



Building Competing Models using Spark DataFrames

Abdulla Al Qawasmeh
Engineering Manager, Machine Learning

Credit Karma

Outline

- Recommendation Problem
- Case Study: John
- Choice of Evaluation Metrics: A Tale of Two Models
- DataFrames & Type Safety
- Gotchas



Recommendation Problem

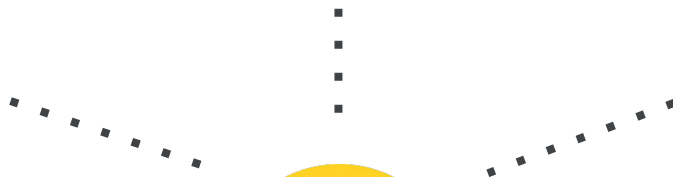
Recommendation problem

Credit Karma helps 60 million members make financial progress

- Revenue stream: Approved financial products
- Verticals: Credit cards, personal loans, auto, mortgages, and tax
- Goal: Recommend products that
 - The member is interested in
 - The member has a high probability of being approved for
 - Have a long term benefit for the member (e.g., pay credit card debt using a cheaper personal loan)

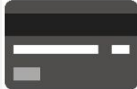




Case Study: John






Searching for the right options for John



Product	Features	P(Interest)	P(Approval)
	2% Rewards \$300 Bonus	Very High	Average
	1% Rewards 0% APR for 6m	High	Average
	0% Rewards \$20 Annual Fee	Very Low	Very High

Long term benefit



Product	Features	P(Interest)	P(Approval)	E[Benefit]
	2% Rewards \$300 Bonus	Very High	Average	Average
	1% Rewards 0% APR for 6m	High		Very High
	0% Rewards \$20 Annual Fee	Very Low	Very High	Very Low

We achieved success!



Did we?



+





=



What about other financial products?



Product	Features	P(Interest)	P(Approval)	E[Benefit]
	Unsecured 7.5% APR	High	High	Very High
	1% Rewards 0% APR for 6m (25% APR after)	High	Average	High



Choice of Evaluation Metrics: A Tale of Two Models

A Tale of Two Models

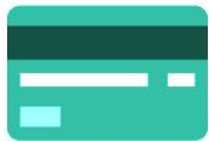


Label	Predicted
0	0.05
0	0.9
1	0.95



Label	Predicted
1	0.95
1	0.55
0	0.05

AUC as a Metric



Label	Predicted
0	0.05
0	0.9
1	0.95

AUC = 1.0



Label	Predicted
1	0.95
1	0.55
0	0.05

What's Wrong?



