Census Income Data

Motivation:

Can we accurately predict a person's income from their demographic information such as age, education, marital status.

Objectives:

Build a model to best predict whether a person's income will be less than or greater than $50K based on their census data.

Data and Tools

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             48842 non-null  int64 
 1   workclass       48842 non-null  object
 2   fnlwgt          48842 non-null  int64 
 3   education       48842 non-null  object
 4   education-num   48842 non-null  int64 
 5   marital-status  48842 non-null  object
 6   occupation      48842 non-null  object
 7   relationship    48842 non-null  object
 8   race            48842 non-null  object
 9   sex             48842 non-null  object
 10  capital-gain    48842 non-null  int64 
 11  capital-loss    48842 non-null  int64 
 12  hours-per-week  48842 non-null  int64 
 13  native-country  48842 non-null  object
 14  income          48842 non-null  int64 
dtypes: int64(7), object(8)
memory usage: 5.6+ MB
```

pandas

scikit learn

seaborn

dmlc XGBoost

matplotlib

Processing Data and Feature Engineering

- Cleaned Data removed 10 lines of nulls
- Scaled Data for K-nearest neighbors
- Created dummy columns to convert categorical data
- Converted >= $50K to 1 and < $50K to 0

age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	workclass_?	workclass_Federal-gov	workclass_Local-gov	workclass_Never-worked	...	native-country_Portugal	native-country_Puerto-Rico
25	226802	7	0	0	40	0	0	0	0	...	0	0
38	89814	9	0	0	50	0	0	0	0	...	0	0
28	336951	12	0	0	40	0	0	1	0	...	0	0
44	160323	10	7688	0	40	0	0	0	0	...	0	0
18	103497	10	0	0	30	1	0	0	0	...	0	0

ows × 108 columns

Model Comparison

	K nearest neighbor	Logistic Regression	Decision Tree	Random Forest	Extra Trees	XG Boost
Optimized for	N_neighbors	Class_weight	Max_depth	Max_depth	Max_depth	Class_weight
Accuracy	84.2%	80.3%	85.5%	86.9%	85.3%	87.5%
F1	63.1%	39.2%	67.1%	69.2%	65.3%	71.4%

