PREDICT CUSTOMER SUBSCRIPTIONS BASED ON BANK MARKETING

Team D04

Tengyue Chen, Guofei Du, Haoxing Zhang

Team Picture

Tengyue Chen
Major: BIA
City: Beijing

Haoxing Zhang
Major: BIA
Home City: Guangzhou

Guofei Du
Major: Data Science
Home city: Tianjin

Project Introduction

- **Background:** Marketing campaigns are planned, strategic initiatives to advance a particular business objective, such as increasing consumer awareness of a new product or gathering customer feedback. They often use a variety of media, including but not limited to email, print advertising, television or radio advertising, pay-per-click, and social media, to reach consumers in a variety of ways.

- **Goal:** attempts to build a model based on the dataset from bank marketing using binary classification to predict whether the client subscribed to a term deposit.

Project Introduction

Platform:

Data Understanding

- UCI Bank Marketing Data Set (http://archive.ics.uci.edu/ml/datasets/Bank+Marketing#)

- The dataset includes 20 variables, including both categorical and numeric variables. And The dataset includes 45,211 observations, or individual customer records.

Column	**Feature**
age	age
job	type of job
marital	marital status
education	education
default	has credit in default?
balance	
housing	has housing loan?
loan	has personal loan?
contact	contact communication type
day	last contact day of the week
duration	last contact duration, in seconds
campaign	number of contacts performed during this campaign and for this client
pdays	number of days that passed by after the client was last contacted from a previous campaign
previous	number of contacts performed before this campaign and for this client
poutcome	outcome of the previous marketing campaign
deposit	has the client subscribed a term deposit?

