In [1]:

```
from pyspark.sql import functions as F
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, count, when, col
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.classification import DecisionTreeClassifier, RandomForestClassifier, LogisticRe
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
VBox()
Starting Spark application
```

ID	Kind	State	Spark UI	Driver log	User	Current session?
6	pyspark	idle	Link		None	✓

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
SparkSession available as 'spark'.
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

In [2]:

```
spark = SparkSession.builder \
    .appName("COVID-19 Data Processing") \
    .config("spark.sql.shuffle.partitions", "100") \
    .config("spark.driver.memory", "8g") \
    .config("spark.executor.memory", "8g") \
    .config("spark.driver.maxResultSize", "2g") \
    .config("spark.sql.execution.arrow.pyspark.enabled", "false") \
    .getOrCreate()

# Read CSV file from S3 and repartition for efficient processing
covid_read = spark.read.csv("s3://covid-data-project-final/Covid Data.csv", header=True, inferSc
covid_read = covid_read.repartition(40)  # Adjust based on your instance capacity
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

This initializes a Spark session with specific memory and partition settings for efficient data processing. Then reads a CSV file from an S3 bucket named "covid-data-project-final". Finally, it repartitions the data into 40 parts to optimize performance based on the instance's processing capacity.

In [3]:

```
# Convert CLASIFFICATION_FINAL to binary classification
covid_read = covid_read.withColumn(
    "CLASIFFICATION_FINAL",
    when(covid_read["CLASIFFICATION_FINAL"] <= 3, 0).otherwise(1)
)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

We converted the CLASIFFICATION_FINAL column in the dataset to a binary classification. If the value in CLASIFFICATION_FINAL is 3 or less, it is set to 0 (indicating "has COVID"). For any other value, it is set to 1 (indicating "no COVID").

Dataset Description: classification: covid test findings. Values 1-3 mean that the patient was diagnosed with covid in different degrees. 4 or higher means that the patient is not a carrier of covid or that the test is inconclusive.

In [4]:

```
# Initial Data Exploration
covid_read.show(truncate=False)
print(f"DATA SET COUNT: {covid_read.count()}")
covid_read.printSchema()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

```
+-----+------------+---+------------+----------+-------+---------+---+--------+--------+----+-----
-+-------+------------+-------------+--------------+-------+-------------+-------+----------------
----+---+
|USMER|MEDICAL_UNIT|SEX|PATIENT_TYPE|DATE_DIED |INTUBED|PNEUMONIA|AGE|PREGNANT|DIABETES|COPD|ASTHM
A|INMSUPR|HIPERTENSION|OTHER_DISEASE|CARDIOVASCULAR|OBESITY|RENAL_CHRONIC|TOBACCO|CLASIFFICATION_F
INAL|ICU|
+-----+------------+---+------------+----------+-------+---------+---+--------+--------+----+-----
-+-------+------------+-------------+--------------+-------+-------------+-------+----------------
----+---+
|2    |4           |2  |1           |9999-99-99|97     |2        |48 |97      |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|2    |4           |2  |1           |9999-99-99|97     |2        |48 |97      |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |2  |1           |9999-99-99|97     |2        |47 |97      |2       |2   |1
|1      |1           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |1  |1           |9999-99-99|97     |99       |36 |2       |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |2  |1           |9999-99-99|97     |2        |22 |97      |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |2  |1           |9999-99-99|97     |2        |22 |97      |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |2  |2           |9999-99-99|99     |99       |7  |97      |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|99 |
|1    |4           |1  |1           |9999-99-99|97     |2        |47 |2       |1       |2   |2
|2      |1           |2            |2             |2      |2            |2      |1
|97 |
|2    |4           |2  |2           |9999-99-99|2      |1        |35 |97      |1       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|2  |
|2    |4           |1  |1           |9999-99-99|97     |2        |50 |2       |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|2    |4           |1  |1           |9999-99-99|97     |2        |50 |2       |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|2    |4           |1  |1           |9999-99-99|97     |2        |50 |2       |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|2    |4           |1  |1           |9999-99-99|97     |2        |50 |2       |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |1  |1           |9999-99-99|97     |2        |33 |2       |2       |2   |2
|2      |2           |1            |2             |2      |2            |2      |1
|97 |
|2    |4           |1  |1           |9999-99-99|97     |2        |64 |2       |1       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |1  |1           |9999-99-99|97     |99       |24 |2       |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
|97 |
|1    |4           |2  |2           |9999-99-99|2      |2        |30 |97      |2       |2   |1
|2      |2           |2            |2             |2      |2            |2      |1
|2  |
|2    |4           |2  |2           |9999-99-99|2      |1        |63 |97      |2       |2   |2
|2      |2           |2            |2             |2      |2            |2      |1
```

