BDAD Summer 2019 Symposium

NYU Courant, Computer Science

August 8, 2019

Big Data Applications Symposium

Project Name: Fair Lending Finder

Team:

- Cody Gilbert
- Fang Han
- Jeremy Lao

Abstract: *We want to help people increase their chances of securing a mortgage related loan. Our application will ask you for your details and provide you with the lender that is most likely to lend to you. We trained our application using publicly available anonymized mortgage application information.*

Motivation

Who are the users of this application?

- General Public
- Banking Regulators

Who will benefit from this application?

- Anyone that is looking for a mortgage loan
- Low to moderate income (LMI) borrowers
- People in states with high loan denial rates

Why is this application important?

- While there have been improvements in the mortgage lending process over the last decade, unconscious bias remains a factor in provisioning credit to average income borrowers. Our application will help borrowers use that unconscious bias in their favor.

Goodness

What steps were taken to assess the ”goodness“ of the analytic itself?

We utilized publicly available Home Mortgage Disclosure Act (HMDA) data from 2007-2017 that contains over 207 million anonymized home mortgage application records to train a machine learning model on ”approved“ or ”denied“ mortgage applications.

We use the following features to train a Naive Bayes model:

- Loan Amount
- Applicant Income
- Race
- Gender
- Lender
- State

Goodness (contd.)

$$\begin{aligned} P(y=k) &= \beta_0 loanAmt_{obs} + \beta_1 applicantIncome_{obs} \\ &\quad + \delta_0 race + \delta_1 gender + \delta_2 lender + \delta_3 state \end{aligned}$$

Where δ are dummy variables for the categorical variables and β are coefficients. The outcome (k), approve or deny, $k \in 0, 1$

MLModel	Training/Test	AUC
Logistic Regression	80/20	60%
SVM	80/20	59%
Naive Bayes	80/20	79%

Table 1: Model Evaluation

Actuation/Remediation

What actuation or remediation actions are/could be performed by this application?

- The loan applicant will use this application to determine the lender that will most likely extend credit, and the applicant can apply directly to that lender.
- A banking regulator can use this to determine the lenders that are least likely to extend credit to LMI and minority communities

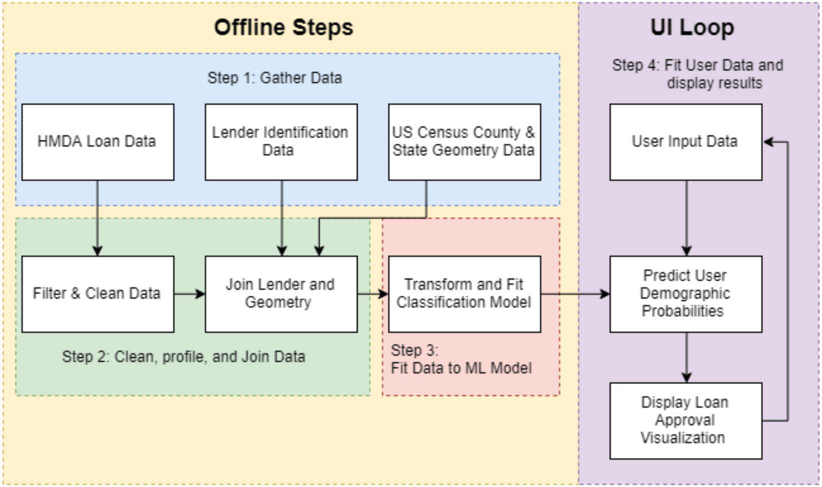
Data Sources

Name	HMDA Data Set
Description	Anonymized mortgage loan application information
Size of data	$>$ 120 GB

Name	Geospatial Data
Description	Latitude and Longitude of States and Counties
Size of data	$>$ 100 MB

Name	HMDA Panel Information
Description	Lender metadata, such as parent ID and head office
Size of data	$>$ 100 MB

Design Diagram


Offline Steps
Step 1: Gather Data
HMDA Loan Data
Lender Identification Data
US Census County & State Geometry Data
Filter & Clean Data
Join Lender and Geometry
Step 2: Clean, profile, and Join Data
Transform and Fit Classification Model
Step 3:
Fit Data to ML Model
UI Loop
Step 4: Fit User Data and display results
User Input Data
Predict User Demographic Probabilities
Display Loan Approval Visualization

Platform(s) on which the application runs:

NYU HPC Cluster (DUMBO)

Code Walkthrough

HMDA

For data profiling, we originally ingested the data into a dataframe. The entire profiling exercise would take 5-7 hours.

```
val dataForAnalysis =
    spark.read.format("csv").option("header", "true").
                 option("inferSchema",
                     "true").load(hdfsPath).
                 select("loan_amount_000s",...)
```

We changed our strategy and leveraged Spark Context RDDs to profile the data, reducing run-time to 1.5 hour:

```
val dataForAnalysis = sc.textFile(hdfsPath)
val reducedLoanAmtData =
    mapReduceFunc(dataForAnalysis, 7)
```

Code Walkthrough (contd.)

HMDA

While dataframes have the .count() function, we had to write a custom function to perform count():

```
def mapReduceFunc(dataForAnalysis : RDD[String], colNum :
    Integer) : RDD[String] = {
  val firstLine = dataForAnalysis.first()
  val data = dataForAnalysis.filter(row => row !=
      firstLine)
  val keyAmt = data.map(_.split(",")). map(c =>
      (c(colNum),1)). reduceByKey((x,y) => x+y)
  val mrAmt = keyAmt.map(x =>
      x._1.stripPrefix("\"").stripSuffix("\"") + "," +
      x._2)
  mrAmt
}
```

Code Walkthrough (contd.)

HMDA
While dataframes preserve column names, you have to manually incorporate them in the RDD before saving as a .csv file:

```
    val header: RDD[String]=
        sc.parallelize(List("loan_amount,frequency"))
     header.union(reducedLoanAmtData).saveAsTextFile(<path>)
```