```
!pip install pyspark
!pip install tensorflowonspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.1
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tensorflowonspark in /usr/local/lib/python3.10/dist-packa
```

```
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.context import SparkContext
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
import tensorflowonspark as tfos
from tensorflowonspark import TFCluster
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import VectorSlicer
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.context import SparkContext
from pyspark.sql.functions import count, when, isnull
import tensorflow as tf
import tensorflowonspark as tfos
import pyspark.sql.functions as F
from pyspark.sql.functions import col
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```
# Create SparkSession
spark = SparkSession.builder \
    .appName("MalwareDetection") \
    .getOrCreate()

# Access SparkContext from SparkSession
sc = spark.sparkContext
```

```
# Load the dataset into a Spark DataFrame
dataset_path = "Android Malware Detection.csv"
df = spark.read.csv(dataset_path, header=True, inferSchema=True)
```

```
# Display the schema of the DataFrame
df.printSchema()
```

```
 |-- Label: double (nullable = true)
```

```
#display first 20 rows
df.show()
```

```
+---+--------------------+----------------------+-------------------------+------------
|_c0|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_PROPERTIES|ACCESS_COARS
+---+--------------------+----------------------+-------------------------+------------
|  0|                 0.0|                   0.0|                      0.0|
|  1|                 0.0|                   0.0|                      0.0|
|  2|                 0.0|                   0.0|                      0.0|
|  3|                 0.0|                   0.0|                      0.0|
|  4|                 0.0|                   0.0|                      0.0|
|  5|                 0.0|                   0.0|                      0.0|
|  6|                 0.0|                   0.0|                      0.0|
|  7|                 0.0|                   0.0|                      0.0|
|  8|                 0.0|                   0.0|                      0.0|
|  9|                 0.0|                   0.0|                      0.0|
| 10|                 0.0|                   0.0|                      0.0|
| 11|                 0.0|                   0.0|                      0.0|
| 12|                 0.0|                   0.0|                      0.0|
| 13|                 0.0|                   0.0|                      0.0|
| 14|                 0.0|                   0.0|                      0.0|
| 15|                 0.0|                   0.0|                      0.0|
| 16|                 0.0|                   0.0|                      0.0|
| 17|                 0.0|                   0.0|                      0.0|
| 18|                 0.0|                   0.0|                      0.0|
| 19|                 0.0|                   0.0|                      0.0|
+---+--------------------+----------------------+-------------------------+------------
only showing top 20 rows
```

```
# Count the number of rows in the dataset
row_count = df.count()
print("Number of rows in the dataset:", row_count)
```

```
Number of rows in the dataset: 4863
```

```
# Summary statistics
print("Summary Statistics:")
df.describe().show()
```

```
Summary Statistics:
+-------+------------------+--------------------+----------------------+--------------
|summary|               _c0|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_
+-------+------------------+--------------------+----------------------+--------------
|  count|              4863|                4862|                  4862|
|   mean|            2431.0|8.227067050596463E-4|   0.001234060057589...|     0.00452488
| stddev|1403.9715096824436|0.028674012029843828|    0.03511111945893242|      0.0671218
|    min|                 0|                 0.0|                   0.0|
|    max|              4862|                 1.0|                   1.0|
```

```
+-------+-----------------+-------------------+----------------------+--------------
```

```
# checking the shape of the dataframe
num_rows = df.count() # Counting the number of rows.
num_columns =len(df.columns) # Length of columns.
print(f"Shape: ({num_rows}, {num_columns})") # Prints the shape of the dataset.