EDA

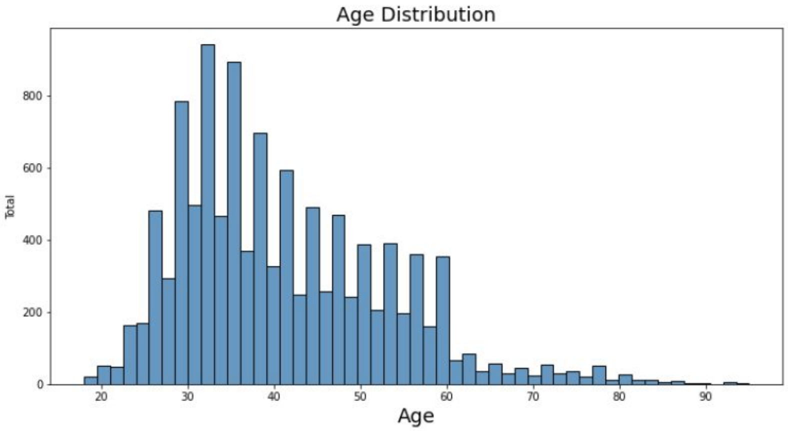


Figure 1. Age Distribution

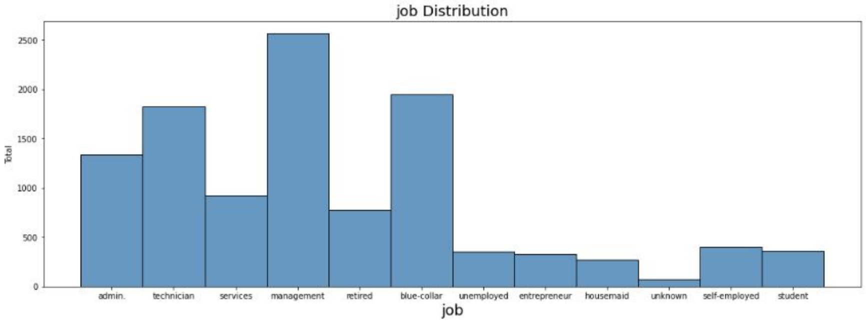


Figure 2. Job Distribution

EDA

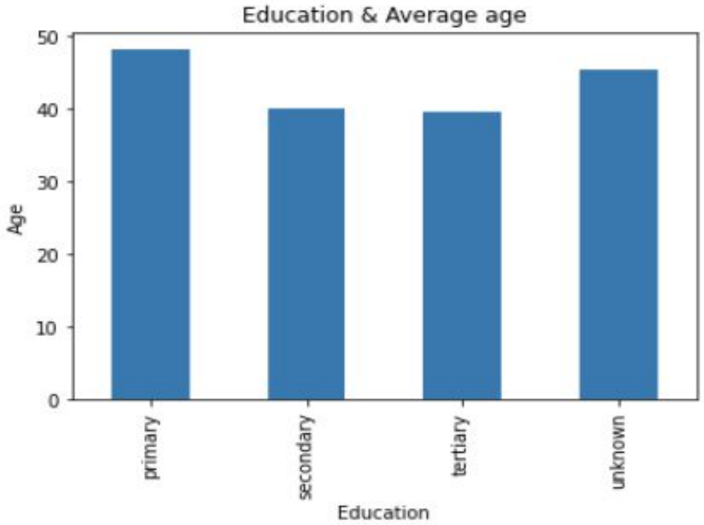


Figure 3. Education & average age

EDA

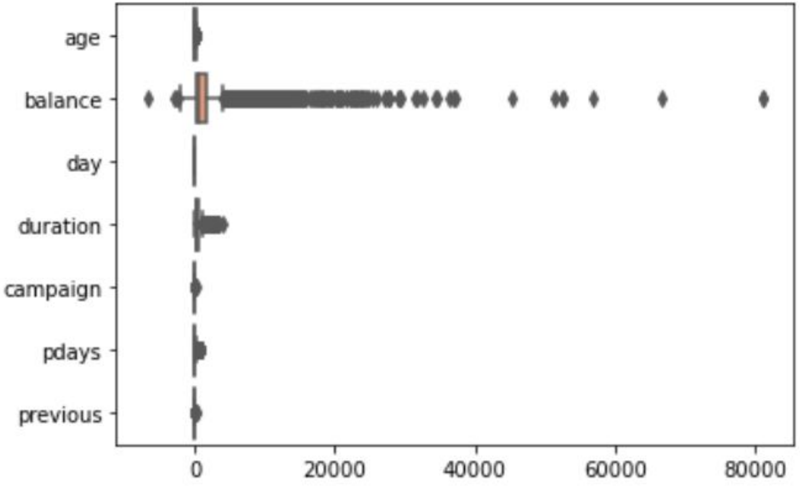


Figure 4. boxplot of dataset

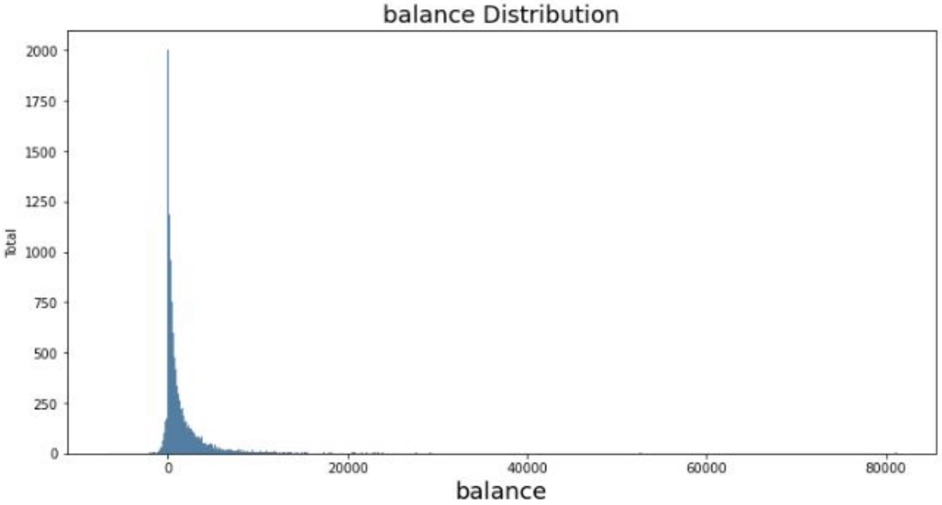


Figure 5. balance distribution

EDA

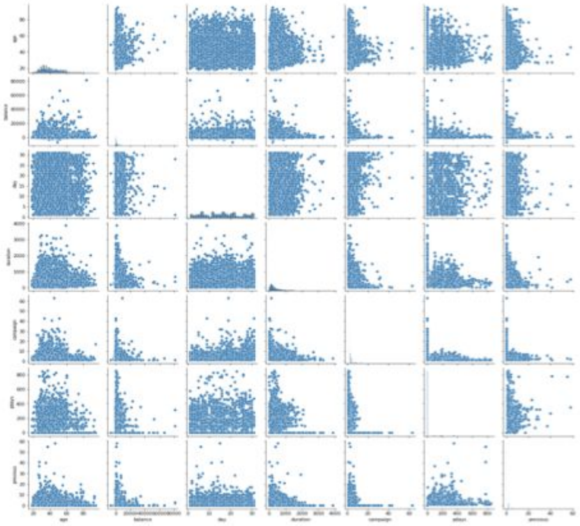


Figure 6. Bi-Variate Analysis Result

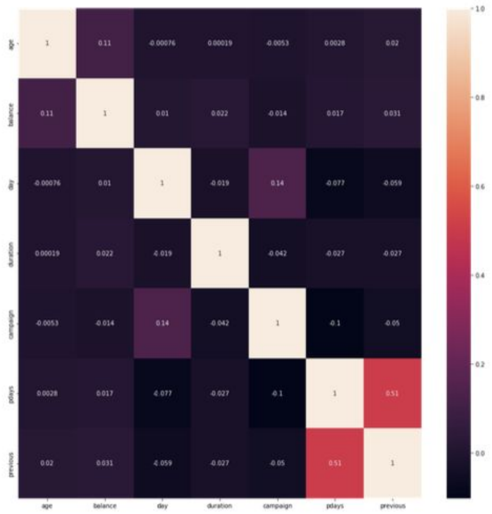


Figure 7. Correlation

Methodology

- Use the PySpark on AWS

```
[4]: from pyspark.ml.feature import OneHotEncoderEstimator
     from pyspark.ml.feature import StringIndexer
     from pyspark.ml.feature import VectorAssembler
```

Last executed at 2022-12-08 03:12:13 in 284ms

```
[5]: df2 =  df.withColumn('age',df['age'].cast('int'))
     df2 =  df2.withColumn('duration',df2['duration'].cast('int'))
     df2 =  df2.withColumn('campaign',df2['campaign'].cast('int'))
     df2 =  df2.withColumn('pdays',df2['pdays'].cast('int'))
     df2 =  df2.withColumn('previous',df2['previous'].cast('int'))
     stages = []
     cateColumns = [
         'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'poutcome'
     ]

     for categ in cateColumns:
         indexer = StringIndexer(inputCol = categ, outputCol = categ + 'Index')
         encoder = OneHotEncoderEstimator(
             inputCols=[indexer.getOutputCol()],
             outputCols=[categ + "classVec"]
         )
         stages += [indexer, encoder]

     label_indexer = StringIndexer(inputCol = 'deposit', outputCol = 'label')
     stages += [label_indexer]
     numericCols = ['age', 'duration', 'campaign', 'pdays', 'previous']
     assemblerInputs = [c + "classVec" for c in cateColumns] + numericCols
     assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
     stages += [assembler]
```

Last executed at 2022-12-08 03:12:14 in 807ms

```
[6]: from pyspark.ml import Pipeline
