```
!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 (from pyspark) (0.10.9.7)
     Requirement already satisfied: tensorflowonspark in /usr/local/lib/python3.10/dist-packages (2.2.5)
     Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflowonspark) (24.0)
     Requirement already satisfied: setuptools>38.0 in /usr/local/lib/python3.10/dist-packages (from tensorflowonspark) (67.7.2)
```

```
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.context import SparkContext
import tensorflow as tf
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
from pyspark.sql.functions import pandas_udf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
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 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()
```

```
 |-- SET_PROCESS_LIMIT: double (nullable = true)
 |-- SET_TIME: double (nullable = true)
 |-- SET_TIME_ZONE: double (nullable = true)
 |-- SET_WALLPAPER: double (nullable = true)
 |-- SET_WALLPAPER_HINTS: double (nullable = true)
 |-- SIGNAL_PERSISTENT_PROCESSES: double (nullable = true)
 |-- STATUS_BAR: double (nullable = true)
 |-- SUBSCRIBED_FEEDS_READ: double (nullable = true)
 |-- SUBSCRIBED_FEEDS_WRITE: double (nullable = true)
 |-- SYSTEM_ALERT_WINDOW: double (nullable = true)
 |-- TRANSMIT_IR: double (nullable = true)
 |-- UPDATE_DEVICE_STATS: double (nullable = true)
 |-- USE_CREDENTIALS: double (nullable = true)
 |-- USE_SIP: double (nullable = true)
 |-- VIBRATE: double (nullable = true)
 |-- WAKE_LOCK: double (nullable = true)
 |-- WRITE_APN_SETTINGS: double (nullable = true)
 |-- WRITE_CALENDAR: double (nullable = true)
 |-- WRITE_CALL_LOG: double (nullable = true)
 |-- WRITE_CONTACTS: double (nullable = true)
 |-- WRITE_EXTERNAL_STORAGE: double (nullable = true)
 |-- WRITE_GSERVICES: double (nullable = true)
 |-- WRITE_MEDIA_STORAGE: double (nullable = true)
 |-- WRITE_PROFILE: double (nullable = true)
 |-- WRITE_SECURE_SETTINGS: double (nullable = true)
 |-- WRITE_SETTINGS: double (nullable = true)
 |-- WRITE_SMS: double (nullable = true)
 |-- WRITE_SOCIAL_STREAM: double (nullable = true)
 |-- WRITE_SYNC_SETTINGS: double (nullable = true)
 |-- WRITE_USER_DICTIONARY: double (nullable = true)
 |-- Label: double (nullable = true)
```

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

```
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
|_c0|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_PROPERTIES|ACCESS_COARSE_LOCATION|ACCESS_FINE_LOCATION|ACCESS_LOCATION_
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
|  0|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  1|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  2|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  3|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  4|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  5|                 0.0|                   0.0|                     0.0|                   1.0|                 0.0|
|  6|                 0.0|                   0.0|                     0.0|                   0.0|                 1.0|
|  7|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  8|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  9|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 10|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 11|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 12|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 13|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 14|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 15|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 16|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 17|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 18|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 19|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
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_PROPERTIES|ACCESS_COARSE_LOCATION|ACCESS_FINE_LC
+-------+------------------+--------------------+----------------------+------------------------+----------------------+-------------
|  count|              4863|                4862|                  4862|                    4862|                  4862|
|   mean|            2431.0|8.227067050596463E-4|   0.001234060057589...|     0.004524886877828055|   0.09563965446318387| 0.09954751131
| stddev|1403.9715096824436|0.028674012029843828|    0.03511111945893242|      0.06712182148679634|   0.29412668044620227| 0.29942652604
|    min|                 0|                 0.0|                    0.0|                       0.0|                     0.0|
|    max|              4862|                 1.0|                    1.0|                       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_COARSE_LOCATION|ACCESS_FINE_LOCATION|ACCESS_LOCATION_
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
|  0|                   1|                     1|                       1|                     1|                   1|
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
```

