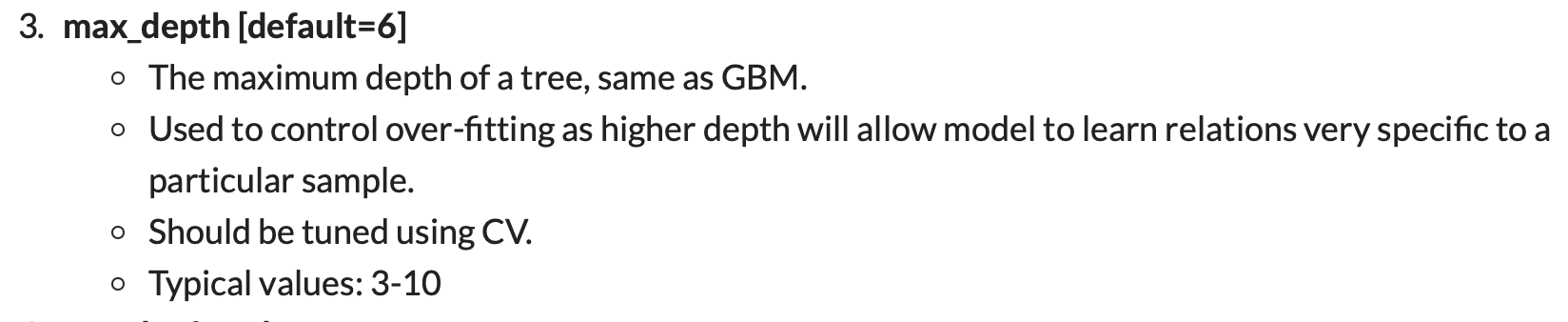
### XGBoost

Next, I am going to use one of the most popular predictive models in machine learning, which is the XGBoost. Boosting is a strong alternative to bagging. Instead of aggregating predictions, boosters turn weak learners into strong learners by focusing on where the individual models (usually Decision Trees) went wrong. In Gradient Boosting, individual models train upon the residuals, the difference between the prediction and the actual results. Instead of aggregating trees, gradient boosted trees learn from errors during each boosting round.[[3]](#footnote-2) XGBoost is short for “eXtreme Gradient Boosting.” The “eXtreme” refers to speed enhancements such as parallel computing which makes XGBoost approximately 10 times faster than traditional Gradient Boosting. In addition, XGBoost includes a unique split-finding algorithm to optimize trees, along with built-in regularization that reduces overfitting. Generally speaking, XGBoost is a faster, more accurate version of Gradient Boosting.

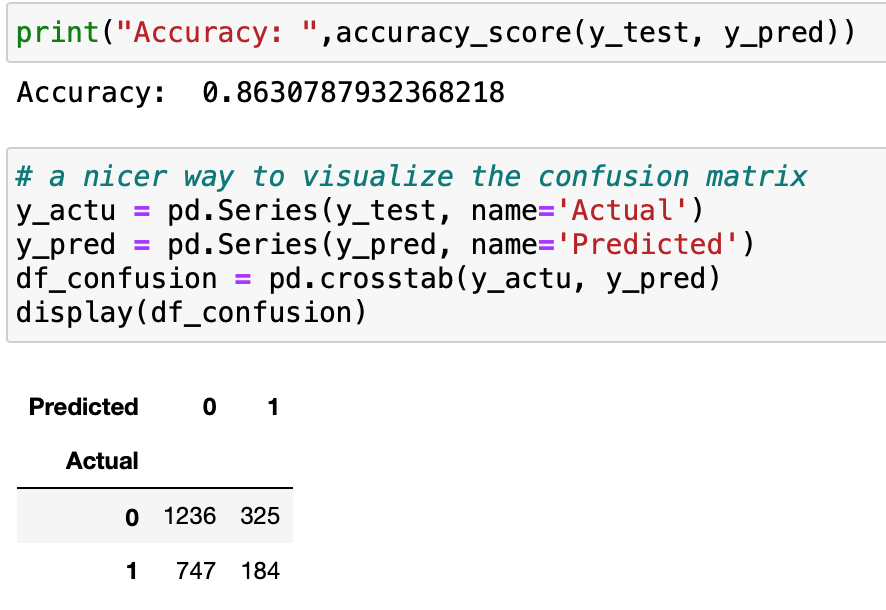
The hyperparameter tuning for XGBoost is a little bit complicated. This algorithm contains three groups of hyperparameters and I learn the details of these from this website.[[4]](#footnote-3)







