LOAN PREDICTION

Yuhao Wang

- Data processing
- Exploratory data analysis
- Building model
- Model evaluation

ISSUES WITH DATA

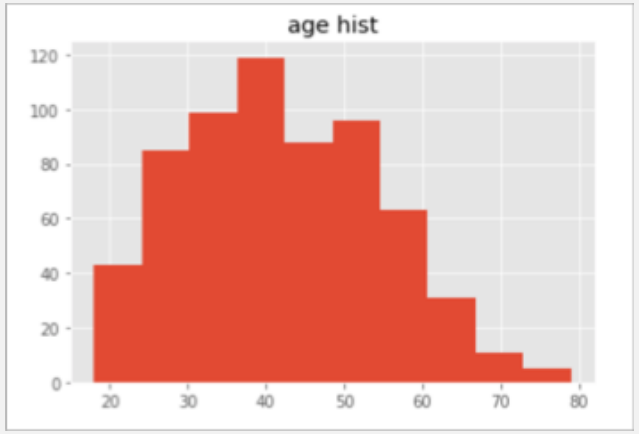
Problems with "X":

- Slightly corrupted
- Missing Value
- Several id unrecognized
- "Useless" variable

Problems with "Y"

- Not matched with "X"
- Clients with more than two loans

DATA PROCESSING



- Create “age” variable
- Extract “email” type
- Missing value imputation
- Data binning
- Create dummy variables

```
40s    170
30s    170
50s    135
20s    106
60s     41
70s     12
10s      6
Name: age, dtype: int64
```

```
hotmail    133
aol        132
bing       130
yahoo      130
gmail      115
Name: email, dtype: int64
```

EXPLORATORY DATA ANALYSIS

After data cleaning:

640 rows × 55 columns

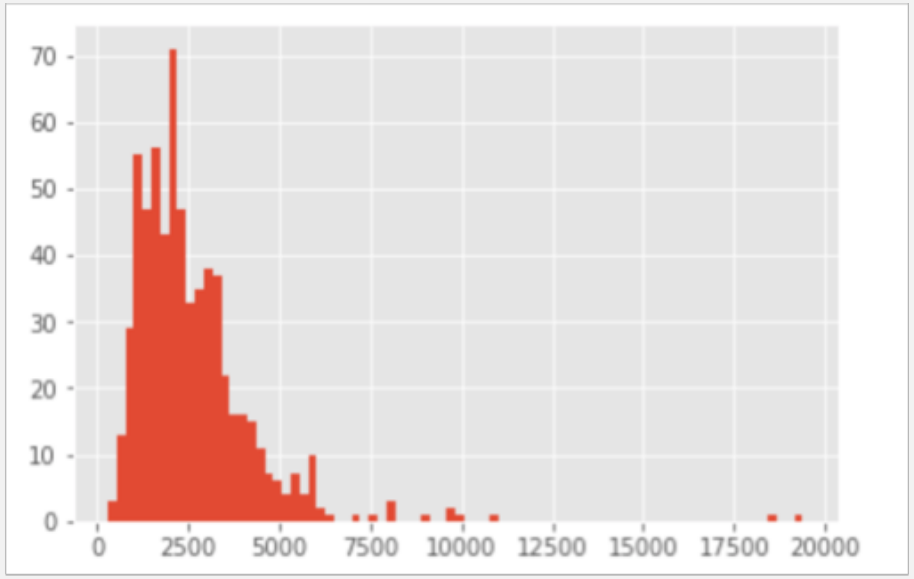
```
0    351
1    289
Name: flgGood, dtype: int64
```