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

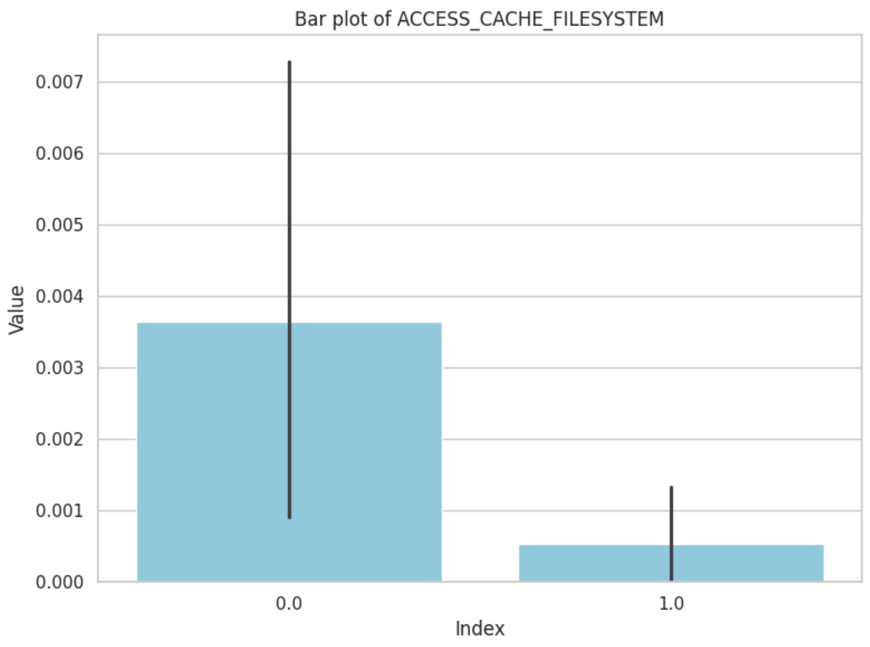
```
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
|_c0|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_PROPERTIES|ACCESS_COARSE_LOCATION|ACCESS_FINE_LOCATION|ACCESS_LOCATION_
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
|  0|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  1|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  2|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  3|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  4|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  5|                 0.0|                   0.0|                     0.0|                   1.0|                 0.0|
|  6|                 0.0|                   0.0|                     0.0|                   0.0|                 1.0|
|  7|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  8|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
|  9|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 10|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 11|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 12|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 13|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 14|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
| 15|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 16|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 17|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 18|                 0.0|                   0.0|                     0.0|                   0.0|                 0.0|
| 19|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
+---+--------------------+----------------------+------------------------+----------------------+--------------------+-----------------
only showing top 20 rows
```

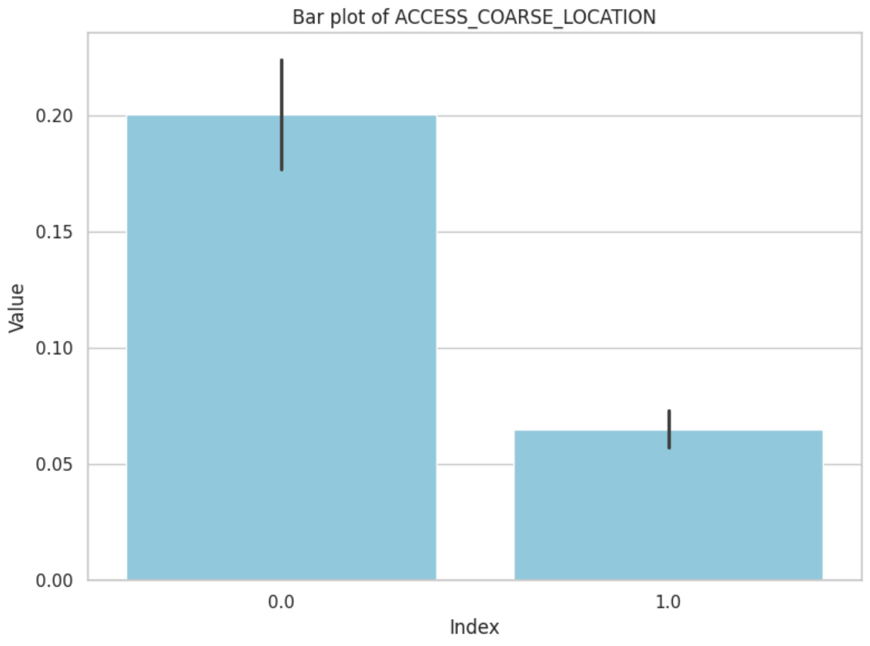
```
# drop unnecessary columns
df = df_cleaned.drop('_c0')
```