     pipeline = Pipeline(stages = stages)
     pipelineModel = pipeline.fit(df2)
```

Last executed at 2022-12-08 03:12:25 in 11.40s

▸ Spark Job Progress

```
[7]: df_pip = pipelineModel.transform(df2)
     df_pip.printSchema()
```

Last executed at 2022-12-08 03:12:26 in 919ms

```
root
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- day: string (nullable = true)
 |-- month: string (nullable = true)
 |-- duration: integer (nullable = true)
 |-- campaign: integer (nullable = true)
 |-- pdays: integer (nullable = true)
 |-- previous: integer (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- deposit: string (nullable = true)
 |-- jobIndex: double (nullable = false)
 |-- jobclassVec: vector (nullable = true)
 |-- maritalIndex: double (nullable = false)
 |-- maritalclassVec: vector (nullable = true)
```

Modeling

- Decision tree

- A type of machine learning model that uses a tree-like structure to make predictions based on the values of multiple attributes

- Use ParamGridBuilder to find the best model

```
[9]: from pyspark.ml.classification import DecisionTreeClassifier
```
Last executed at 2022-12-08 03:12:27 in 64ms

```
[10]: dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDepth = 3)
model_dt = dt.fit(training)
pred_dt = model_dt.transform(test)
```
Last executed at 2022-12-08 03:12:34 in 7.37s

▸ Spark Job Progress

```
[11]: from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator()
rmse_dt = evaluator.evaluate(pred_dt,{evaluator.metricName: "rmse"})
rmse_dt
```
Last executed at 2022-12-08 03:12:36 in 1.64s

▸ Spark Job Progress

0.4916988552946748

```
[12]: from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
param_grid_dt = ParamGridBuilder()\
            .addGrid(dt.maxDepth, [1,2,3,4,5])\
            .build()
crossvalidate = CrossValidator(estimator=dt,estimatorParamMaps=param_grid_dt, evaluator=evaluator,numFolds=5)
tuned_model_dt = crossvalidate.fit(training)
```
Last executed at 2022-12-08 03:13:20 in 44.48s