```

```
Shape: (4863, 150)
```

```
# checking for missing data
missing_data = df.agg(*[count(when(isnull(c), c)).alias(c) for c in df.columns])
missing_data.show()
```

```
+---+--------------------+----------------------+-------------------------+------------
|_c0|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_PROPERTIES|ACCESS_COARS
+---+--------------------+----------------------+-------------------------+------------
|  0|                   1|                     1|                        1|
+---+--------------------+----------------------+-------------------------+------------
```

```
# Handling missing values
df_cleaned = df.na.drop()
# Show the cleaned DataFrame
df_cleaned.show()
```

```
+---+--------------------+----------------------+-------------------------+------------
|_c0|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_PROPERTIES|ACCESS_COARS
+---+--------------------+----------------------+-------------------------+------------
|  0|                 0.0|                   0.0|                      0.0|
|  1|                 0.0|                   0.0|                      0.0|
|  2|                 0.0|                   0.0|                      0.0|
|  3|                 0.0|                   0.0|                      0.0|
|  4|                 0.0|                   0.0|                      0.0|
|  5|                 0.0|                   0.0|                      0.0|
|  6|                 0.0|                   0.0|                      0.0|
|  7|                 0.0|                   0.0|                      0.0|
|  8|                 0.0|                   0.0|                      0.0|
|  9|                 0.0|                   0.0|                      0.0|
| 10|                 0.0|                   0.0|                      0.0|
| 11|                 0.0|                   0.0|                      0.0|
| 12|                 0.0|                   0.0|                      0.0|
| 13|                 0.0|                   0.0|                      0.0|
| 14|                 0.0|                   0.0|                      0.0|
| 15|                 0.0|                   0.0|                      0.0|
| 16|                 0.0|                   0.0|                      0.0|