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

```
+--------------------+----------------------+------------------------+----------------------+--------------------+--------------------
|ACCESS_ALL_DOWNLOADS|ACCESS_CACHE_FILESYSTEM|ACCESS_CHECKIN_PROPERTIES|ACCESS_COARSE_LOCATION|ACCESS_FINE_LOCATION|ACCESS_LOCATION_EXTR
+--------------------+----------------------+------------------------+----------------------+--------------------+--------------------
|                 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|                   1.0|                 0.0|
|                 0.0|                   0.0|                     0.0|                   0.0|                 1.0|
|                 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|                   1.0|                 1.0|
|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
|                 0.0|                   0.0|                     0.0|                   1.0|                 1.0|
|                 0.0|                    0.0|                        0.0|                      0.0|                    0.0|
|                 0.0|                    0.0|                        0.0|                      1.0|                    1.0|
|                 0.0|                    0.0|                        0.0|                      1.0|                    1.0|
|                 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|                      1.0|                    1.0|
+--------------------+-----------------------+---------------------------+-------------------------+----------------------+---------------------
only showing top 20 rows
```

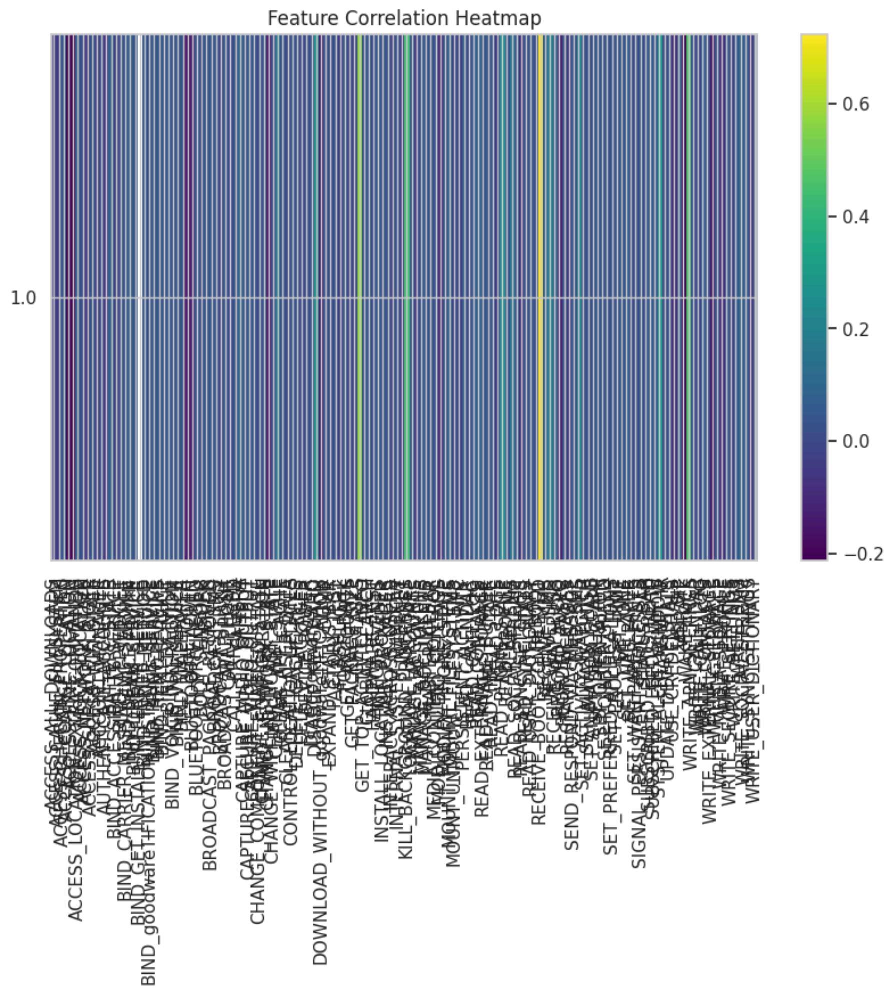
```
# Plot bar plots for selected features
selected_features = ["ACCESS_ALL_DOWNLOADS", "ACCESS_CACHE_FILESYSTEM", "ACCESS_COARSE_LOCATION"]
for feature in selected_features:
    plt.figure(figsize=(8, 6))
    sns.barplot(x=df.select('Label').rdd.flatMap(lambda x: x).collect(), y=df.select(feature).rdd.flatMap(lambda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


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


Bar plot of ACCESS_COARSE_LOCATION
Value
0.00
0.05
0.10
0.15
0.20
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]).toPandas().set_index('Label')
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='vertical')
plt.yticks(range(len(correlation_matrix.index)), correlation_matrix.index)
plt.title('Feature Correlation Heatmap')
plt.show()
```