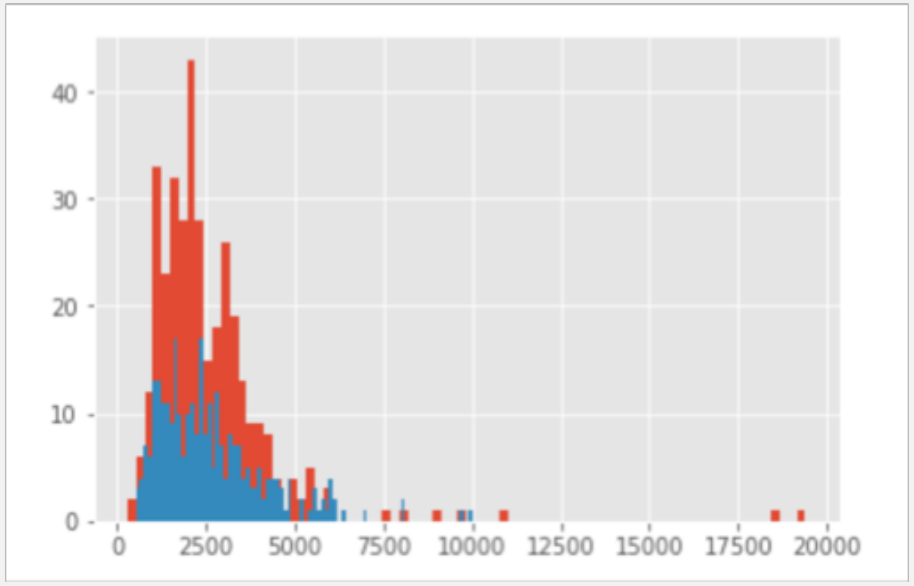
amount_requested	0
monthly_rent_amount	0
loan_duration	0
num_payments	0
payment_amount	0
amount_approved	0
duration_approved	0
payment_amount_approved	0
address_zip	0
bank_routing_number	0
monthly_income_amount	0
raw_l2c_score	0
raw_FICO_telecom	0
raw_FICO_retail	0
raw_FICO_bank_card	0
raw_FICO_money	0
flgGood	0
age_20s	0
age_30s	0
age_40s	0

...

MONTHLY_INCOME_AMOUNT

```
count       640.000000
mean       2616.026563
std        1712.436920
min         300.000000
25%        1503.000000
50%        2245.000000
75%        3200.000000
max       19392.000000
Name: monthly_income_amount, dtype: float64
```