| 17|                 0.0|                   0.0|                      0.0|
| 18|                 0.0|                   0.0|                      0.0|
| 19|                 0.0|                   0.0|                      0.0|
+---+--------------------+----------------------+-------------------------+------------
```

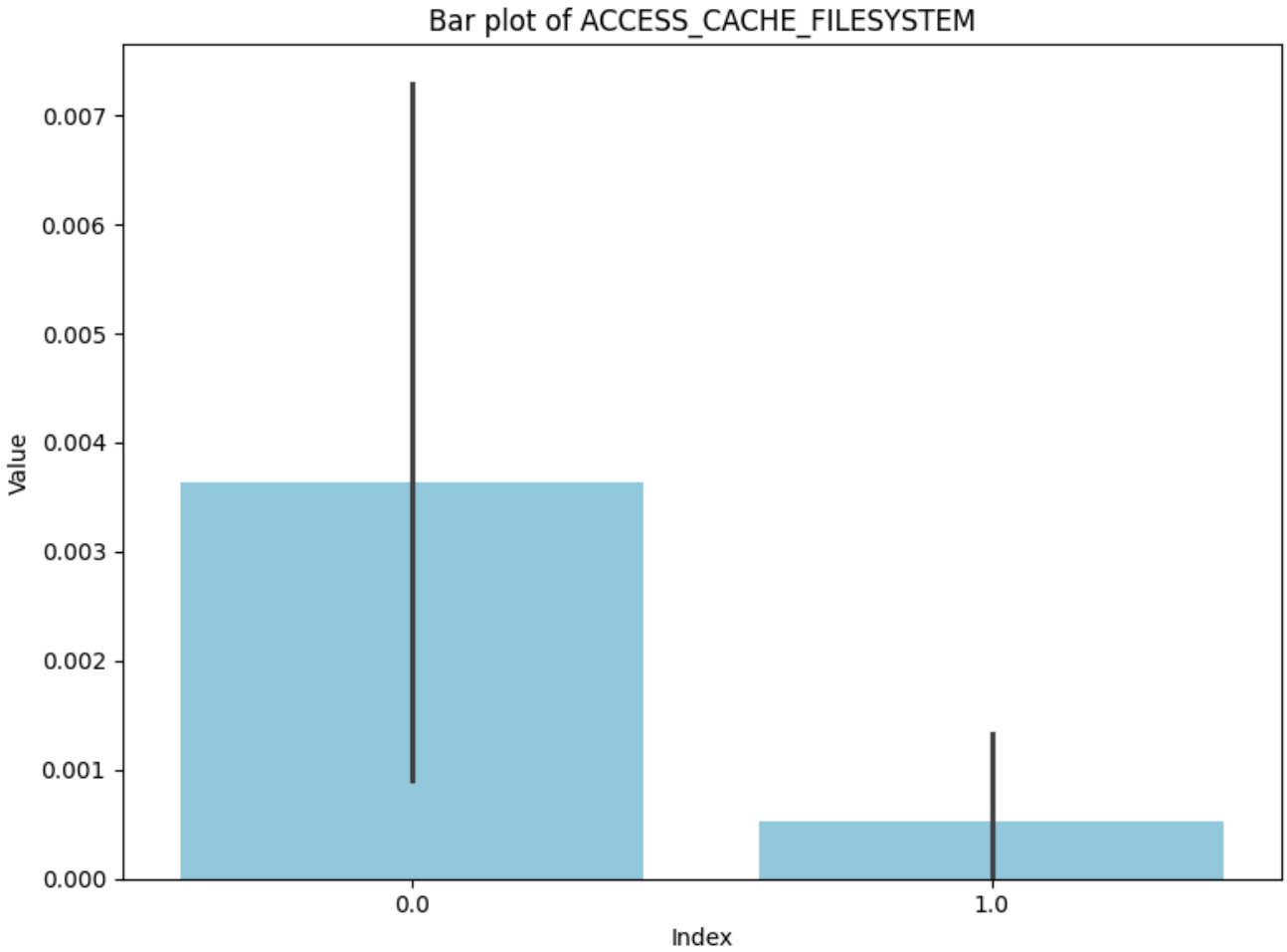
```
only showing top 20 rows
```

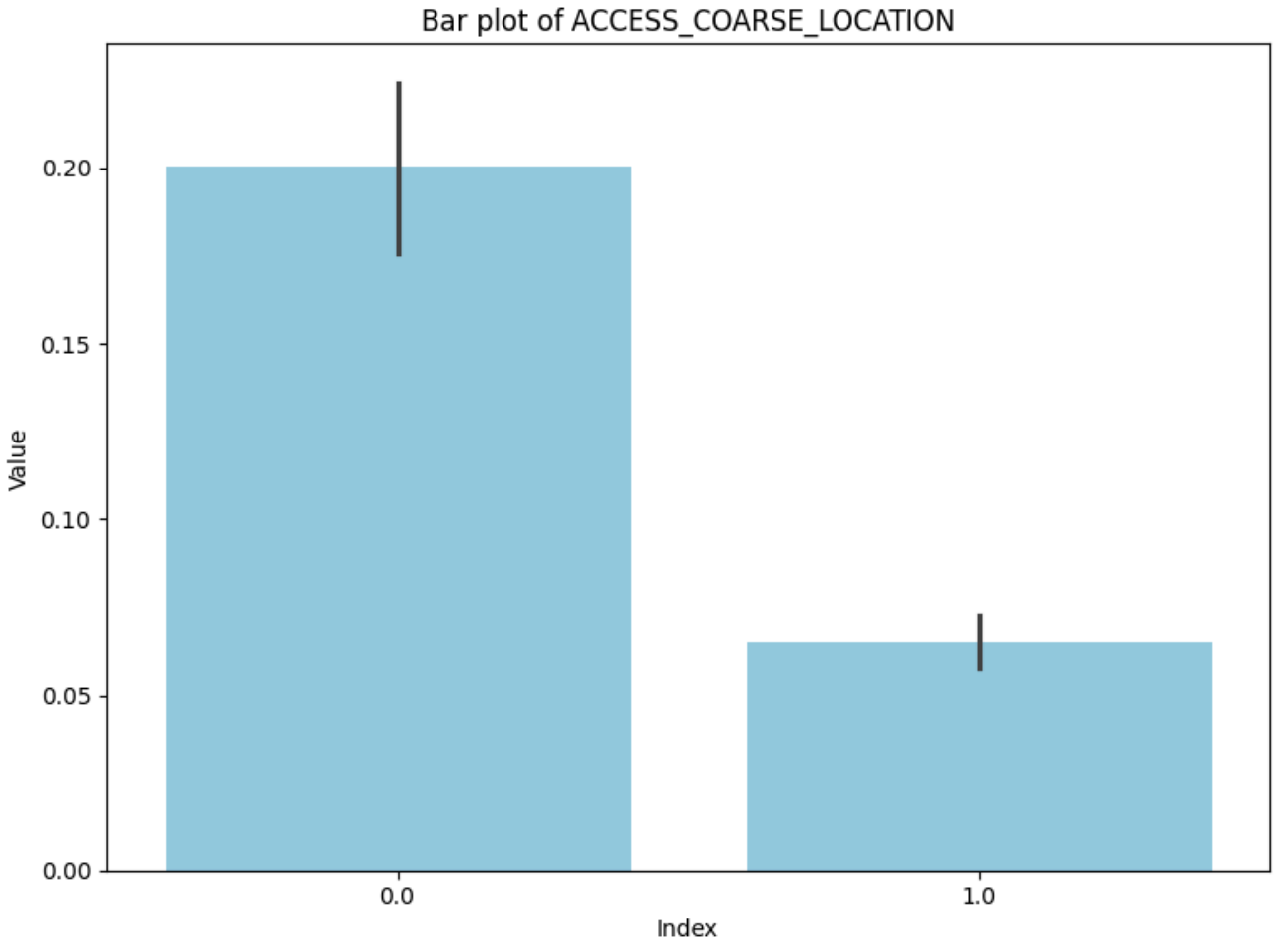
```
# drop unnecessary columns
df = df_cleaned.drop('_c0')
```

```
# display new df
df.show()
```

```
+--------------------+----------------------+-------------------------+------------------
|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_PROPERTIES|ACCESS_COARSE_LC
+--------------------+----------------------+-------------------------+------------------
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
|                 0.0|                   0.0|                      0.0|
+--------------------+----------------------+-------------------------+------------------
only showing top 20 rows
```

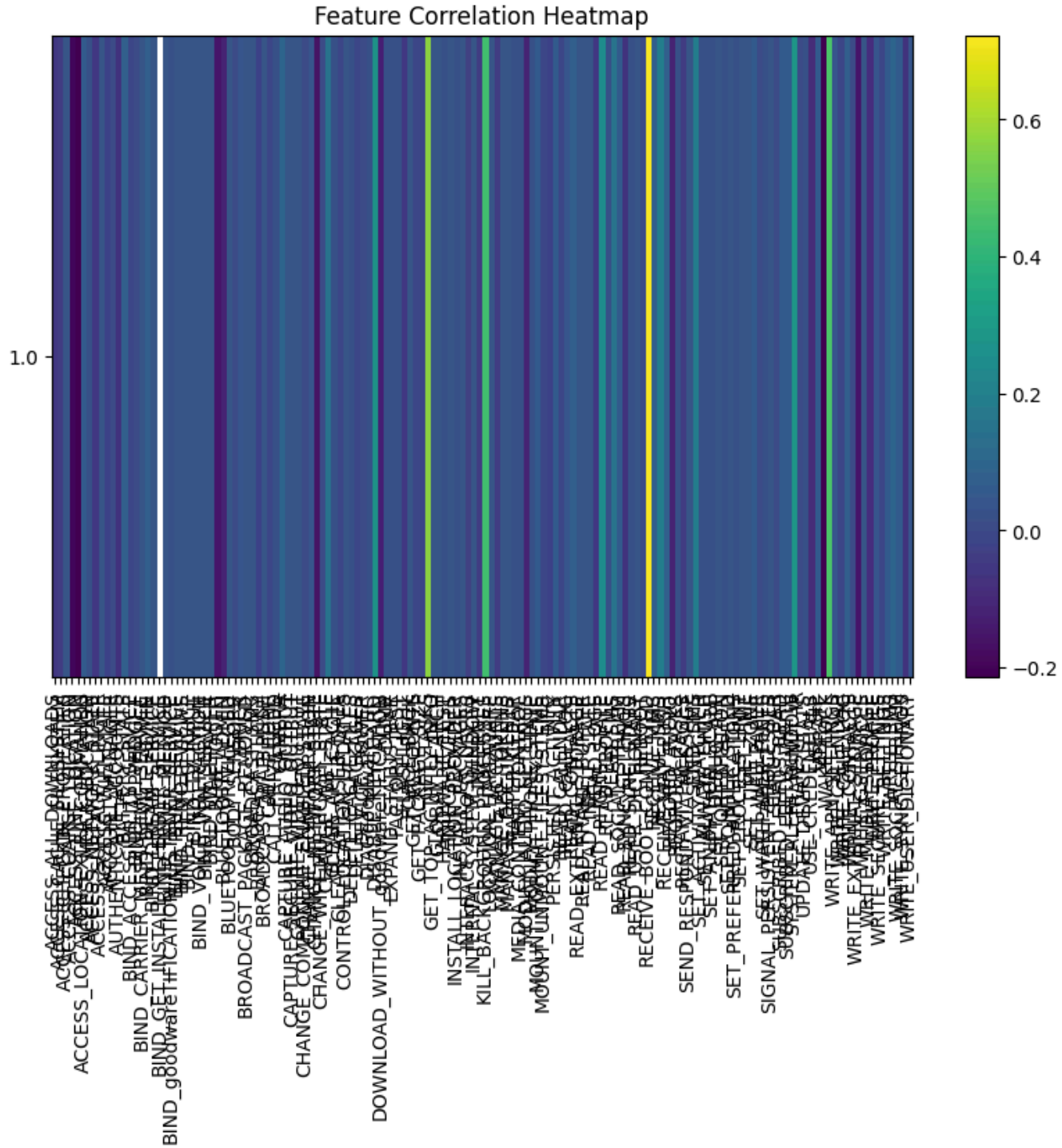