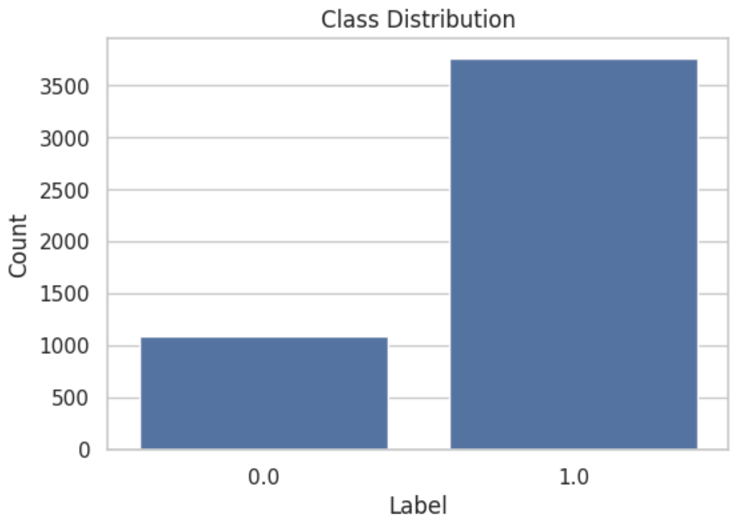
```
Feature Correlations:
```



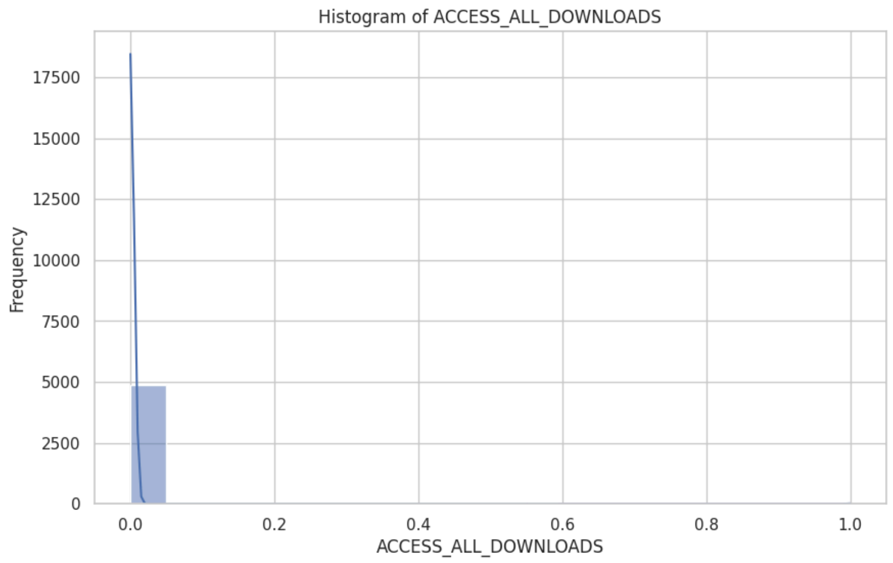
```
# Check the class distribution
class_distribution = df.groupBy("Label").count().orderBy("Label")
class_distribution.show()
```

```
+-----+-----+
|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()
```