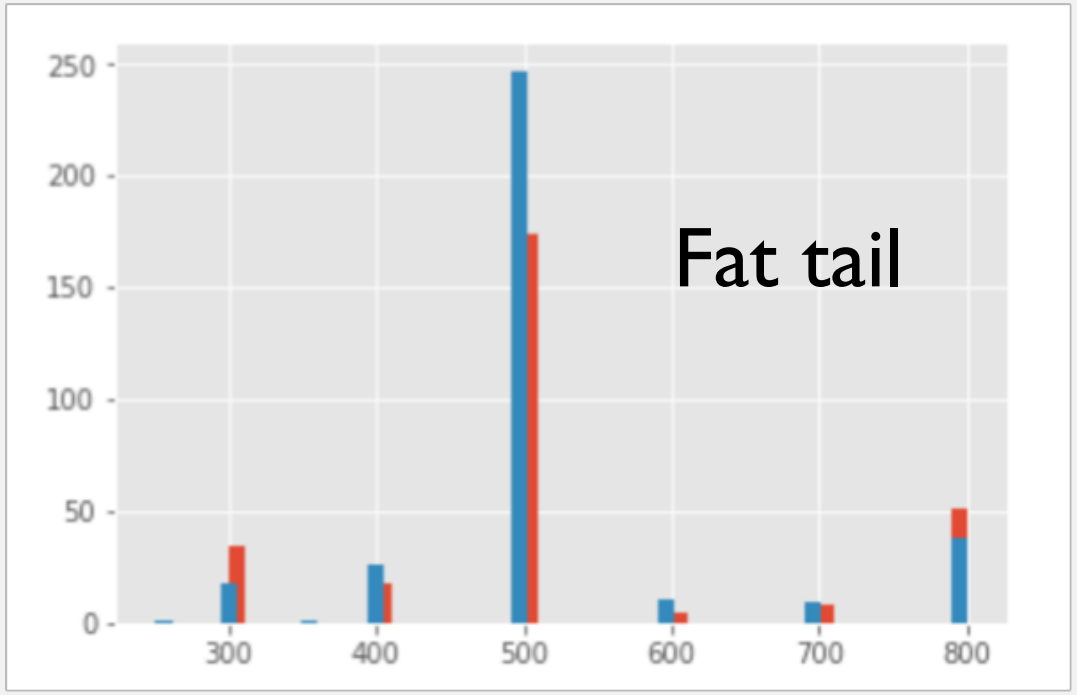


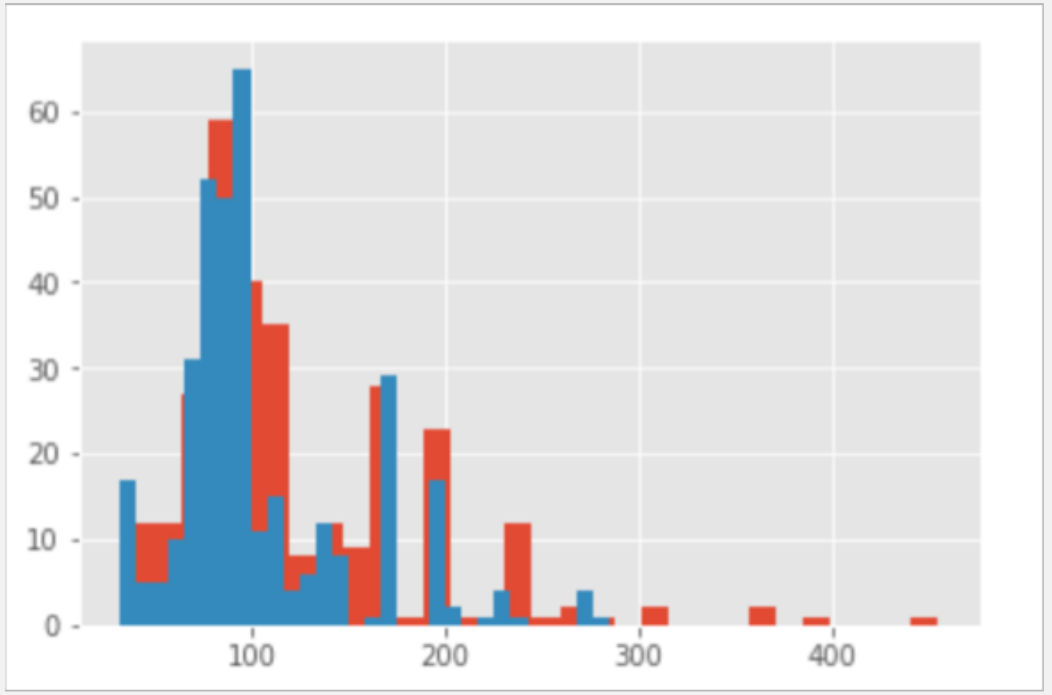


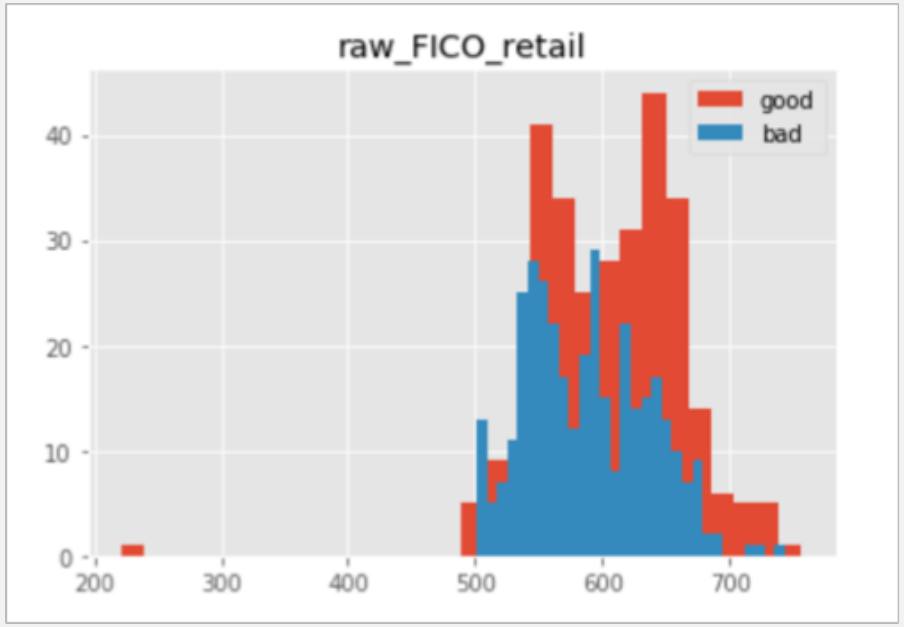