I focus on **eta(learning rate)** and **max\_depth** and choose eta=0.05 and max\_depth as the most appropriate parameters. The **number of trees** can be set via the “n\_estimators” argument and defaults to 100. Indeed, the accuracy by using XGBoost is the highest, which is around 86.3%, higher than decision tree and random forest. If prediction is our main goal, we should choose XGBoost. For the same reason, XGBoost is not good at interpretation.



|  | Accuracy(test set) | Interpretation V.S. Prediction | Hyperparameter notes |
| --- | --- | --- | --- |
| Random Forest(Bagging) | 85.2% | Prediction | random\_state=1,n\_estimators = 400,max\_depth = 13,min\_samples\_split = 1000 |
| XGBoost | 86.3% | Prediction | objective="binary:logistic", random\_state=42,eta = 0.05,max\_depth=7,n\_estimators=400 |
| Decision Tree | 84.9% | Interpretation | criterion = 'gini',  max\_depth = 6,  min\_samples\_split = 1000 |

In conclusion, for getting a more accurate result of predicting income level based on census demographic information, we choose XGBoost as our ideal model.

**More thoughts**: Identifying low/high income customers is essential for business or loan-related companies, especially usually customers are not willing to self-report their income. After knowing their income level and some of their personal banking history, we can predict appropriate interest rates for microloan customers and finally make this process highly automated. Also, the income level information might help us conduct further customer segmentation. Maybe increase response rates, customer loyalty, and return on investment by engaging the right customers, reducing marketing costs by targeting the most valuable customers.

## Goal 3: Anomaly Detection

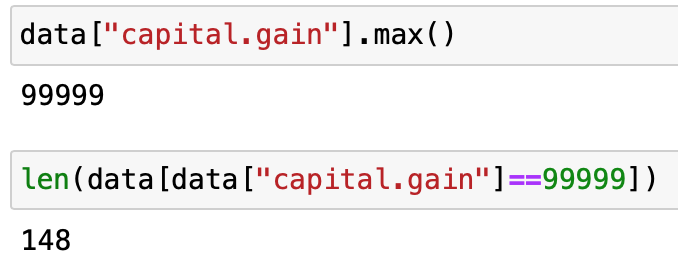
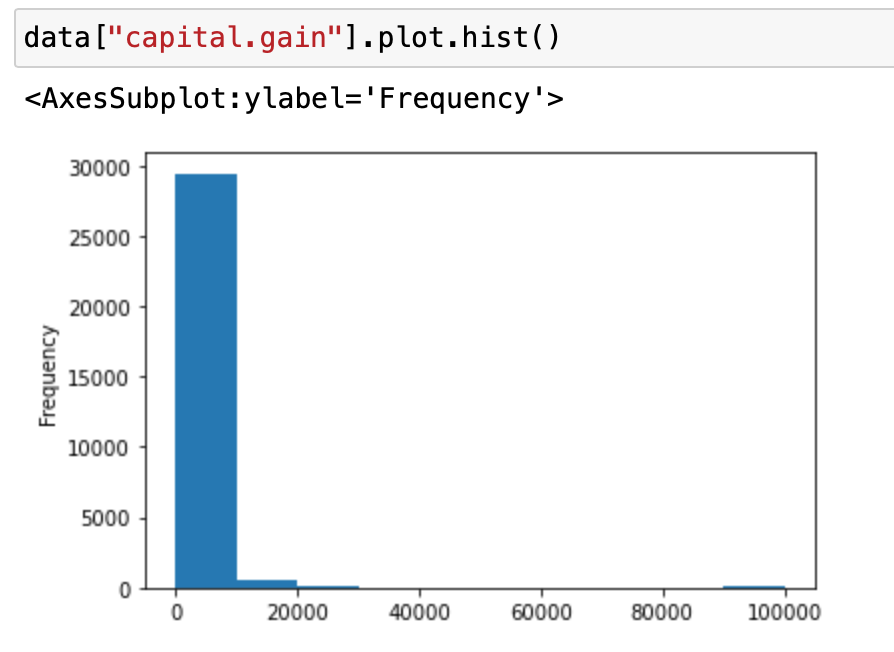
Any observation that deviates significantly from the other observations is called an Outlier. Anomaly detection is important and finds its application in various domains like detection of fraudulent bank transactions, network intrusion detection, sudden rise/drop in sales, change in customer behavior, etc.

Many techniques were developed to detect anomalies in the data and I will mainly look at the implementation of Isolation Forests – an unsupervised anomaly detection technique[[5]](#footnote-4).

