BIG DATA IN AMERICAN EXPRESS

CONTENT

DISCUSSION FLOW

- What is Big Data?
- About American Express
- How does AmEx use Big Data?
- Fraud Detection Method
- Problem Statement
- Future scopes

WHAT IS BIG DATA?

Big data is a combination of structured, semistructured, and unstructured data collected by organizations that can be mined for information and used in machine learning projects, predictive modeling, and other advanced analytics applications.

Systems that process and store big data have become a common component of data management architectures in organizations, combined with tools that support big data analytics uses.

Big data is often characterized by the four V's:

- the large volume of data in many environments;
- the wide variety of data types frequently stored in big data systems;
- the veracity that refers to the trustworthiness of data;
- and the velocity at which much of the data is generated, collected, and processed.

ABOUT AMEX

The American Express Company is an American multinational corporation specializing in payment card services headquartered at 200 Vesey Street in the Battery Park City neighborhood of Lower Manhattan in New York City. The company was founded in 1850 and is one of the 30 components of the Dow Jones Industrial Average.

SOME FACTS

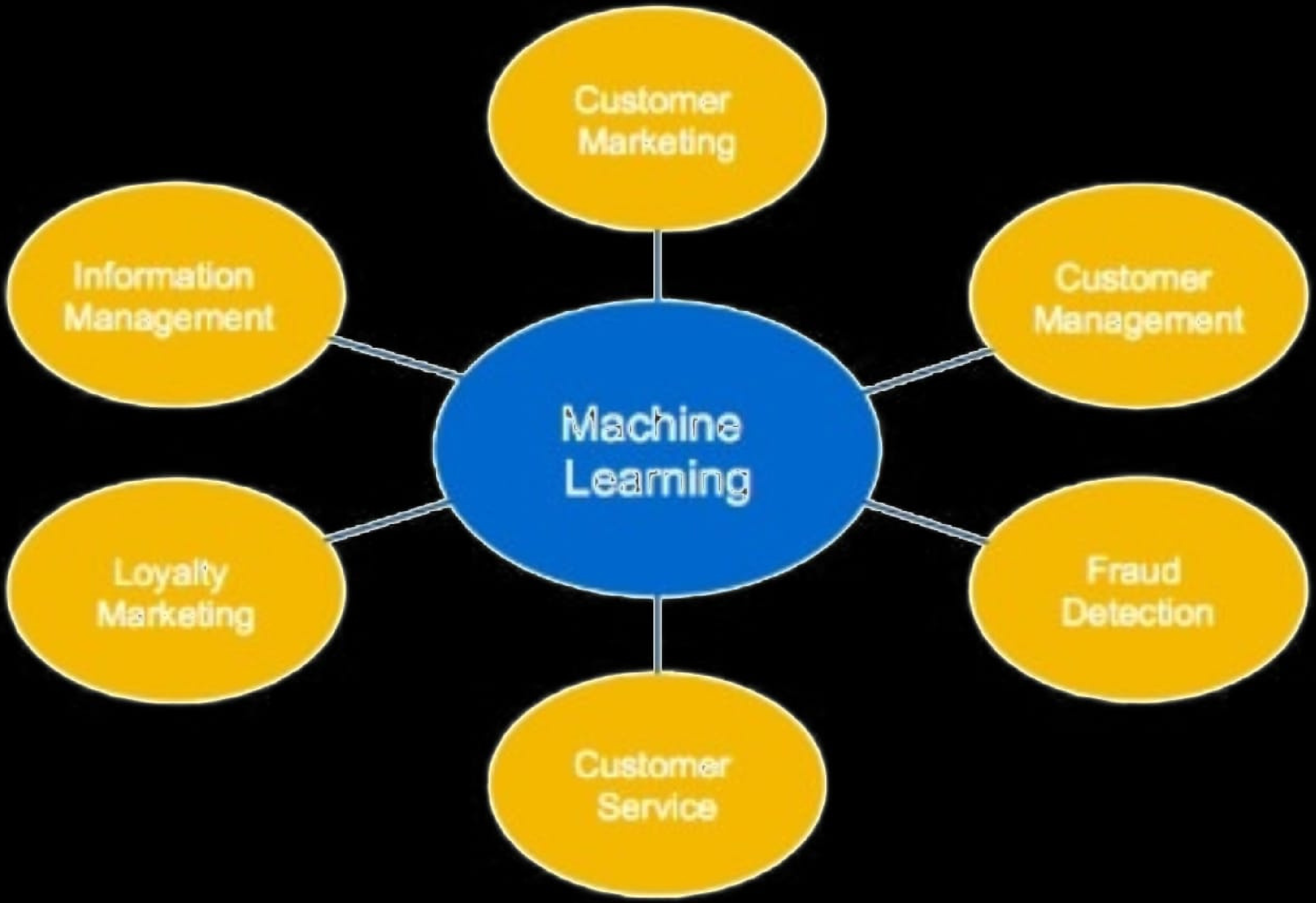
- American Express was the first to introduce plastic credit cards.
- In 1966, American Express introduced the Gold Card, and in 1984 the Platinum Card. The Platinum Card had a $250 annual fee, today it is $550. It was only offered to trusted customers. Today customers can apply for it.
- American Express was granted the contract to run the official currency exchange office on Ellis Island
- American Express continues to be one of the world's biggest financial companies.
-