```
# Plot bar plots for selected features
selected_features = ["ACCESS_ALL_DOWNLOADS", "ACCESS_CACHE_FILESYSTEM", "ACCESS_COARSE_LOCAT
for feature in selected_features:
    plt.figure(figsize=(8, 6))
    sns.barplot(x=df.select('Label').rdd.flatMap(lambda x: x).collect(), y=df.select(feature
    plt.title(f'Bar plot of {feature}')
    plt.xlabel('Index')
    plt.ylabel('Value')
    plt.tight_layout()
    plt.show()
```


Bar plot of ACCESS_CACHE_FILESYSTEM
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Value
0.0
1.0
Index


Bar plot of ACCESS_COARSE_LOCATION
0.20
0.15
0.10
0.05
0.00
Value
0.0
1.0
Index

```
# Feature correlations
print("Feature Correlations:")
correlation_matrix = df.select([F.corr(col, 'Label').alias(col) for col in df.columns]).toPa
plt.figure(figsize=(10, 6))
plt.imshow(correlation_matrix, cmap='viridis', interpolation='nearest', aspect='auto')
plt.colorbar()
plt.xticks(range(len(correlation_matrix.columns)), correlation_matrix.columns, rotation='ver
plt.yticks(range(len(correlation_matrix.index)), correlation_matrix.index)
plt.title('Feature Correlation Heatmap')