▸ Spark Job Progress

```
[13]: model_dt_best = tuned_model_dt.bestModel
```

Modeling

- RMSE = 0.49
- ROC = 0.69
- PR = 0.71

[14]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
pred_dt_best = model_dt_best.transform(test)
pred_dt_best
rmse_dt = evaluator.evaluate(pred_dt_best,{evaluator.metricName: "rmse"})
rmse_dt
```
Last executed at 2022-12-08 03:13:22 in 1.40s

▸ Spark Job Progress

0.45793599183399464

[15]:
```
roc_evaluator = BinaryClassificationEvaluator()
dtROC = roc_evaluator.evaluate(pred_dt_best, {roc_evaluator.metricName: "areaUnderROC"})
dtROC
```
Last executed at 2022-12-08 03:13:24 in 2.52s

▸ Spark Job Progress

0.6980700526239559

[16]:
```
dtPR = roc_evaluator.evaluate(pred_dt_best, {roc_evaluator.metricName: "areaUnderPR"})
dtPR
```
Last executed at 2022-12-08 03:13:26 in 1.46s

▸ Spark Job Progress

0.7121474522922859

Modeling

- Gradient-boosted trees(GBTs)
- GBTs work by training multiple decision trees on the data, and using the errors made by the first tree to train the second tree and next tree
- Use ParamGridBuilder to find the best model

```
[17]: from pyspark.ml.classification import GBTClassifier
Last executed at 2022-12-08 03:13:26 in 318ms
```

```
[18]: gbt = GBTClassifier(maxIter=5, maxDepth=2)
model_gbt = gbt.fit(training)
pred_gbt = model_gbt.transform(test)
Last executed at 2022-12-08 03:13:32 in 5.60s
```

▸ Spark Job Progress

```
[19]: rmse_gbt = evaluator.evaluate(pred_gbt,{evaluator.metricName: "rmse"})
rmse_gbt
Last executed at 2022-12-08 03:13:33 in 1.10s
```

▸ Spark Job Progress

```
0.4925792478623212
```

```
[20]: param_grid_gbt = ParamGridBuilder()\
            .addGrid(gbt.maxIter, [1,2,3,4,5])\
            .addGrid(gbt.maxDepth, [1,2,3,4,5])\
            .build()
crossvalidate = CrossValidator(estimator=gbt,estimatorParamMaps=param_grid_gbt, evaluator=evaluator,numFolds=5)
tuned_model_gbt = crossvalidate.fit(training)
Last executed at 2022-12-08 03:16:50 in 3m 17.28s
```

Modeling

- RMSE = 0.452
- ROC = 0.874
- PR = 0.829

```
[21]: model_gbt_best = tuned_model_gbt.bestModel
      pred_gbt_best = model_gbt_best.transform(test)
      rmse_gbt = evaluator.evaluate(pred_gbt_best,{evaluator.metricName: "rmse"})
      rmse_gbt
```

Last executed at 2022-12-08 03:16:51 in 770ms

▸ Spark Job Progress

```
0.45222346893752163
```

```
[22]: gbtROC = roc_evaluator.evaluate(pred_gbt_best, {roc_evaluator.metricName: "areaUnderROC"})
      gbtROC
```

Last executed at 2022-12-08 03:16:52 in 777ms

▸ Spark Job Progress