AMOUNT_APPROVED

PAYMENT_AMOUNT_APPROVED

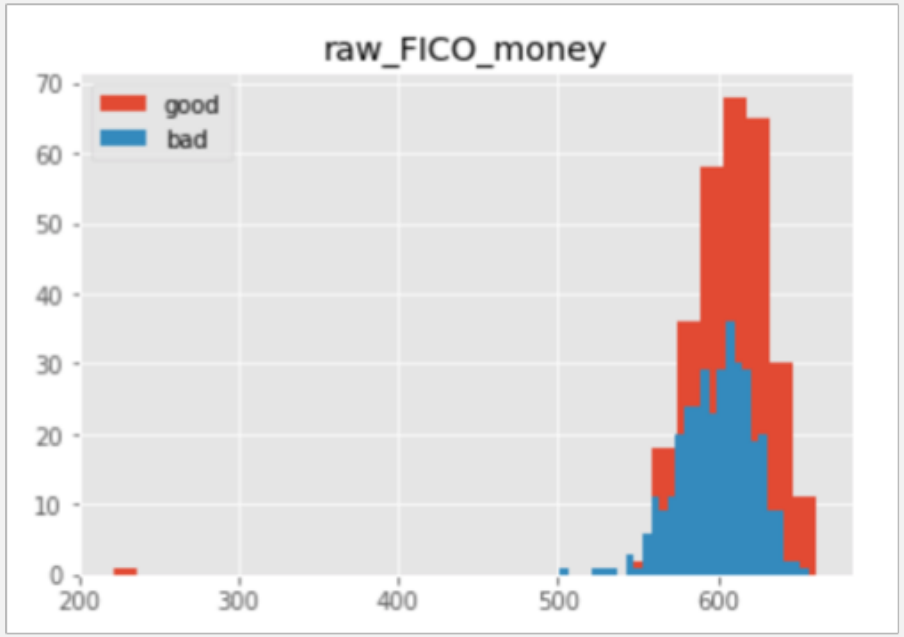
"Loss distribution"

