Machine Learning with Home Mortgage Data

Nikeeta Akbari, Nikita Dhanraj Marathe, Rohit Tiwari, Zeeshan Khan
Department of Information Systems, California State University
Los Angeles
e-mail : nakbari2@calstatela.edu, nmarath@calstatela.edu, rtiwari2@calstatela.edu,
zkhan7@calstatela.edu

Abstract: Our dataset contains information regarding "Home mortgage." It includes home loan data of the years 2016 and 2017 for states California and Washington. Our aim behind choosing this dataset is to provide an idea of whether a person gets loan approval or not before applying to the bank or financial organization for a mortgage application. Data which we have used for this project is publicly available under "Home Mortgage Disclosure Act." Dataset gives insights about whether loan gets approved or denied, a reason behind denial such as credit history, debt to income ratio, etc. The dataset also contains data about the type of property, location of the property, applicant's detailed information, Loan amount, Loan type such as home Loan or refinancing, etc. We have used machine learning's Classification algorithm to create our models both in Microsoft Azure and Databricks.

1. Introduction

Thousands of banks and financial institutions report data about their mortgages applications to the public every year under "Home Mortgage Disclosure Act," or "HMDA." This act was enacted by Congress in 1975 and implemented in 2011. This public loan data is used to determine whether financial institutions are serving housing needs of communities and identifying possible biased lending patterns.

The primary objective of our project is to create a platform/ machine learning model which analyze and predict if the home mortgage application will be approved or denied for the future applicant. Our predictive model will predict whether a loan application will be approved or not based on some factors such as population, loan_amount, the income of applicant, agency name, loan type, and applicant's race, which will be discussed in detail in upcoming sections.

2. Work Flow

We discovered data on Kaggle.com on HMDA Home Loans and used the link provided to the data source, which is available on Consumer Financial Protection Bureau, an official website of the United State Goverment. We have downloaded HMDA data available on Data and Research page. The dataset we got is in Microsoft Excel compatible (CSV format) format.

We intend to make a predictive analysis on loan approval or denial based on factors essential for building the models in AzureML studio and Databricks. We have developed supervised learning model using classification technique. The following diagram shows step by step workflow carried out for this project.

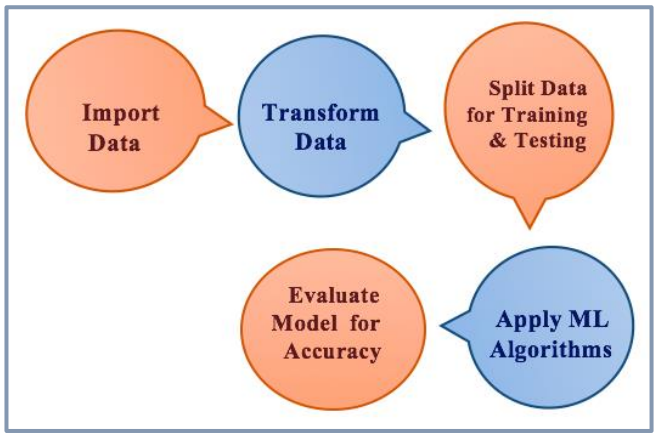

Figure 1: Workflow

3. Platform Specifications

To begin our project, we have used Microsoft Azure ML Studio and Databricks as they are popular to built machine learning model on massive datasets.

Microsoft Azure ML Studio: We have used the free version of the Machine Learning Studio for which Azure subscription is not needed. Below is the Microsoft Azure ML Studio system details: -

- Max number of modules per experiment – 100
- Max storage space – 10 GB
- Execution/performance – Single Node

Databricks: We have used the Databricks community edition, a free version of the cloud-based Spark Platform. Below is the cluster details: -

- Memory – 6GB Memory , 0.88 Cores, 1 DBU
- DataBricks Runtime Version – 4.0 (includes Apache Spark 2.3.0, Scala 2.11)
- Python Version – 2

4. Azure Machine Learning Model

The Azure Machine Learning Studio is a cloud-based development environment which allows a user to build, test and deploy predictive models in the cloud. It has set of machine-learning algorithm library which makes it easier to create a predictive model. To create these models different categories of modules are provided. These modules are set of codes that perform specific machine learning tasks based on given inputs. We have built two Classification models with different Classification machine learning algorithms for this purpose we have used Data transformation modules, Feature selection modules, and Machine Learning modules in the Azure ML Studio.

4.1 The Data Transformation Modules

We have used these category modules to convert data into a dataset suitable to be used by the machine learning algorithms. Under this category, we have used different modules which are as follows:-

- Clean Missing Data – This module was used to remove all missing values from the dataset.
- Select Column in Dataset – This module was used to select a subset of columns from original dataset.
- Split Data – This module was used to split our dataset into two groups for training and testing purpose. We have used 70% of data for training and 30% for testing our model.
- Normalize Data – This module was used to rescale our numeric data.