Tuning the XGBoost model

- Additional tuning Boost included learning_rate = .2, gamma = 7, scale_pos_weight = 2
- Randomized Search included learning_rate, n_estimators, min_child_weight, gamma, subsample, max_depth, and scale_pos_weight

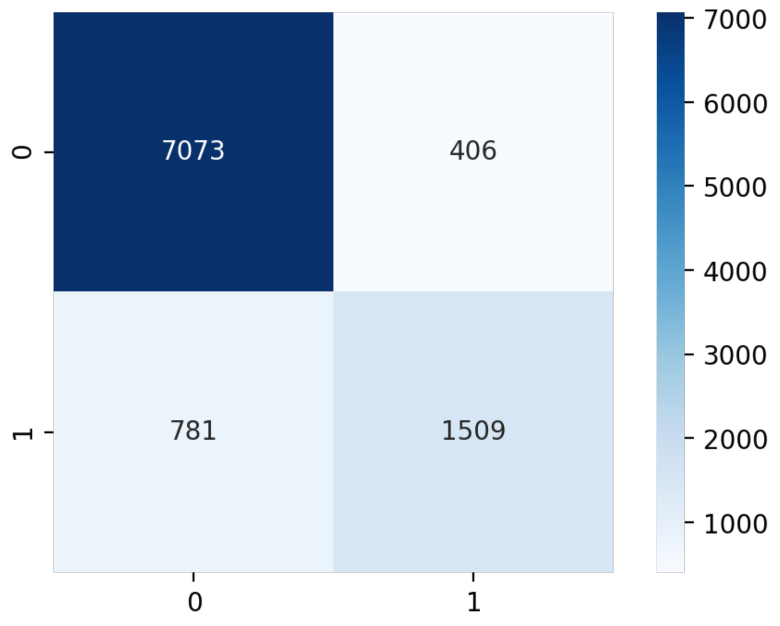
	XG Boost	XG Boost Additional Tuning	XG Boost Randomized Search with F1
Accuracy	87.5%	87.8%	85.6%
F1	71.4%	71.8%	72.1%

Next Steps

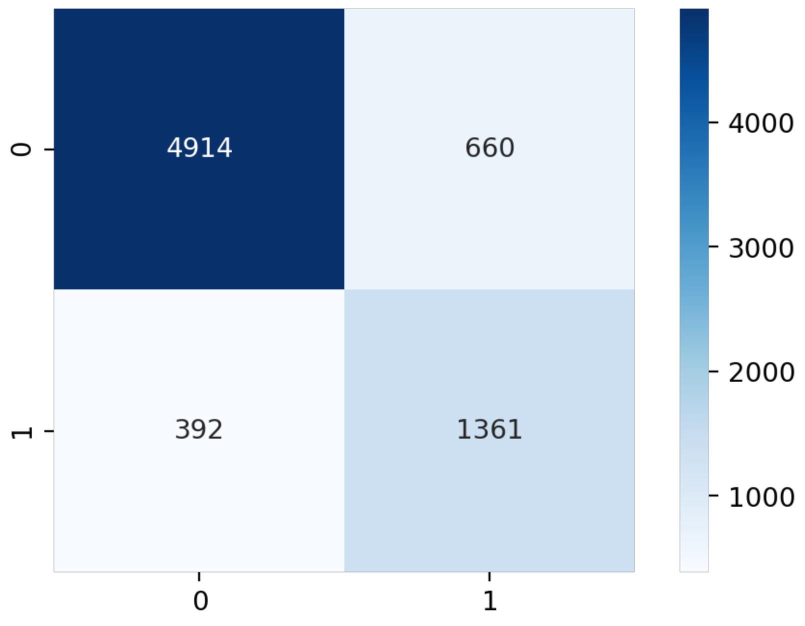
- Review column fnlwgt and how that weighting could affect the model.
- Explore other ways to tune the Random Forest model.
- Implement an Object Oriented Programming approach.
- Standardize the train test validate approach used.