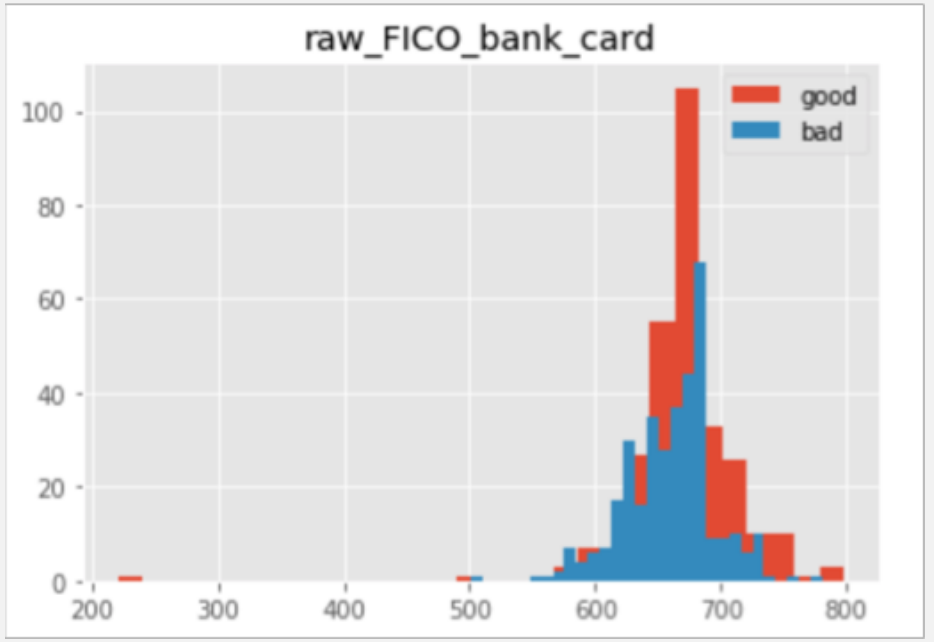





raw_FICO_retail
good
bad
0
10
20
30
40
200
300
400
500
600
700


raw_l2c_score
bad
good
0
10
20
30
40
50
60
100
200
300
400
500
600
700
800


raw_FICO_money
good
bad
0
10
20
30
40
50
60
70
200
300
400
500
600


raw_FICO_bank_card
good
bad
0
20
40
60
80
100
200
300
400
500
600
700
800

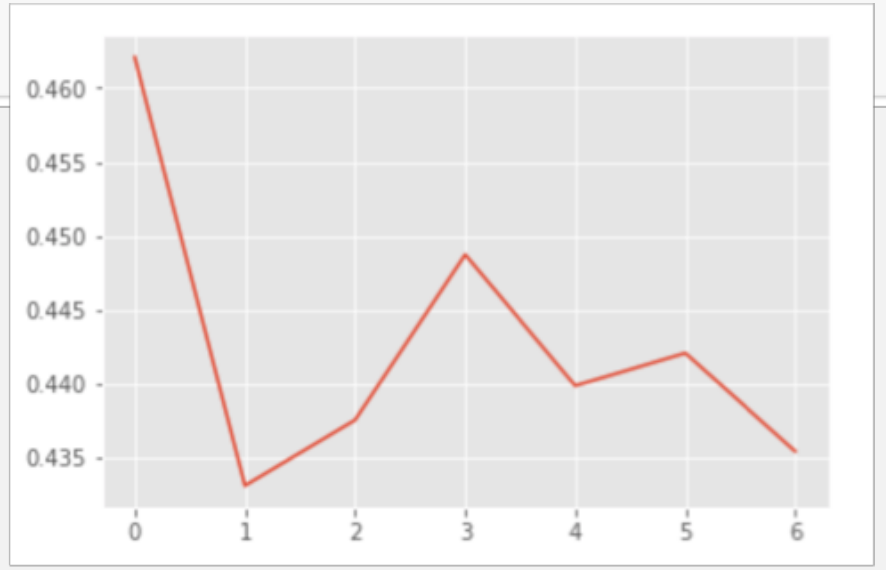
L1-REGULARIZED LOGISTIC REGRESSION

```
# k-fold: choosing best hyperparameter
log_reg_Cs = [1, 2, 3, 4, 5, 6, 7]
CVs = []

for i in log_reg_Cs:
    tmp_lr = LogisticRegression(penalty = "l1", solver = "liblinear", max_iter=1000, C = i)
    tmp_cv = cross_val_score(tmp_lr, train[predictors], train["flgGood"], cv=4, scoring="accuracy")
    CVs.append(1-np.mean(tmp_cv))

plt.plot(CVs)
plt.show()
```

Choose hyperparameter C=1



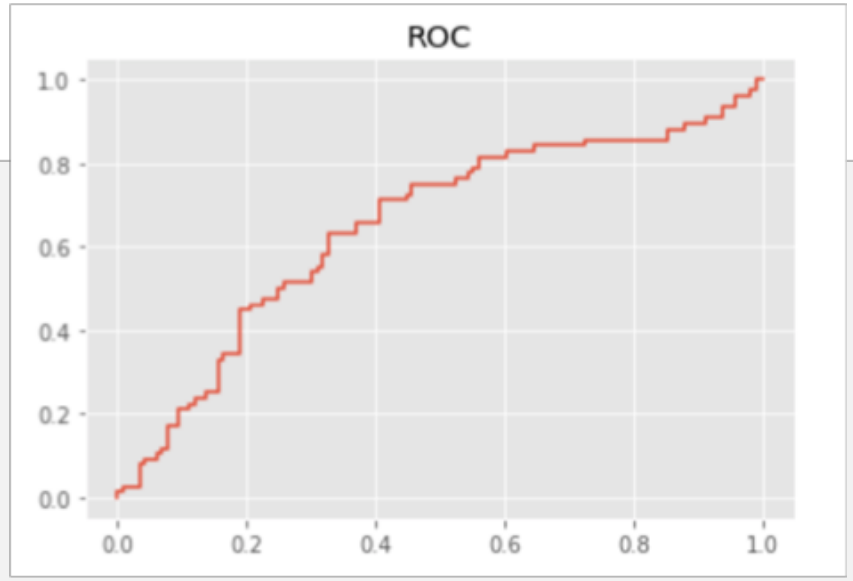
EVALUATION

Final model:

```
log_reg = LogisticRegression(penalty = "l1", solver = "liblinear", max_iter=200, C = 1)
```

```
log_reg = LogisticRegression(penalty = "l1", solver = "liblinear", max_iter=200, C = 1)
log_reg.fit(train[predictors], train["flgGood"])
print("Accuracy: ", np.mean(log_reg.predict(test[predictors]) == test["flgGood"]))
fpr, tpr, thresholds = metrics.roc_curve(test["flgGood"], log_reg.predict_proba(test[predictors])[:, 1])
print("AUC: ", metrics.auc(fpr, tpr))
print("F-1 score: ", metrics.f1_score(test["flgGood"], log_reg.predict(test[predictors])))
```

```
Accuracy:  0.640625
AUC:  0.6493874773139745
F-1 score:  0.5660377358490565
```



