Machine learning

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

More Resources

Working with Notebooks in Colab

- Overview of Colab
- Guide to Markdown
- Importing libraries and installing dependencies
- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets

Working with Data

- Loading data: Drive, Sheets, and Google Cloud Storage
- Charts: visualizing data
- Getting started with BigQuery

Machine Learning

These are a few of the notebooks related to Machine Learning, including Google's online Machine Learning course. See the full course website for more.

- Intro to Pandas DataFrame
- Intro to RAPIDS cuDF to accelerate pandas
- Getting Started with cuML's accelerator mode
- Linear regression with tf.keras using synthetic data

Using Accelerated Hardware

- TensorFlow with GPUs
- TPUs in Colab

Featured examples

- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB movie reviews as either *positive* or *negative*.
- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
```

```
# Step 2: Start SparkSession
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder \
    .appName("Task2_ML_Project") \
    .getOrCreate()


# Step 3: Load your dataset into a Spark DataFrame (Friend)
df = spark.read.csv("/content/college_student_management_data.csv", header=True, inferSchema=True)

# Show first few rows
df.show(5)
```

```
+----------+---+------+----------------+----+----------+----------------+--------------+-----------------+----------------------+------
|student_id|age|gender|           major| GPA|course_load|avg_course_grade|attendance_rate|enrollment_status|lms_logins_past_month|avg_se
+----------+---+------+----------------+----+----------+----------------+--------------+-----------------+----------------------+------
|      S001| 24| Other|Computer Science|2.42|         5|            67.2|          0.71|        Graduated|                    32|
|      S002| 21|  Male|            Arts|3.73|         6|            64.4|          0.84|           Leave|                    29|
|      S003| 22|  Male|Computer Science| 2.8|         3|            95.3|          0.89|        Graduated|                    34|
|      S004| 24|  Male|            Arts|2.59|         4|            73.7|          0.98|        Graduated|                    22|
|      S005| 20| Other|Computer Science| 2.3|         4|            87.4|          0.95|           Active|                     9|
+----------+---+------+----------------+----+----------+----------------+--------------+-----------------+----------------------+------
only showing top 5 rows
```

```
df.printSchema()
```

```
root
 |-- student_id: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- gender: string (nullable = true)
 |-- major: string (nullable = true)
 |-- GPA: double (nullable = true)
 |-- course_load: integer (nullable = true)
 |-- avg_course_grade: double (nullable = true)
 |-- attendance_rate: double (nullable = true)
 |-- enrollment_status: string (nullable = true)
 |-- lms_logins_past_month: integer (nullable = true)
 |-- avg_session_duration_minutes: integer (nullable = true)
 |-- assignment_submission_rate: double (nullable = true)
 |-- forum_participation_count: integer (nullable = true)
 |-- video_completion_rate: double (nullable = true)
 |-- risk_level: string (nullable = true)
```

