Australian Credit Card Approval
Math 5671
Daniel O’Connell and Linfei Xie
Introduction and Data Observations
Introduction

• This project is to create a model to classify credit card applications as either 0 or 1
• All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data
Dataset

• 14 predictor variables, 6 numerical and 8 categorical. The predictors are labeled A1 through A14, A15 is the classification
• 482 observations, 265 with A15=0 and 218 with A15=1
• Not a very large number of observations, so some variables with many categories will be unreliable for prediction
A4 and A5
A7

```
[ ] training_0['A7'].describe()

count    265.000000
       mean    1.316415
        std    2.202311
         min    0.000000
        25%    0.125000
        50%    0.500000
        75%    1.500000
       max    13.500000
Name: A7, dtype: float64
```

```
[ ] training_1['A7'].describe()

count    218.000000
       mean    3.627087
        std    4.419074
         min    0.000000
        25%    0.970000
        50%    2.000000
        75%    5.000000
       max   28.500000
Name: A7, dtype: float64
```
A8 and A9

A8 Grouped by Label

A9 Grouped by Label
A10

```
[ ] training_0['A10'].describe()

count       265.000000
mean         0.615694
std          2.034494
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          20.000000
Name: A10, dtype: float64
```

```
[ ] training_1['A10'].describe()

count       218.000000
mean         5.058459
std          6.888996
min          0.000000
25%          0.000000
50%          3.000000
75%          8.000000
max          67.000000
Name: A10, dtype: float64
```
Correlation Matrix

<table>
<thead>
<tr>
<th></th>
<th>A8/A15</th>
<th>A9/A10</th>
<th>A9/A15</th>
<th>A10/A15</th>
</tr>
</thead>
<tbody>
<tr>
<td>A8/A15</td>
<td>0.7148</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9/A10</td>
<td></td>
<td>0.5617</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9/A15</td>
<td></td>
<td>0.4927</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A10/A15</td>
<td></td>
<td></td>
<td>0.4137</td>
<td></td>
</tr>
</tbody>
</table>
Why Use Trees

• This dataset has many categorical variables, which are handled well by tree based models.
• Trees do not assume anything about normality or linearity, so they generally require less preprocessing than a regression model.
• We trained a regular decision tree, a random forest, and a gradient boosted tree.
• We compared them based on the accuracy and area under ROC curve.
Decision Trees

1. **Root Node:** It represents entire sample and this further gets divided into two or more sets.

2. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.

3. **Terminal Node:** Nodes do not split is called Leaf or Terminal node.
Benefits of Trees

Trees have the following advantages over regression models:

• More closely represent the process of human decision making.
• Can be visualized, and therefore can be interpreted and understood easily.
• Can handle categorical and ordinal variables without needing to create dummy variables.
Aggregate Tree Models

- A single decision tree is usually not the best predictor, it cannot recreate the complexity of the relationships between input variables and output.
- There are various methods to aggregate multiple trees in order to improve accuracy.
- These methods differ in how they make the new trees.
Types of Aggregate Trees Used

• Random Forest
  • Make multiple trees based on a subset of observations and a subset of predictor variables. The average (or mode) of the output from all trees is used as the prediction.

• Gradient Boosted Tree
  • Make a tree, calculate the residuals for the predictions on this tree, and make a new tree to predict the residuals of the first tree. Iteratively make trees to predict the residuals from the previous trees. The final prediction is the prediction of the first tree, plus all the predicted residuals.
Methods

• Calculations are done in Spark using the Pyspark, Machine Learning, classification library.

```
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import GBTClassifier
```

• The pictures were generated in Google colab, but the final calculations were done using the Google Cloud Platform
from pyspark.ml.classification import DecisionTreeClassifier

dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDepth = 3)
dtModel = dt.fit(train)
predict_dt = dtModel.transform(test)

from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
accuracy = MulticlassClassificationEvaluator()

print("Test Area Under ROC: " + str(evaluator.evaluate(predict_dt, {evaluator.metricName: "areaUnderROC"})))

Test Area Under ROC: 0.8989547038327526

print("Accuracy is " + str(accuracy.evaluate(predict_dt, {accuracy.metricName: 'accuracy'})))

Accuracy is 0.8355263157894737
Random Forest

```python
from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'label')
rfModel = rf.fit(train)
predict_rf = rfModel.transform(test)

print("Train Area Under ROC: " + str(evaluator.evaluate(predict_rf, {evaluator.metricName: "areaUnderROC"})))

Train Area Under ROC: 0.9395470383275263

print("Accuracy is " + str(accuracy.evaluate(predict_rf, {accuracy.metricName: 'accuracy'})))

Accuracy is 0.881578947368421
```
Gradient-Boosted Tree Classifier

```py
from pyspark.ml.classification import GBTClassifier

gbt = GBTClassifier(maxIter=10)
gbtModel = gbt.fit(train)
predict_gbt = gbtModel.transform(test)

print("Train Area Under ROC: " + str(evaluator.evaluate(predict_gbt, {evaluator.metricName: "areaUnderROC"})))
```

```
Train Area Under ROC: 0.9094076655052273
```

```py
print("Accuracy is " + str(accuracy.evaluate(predict_gbt, {accuracy.metricName: 'accuracy'])))
```

```
Accuracy is 0.8552631578947368
```
Model Evaluation

• As expected, the single tree model was the worst performing
• The two aggregate models were very close
• Random Forest classifier has better accuracy and area under ROC

Final choice: random forest

```python
In [0]: final_predict = rfModel.transform(test_final)
output = final_predict.select('ID', 'prediction')
```
What is better: gradient-boosted trees, or a random forest?

Random Forest

• Free to tuning
• biased in favor of those attributes with more levels

Gradient boosted tree

• More hyper parameters, hard to tune
• Better performance
• Sensitive to noisy