Confusion Matrixes – XG Boost

XG Boost Additional Tuning

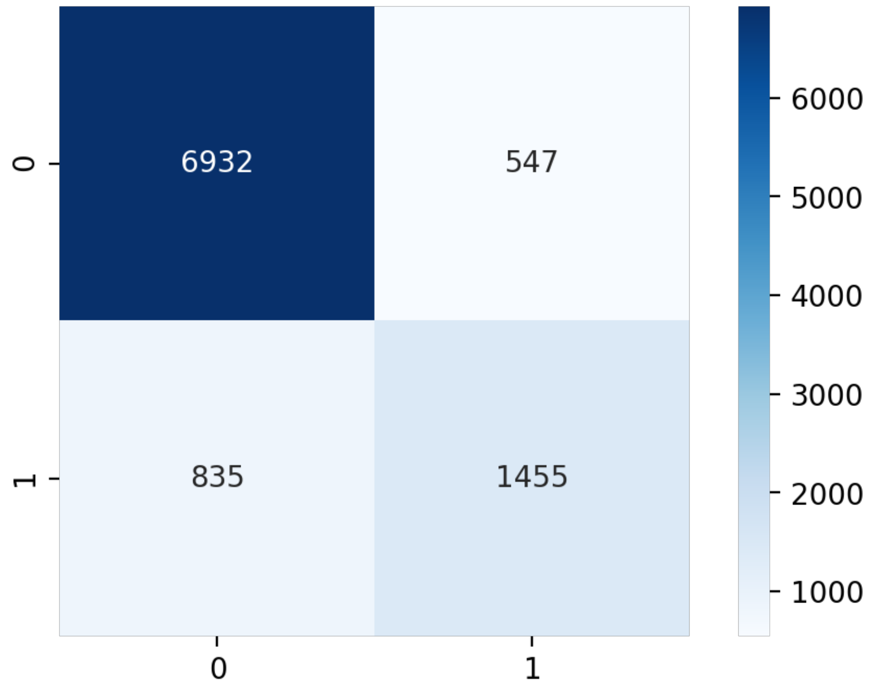


XG Boost Randomized Search F1

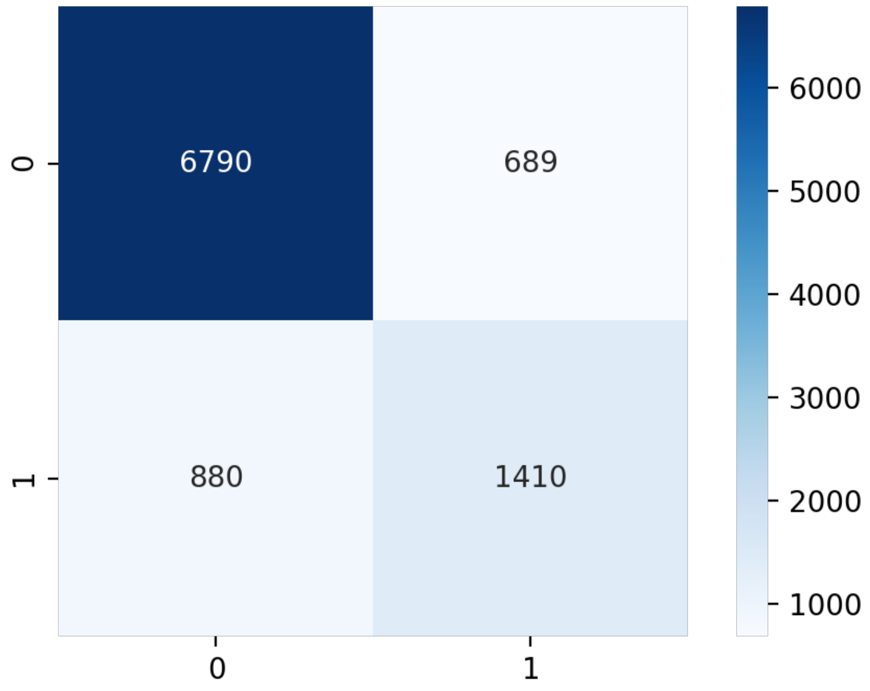


Confusion Matrixes – Random Forest and Extra Trees

Random Forest

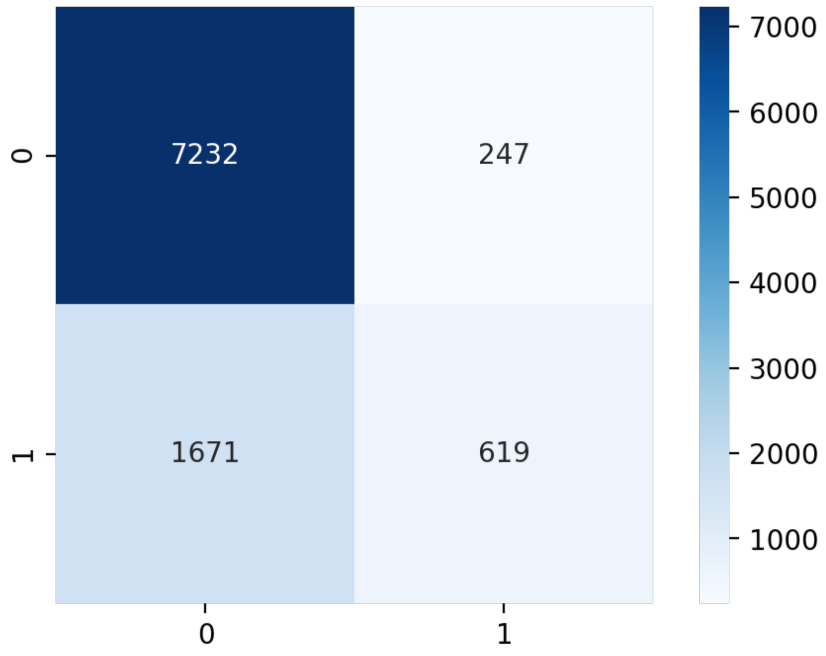


Extra Trees

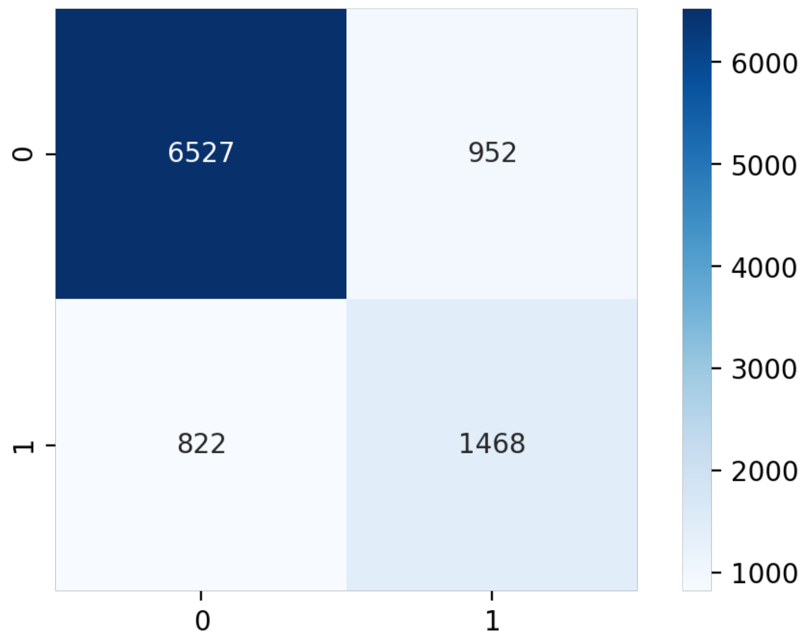


Confusion Matrixes – Logistic Regression and Decision Trees

Logistic Regression

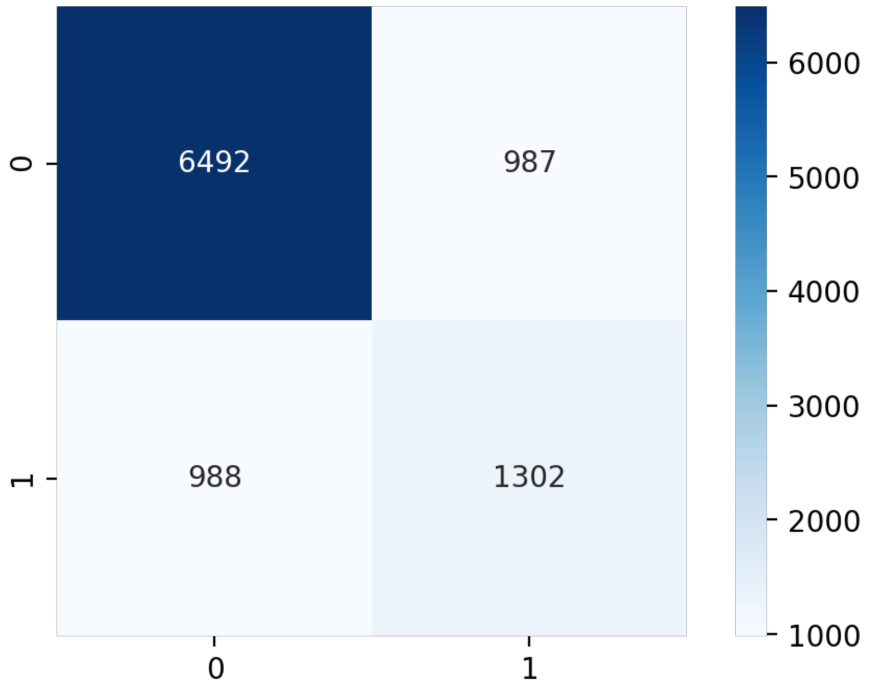


Decision Trees

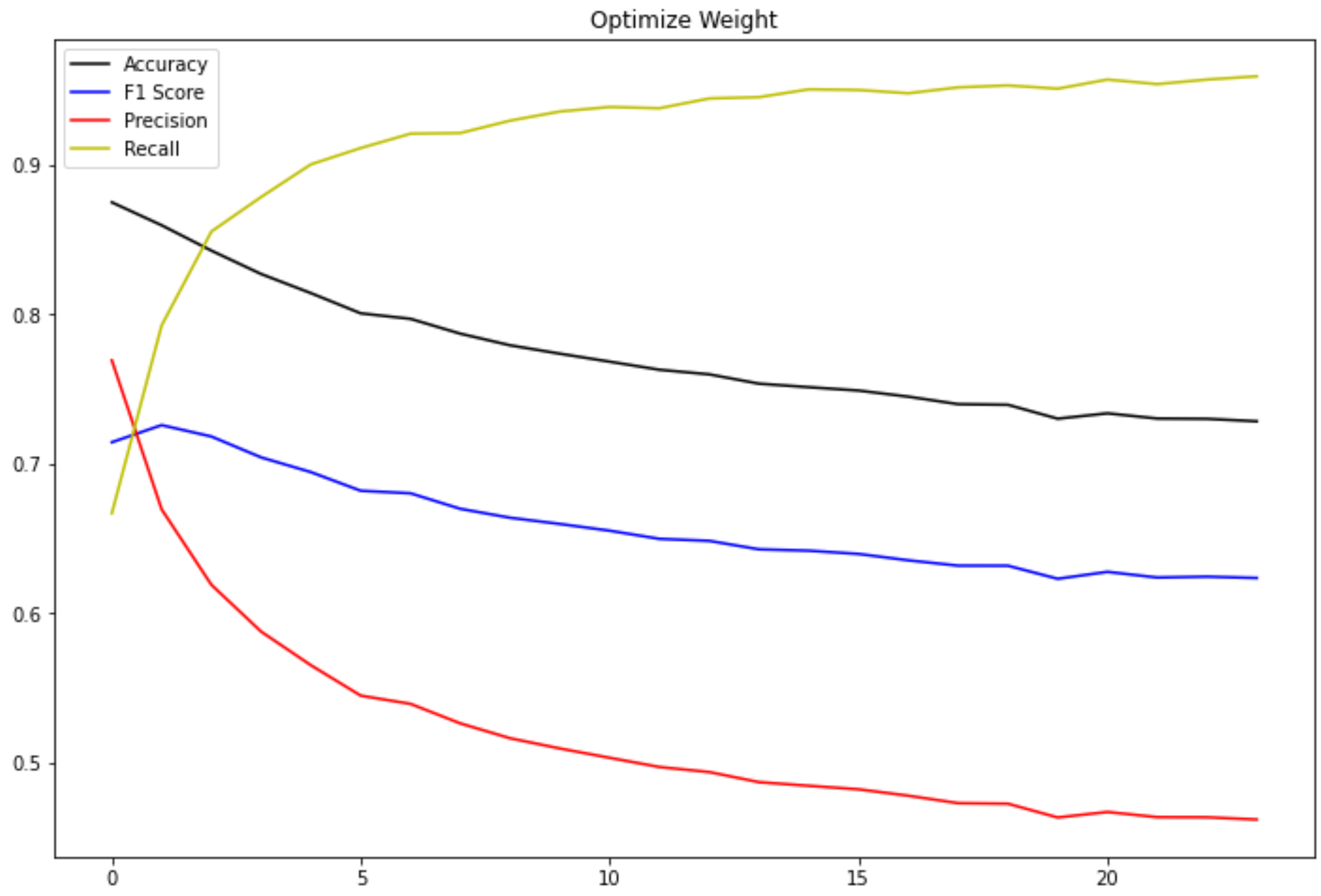


Confusion Matrixes – Knn

Knn



Metrics with Optimize Weight in XGBoost



Metrics with Optimize Weight – Log Regression