```
|2  |
|1    |4           |2  |2           |9999-99-99|2      |1         |39 |97       |1        |2   |2
|2      |1            |2             |2              |2       |2             |2       |1
|2  |
|2    |4           |2  |1           |9999-99-99|97     |2         |33 |97       |2        |2   |2
|2      |2            |1             |2              |1       |2             |1       |1
|97 |
+-----+------------+---+------------+----------+-------+---------+---+--------+--------+----+------
-+--------+------------+-------------+--------------+-------+-------------+-------+----------------
----+---+
only showing top 20 rows

DATA SET COUNT: 1048575
root
 |-- USMER: integer (nullable = true)
 |-- MEDICAL_UNIT: integer (nullable = true)
 |-- SEX: integer (nullable = true)
 |-- PATIENT_TYPE: integer (nullable = true)
 |-- DATE_DIED: string (nullable = true)
 |-- INTUBED: integer (nullable = true)
 |-- PNEUMONIA: integer (nullable = true)
 |-- AGE: integer (nullable = true)
 |-- PREGNANT: integer (nullable = true)
 |-- DIABETES: integer (nullable = true)
 |-- COPD: integer (nullable = true)
 |-- ASTHMA: integer (nullable = true)
 |-- INMSUPR: integer (nullable = true)
 |-- HIPERTENSION: integer (nullable = true)
 |-- OTHER_DISEASE: integer (nullable = true)
 |-- CARDIOVASCULAR: integer (nullable = true)
 |-- OBESITY: integer (nullable = true)
 |-- RENAL_CHRONIC: integer (nullable = true)
 |-- TOBACCO: integer (nullable = true)
 |-- CLASIFFICATION_FINAL: integer (nullable = false)
 |-- ICU: integer (nullable = true)
```

Here, we can see the dataset's count and schema, showing the column names and data types for each field.

In [5]:

```
# Count missing values in each column
missing_counts = covid_read.select([(count(when(col(c).isNull(), c)).alias(c)) for c in covid_rea
missing_counts.show()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
+-----+------------+---+------------+---------+-------+---------+---+--------+--------+----+------
+-------+------------+-------------+--------------+-------+-------------+-------+-----------------
---+---+
|USMER|MEDICAL_UNIT|SEX|PATIENT_TYPE|DATE_DIED|INTUBED|PNEUMONIA|AGE|PREGNANT|DIABETES|COPD|ASTHMA
|INMSUPR|HIPERTENSION|OTHER_DISEASE|CARDIOVASCULAR|OBESITY|RENAL_CHRONIC|TOBACCO|CLASIFFICATION_FI
NAL|ICU|
+-----+------------+---+------------+---------+-------+---------+---+--------+--------+----+------
+-------+------------+-------------+--------------+-------+-------------+-------+-----------------
---+---+
|    0|           0|  0|           0|        0|      0|        0|  0|       0|       0|   0|     0
|      0|           0|            0|             0|      0|            0|      0|
0|  0|
+-----+------------+---+------------+---------+-------+---------+---+--------+--------+----+------
+-------+------------+-------------+--------------+-------+-------------+-------+-----------------
---+---+
```

In [6]:

```
# Find and drop duplicate rows
covid_data_final = covid_read.dropDuplicates()