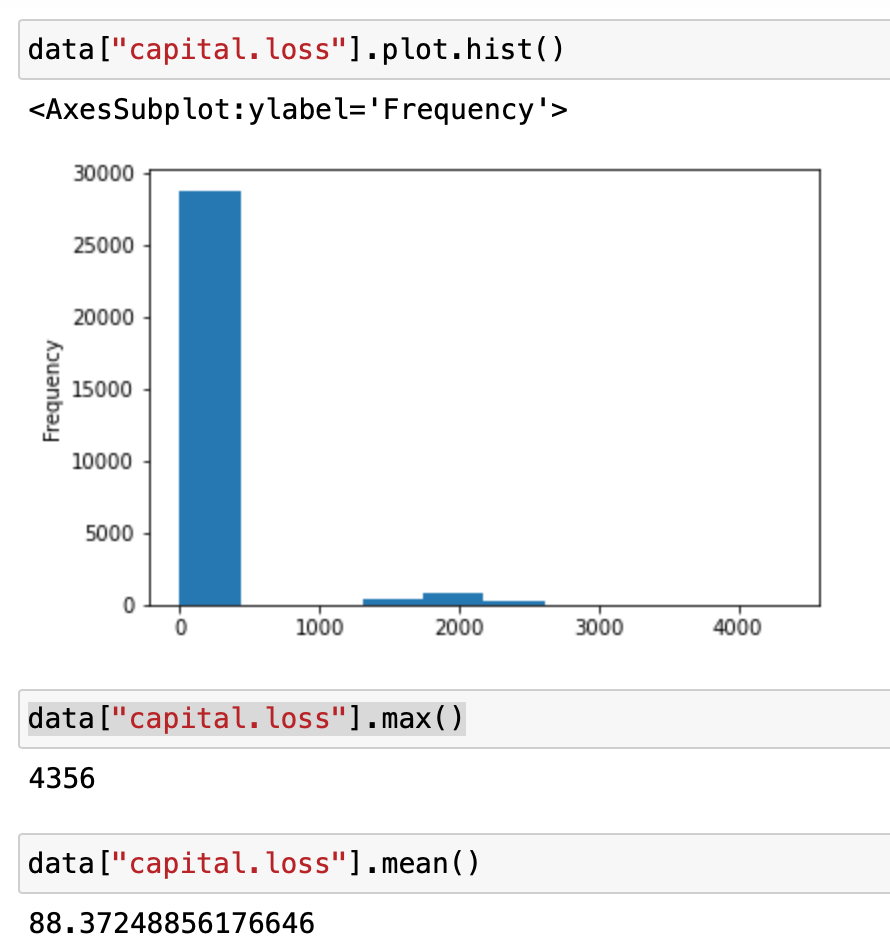
### EDA

I first conduct some exploratory data analysis(EDA) and see which features are having the potential risks of containing outliers. I checked “age” and “education number”, and it seems that their distribution is appropriate and there are no unreasonable outliers. However, for "capital.gain" and "capital.loss", the distribution is risky. Here are the histograms of these two predictors and how large their maximum values are.

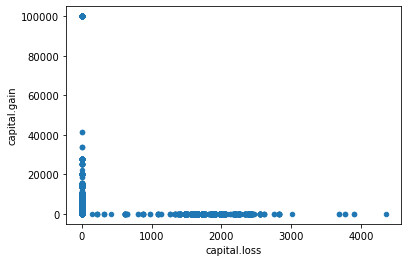
For "capital.gain", the mean is around 1092 but the max value is 99999, and 148 rows have this number. It is unusual and might be the candidate of outliers due to census input error.



For "capital.loss", the mean is around 88 but the max value is 4356, which is also unusual and might be the candidate of outliers due to census input error.