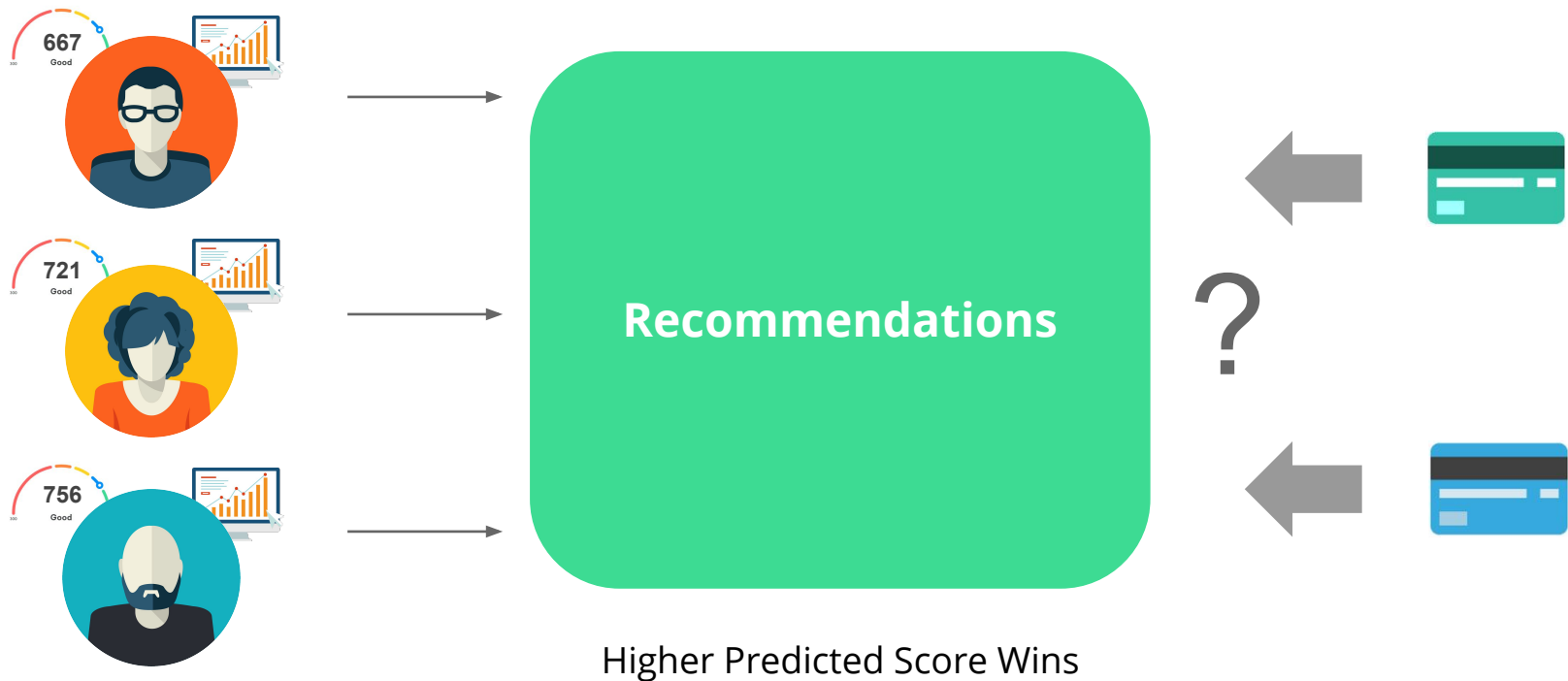
Label	Predicted
0	0.05
0	0.9
1	0.95

AUC = 1.0



Label	Predicted
1	0.95
1	0.55
0	0.05

Impression Distribution



Which metric?



Label	Predicted
0	0.05
0	0.9
1	0.95

AUC = 1.0



Label	Predicted
1	0.95
1	0.55
0	0.05

LogLoss Instead of AUC



Label	Predicted
0	0.05
0	0.9
1	0.95

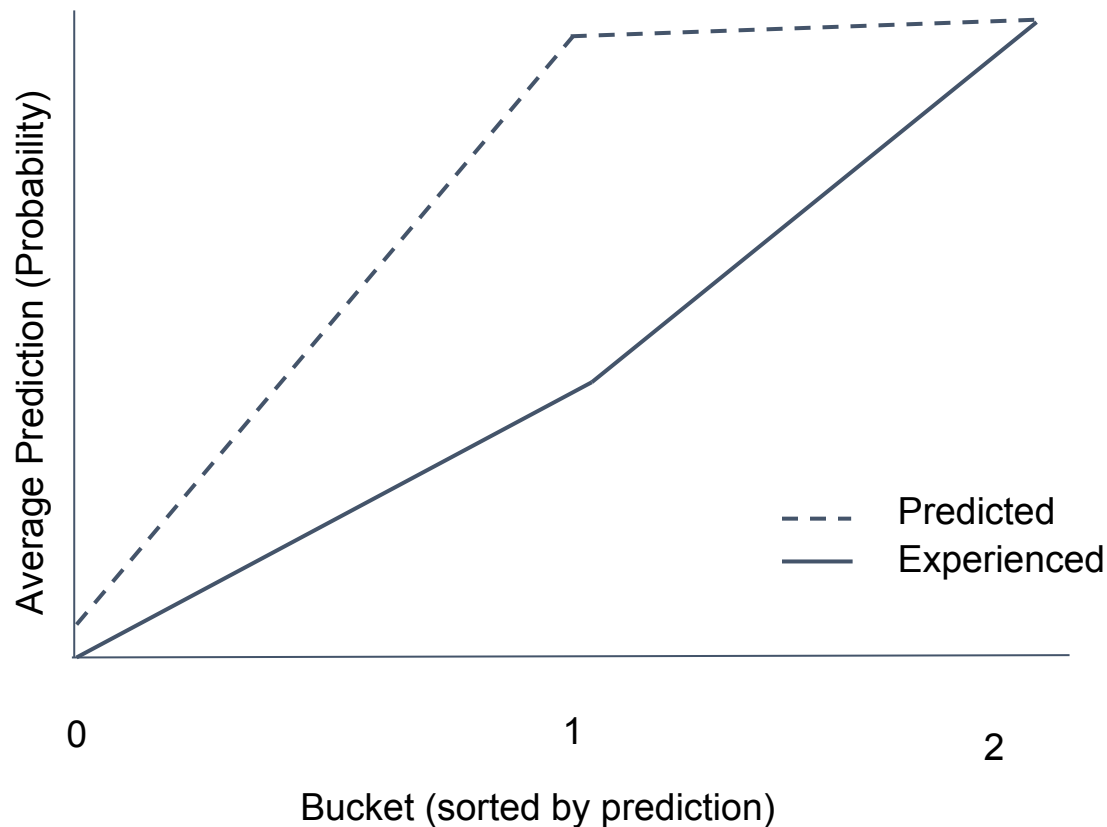
LogLoss = 0.8



Label	Predicted
1	0.95
1	0.55
0	0.05

LogLoss = 0.2

Calibration



Average Label	Predicted
0	0.05
0.45	0.9
0.95	0.95

AUC = 1.0

LogLoss = 0.80



DataFrames & Type Safety

DataFrames

DataFrames are flexible data types

- Any number of columns
- Any types
- No type checking done at compile time

Some checking done at runtime

- E.g., using a schema

DataFrames - Type Safety Problem

```
val featuresLabelDF: DataFrame

val featuresOnlyDF: DataFrame

def doTraining(featuresLabelDF: DataFrame) = {
  ...
}

...

doTraining(featuresOnlyDF)
```