# Drop rows with null values
covid_data_final_cleaned = covid_read.na.drop()
covid_data_final_cleaned.show()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

```
+-----+------------+---+------------+----------+-------+---------+---+--------+--------+----+-----
-+-------+------------+-------------+--------------+-------+-------------+-------+----------------
----+---+
|USMER|MEDICAL_UNIT|SEX|PATIENT_TYPE| DATE_DIED|INTUBED|PNEUMONIA|AGE|PREGNANT|DIABETES|COPD|ASTHM
A|INMSUPR|HIPERTENSION|OTHER_DISEASE|CARDIOVASCULAR|OBESITY|RENAL_CHRONIC|TOBACCO|CLASIFFICATION_F
INAL|ICU|
+-----+------------+---+------------+----------+-------+---------+---+--------+--------+----+-----
-+-------+------------+-------------+--------------+-------+-------------+-------+----------------
----+---+
|    2|           4|  2|           1|9999-99-99|     97|        2| 48|      97|       2|   2|
2|      2|           2|            2|             2|      2|            2|      2|
1| 97|
|    2|           4|  2|           1|9999-99-99|     97|        2| 48|      97|       2|   2|
2|      2|           2|            2|             2|      2|            2|      2|
1| 97|
|    2|           4|  1|           1|9999-99-99|     97|        2| 62|       2|       2|   2|
2|      2|           1|            2|             2|      2|            2|      2|
0| 97|
|    1|           4|  1|           1|9999-99-99|     97|        2| 47|       2|       2|   2|
2|      2|           2|            2|             2|      1|            2|      2|
0| 97|
|    2|           4|  1|           1|9999-99-99|     97|        2| 26|       2|       2|   2|
1|      2|           2|            2|             2|      2|            2|      2|
1| 97|
|    1|           4|  2|           1|9999-99-99|     97|        2| 30|      97|       2|   2|
2|      2|           2|            2|             2|      2|            2|      2|
1| 97|
|    1|           4|  1|           1|9999-99-99|     97|       99| 30|       2|       2|   2|
2|      2|           2|            2|             2|      2|            2|      2|
1| 97|
|    1|           4|  1|           1|9999-99-99|     97|        2| 47|       2|       1|   2|
2|      2|           1|            2|             2|      2|            2|      2|
1| 97|
|    2|           4|  2|           1|9999-99-99|     97|        2| 27|      97|       2|   2|
2|      2|           2|            2|             2|      1|            2|      2|
0| 97|
|    1|           4|  2|           2|9999-99-99|      2|        1| 65|      97|       2|   2|
2|      2|           2|            2|             2|      2|            2|      2|
0|  2|
|    2|           4|  2|           2|9999-99-99|      2|        2| 51|      97|       1|   2|
2|      2|           1|            1|             2|      2|            2|      2|
0|  2|
|    1|           4|  1|           1|9999-99-99|     97|        2| 46|       2|       2|   2|
2|      2|           1|            2|             2|      2|            2|      2|
0| 97|
|    2|           4|  2|           1|9999-99-99|     97|        1| 33|      97|       2|   2|
2|      2|           2|            2|             2|      2|            2|      2|
0| 97|
|    2|           4|  1|           1|9999-99-99|     97|        2| 44|       2|       2|   2|
2|      2|           2|            2|             2|      1|            2|      2|
0| 97|
|    1|           4|  1|           1|9999-99-99|     97|        2| 36|       2|       2|   2|
2|      2|           2|            2|             2|      2|            2|      1|
0| 97|
|    1|           4|  2|           1|9999-99-99|     97|        2| 57|      97|       1|   2|
2|      2|           1|            1|             2|      2|            2|      2|
0| 97|
|    2|           4|  2|           1|9999-99-99|     97|        2| 53|      97|       2|   2|
2|      2|           1|            2|             2|      2|            2|      2|
0| 97|
|    2|           4|  2|           1|9999-99-99|     97|        2| 38|      97|       2|   2|
2|      2|           2|            2|             2|      1|            2|      2|
0| 97|
|    2|           4|  1|          2|9999-99-99|      2|        2| 53|        2|        2|   2|
1|      2|            2|            2|              2|       2|             2|       2|
0|  2|
|    2|           4|  1|          1|9999-99-99|     97|        2| 37|        2|        2|   2|
2|      2|            2|            2|              2|       1|             2|       2|
0| 97|
+-----+------------+---+------------+----------+-------+---------+---+--------+--------+----+-----
-+--------+------------+------------+--------------+-------+-------------+-------+----------------
----+---+
only showing top 20 rows
```

In [7]:

```python
# Numeric Columns for EDA
numeric_cols = ["AGE", "INTUBED", "PNEUMONIA", "OBESITY", "DIABETES"]

# Visualize distributions of numeric columns (convert to Pandas for plotting)
covid_pd = covid_data_final_cleaned.toPandas()

# EDA: Descriptive Statistics
covid_data_final_cleaned.describe(numeric_cols).show()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
+-------+------------------+------------------+------------------+-----------------+--------------
----+
|summary|               AGE|           INTUBED|         PNEUMONIA|          OBESITY|             DIAB
ETES|
+-------+------------------+------------------+------------------+-----------------+--------------
----+
|  count|           1048575|           1048575|           1048575|          1048575|               104
8575|
|   mean|41.794102472403026| 79.52287533080609|3.3468306988055216|2.125175595450969|2.186404405979
5438|
| stddev|16.907389199431204|36.868886275044304|11.912881086507994|5.175445110188411|5.424241787888
3415|
|    min|                 0|                 1|                 1|                1|
1|
|    max|               121|                99|                99|               98|
98|
+-------+------------------+------------------+------------------+-----------------+--------------
----+
```

Here, It displays descriptive statistics (such as count, mean, standard deviation, min, max) for the selected numeric columns ["AGE", "INTUBED", "PNEUMONIA", "OBESITY", "DIABETES"].