HOW AMEX USES BIG DATA?

American Express has a rich history of using data and analytics to create deeper relationships with potential and current customers, but it's the advent of machine learning that has allowed its scientists to harness the full power of their data. American Express' Risk & Information Management team in partnership with the company's Technology group embarked on a journey to build world-class Big Data capabilities nearly five years ago. Big data analytics helps American Express drive commerce, service their customers more effectively and detect fraud

American Express is used to operating at a large scale. In business for 165 years, it has continued to transform itself to keep up with changing demand. It has gone from being primarily a shipping company, then a travel business, and now a major credit card issuer, handling over 25% of US credit card spending. And in 2014, the company reached a milestone: one trillion dollars in transactions. The nature of the company allows it to see data from both the customer and merchant side of the business, in fact, from millions of sellers and millions of buyers. One thing American Express is never short of is data.

Machine Learning at American Express



Data volume is not only increasing, but data sources are also changing. More people do business online or via their mobile devices. As a part of American Express’s ongoing journey, they must keep up with these changes in the style of interactions as well as with the increasing volume. Part of that involves making a huge number of decisions, millions every day. If American Express can become just a little bit smarter in these decisions, it can have a huge advantage for customers and the company. That’s why they are expanding how they use machine learning on a large scale. With access to big data, machine learning models can produce superior discrimination and thus better understand customer behavior.

FRAUD DETECTION METHOD

In the case of fraud detection and prevention, machine learning has been helpful to improve American Express's already excellent track record, including their online business interactions. To do this, modeling methods make use of a variety of data sources including card membership information, spending details, and merchant information. The goal is to stop fraudulent transactions before a substantial loss is incurred while allowing normal business transactions to proceed on time. A customer has swiped their card to make a purchase, for instance, and expects to get an approval immediately. In addition to accurately finding fraud, the fraud detection system is required to have these two characteristics:

- Detect suspicious events early
- Make decisions in a few milliseconds against a vast dataset

