HOME MORTGAGE PREDICTION MODEL

BY GROUP G:-
Nikeeta Akbari
Nikita Marathe
Rohit Tiwari
Zeeshan Khan

CONTENT

- Introduction
- Objective
- Dataset Information
- Hardware Specifications
- Machine Learning Model in Azure studio
  - Data Transformation – Manipulation
  - Feature Selection
- Machine Learning Model in DataBricks
- Summary Table
- References

INTRODUCTION

What is HMDA data?

- Each year thousands of banks and other financial institutions report data about mortgages to the public, thanks to the Home Mortgage Disclosure Act, or "HMDA" for short.

What you can find in HMDA data?

- Dataset includes information about the loan - the loan amount, the type of loan, loan purpose name (buying a home, refinancing an existing mortgage, or for home improvements).
- Secondly, dataset also includes demographic information on applicants' race, ethnicity, and sex.
- Thirdly, dataset also has information about the property, the location of property and data about the lender.
- Finally, dataset has information about action taken on the application – Approved or denied.

OBJECTIVE

- Our goal is to build a machine learning predictive model to predict if the home mortgage application will be approved or denied.
- This model will help to show if an applicant is fit to apply for home mortgage.
- Also, it shows whether lenders are serving the housing needs of their communities.
- Shed light on lending patterns that could be discriminatory.

DATASET INFORMATION

- Name - Home Mortgage Disclosure Act (HMDA)
- Dataset Source - https://www.consumerfinance.gov/data-research/hmda/explore
- Size – 2.21 GB
- Years – 2017 and 2016
- States – California and Washington
- Dataset Format - CSV

HARDWARE SPECIFICATIONS

- **Azure ML Studio Hardware Specifications**
  - Max number of modules per experiment – 100
  - Max storage space – 10 GB
  - Execution/performance – Single Node
- **DataBricks Cluster Hardware Specifications**
  - 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

MACHINE LEARNING MODEL IN AZURE ML STUDIO

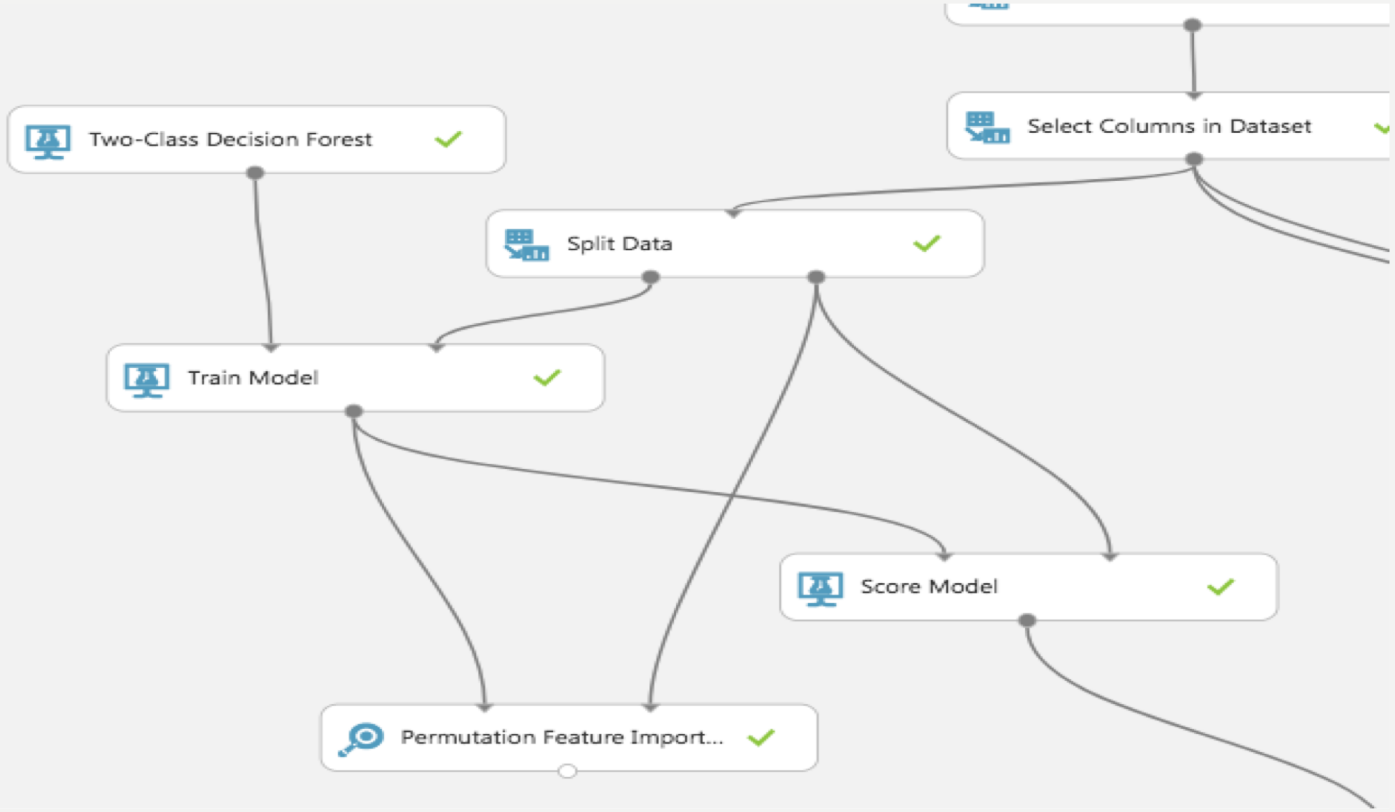
- Size of Dataset – 46.1 MB
- Algorithms used – Two-Class Logistic Regression and Two-Class Decision Forest
- Label Column – Action Taken Name (contains decision like approved or denied)
- Split – 70% train and 30% test