```
from pyspark.sql.functions import col, isnan, when, count

df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

```
+----------+---+------+-----+---+----------+----------------+---------------+-----------------+----------------------+------------------
|student_id|age|gender|major|GPA|course_load|avg_course_grade|attendance_rate|enrollment_status|lms_logins_past_month|avg_session_durati
+----------+---+------+-----+---+----------+----------------+---------------+-----------------+----------------------+------------------
|         0|  0|     0|    0|  0|          0|               0|              0|                0|                     0|
+----------+---+------+-----+---+----------+----------------+---------------+-----------------+----------------------+------------------
```

```
df = df.dropna()


df = df.dropDuplicates()


df.show(5)
```

```
+----------+---+------+----------------+----+----------+----------------+--------------+-----------------+----------------------+------
|student_id|age|gender|           major| GPA|course_load|avg_course_grade|attendance_rate|enrollment_status|lms_logins_past_month|avg_se
+----------+---+------+----------------+----+----------+----------------+--------------+-----------------+----------------------+------
|      S176| 23|  Male|        Business|2.41|         6|            82.3|          0.92|        Graduated|                     8|
|      S314| 24| Other|Computer Science|2.51|         4|            89.5|          0.65|           Leave|                     4|
|      S495| 22|Female|     Engineering|2.14|         3|            73.7|          0.68|           Active|                    12|
|      S513| 24| Other|        Business| 2.9|         5|            74.9|          0.92|        Graduated|                    19|
|      S603| 21| Other|            Arts|2.56|         5|            99.5|          0.87|           Active|                    37|
+----------+---+------+----------------+----+----------+----------------+--------------+-----------------+----------------------+------
only showing top 5 rows
```

```
df.describe().show()
```

```
+-------+----------+------------------+------+----------+------------------+------------------+----------------+------------------+---
|summary|student_id|               age|gender|     major|               GPA|       course_load|avg_course_grade|    attendance_rate|enr
```

```
+-------+----------+------------------+------+----------+------------------+------------------+---------------+-------------------+---
|  count|      1545|              1545|  1545|      1545|              1545|              1545|           1545|               1545|
|   mean|      NULL| 21.48284789644013|  NULL|      NULL|3.0123236245954677| 4.487378640776699|79.9148867313916| 0.7944595469255684|
| stddev|      NULL|2.3002865710826996|  NULL|      NULL| 0.571394620466219|1.0981625856128319|11.5286250411808|0.11514176248178233|
|    min|      S001|                18|Female|      Arts|               2.0|                 3|           60.0|                0.6|
|    max|      S999|                25| Other|Engineering|               4.0|                 6|          100.0|                1.0|
+-------+----------+------------------+------+----------+------------------+------------------+---------------+-------------------+---
```

```
df.count()
```

```
1545
```

```
df.groupBy("risk_level").count().show()
```

```
+----------+-----+
|risk_level|count|
+----------+-----+
|      High|  805|
|       Low|  284|
|    Medium|  456|
+----------+-----+
```

```
df.toPandas().to_csv("cleaned_dataset.csv", index=False)
```

```
from google.colab import files
files.download("cleaned_dataset.csv")
```

```
df.columns
```

```
['student_id',
 'age',
 'gender',
 'major',
 'GPA',
 'course_load',
 'avg_course_grade',
 'attendance_rate',
 'enrollment_status',
 'lms_logins_past_month',
 'avg_session_duration_minutes',
 'assignment_submission_rate',
 'forum_participation_count',
 'video_completion_rate',
 'risk_level']
```

```
from pyspark.ml.feature import StringIndexer

indexer = StringIndexer(inputCol="risk_level", outputCol="label")
df = indexer.fit(df).transform(df)
df.select("risk_level", "label").show(5)
```

```
+----------+-----+
|risk_level|label|
+----------+-----+
|      High|  0.0|
|      High|  0.0|
|      High|  0.0|
|    Medium|  1.0|
|    Medium|  1.0|
+----------+-----+
only showing top 5 rows
```

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer

cat_cols = ["gender", "major"]
indexers = [StringIndexer(inputCol=col, outputCol=col+"_indexed") for col in cat_cols]
pipeline = Pipeline(stages=indexers)
df = pipeline.fit(df).transform(df)

df.select("gender", "gender_indexed").show(3)
```

```
+------+--------------+
|gender|gender_indexed|
+------+--------------+
|  Male|           0.0|
| Other|           2.0|
|Female|           1.0|
+------+--------------+
only showing top 3 rows
```

```
from pyspark.ml.feature import VectorAssembler

input_features = [
 "age",
 "GPA",
 "course_load",
 "avg_course_grade",
 "attendance_rate",
 "assignment_submission_rate",
 "video_completion_rate",
 "gender_indexed",
 "major_indexed"
]

assembler = VectorAssembler(inputCols=input_features, outputCol="features")
df = assembler.transform(df)

df.select("features", "label").show(3, truncate=False)
```

```
+-----------------------------------------+-----+
|features                                 |label|
+-----------------------------------------+-----+
|[23.0,2.41,6.0,82.3,0.92,0.51,0.75,0.0,3.0]|0.0  |
|[24.0,2.51,4.0,89.5,0.65,0.85,0.53,2.0,1.0]|0.0  |
|[22.0,2.14,3.0,73.7,0.68,0.95,0.65,1.0,2.0]|0.0  |
+-----------------------------------------+-----+
only showing top 3 rows
```

```
# Split into 80% train and 20% test
train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)

print("Training rows:", train_data.count())
print("Testing rows:", test_data.count())
```

```
Training rows: 1282
Testing rows: 263
```

```
from pyspark.ml.classification import DecisionTreeClassifier

# Create Decision Tree model
dt = DecisionTreeClassifier(featuresCol="features", labelCol="label")

# Train the model
dt_model = dt.fit(train_data)

# Apply model on test data
predictions = dt_model.transform(test_data)

# Show a few predictions
predictions.select("prediction", "label", "features").show(5)
```

```
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       1.0|  1.0|[22.0,2.8,3.0,95....|
|       0.0|  0.0|[22.0,2.04,3.0,87...|
```