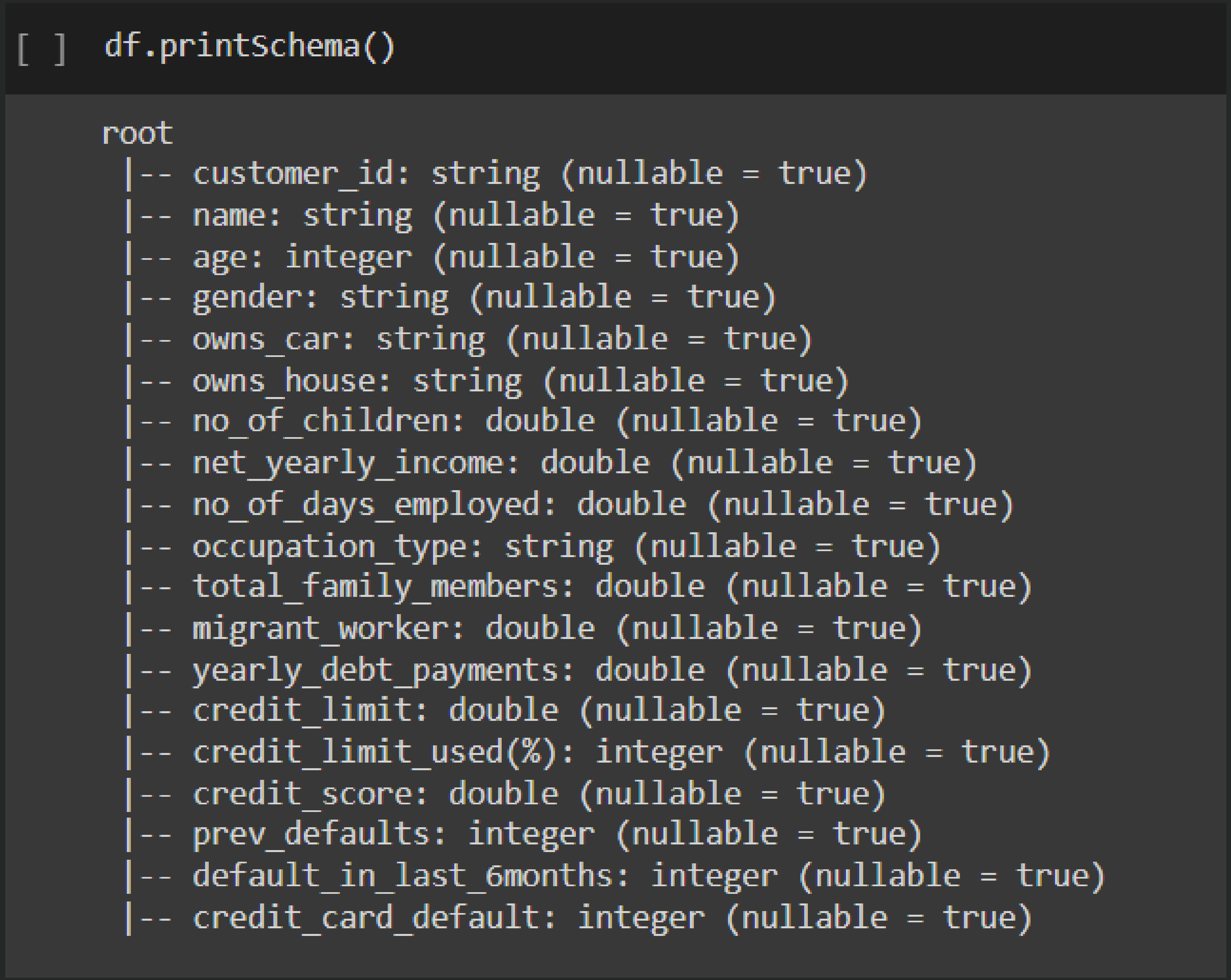
PROBLEM STATEMENT

- Credit card default risk is the chance that companies or Individuals will not be able to return the money lent on time.

- We used the PySpark method to build a machine learning model that can predict if there will be a credit card default.

- Problem solution:- https://colab.research.google.com/drive/1Ay7NewQi2htY4osyN0gQXwEuX_h81WVc?usp=sharing

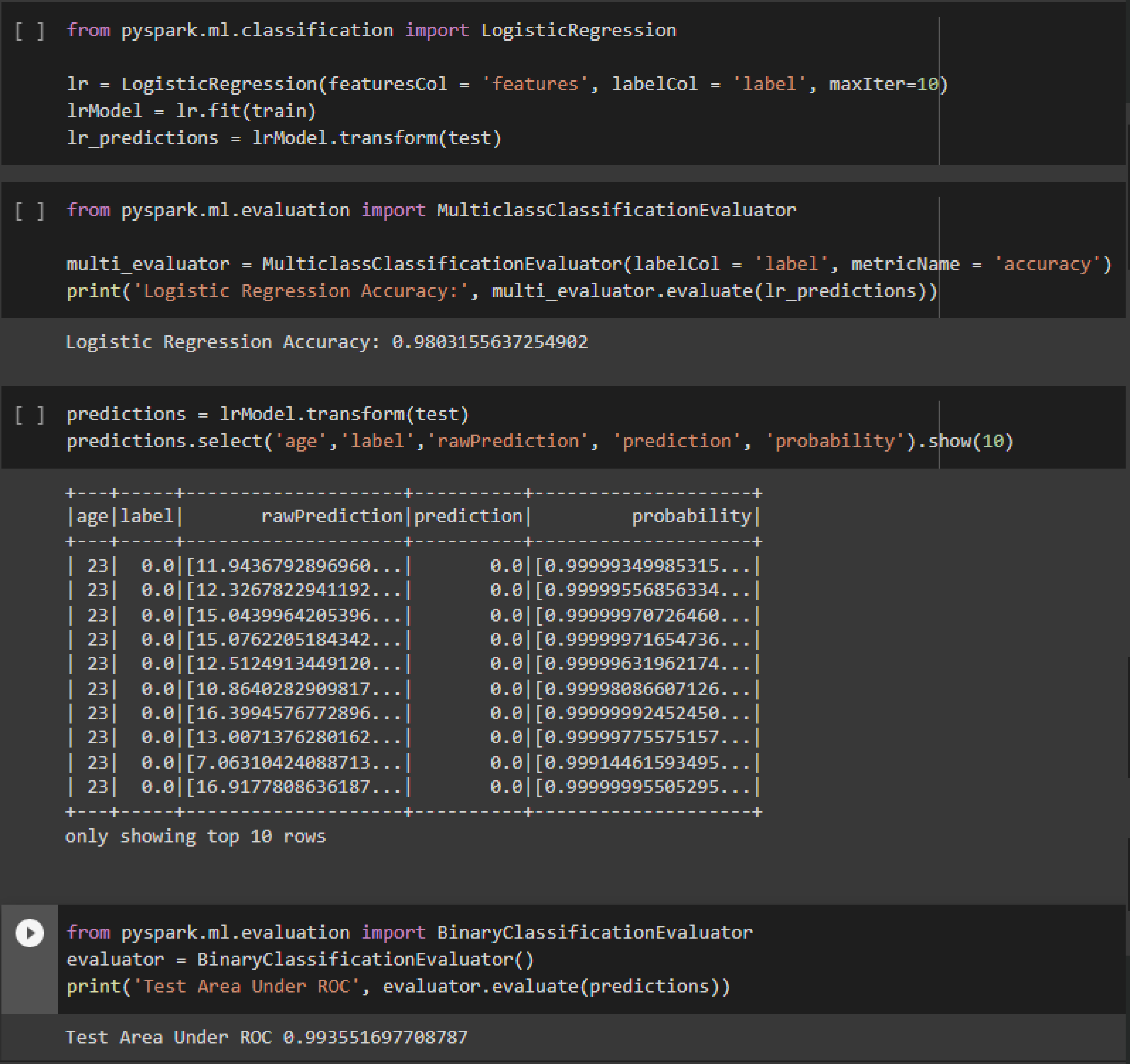
DATASET DESCRIPTION



```
[ ]  df.printSchema()
```

```
root
 |-- customer_id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- gender: string (nullable = true)
 |-- owns_car: string (nullable = true)
 |-- owns_house: string (nullable = true)
 |-- no_of_children: double (nullable = true)
 |-- net_yearly_income: double (nullable = true)
 |-- no_of_days_employed: double (nullable = true)
 |-- occupation_type: string (nullable = true)
 |-- total_family_members: double (nullable = true)
 |-- migrant_worker: double (nullable = true)
 |-- yearly_debt_payments: double (nullable = true)
 |-- credit_limit: double (nullable = true)
 |-- credit_limit_used(%): integer (nullable = true)
 |-- credit_score: double (nullable = true)
 |-- prev_defaults: integer (nullable = true)
 |-- default_in_last_6months: integer (nullable = true)
 |-- credit_card_default: integer (nullable = true)
```