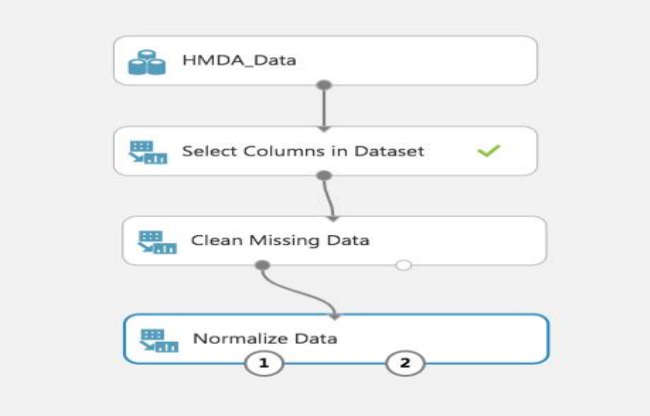

Figure 2: Data Transformation Modules

4.2 The Feature Selection Modules

We have used this category of module to obtain useful set of features from our dataset to build the model. Under this category we have used module that is as follows:-

- Permutation Feature Importance – This module was used to obtain scores of the columns, using this information we could select columns exteremly useful for our model. Below is the screenshot of the feature columns and score given to them.

rows 15 columns 2

view as

purchaser_type_name	0.349984
agency_name	0.006387
agency_abbr	0.00514
applicant_income_000s	0.002445
tract_to_msamd_income	0.001871
preapproval_name	0.001747
co_applicant_sex_name	0.001248
loan_amount_000s	0.001023
co_applicant_ethnicity_name	0.000349
loan_type_name	0.000125
population	0
lien_status_name	-0.000075
co_applicant_race_name_1	-0.000724
owner_occupancy_name	-0.001023
minority_population	-0.001672

Figure 3: Feature Score of Logistic Regression Algorithm Model

rows 15 columns 2

view as

Feature	Score
purchaser_type_name	0.360994
applicant_income_000s	0.003342
loan_amount_000s	0.003035
agency_name	0.002591
agency_abbr	0.002591
tract_to_msamd_income	0.000021
population	0
minority_population	0
preapproval_name	0
owner_occupancy_name	0
loan_type_name	0
lien_status_name	0
co_applicant_sex_name	0
co_applicant_race_name_1	0
co_applicant_ethnicity_na	0

Figure 4: Feature Score of Two-Class Decision Forest Algorithm Model

4.3 The Machine Learning Modules

We have used these category modules to build, evaluate, and tune our models. Under this category, we have used different modules which are as follows:-

- Train Model- This module was used to train our Classification models.
- Score Model – This module was used to score predictions for our models.
- Evaluate Model - This module was used to measure accuracy for our models.
- Cross-Validate - This module was used to assess the reliability of our models.
- Tune Model Hyperparameters - This module was used to determine optical hyperparameters of our models.

4.4 Classification Algorithms

We have used two Classification algorithms to build two separate predictive models. The Logistic Regression Algorithm and Two-Class Decision Forest Algorithms are used. The motive behind using two algorithms is that we wanted to assess which algorithm is best for our dataset.

4.5 Evaluation Results

The evaluation metrics we have used are Area Under the curve, precision, and recall. From the results, it was quite evident that the best classification algorithm for our dataset is Two-Class Decision Forest algorithm.

	Two-Class Decision Algorithm	*Logistic Regression Algorithm*
AUC	0.922	0.878
Precision	0.951	0.952
Recall	0.761	0.736