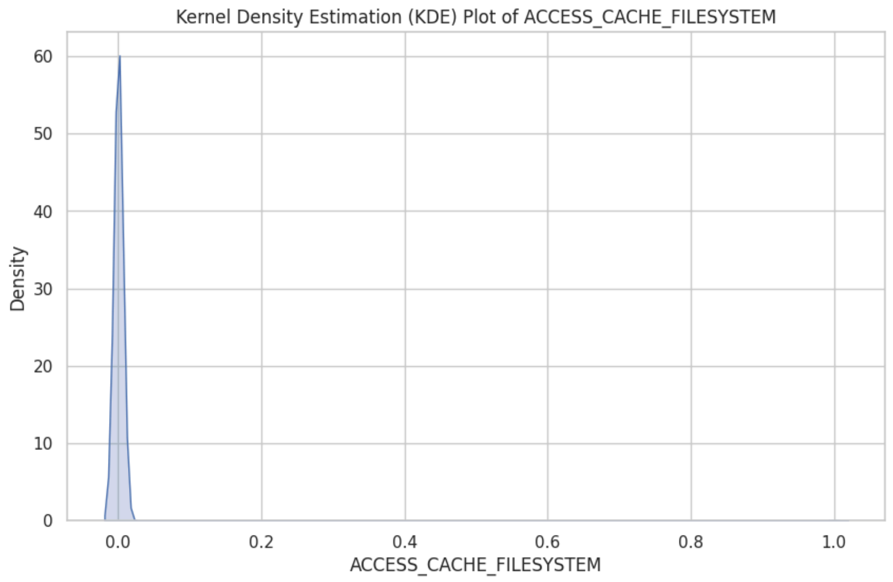


```
# Histograms
plt.figure(figsize=(10, 6))
sns.histplot(df.select('ACCESS_ALL_DOWNLOADS').rdd.flatMap(lambda x: x).collect(), bins=20, kde=True)
plt.title('Histogram of ACCESS_ALL_DOWNLOADS')
plt.xlabel('ACCESS_ALL_DOWNLOADS')
plt.ylabel('Frequency')
plt.show()
```

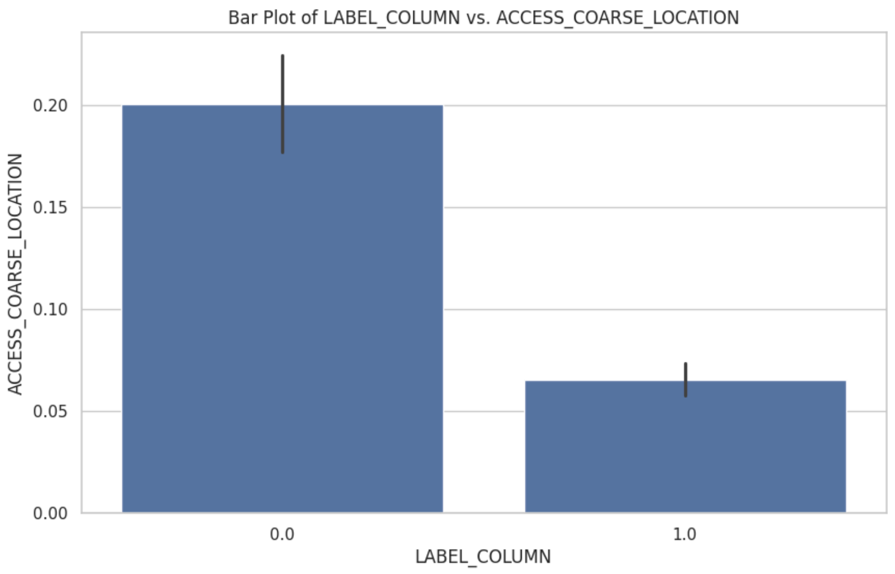


```
# KDE Plots
plt.figure(figsize=(10, 6))
sns.kdeplot(df.select('ACCESS_CACHE_FILESYSTEM').rdd.flatMap(lambda x: x).collect(), shade=True)
plt.title('Kernel Density Estimation (KDE) Plot of ACCESS_CACHE_FILESYSTEM')
plt.xlabel('ACCESS_CACHE_FILESYSTEM')
plt.ylabel('Density')
plt.show()
```