LOGISTIC REGRESSION



```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
lrModel = lr.fit(train)
lr_predictions = lrModel.transform(test)
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'label', metricName = 'accuracy')
print('Logistic Regression Accuracy:', multi_evaluator.evaluate(lr_predictions))
```

```
Logistic Regression Accuracy: 0.9803155637254902
```

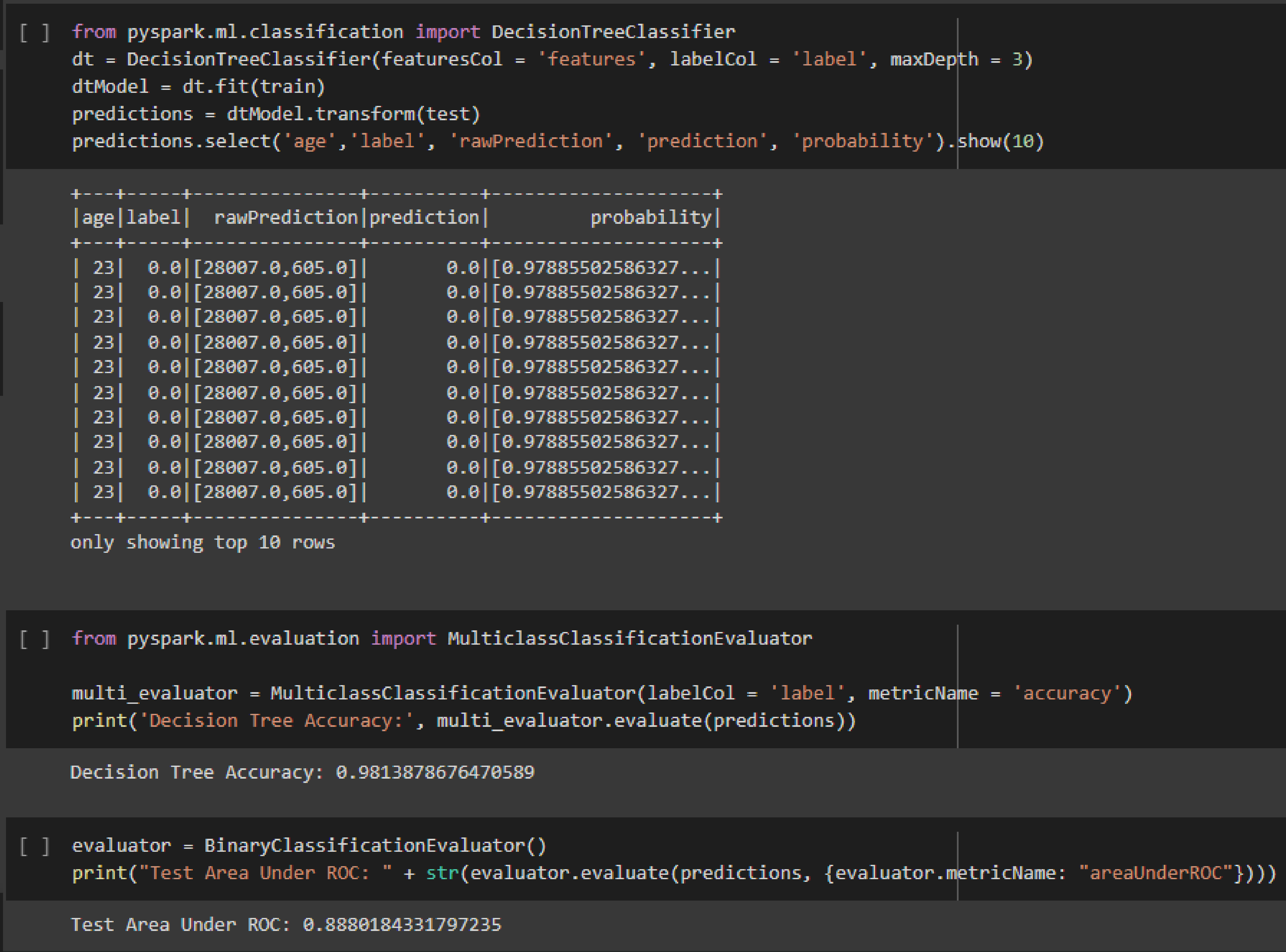
```
predictions = lrModel.transform(test)
predictions.select('age','label','rawPrediction', 'prediction', 'probability').show(10)
```

```
+---+-----+--------------------+----------+--------------------+
|age|label|       rawPrediction|prediction|         probability|
+---+-----+--------------------+----------+--------------------+
| 23|  0.0|[11.9436792896960...|       0.0|[0.99999349985315...|
| 23|  0.0|[12.3267822941192...|       0.0|[0.99999556856334...|
| 23|  0.0|[15.0439964205396...|       0.0|[0.99999970726460...|
| 23|  0.0|[15.0762205184342...|       0.0|[0.99999971654736...|
| 23|  0.0|[12.5124913449120...|       0.0|[0.99999631962174...|
| 23|  0.0|[10.8640282909817...|       0.0|[0.99998086607126...|
| 23|  0.0|[16.3994576772896...|       0.0|[0.99999992452450...|
| 23|  0.0|[13.0071376280162...|       0.0|[0.99999775575157...|
| 23|  0.0|[7.06310424088713...|       0.0|[0.99914461593495...|
| 23|  0.0|[16.9177808636187...|       0.0|[0.99999995505295...|
+---+-----+--------------------+----------+--------------------+
only showing top 10 rows
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
print('Test Area Under ROC', evaluator.evaluate(predictions))
```

```
Test Area Under ROC 0.993551697708787
```

DECISION TREE



```
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDepth = 3)
dtModel = dt.fit(train)
predictions = dtModel.transform(test)
predictions.select('age','label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+---+-----+---------------+----------+--------------------+
|age|label|  rawPrediction|prediction|         probability|
+---+-----+---------------+----------+--------------------+
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
| 23|  0.0|[28007.0,605.0]|       0.0|[0.97885502586327...|
+---+-----+---------------+----------+--------------------+
only showing top 10 rows
```