Table 1: Evaluation Result Overview

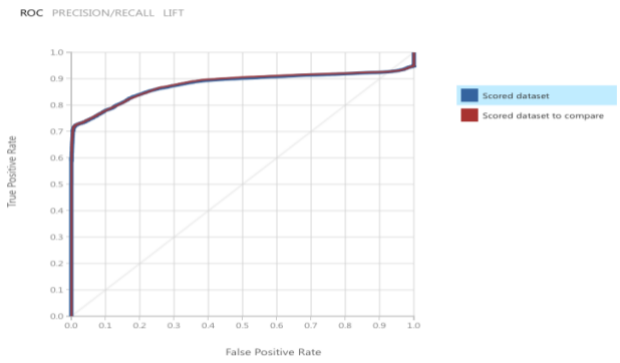

Figure 5: AUC graph of Logistic Regression Algorithm

Figure 6: Precision and Recall of Logistic Regression Algorithm

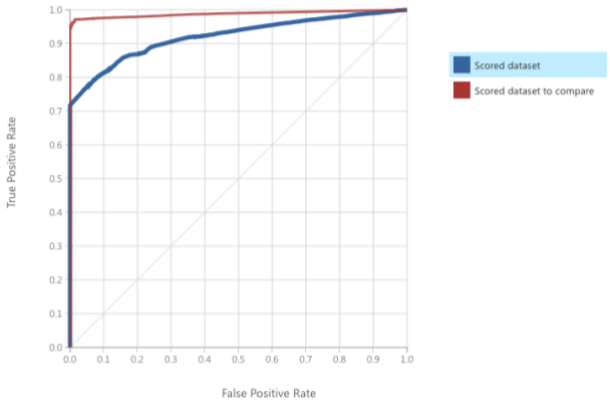

Figure 7: AUC graph of Two-Class Decision Algorithm

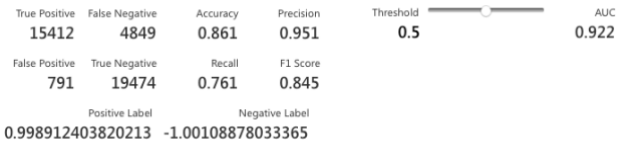

Figure 8: Precision and Recall of Two-Class Decision Algorithm

5. Databricks Machine Learning Model

We have used the free cloud-based version of Databricks containing the Spark platform. We have used IPython notebook environment to build our models. We have used two algorithms to build two separate predictive models, one of the algorithms is Logistic-Regression and the another is Decision Tree Classifier.

5.1 Steps to Create Our Models

We have followed a certain number of steps to build our model in the Databricks. The steps are as follows:-

- We first uploaded our CSV file into the database of the Databricks.
- After uploading the file, we have to read the read the CSV file using Sparl SQL command.
- Data Manipulation was performed by selecting columns suitable for our model and by droping null value rows using dropna() function from the original dataset.
- Then, we have spilt our data into training (70%) and testing (30%) by using data.randomSplit() function.
- After splitting, we have define our pipeline which created vector of features using VectorAssembler function and trained our model.
- Later, we have performed testing by inserting test data to our model.
- Finally to assess our model we have measured AUC using BinaryClassificationEvalutor() function. Also, measured the precision and recall metrics.

5.2 Evaluation Results

The evaluation metrics we have used are Area Under the curve, precision, and recall. From the results, it was quite evident that the best classification algorithm for our dataset is Decision Tree Classifier algorithm.

	Decision Tree Classifier Algorithm	*Logistic Regression Algorithm*
AUC	0.898	0.846
Precision	0.982	0.954
Recall	0.812	0.735

Table 2: Evaluation Result Overview

```
evaluator = BinaryClassificationEvaluator(labelCol="trueLabel",
rawPredictionCol="prediction", metricName="areaUnderROC")
auc = evaluator.evaluate(prediction)
print "AUC =", auc
```

```
▸ (4) Spark Jobs
AUC = 0.898579780922
```

Figure 9: AUC of Decision Tree Classifier Algorithm

```
tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())
metrics = spark.createDataFrame([("Precision", tp / (tp + fp)), ("Recall", tp / (tp
+ fn))],["metric", "value"])
metrics.show()
```

```
▸ (7) Spark Jobs
▸ metrics: pyspark.sql.dataframe.DataFrame = [metric: string, value: double]
+---------+------------------+
|   metric|             value|
+---------+------------------+
|Precision|0.9821663674446439|
|   Recall|0.8122741895570403|
+---------+------------------+
```

Figure 10: Precision and Recall of Decision Tree Classifier Algorithm

```
evaluator = []
for i in range(2):
  evaluator.insert(i, BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC"))
  auc = evaluator[i].evaluate(prediction[i])
  print "AUC ", i, " = ", auc
```

```
▸ (8) Spark Jobs
AUC  0  =  0.734014728022
AUC  1  =  0.846737928678
```

Figure 11: AUC of Logistic Regression Algorithm

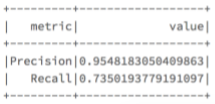

Figure 12: Precision and Recall of Logistic Regression Algorithm

6. Comparision Between Azure ML and Databricks

Below is the overview table of the measured metrics to evaluate our models both in Azure ML and Databricks.

	Two-Class Decision Algorithm	*Logistic Regression Algorithm*
AUC	0.922	0.878
Precision	0.951	0.952
Recall	0.761	0.736

Table 3: Azure ML Evaluation Result Overview

	Decision Tree Classifier Algorithm	*Logistic Regression Algorithm*
AUC	0.898	0.846
Precision	0.982	0.954
Recall	0.812	0.735

Table 4: Databricks Evaluation Result Overview

7. Summary

We successfully build classification models in both Azure ML Studio and Databricks. On comparing models of both the platforms, the evaluation results were quite similar for Logistic Regression Algorithm. However, the evaluation results were quite different for Decision Tree Classifier and Two-Class Decision Algorithm.

8. GitHub Link

https://github.com/Zeebag/CIS-5560-Big-Data-Project.

9. References

[1] https://www.consumerfinance.gov/data-research/hmda/explore
[2] https://studio.azureml.net/
[3] https://community.cloud.databricks.com/login.html
[4] https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/machine-learning-studio-algorithm-and-module-help