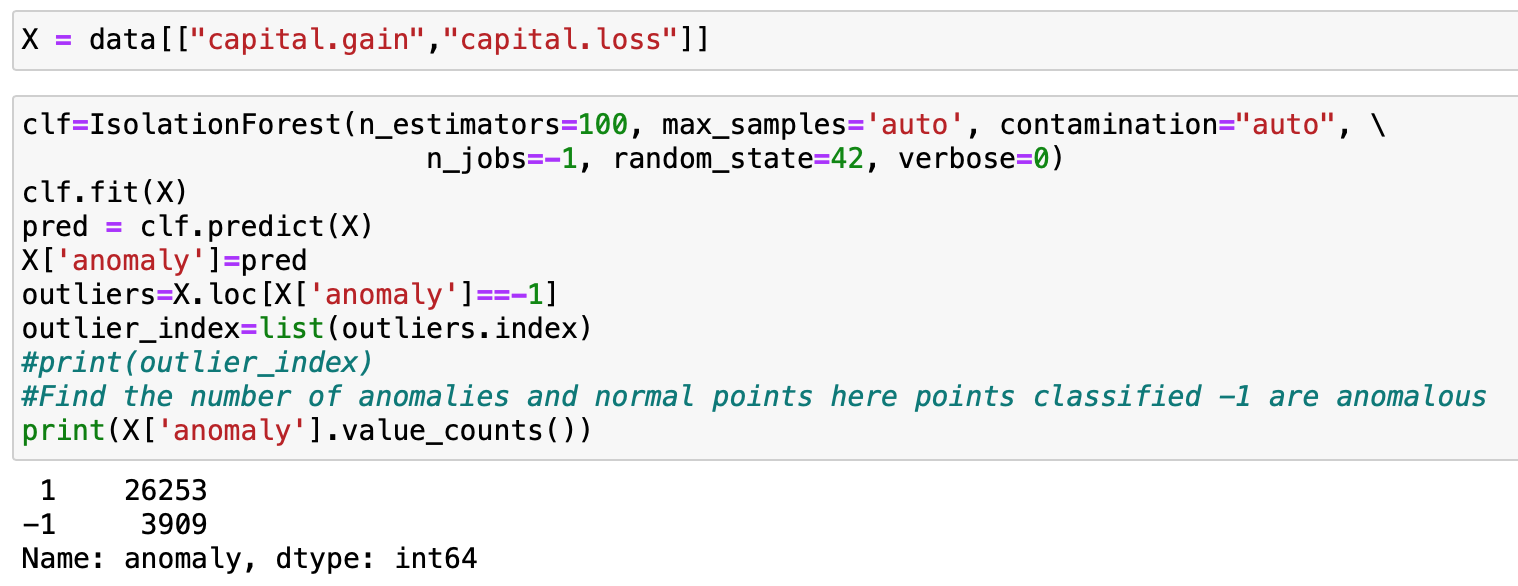


I also draw the scatter plot and it is apparent that some outliers exist here.