In [8]:

```python
# Categorical data count
categorical_cols = ['SEX', 'PATIENT_TYPE', 'PNEUMONIA', 'DIABETES', 'COPD', 'ASTHMA',
                    'INMSUPR', 'HIPERTENSION', 'OTHER_DISEASE', 'CARDIOVASCULAR',
                    'OBESITY', 'RENAL_CHRONIC', 'TOBACCO', 'ICU']

# Count unique values in categorical columns
for col in categorical_cols:
    print(f"{col}: {covid_data_final_cleaned.select(col).distinct().count()} unique values")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

```
SEX: 2 unique values
PATIENT_TYPE: 2 unique values
PNEUMONIA: 3 unique values
DIABETES: 3 unique values
COPD: 3 unique values
ASTHMA: 3 unique values
INMSUPR: 3 unique values
HIPERTENSION: 3 unique values
OTHER_DISEASE: 3 unique values
CARDIOVASCULAR: 3 unique values
OBESITY: 3 unique values
RENAL_CHRONIC: 3 unique values
TOBACCO: 3 unique values
ICU: 4 unique values
```

Then we iterated over each column in the list of categorical_cols - ['SEX', 'PATIENT_TYPE', 'PNEUMONIA', 'DIABETES', 'COPD', 'ASTHMA', 'INMSUPR', 'HIPERTENSION', 'OTHER_DISEASE', 'CARDIOVASCULAR', 'OBESITY', 'RENAL_CHRONIC', 'TOBACCO', 'ICU'] - and print the number of unique values found for each column within the covid_data_final_cleaned DataFrame.

In [9]:

```python
from pyspark.sql.functions import col

covid_data_final_cleaned = covid_data_final_cleaned.withColumn(
    'AGE_GROUP',
    F.when(col('AGE') < 20, 'Under 20')
     .when((col('AGE') >= 20) & (col('AGE') < 40), '20-39')
     .when((col('AGE') >= 40) & (col('AGE') < 60), '40-59')
     .when(col('AGE') >= 60, '60 and above')
)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

Then we added an AGE_GROUP column, which categorizes patients based on age ranges for age-based analysis. Specifically, it assigns individuals under 20 to the "Under 20" category, those aged 20-39 to the "20-39" group, those aged 40-59 to the "40-59" category, and anyone aged 60 or above to the "60 and above" group.

In [10]:

```python
# Re-evaluate nulls after feature engineering
covid_data_final_cleaned.select(
    [count(when(col(column).isNull(), column)).alias(column) for column in covid_data_final_clea
).show()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

```
+-----+------------+---+------------+---------+-------+---------+---+--------+--------+----+------
+-------+------------+-------------+--------------+-------+------------+-------+----------------
---+---+---------+
|USMER|MEDICAL_UNIT|SEX|PATIENT_TYPE|DATE_DIED|INTUBED|PNEUMONIA|AGE|PREGNANT|DIABETES|COPD|ASTHMA
|INMSUPR|HIPERTENSION|OTHER_DISEASE|CARDIOVASCULAR|OBESITY|RENAL_CHRONIC|TOBACCO|CLASIFFICATION_FI
NAL|ICU|AGE_GROUP|
+-----+------------+---+------------+---------+-------+---------+---+--------+--------+----+------
+-------+------------+-------------+--------------+-------+------------+-------+----------------
---+---+---------+
|    0|           0|  0|           0|        0|      0|        0|  0|       0|       0|   0|     0
|      0|           0|            0|             0|      0|           0|      0|
0|  0|        0|
+-----+------------+---+------------+---------+-------+---------+---+--------+--------+----+------
+-------+------------+-------------+--------------+-------+------------+-------+----------------
---+---+---------+
```

In [11]:

```python
# StringIndexer for categorical columns
indexers = [StringIndexer(inputCol=col, outputCol=col+"_index").fit(covid_data_final_cleaned) fo
feature_cols = [col+"_index" for col in categorical_cols] + ['AGE', 'AGE_GROUP']

# Assemble features
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

This will create a StringIndexer objects for each categorical column, transforming them into numeric indices. And then we use a VectorAssembler to combine these feature columns into a single feature vector column named features, which converts all variables into a format that machine learning models can process.