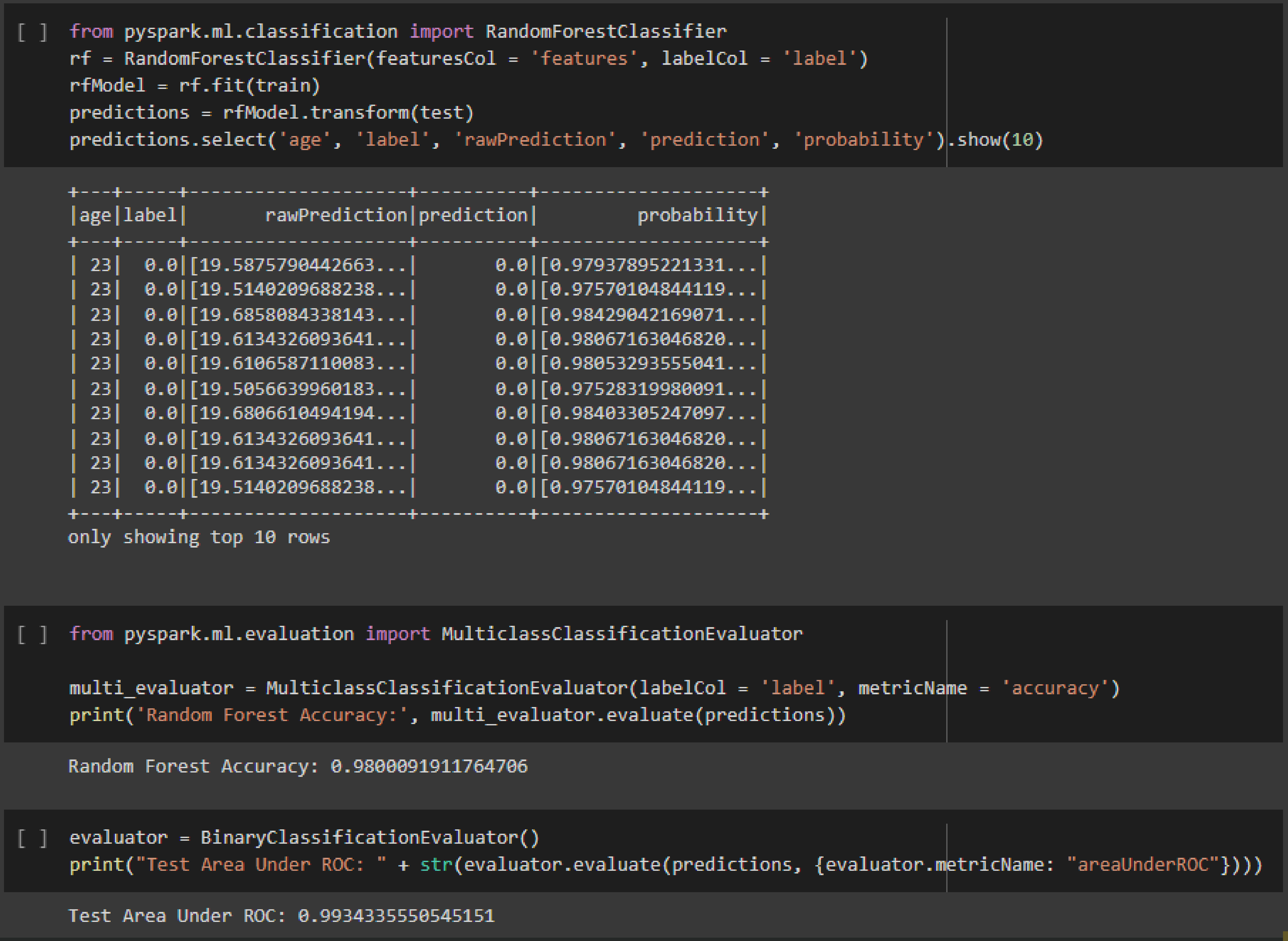
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'label', metricName = 'accuracy')
print('Decision Tree Accuracy:', multi_evaluator.evaluate(predictions))
```