Data manipulation tasks Performed

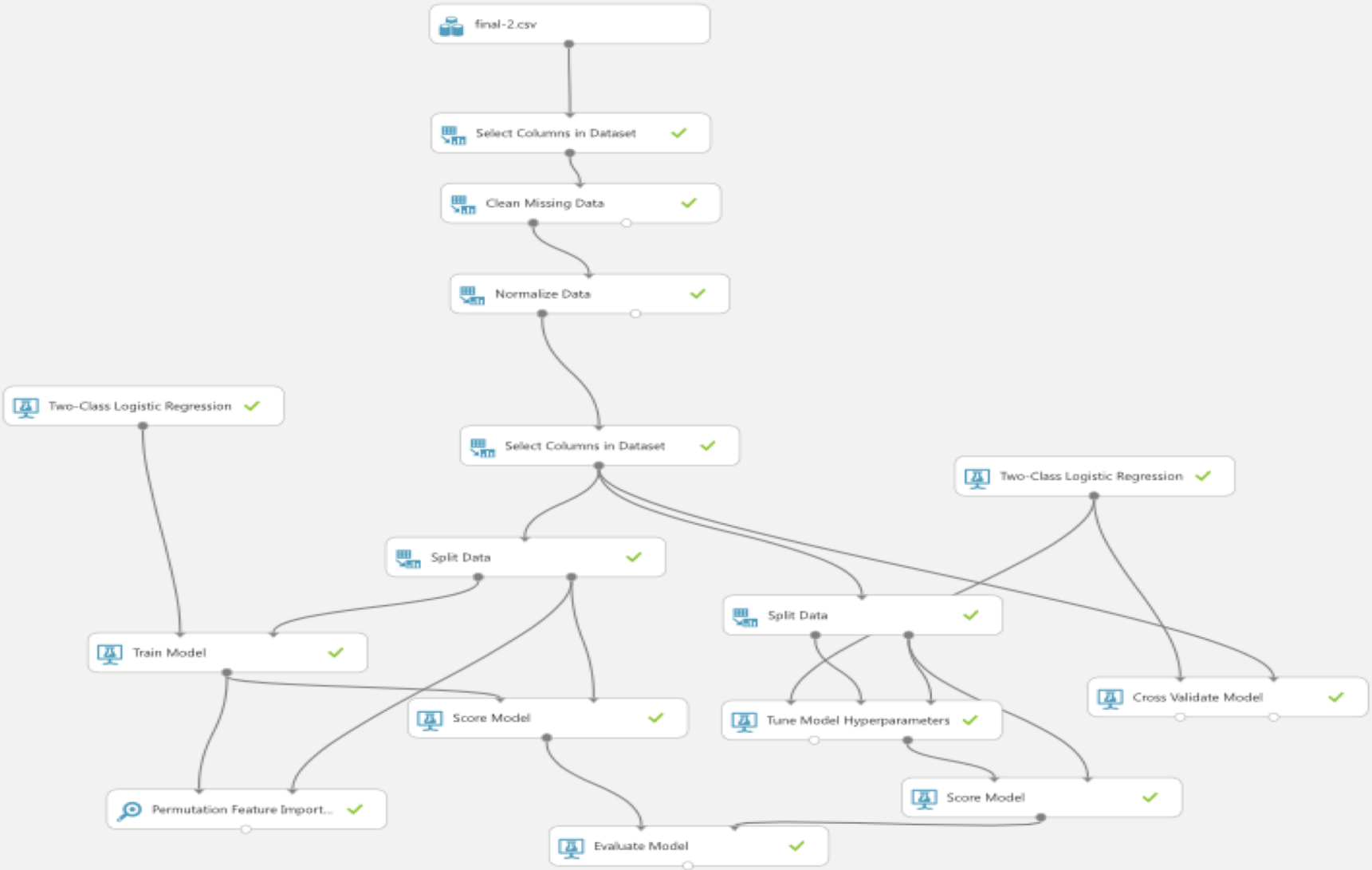
- Clean Missing Data Module - Specifies how to handle values that are missing from a dataset.
- Select Columns in Dataset Module - Selects columns to include in a dataset or exclude from a dataset in an operation.