plt.show()
```

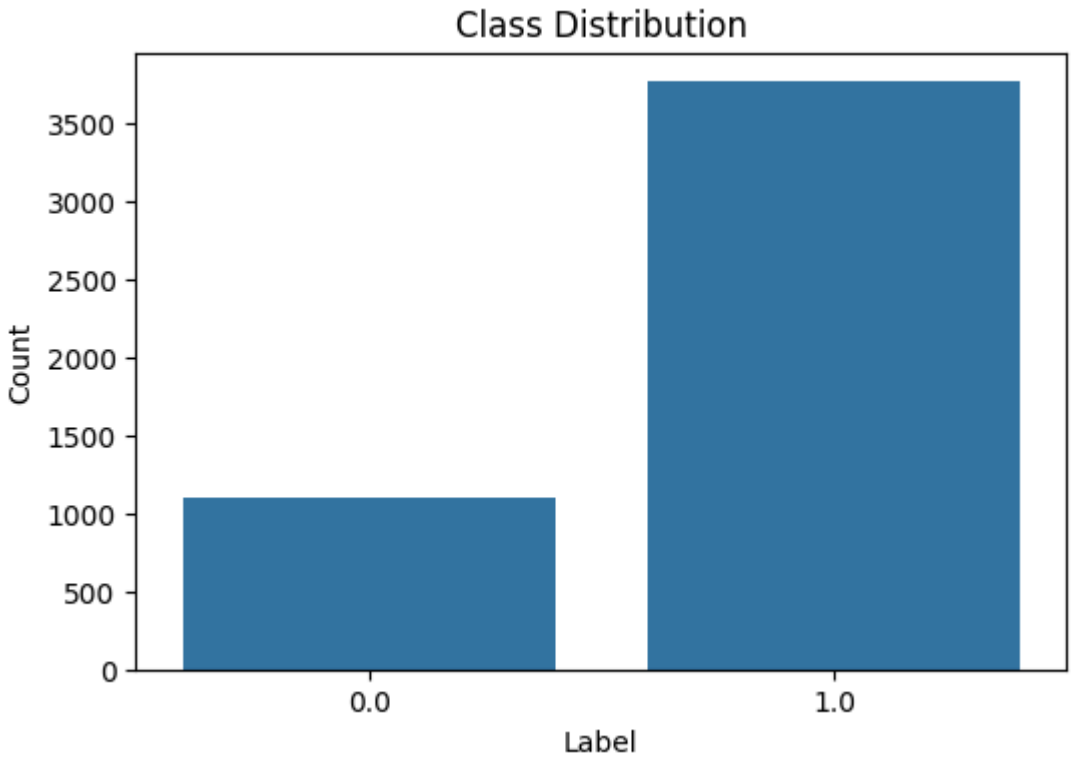
Feature Correlations:



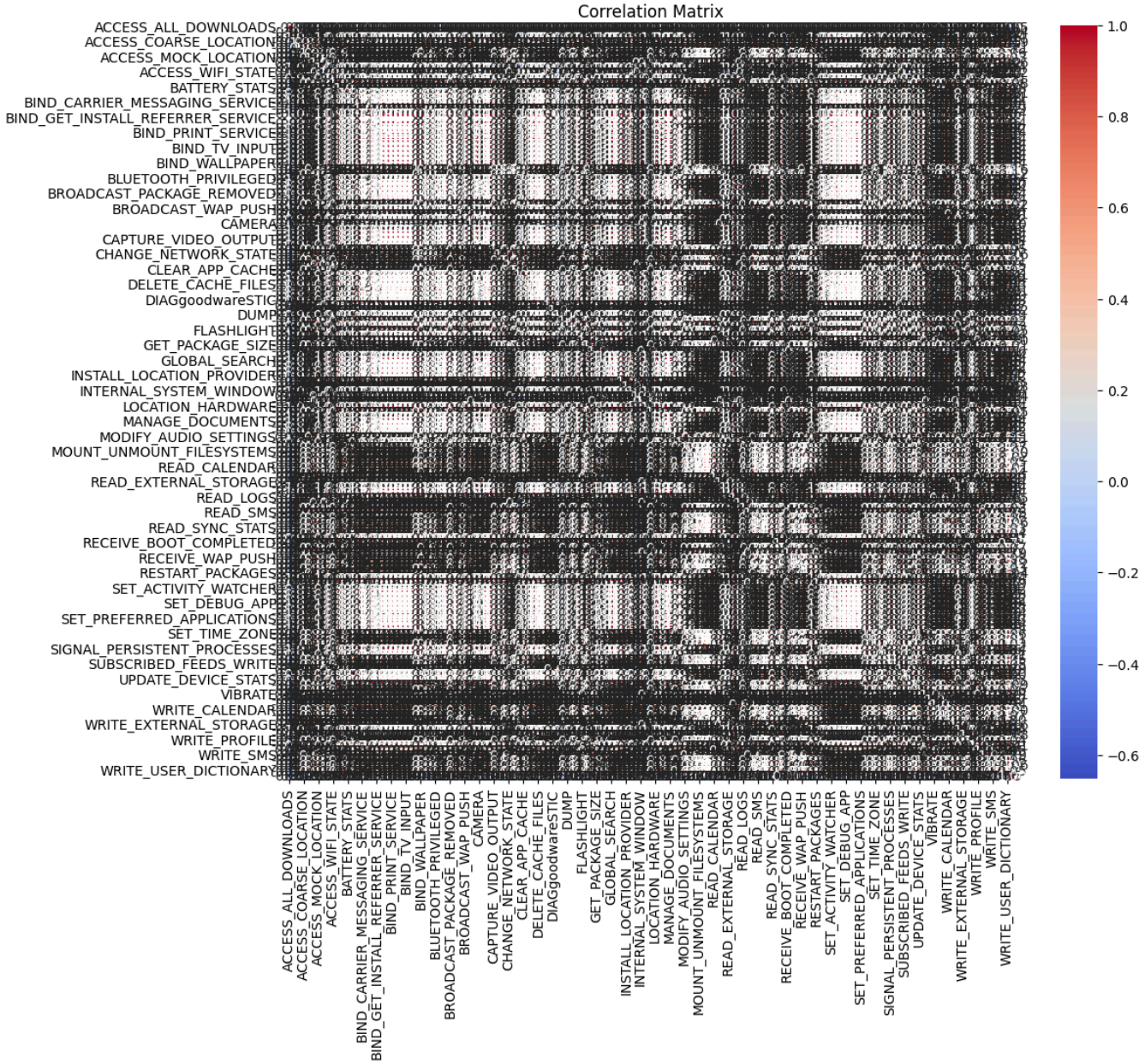

```
# Check the class distribution
class_distribution = df.groupBy("Label").count().orderBy("Label")
class_distribution.show()
```

```
+-----+-----+
|Label|count|
+-----+-----+
|  0.0| 1098|
|  1.0| 3764|
+-----+-----+
```

```
# Visualize the class distribution
class_distribution_pd = class_distribution.toPandas()
plt.figure(figsize=(6, 4))
sns.barplot(x="Label", y="count", data=class_distribution_pd)
plt.title("Class Distribution")
plt.xlabel("Label")
plt.ylabel("Count")
plt.show()
```




```
# Correlation Analysis
correlation_matrix = df.drop("_c0").toPandas().corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

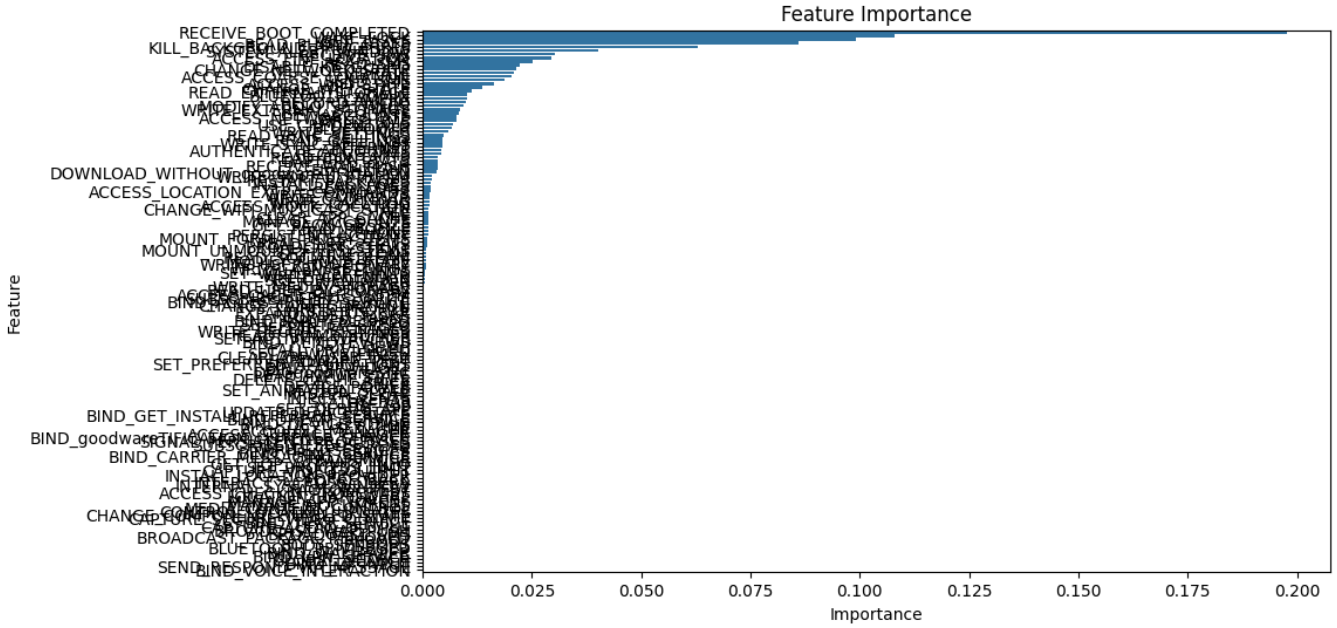

Correlation Matrix
ACCESS_ALL_DOWNLOADS
ACCESS_COARSE_LOCATION
ACCESS_MOCK_LOCATION
ACCESS_WIFI_STATE
BATTERY_STATS
BIND_CARRIER_MESSAGING_SERVICE
BIND_GET_INSTALL_REFERRER_SERVICE
BIND_PRINT_SERVICE
BIND_TV_INPUT
BIND_WALLPAPER
BLUETOOTH_PRIVILEGED
BROADCAST_PACKAGE_REMOVED
BROADCAST_WAP_PUSH
CAMERA
CAPTURE_VIDEO_OUTPUT