```
Decision Tree Accuracy: 0.9813878676470589
```

```
evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
```

```
Test Area Under ROC: 0.8880184331797235
```

RANDOM FOREST



```python
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'label')
rfModel = rf.fit(train)
predictions = rfModel.transform(test)
predictions.select('age', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+---+-----+--------------------+----------+--------------------+
|age|label|       rawPrediction|prediction|         probability|
+---+-----+--------------------+----------+--------------------+
| 23|  0.0|[19.5875790442663...|       0.0|[0.97937895221331...|
| 23|  0.0|[19.5140209688238...|       0.0|[0.97570104844119...|
| 23|  0.0|[19.6858084338143...|       0.0|[0.98429042169071...|
| 23|  0.0|[19.6134326093641...|       0.0|[0.98067163046820...|
| 23|  0.0|[19.6106587110083...|       0.0|[0.98053293555041...|
| 23|  0.0|[19.5056639960183...|       0.0|[0.97528319980091...|
| 23|  0.0|[19.6806610494194...|       0.0|[0.98403305247097...|
| 23|  0.0|[19.6134326093641...|       0.0|[0.98067163046820...|
| 23|  0.0|[19.6134326093641...|       0.0|[0.98067163046820...|
| 23|  0.0|[19.5140209688238...|       0.0|[0.97570104844119...|
+---+-----+--------------------+----------+--------------------+
only showing top 10 rows
```

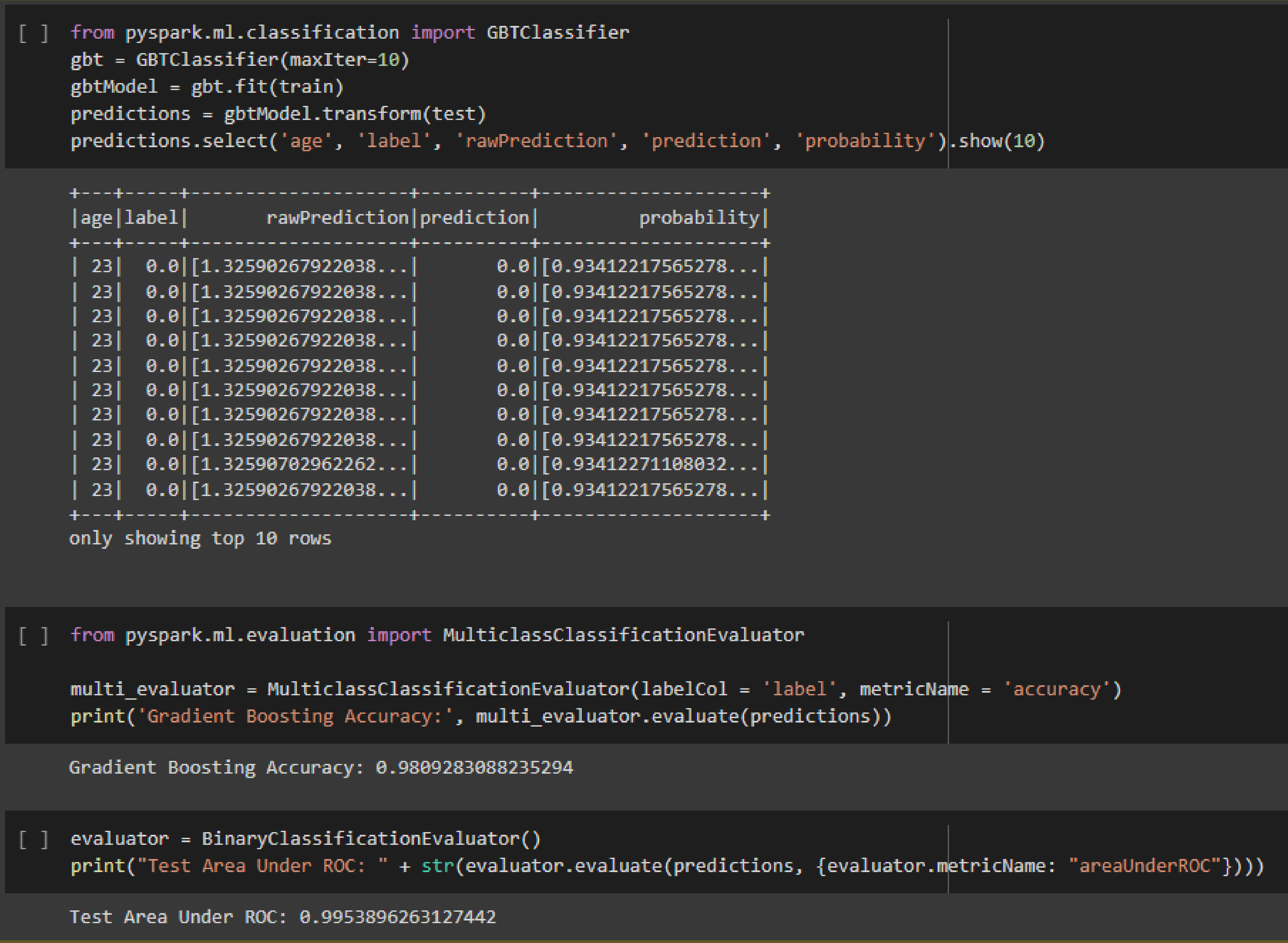
```python
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'label', metricName = 'accuracy')
print('Random Forest Accuracy:', multi_evaluator.evaluate(predictions))
```