Feature Selection tasks Performed

- *The permutation feature importance - Computes the permutation feature importance scores of feature variables given a trained model and a test dataset.*

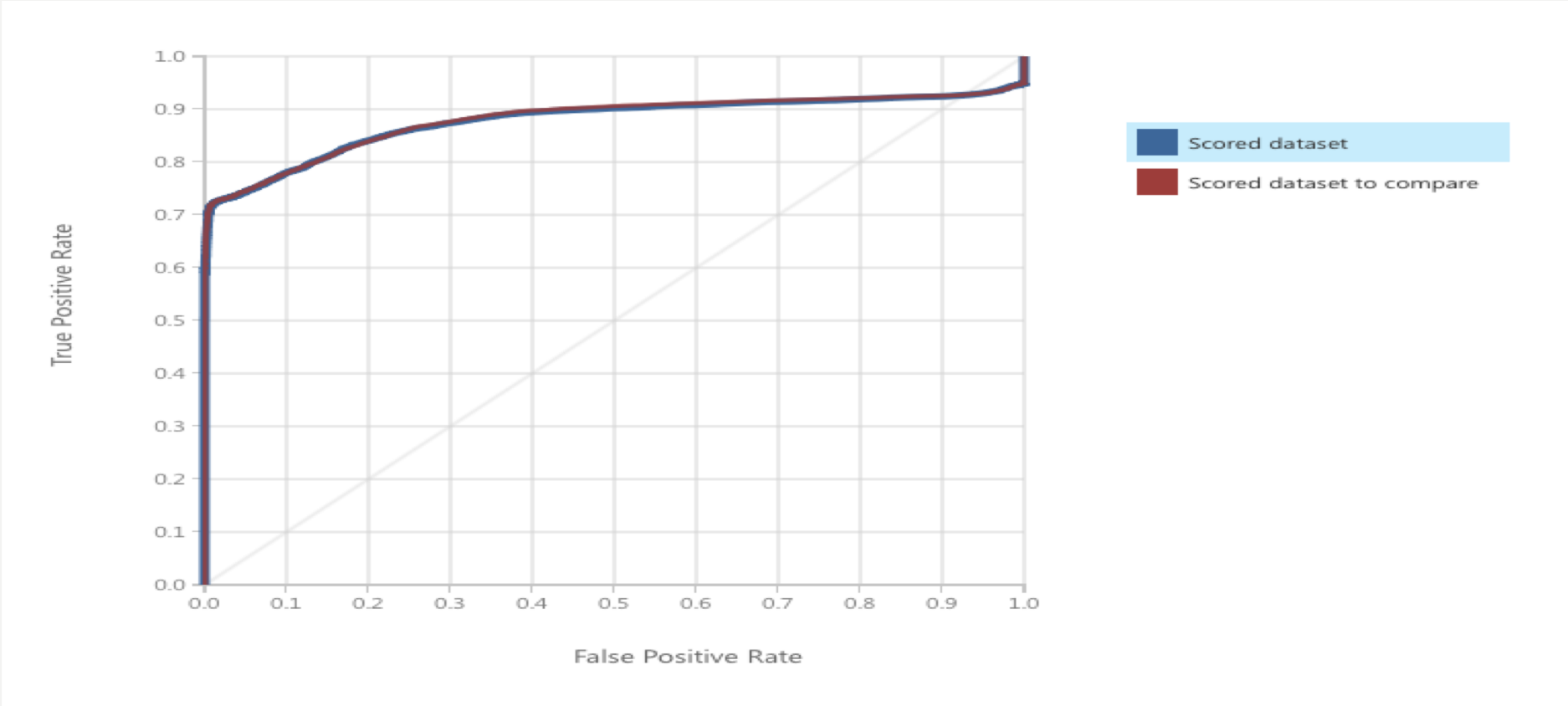