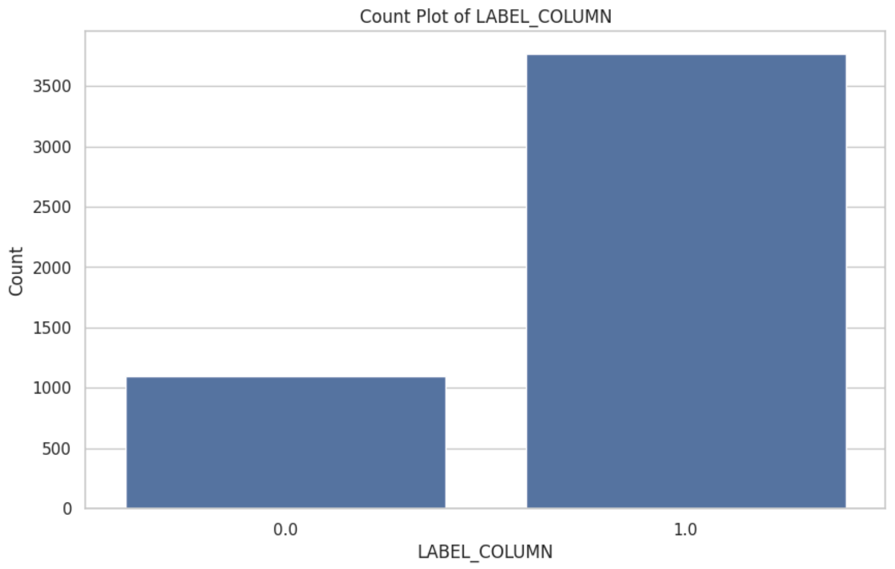
```
<ipython-input-142-6ad600ba46de>:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df.select('ACCESS_CACHE_FILESYSTEM').rdd.flatMap(lambda x: x).collect(), s
```



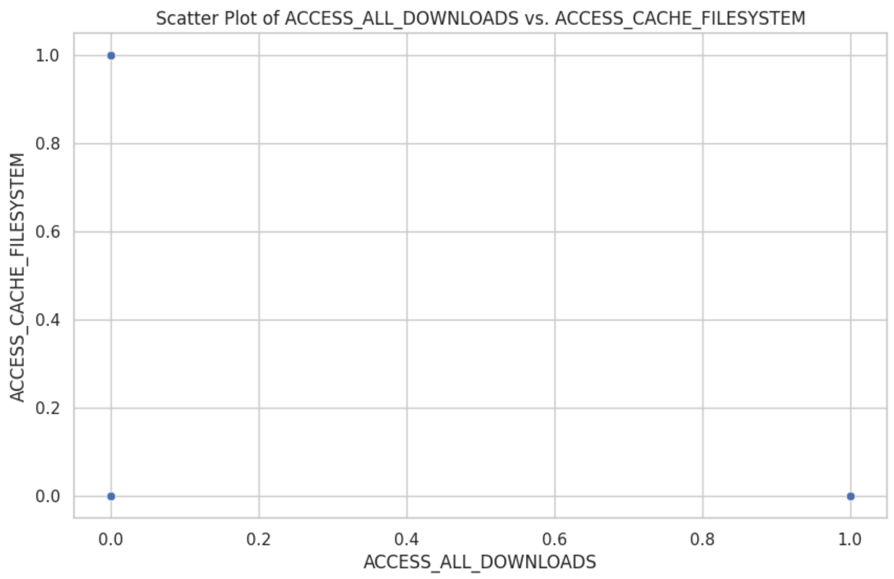
```
# Bar Plots
plt.figure(figsize=(10, 6))
sns.barplot(x='Label', y='ACCESS_COARSE_LOCATION', data=df.toPandas())
plt.title('Bar Plot of LABEL_COLUMN vs. ACCESS_COARSE_LOCATION')
plt.xlabel('LABEL_COLUMN')
plt.ylabel('ACCESS_COARSE_LOCATION')
plt.show()
```



```
# Count Plots
plt.figure(figsize=(10, 6))
sns.countplot(x='Label', data=df.toPandas())
plt.title('Count Plot of LABEL_COLUMN')
plt.xlabel('LABEL_COLUMN')
plt.ylabel('Count')
plt.show()
```