Type Safety Wrappers

- Row Wrapper
- DataFrame Wrapper

Type Safety Wrappers

Row Wrapper

```
final case class FeaturesLabelDataRow(facts: Facts, label: Double)
extends DataRow {
  def toRow: Row
}
object FeaturesLabelDataRow {
  def parseRow(row: Row): FeaturesLabelDataRow
}
```


Type Safety Wrappers

Row Wrapper

```
final case class FeaturesLabelDataRow(facts: Facts, label: Double)
  extends DataRow {
  def toRow: Row
}
object FeaturesLabelDataRow {
  def parseRow(row: Row): FeaturesLabelDataRow
}
```

DataFrame Wrapper

```
final case class FeaturesLabelData(dataFrame: DataFrame)
object FeaturesLabelData {
  def apply(rowDataRDD: RDD[FeaturesLabelDataRow]): FeaturesLabelsData = {
    val schema : StructType = FeaturesLabelsDataRow.getSchema
    FeaturesLabelsData(sqlContext.createDataFrame(rowDataRDD, schema))
  }
}
```

DataFrames - Type Safety Solution

```
val featuresLabelData: FeaturesLabelData

val featuresOnlyData: FeaturesData

def doTraining(featuresLabelData: FeaturesLabelData) = {
  ...
}

...

doTraining(featuresOnlyData) // Compile time error!
```



Gotchas

DataFrames Left Join

ImpressionID	ClickID	...
1	None	
2	1	
3	None	
...		

ClickID	...
1	
...	

```
val impDF: DataFrame
val clickDF: DataFrame

val joined = impDF.join(clickDF, impDF("ClickID") === clickDF("ClickID"), "left_outer")
```

DataFrames Left Join - Solution

ImpressionID	ClickID	...
1	None	
2	1	
3	None	
...		

ClickID	...
1	
...	

```
val impDF: DataFrame
val clickDF: DataFrame

val impClicks = impDF.where(impDF("ClickID").isNotNull())
val impOnly = impDF.where(impDF("ClickID").isNull())

val joined = impClicks.join(clickDF, impClicks("ClickID") === clickDF("ClickID")).unionAll(impOnly)
```

StackOverflow

```
new GBTRegressor().fit(df)
```

StackOverflow

```
new GBTRegressor().fit(df)
```

```
Exception in thread "main" java.lang.StackOverflowError at  
org.apache.spark.sql.catalyst...
```

StackOverflow

```
new GBTRegressor().fit(df)
```

```
Exception in thread "main" java.lang.StackOverflowError at  
org.apache.spark.sql.catalyst...
```

```
new GBTRegressor().setCheckpointInterval(metaData.checkpointInterval)
```


StackOverflow

```
new GBTRegressor().fit(df)
```

```
Exception in thread "main" java.lang.StackOverflowError at  
org.apache.spark.sql.catalyst...
```

```
new GBTRegressor().setCheckpointInterval(metaData.checkpointInterval)
```

```
-Dspark.executor.extraJavaOptions='-XX:ThreadStackSize=81920'
```

Random Split Janino Error

```
val df: DataFrame // 500 columns, 1000 rows
```

```
val Array(left, right) = df.randomSplit(Array(.8,.2))
```

```
// This crashes  
left.count
```

```
Caused by: org.codehaus.janino.JaninoRuntimeException: Code of method  
"(Lorg/apache/spark/sql/catalyst/InternalRow;Lorg/apache/spark/sql/catalyst/  
InternalRow;)I" of class "org.apache.spark.sql.catalyst.ex  
pressions.GeneratedClass$SpecificOrdering" grows beyond 64 KB
```

Random Split Janino Error

```
val df: DataFrame // 500 columns, 1000 rows
```

```
val Array(left, right) = df.randomSplit(Array(.8,.2))
```

```
// This crashes  
left.count
```

```
Caused by: org.codehaus.janino.JaninoRuntimeException: Code of method  
"(Lorg/apache/spark/sql/catalyst/InternalRow;Lorg/apache/spark/sql/catalyst/  
InternalRow;)I" of class "org.apache.spark.sql.catalyst.ex  
pressions.GeneratedClass$SpecificOrdering" grows beyond 64 KB
```

Upgrade to: 1.6.4, 2.0.3, 2.1.1, or 2.2.0

<https://issues.apache.org/jira/browse/SPARK-16845>

Takeaways

- Choice of evaluation metric is very important. Especially for competing models
- There are ways to introduce type safety to dynamic types



Thank You. Questions?

<https://engineering.creditkarma.com>

abdulla.alqawasmeh@creditkarma.com

<https://www.linkedin.com/in/aalqawasmeh>