```
0.8742558266547577
```

```
[23]: gbtPR = roc_evaluator.evaluate(pred_gbt_best, {roc_evaluator.metricName: "areaUnderPR"})
      gbtPR
```

Last executed at 2022-12-08 03:16:53 in 774ms

▸ Spark Job Progress

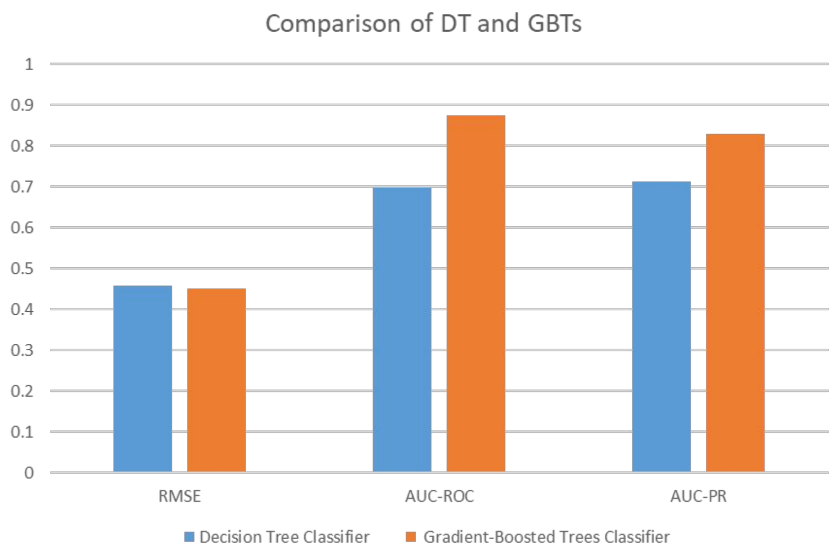
```
0.8293456012152072
```

Result

Compare two algorithms: DT and GBTs

- RMSE: DT ≈ GBTs
- AUC-ROC: DT < GBTs
- AUC-PR: DT < GBTs

	Decision Tree Classifier	Gradient-Boosted Trees Classifier
RMSE	0.458	0.452
AUC-ROC	0.698	0.874
AUC-PR	0.712	0.829



Result

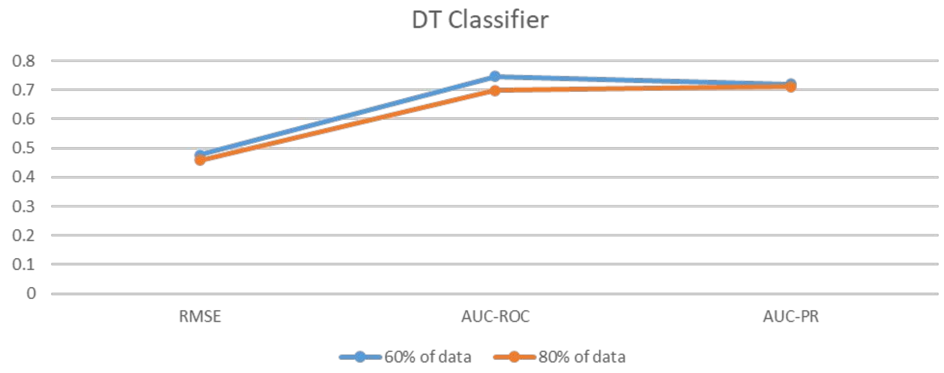
Compare two scales: 60% data and 80% data

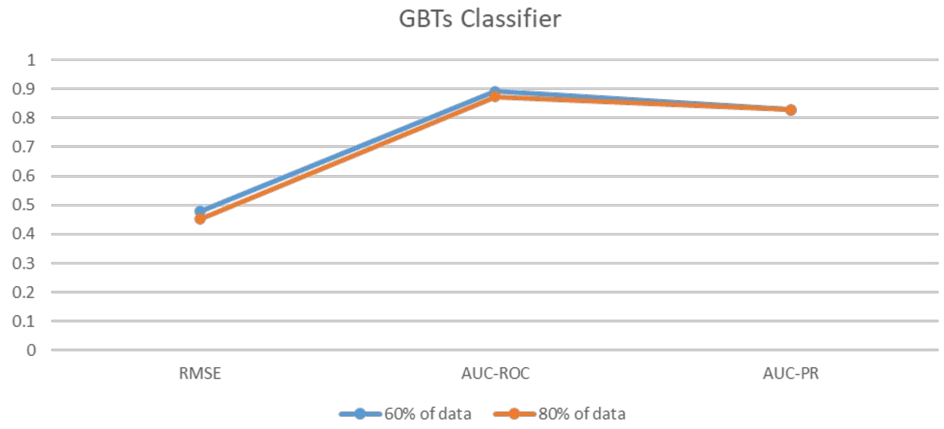
DT Classifier:

- RMSE: 60% data ≈ 80% data
- AUC-ROC: 60% data > 80% data
- AUC-PR: 60% data ≈ 80% data

GBTs Classifier:

- RMSE: 60% data ≈ 80% data
- AUC-ROC: 60% data ≈ 80% data
- AUC-PR: 60% data ≈ 80% data





Conclusion

We were able to build a classification model based on the dataset from bank marketing using binary classification to predict whether a client will subscribe to a term deposit.

- Gradient-Boosted Trees Classifier shows a better performance on this job.
- Decision Tree Classifier requires further hyperparameter tuning that might improve its performance.
- Our model can be applied to larger datasets.

Future Research:

- The models can be tested on different platforms.
- Adding more data.