```
|       1.0|  1.0|[24.0,3.68,6.0,62...|
|       0.0|  0.0|[20.0,2.83,6.0,65...|
|       1.0|  1.0|[20.0,3.64,6.0,64...|
+----------+-----+--------------------+
only showing top 5 rows
```

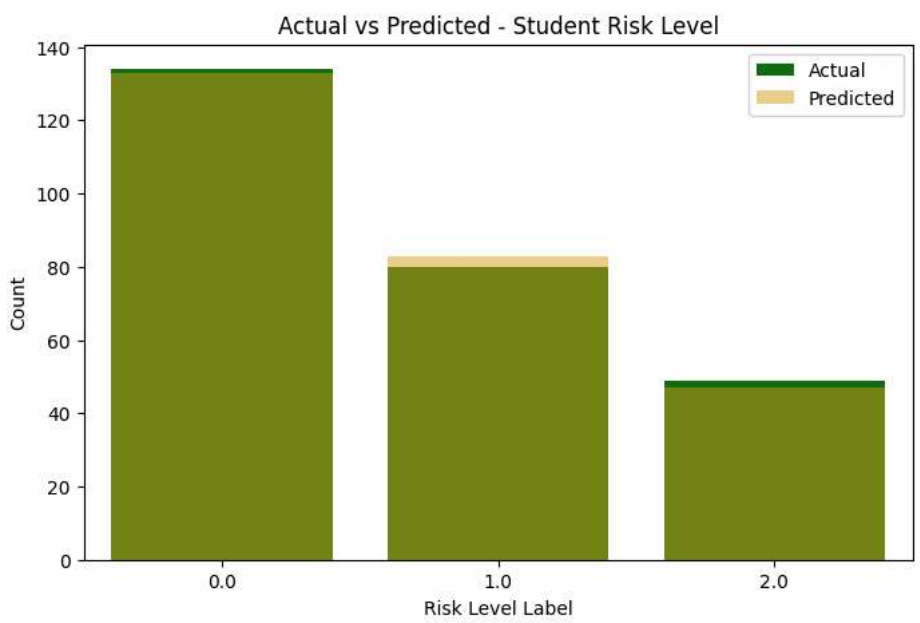
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(
    labelCol="label", predictionCol="prediction", metricName="accuracy")

accuracy = evaluator.evaluate(predictions)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
```

```
Model Accuracy: 98.86%
```

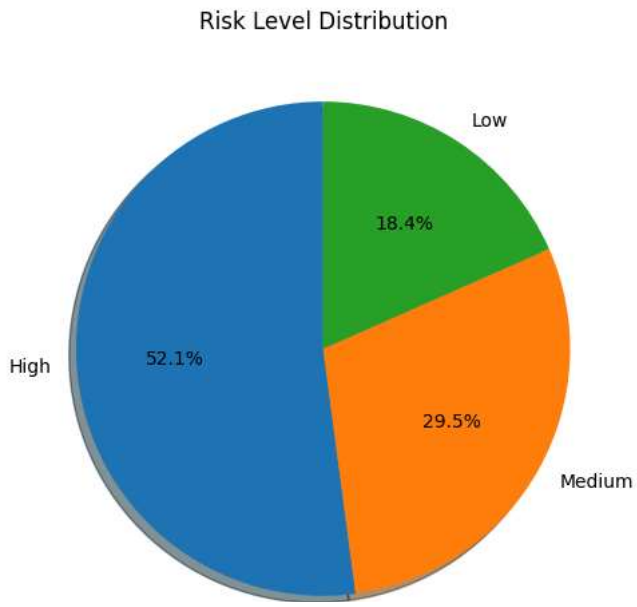
```
pdf = predictions.select("prediction", "label").toPandas()
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.figure(figsize=(8, 5))
sns.countplot(data=pdf, x='label', color='green', label='Actual')
sns.countplot(data=pdf, x='prediction', color='orange', alpha=0.5, label='Predicted')
plt.title('Actual vs Predicted - Student Risk Level')
plt.xlabel('Risk Level Label')
plt.ylabel('Count')
plt.legend()
plt.show()
```




```
df.toPandas()['risk_level'].value_counts().plot.pie(
    autopct='%1.1f%%', shadow=True, startangle=90, figsize=(6, 6), title='Risk Level Distribution')
plt.ylabel("")
plt.show()
```


Risk Level Distribution
Low
18.4%
High
52.1%
29.5%
Medium