CHANGE_NETWORK_STATE
CLEAR_APP_CACHE
DELETE_CACHE_FILES
DIAGgoodwareSTIC
DUMP
FLASHLIGHT
GET_PACKAGE_SIZE
GLOBAL_SEARCH
INSTALL_LOCATION_PROVIDER
INTERNAL_SYSTEM_WINDOW
LOCATION_HARDWARE
MANAGE_DOCUMENTS
MODIFY_AUDIO_SETTINGS
MOUNT_UNMOUNT_FILESYSTEMS
READ_CALENDAR
READ_EXTERNAL_STORAGE
READ_LOGS
READ_SMS
READ_SYNC_STATS
RECEIVE_BOOT_COMPLETED
RECEIVE_WAP_PUSH
RESTART_PACKAGES
SET_ACTIVITY_WATCHER
SET_DEBUG_APP
SET_PREFERRED_APPLICATIONS
SET_TIME_ZONE
SIGNAL_PERSISTENT_PROCESSES
SUBSCRIBED_FEEDS_WRITE
UPDATE_DEVICE_STATS
VIBRATE
WRITE_CALENDAR
WRITE_EXTERNAL_STORAGE
WRITE_PROFILE
WRITE_SMS
WRITE_USER_DICTIONARY
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
# Assemble features
feature_cols = df.columns[1:-1]  # Exclude _c0 and Label columns
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data_assembled = assembler.transform(df)
```

```
# Train a RandomForestClassifier
rf = RandomForestClassifier(labelCol="Label", featuresCol="features", numTrees=100)
model = rf.fit(data_assembled)
```

```
# Get feature importances
feature_importances = model.featureImportances.toArray()
```

```
# Visualize feature importances
feature_importance_pd = pd.DataFrame({"Feature": feature_cols, "Importance": feature_importa
feature_importance_pd_sorted = feature_importance_pd.sort_values(by="Importance", ascending=
plt.figure(figsize=(10, 6))
sns.barplot(x="Importance", y="Feature", data=feature_importance_pd_sorted)
plt.title("Feature Importance")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```




```
# Get the indices of the top 20 features based on their importances
top_20_indices = feature_importances.argsort()[-20:][::-1]

# Select only the top 20 features from the feature vector
slicer = VectorSlicer(inputCol="features", outputCol="selected_features", indices=top_20_inc
data_selected = slicer.transform(data_assembled)

# Train the RandomForestClassifier model using only the top 20 features
rf_selected = RandomForestClassifier(labelCol="Label", featuresCol="selected_features", numT
model_selected = rf_selected.fit(data_selected)
```

```
# Split the data into training and testing sets
train_data_selected, test_data_selected = data_selected.randomSplit([0.8, 0.2], seed=42)
```