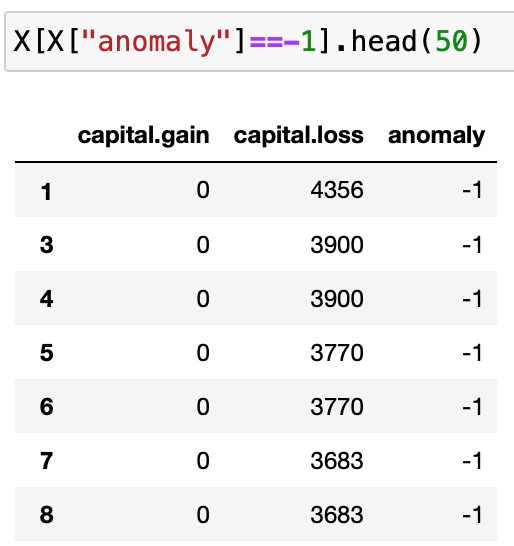


### Isolation Forest

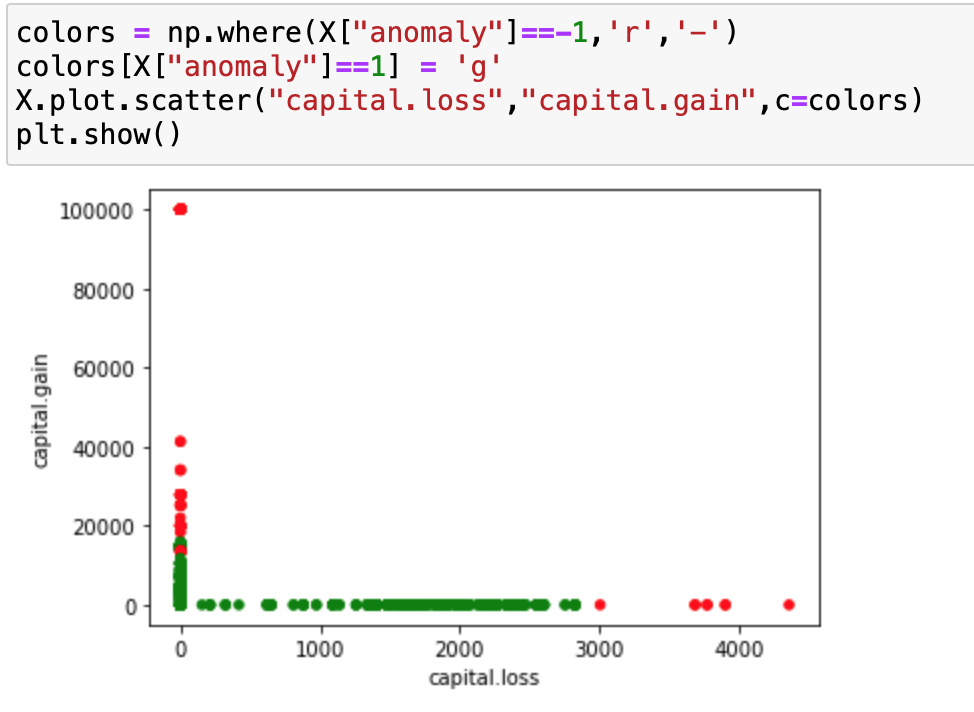
Next, we are going to use an Isolation forest to help detect the outliers. We assign -1/1 to each observation and -1 indicates it is an outlier. We write this information back to our dataset and compare if the anomaly detection is reasonable. The “contamination” in the isolation forest classifier indicates the proportion of outliers. I do not specify this because I want this model to help identify them. Around 13% of data points are classified as outliers.



I also check the first several rows of observations which are classified as outliers. It makes sense because those potential outliers are either with a high capital gain or a high capital loss.



I plot the scatter using red color showing the points which are considered as outliers. It makes perfect sense! We might consider remove those data points and refit our model to see if the accuracy get increased.



# Exercise Theory

See the [individual file](https://docs.google.com/document/d/10ReaBbyuWVgo7_b4tSaF-3Ccp88L9ejn/edit) under the same final DMP folder

# Self-assessment

Overall, I believe I develop a comprehensive understanding of all data mining topics we learned this semester. I choose both supervised learning(model interpretation and model classification) and unsupervised learning(Anomaly detection) for this final DMP and demonstrate my learning by showing how modeling helps solve our questions.

Most of the modeling techniques I used in this DMP are those I never used in my previous DMP, such as bagging, boosting, and isolation forest. I love learning bagging and boosting because they are great iterative skills for improving the model performance of decision trees and random forests. The concepts are easy to understand but the code implementation is tricky. I will definitely explore more about the coding logic of XGBoost and bagging classifiers.

Some biggest takeaways:

* The trade-off between model prediction accuracy and interpretation ability
  + Use train/test set to avoid overfitting
  + K-fold method/LOOCV
* Clustering algorithms are useful in many real-world applications
  + Business: customer segmentation/identify valuable customers
* Anomaly detection for identifying outliers
  + Before taking this class, the only way I know for identifying outliers is to draw some boxplot and find the outliers
  + Now, I have plenty of methods to detection abnormal points, especially in a high dimensional dataset
* Tree structure
  + There are many ways to improve the prediction accuracy for tree-like algorithms, such as bagging and boosting.
  + Random forest is powerful but hard to interpret.
* The application of Association analysis
  + How it gets used in the market dataset
  + Cross support patterns happen a lot in real-life analysis!

I am glad that you offer the data mining class this semester. I really learned a lot and enjoyed those topics. To be honest, I was so lost for the first DMP because I had no idea about what I should include for demonstrating my learning and understanding of the topics. However, I am getting more familiar with the pace of this class and know better about how to prepare for the assessment as the semester goes. All my previous graded items are A- and I got A+ for the last summative assessment. My expected grade for this semester would be something around A.

You are an awesome professor and I enjoy asking questions and chatting with you during your office hours. I am planning to pursue a data science related master’s program in the future and this class will definitely be a stepping stone for my machine learning exploration. Hope you enjoy reading my DMP and enjoy the winter break!

1. https://webfocusinfocenter.informationbuilders.com/wfappent/TLs/TL\_rstat/source/DecisionTree47.htm [↑](#footnote-ref-0)
2. https://machinelearningmastery.com/calculate-feature-importance-with-python/ [↑](#footnote-ref-1)
3. https://towardsdatascience.com/getting-started-with-xgboost-in-scikit-learn-f69f5f470a97 [↑](#footnote-ref-2)
4. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/ [↑](#footnote-ref-3)
5. https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/ [↑](#footnote-ref-4)