GRADIENT BOOSTING

```
# grid search best tree-specific parameters using k-fold
param_test1 = {'n_estimators': range(20, 81, 10), 'max_depth':range(5,16,2), 'min_samples_split':range(20,61,10),
               'min_samples_leaf':range(10,31,10), 'max_features':range(7,20,2)}
gsearch1 = GridSearchCV(estimator = GradientBoostingClassifier(),
                        param_grid = param_test1, scoring='accuracy',n_jobs=10,iid=False, cv=5)
gsearch1.fit(train[predictors], train["flgGood"])
```

```
# grid search best boosting parameters using k-fold
param_test2 = {'subsample':[0.6,0.7,0.75,0.8,0.85,0.9], 'learning_rate': [0.005, 0.05, 0.01]}
gsearch2 = GridSearchCV(estimator = GradientBoostingClassifier(n_estimators=50, max_depth=7, max_features=15,
                                                               min_samples_leaf=20, min_samples_split=60),
                        param_grid = param_test2, scoring='accuracy', n_jobs=-1, iid=False, cv=5)
gsearch2.fit(train[predictors], train["flgGood"])
```

```
{'max_depth': 7,
 'max_features': 15,
 'min_samples_leaf': 20,
 'min_samples_split': 60,
 'n_estimators': 50}
```

```
gsearch2.best_params_
```

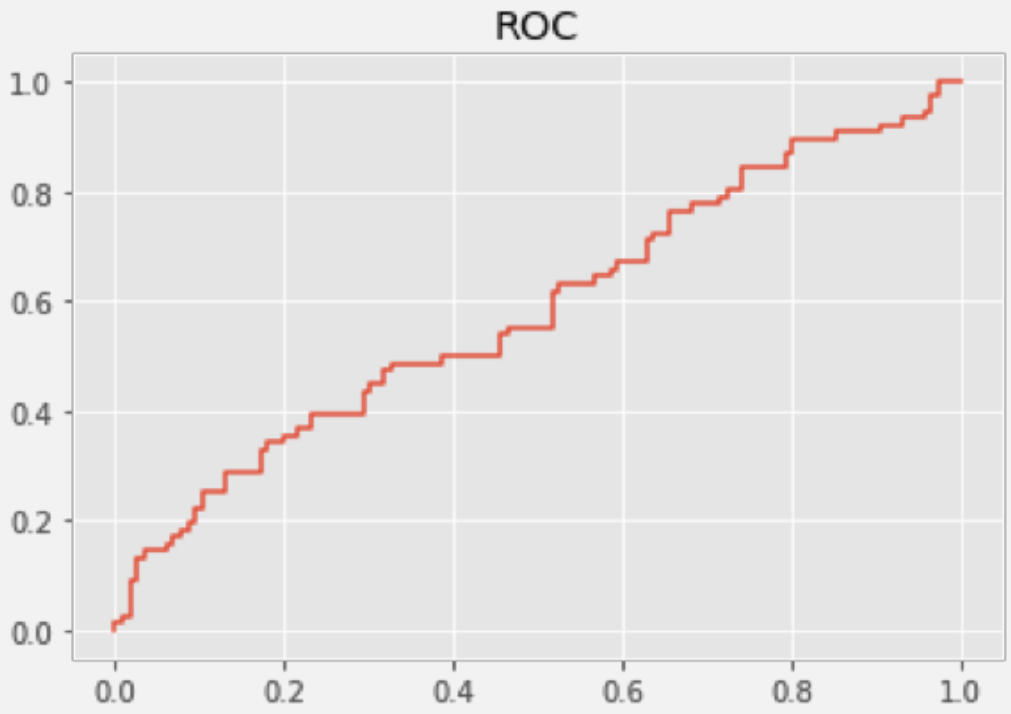
```
{'learning_rate': 0.01, 'subsample': 0.6}
```

EVALUATION

Final model:

```
gb = GradientBoostingClassifier(n_estimators=50, max_depth=7, max_features=15, min_samples_leaf=20,
                                min_samples_split=60, learning_rate=0.01, subsample=0.6)
```

```
Accuracy:  0.5989583333333334
AUC:  0.5835980036297641
F-1 score:  0.43795620437956206
```



THANK YOU!