Two-Class Logistic Regression Model

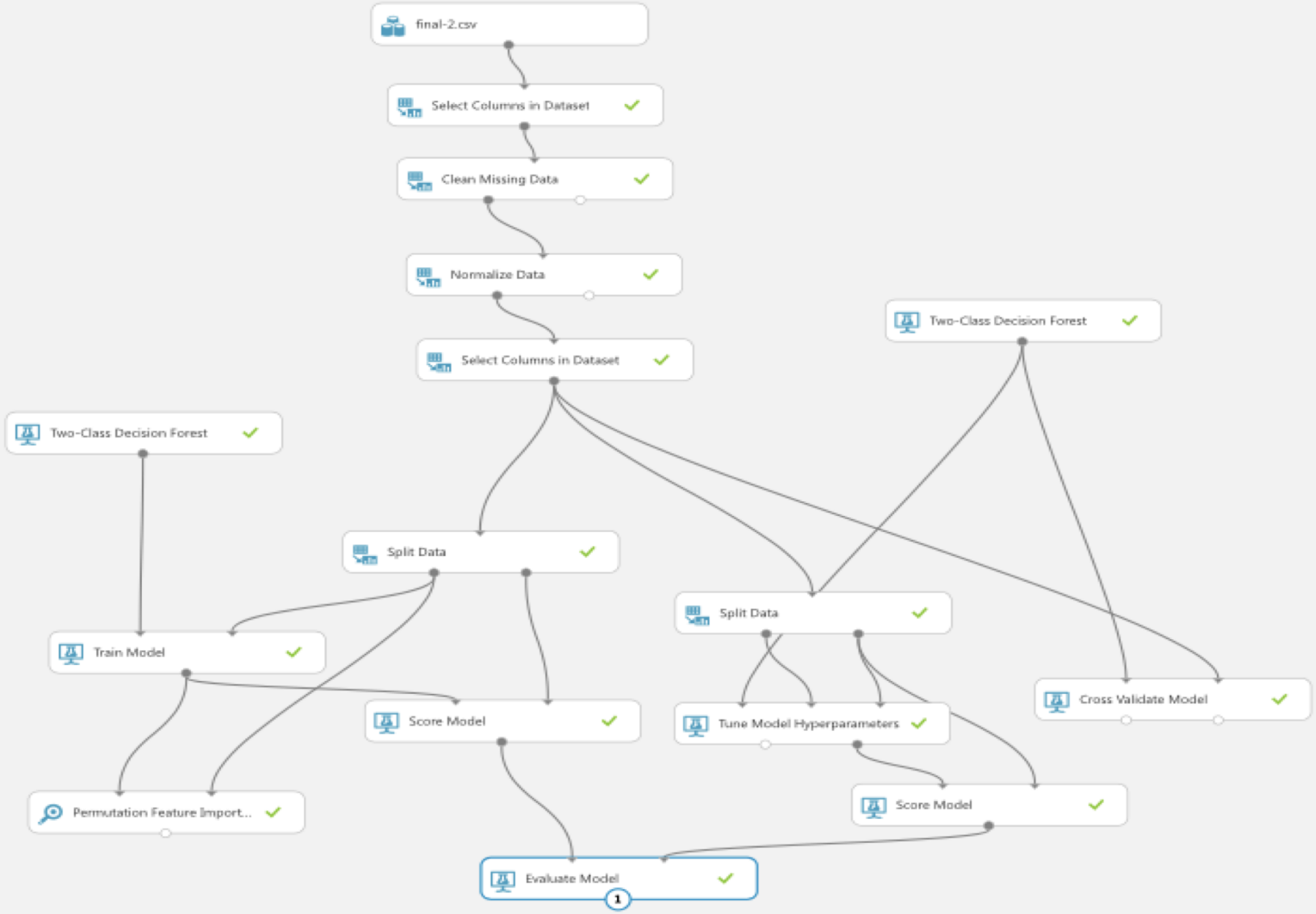


MODEL PERFORMANCE

- To assess the performance of the models we measured the area under a ROC curve which is 0.878