In [12]:

```python
# Define categorical columns, including the new AGE_GROUP column
categorical_cols = ['SEX', 'PATIENT_TYPE', 'PNEUMONIA', 'DIABETES', 'COPD', 'ASTHMA',
                    'INMSUPR', 'HIPERTENSION', 'OTHER_DISEASE', 'CARDIOVASCULAR',
                    'OBESITY', 'RENAL_CHRONIC', 'TOBACCO', 'ICU', 'AGE_GROUP']

# Apply StringIndexer to all categorical columns
indexers = [StringIndexer(inputCol=col, outputCol=col+"_index").fit(covid_data_final_cleaned) fo

# Define feature columns for VectorAssembler
feature_cols = [col+"_index" for col in categorical_cols] + ['AGE']  # AGE is numeric, so no nee

# Assemble features
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

In [13]:

```python
# function to evaluate models and print accuracy and F1 score
def evaluate_model(predictions, model_name):
    accuracy = accuracy_evaluator.evaluate(predictions)
    f1_score = f1_evaluator.evaluate(predictions)
    print(f"{model_name} - Accuracy: {accuracy * 100:.2f}%, F1 Score: {f1_score * 100:.2f}%")

# Split data into training and testing sets
train_data, test_data = covid_data_final_cleaned.randomSplit([0.8, 0.2], seed=42)

# Initialize evaluators for accuracy and F1 score
#accuracy_evaluator = MulticlassClassificationEvaluator(labelCol="CLASIFFICATION_FINAL", predict
#f1_evaluator = MulticlassClassificationEvaluator(labelCol="CLASIFFICATION_FINAL", predictionCol=
accuracy_evaluator = BinaryClassificationEvaluator(labelCol="CLASIFFICATION_FINAL", rawPredictio
f1_evaluator = BinaryClassificationEvaluator(labelCol="CLASIFFICATION_FINAL", rawPredictionCol="
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

We then split the data into training (60%) and testing (40%) sets. Then Initialized evaluators for accuracy (using ROC area) and F1 score (using Precision-Recall area) for binary classification. And the evaluate_model function takes predictions and prints the accuracy and F1 score for a given model name.

In [14]:

```
# 1. Random Forest Classifier
rf_classifier = RandomForestClassifier(labelCol="CLASIFFICATION_FINAL", featuresCol="features")
rf_pipeline = Pipeline(stages=indexers + [assembler, rf_classifier])

# Parameter grid for Random Forest
rf_paramGrid = ParamGridBuilder().addGrid(rf_classifier.numTrees, [50, 100]).addGrid(rf_classifi
rf_crossval = CrossValidator(estimator=rf_pipeline, estimatorParamMaps=rf_paramGrid, evaluator=a

# Train and evaluate Random Forest
rf_model = rf_crossval.fit(train_data)
rf_predictions = rf_model.transform(test_data)
evaluate_model(rf_predictions, "Random Forest Classifier")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

```
Exception in thread cell_monitor-14:
Traceback (most recent call last):
  File "/usr/lib64/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/usr/lib64/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in cell_monitor
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in <listcomp>
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
KeyError: 'jobGroup'
```

```
Random Forest Classifier - Accuracy: 58.83%, F1 Score: 67.29%
```

This block of code sets up a RandomForestClassifier pipeline with preprocessing steps. A parameter grid is defined to test different combinations of tree counts (50 and 100) and maximum tree depths (5 and 10) and a CrossValidator with 10 folds is used. The model is then trained on the training data, predictions are made on the test data.

In [15]:

```
# 2. Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(labelCol="CLASIFFICATION_FINAL", featuresCol="features")
dt_pipeline = Pipeline(stages=indexers + [assembler, dt_classifier])

# Parameter grid for Decision Tree
dt_paramGrid = ParamGridBuilder().addGrid(dt_classifier.maxDepth, [5, 10, 15]).build()
dt_crossval = CrossValidator(estimator=dt_pipeline, estimatorParamMaps=dt_paramGrid, evaluator=a

# Train and evaluate Decision Tree
dt_model = dt_crossval.fit(train_data)
dt_predictions = dt_model.transform(test_data)
evaluate_model(dt_predictions, "Decision Tree Classifier")
```