```
Random Forest Accuracy: 0.9800091911764706
```

```python
evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
```

```
Test Area Under ROC: 0.9934335550545151
```

GRADIENT-BOOSTING



```python
from pyspark.ml.classification import GBTClassifier
gbt = GBTClassifier(maxIter=10)
gbtModel = gbt.fit(train)
predictions = gbtModel.transform(test)
predictions.select('age', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+---+-----+--------------------+----------+--------------------+
|age|label|       rawPrediction|prediction|         probability|
+---+-----+--------------------+----------+--------------------+
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
| 23|  0.0|[1.32590702962262...|       0.0|[0.93412271108032...|
| 23|  0.0|[1.32590267922038...|       0.0|[0.93412217565278...|
+---+-----+--------------------+----------+--------------------+
only showing top 10 rows
```

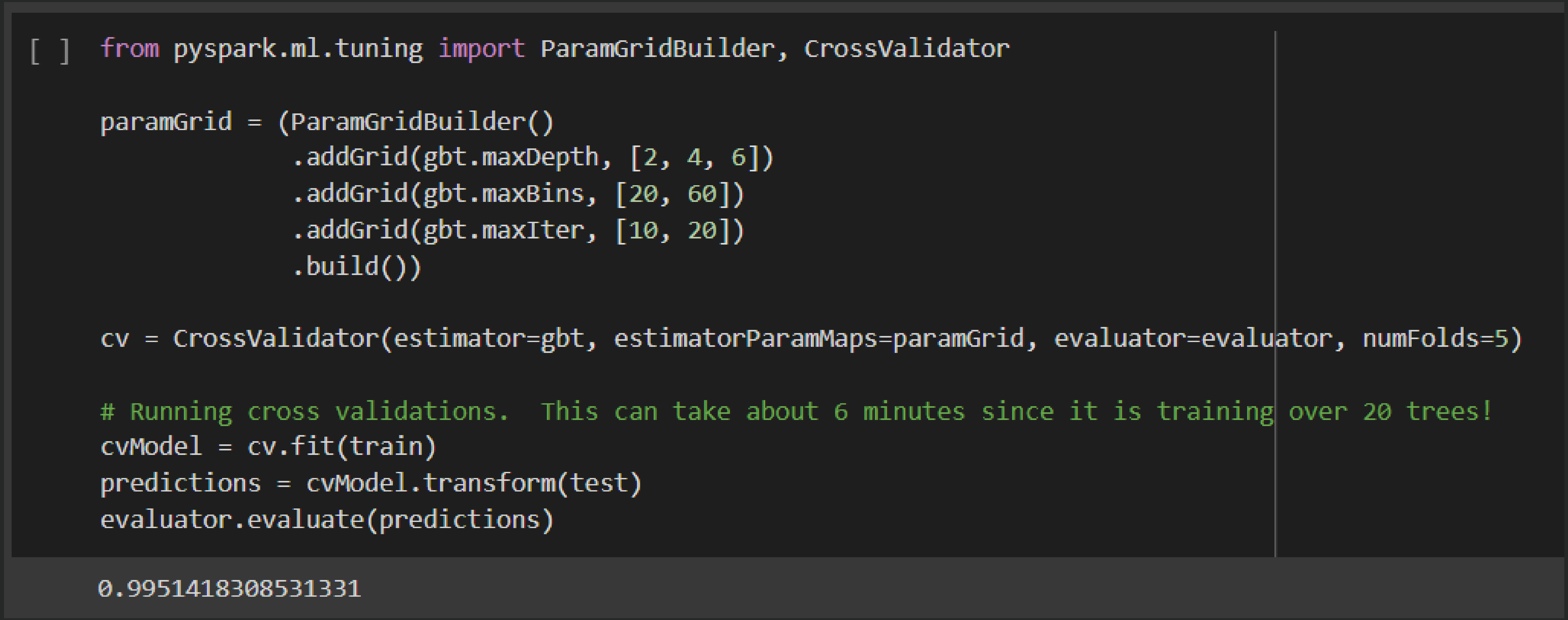
```python
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'label', metricName = 'accuracy')
print('Gradient Boosting Accuracy:', multi_evaluator.evaluate(predictions))
```

```
Gradient Boosting Accuracy: 0.9809283088235294
```

```python
evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
```

```
Test Area Under ROC: 0.9953896263127442
```

FINAL PREDICTION



```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

paramGrid = (ParamGridBuilder()
             .addGrid(gbt.maxDepth, [2, 4, 6])
             .addGrid(gbt.maxBins, [20, 60])
             .addGrid(gbt.maxIter, [10, 20])
             .build())

cv = CrossValidator(estimator=gbt, estimatorParamMaps=paramGrid, evaluator=evaluator, numFolds=5)

# Running cross validations.  This can take about 6 minutes since it is training over 20 trees!
cvModel = cv.fit(train)
predictions = cvModel.transform(test)
evaluator.evaluate(predictions)
```

```
0.9951418308531331
```

FUTURE SCOPES

Large-scale machine learning techniques done correctly are able to meet these criteria and offer an improvement over traditional linear regression methods, taking the precision of predictions to a new level.

The large-scale machine learning deployment is adding far greater precision to the systems’ predictive capabilities.

AMERICAN EXPRESS
7997
3759 876543 21001
Member Since
21
SUSANNA K LEE

THANK YOU!