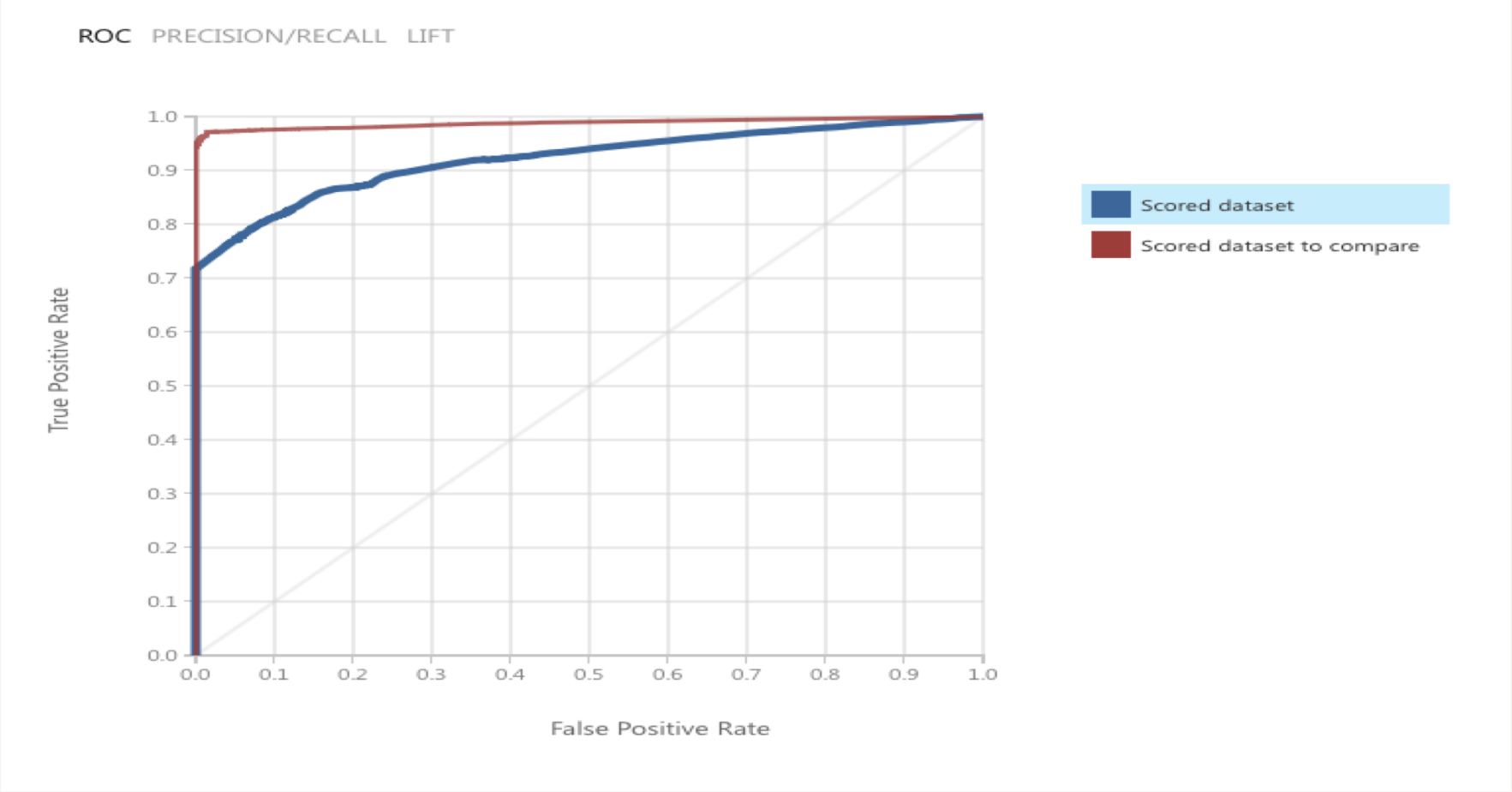


TWO-CLASS DECISION FOREST MODEL



MODEL PERFORMANCE

- To assess the performance of the models we measured the area under a ROC curve which is 0.922



MACHINE LEARNING MODEL IN DATABRICKS

- Algorithms used – Logistic Regression
- Label Column – Action Taken Name (contains decision like approved or denied).
- Split – 70% train and 30% test

```
# Split the data
splits = data2.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
```

EVALUATION

Tune Parameters

- We used TrainValidationSplit with Threshold parameters and elastic-net parameters to evaluate each combination of parameters defined in a ParameterGrid against a subset of the training data in order to find the best performing parameters.

```
paramGrid2 = (ParamGridBuilder()
.addGrid(lr[0].regParam, [0.01, 0.5, 2.0])
.addGrid(lr[0].elasticNetParam, [0.0, 0.5, 1])
.addGrid(lr[0].maxIter, [1, 5]).build())
```

```
paramGrid = (ParamGridBuilder() \
             .addGrid(lr[0].regParam, [0.01, 0.5, 2.0]) \
             .addGrid(lr[0].threshold, [0.30, 0.35]) \
             .addGrid(lr[0].maxIter, [1, 5]) \
             .build())
```

MODEL PERFORMANCE

- To assess the performance of the models with threshold parameters TrainValidationSplit and elastic-net parameters TrainValidationSplit, we measured the area under a ROC curve for both these models.

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

```
▸ (8) Spark Jobs
AUC  0  =  0.734014728022
AUC  1  =  0.846737020678
```

SUMMARY TABLE

Azure ML Studio VS Spark in Databricks

Azure ML Studio : Two-Class Logistic Regression	Spark : Two-Class Logistic Regression
AUC – 0.87	AUC – 0.85

Azure ML Studio Two-Class Logistic Regression VS Azure ML Studio Two-Class Decision Forest

Azure ML Studio : Two-Class Logistic Regression	Azure ML Studio : Two-Class Decision Forest
AUC – 0.87	AUC – 0.92

REFERENCES

- Dataset Source - https://www.consumerfinance.gov/data-research/hmda/explore
- Azure ML Studio - https://studio.azureml.net/
- Azure Microsoft Docs - https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/machine-learning-studio-algorithm-and-module-help
- DataBricks - https://community.cloud.databricks.com/login.html

THANK YOU!

ANY QUESTIONS?