```
VBox()
```

```
Exception in thread cell_monitor-15:
Traceback (most recent call last):
  File "/usr/lib64/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/usr/lib64/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in cell_monitor
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in <listcomp>
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
KeyError: 'jobGroup'
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
Decision Tree Classifier - Accuracy: 59.08%, F1 Score: 67.51%
```

This block of code sets up a DecisionTreeClassifier pipeline. A parameter grid is defined to test different maximum tree depths (5, 10, and 15), and a CrossValidator with 10 folds is used. The model is then trained on the training data, and predictions are made on the test data.

In [16]:

```
# 3. Logistic Regression
lr_classifier = LogisticRegression(labelCol="CLASIFFICATION_FINAL", featuresCol="features", maxI
lr_pipeline = Pipeline(stages=indexers + [assembler, lr_classifier])

# Parameter grid for Logistic Regression
lr_paramGrid = ParamGridBuilder().addGrid(lr_classifier.regParam, [0.01, 0.1, 0.5]).build()
lr_crossval = CrossValidator(estimator=lr_pipeline, estimatorParamMaps=lr_paramGrid, evaluator=a

# Train and evaluate Logistic Regression
lr_model = lr_crossval.fit(train_data)
lr_predictions = lr_model.transform(test_data)
evaluate_model(lr_predictions, "Logistic Regression")
```

```
VBox()
```

```
Exception in thread cell_monitor-16:
Traceback (most recent call last):
  File "/usr/lib64/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/usr/lib64/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in cell_monitor
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in <listcomp>
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
KeyError: 'jobGroup'
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
Logistic Regression - Accuracy: 57.86%, F1 Score: 66.74%
```

This block of code sets up a LogisticRegression pipeline. A parameter grid is defined to test different regularization parameters (0.01, 0.1, and 0.5), and a CrossValidator with 10 folds is used. The model is then trained on the training data, and predictions are made on the test data.

In [17]:

```
# 4. Gradient-Boosted Tree Classifier
gbt_classifier = GBTClassifier(labelCol="CLASIFFICATION_FINAL", featuresCol="features", maxIter=
gbt_pipeline = Pipeline(stages=indexers + [assembler, gbt_classifier])

# Parameter grid for Gradient-Boosted Trees
gbt_paramGrid = ParamGridBuilder().addGrid(gbt_classifier.maxDepth, [5, 10]).addGrid(gbt_classif
gbt_crossval = CrossValidator(estimator=gbt_pipeline, estimatorParamMaps=gbt_paramGrid, evaluato

# Train and evaluate Gradient-Boosted Trees
gbt_model = gbt_crossval.fit(train_data)
gbt_predictions = gbt_model.transform(test_data)
evaluate_model(gbt_predictions, "Gradient-Boosted Tree Classifier")
```

```
VBox()
```

```
Exception in thread cell_monitor-17:
Traceback (most recent call last):
  File "/usr/lib64/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/usr/lib64/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in cell_monitor
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in <listcomp>
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
KeyError: 'jobGroup'
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
Gradient-Boosted Tree Classifier - Accuracy: 58.93%, F1 Score: 67.36%
```

This block of code sets up a GBTClassifier pipeline. A parameter grid is defined to test different maximum tree depths (5 and 10) and iterations (10 and 20), and a CrossValidator with 3 folds is used. The model is then trained on the training data, and predictions are made on the test data

In [18]:

```
model_path = "s3://covid-data-project-final/model/"
dt_model.write().overwrite().save(model_path)
```

```
VBox()
```

```
Exception in thread cell_monitor-18:
Traceback (most recent call last):
  File "/usr/lib64/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/usr/lib64/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in cell_monitor
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
  File "/home/hadoop/spark-magic/spark_monitoring_widget/src/awseditorssparkmonitoringwidget/cellm
onitor.py", line 157, in <listcomp>
    job_group_filtered_jobs = [job for job in jobs_data if job['jobGroup'] == str(statement_id)]
KeyError: 'jobGroup'
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', w
idth='50%'),…
```

This code block saves the model to S3 - s3://covid-data-project-final/model/

In [ ]:

```
# Plotting Histograms
plt.figure(figsize=(15, 10))
for i, col in enumerate(numeric_cols):
    plt.subplot(3, 2, i + 1)
    sns.histplot(covid_pd[col], kde=True)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```

In [ ]:

```
# Check correlations
plt.figure(figsize=(10, 8))
correlation_matrix = covid_pd[numeric_cols].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```