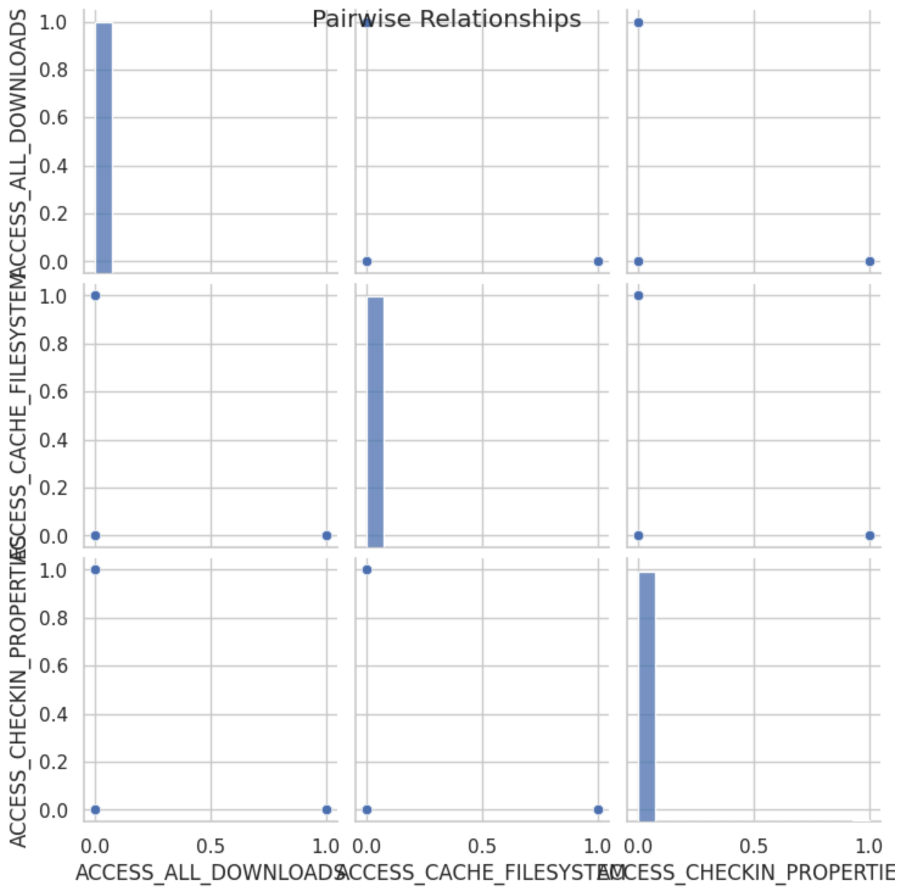


```
# Scatter Plots
plt.figure(figsize=(10, 6))
sns.scatterplot(x='ACCESS_ALL_DOWNLOADS', y='ACCESS_CACHE_FILESYSTEM', data=df.toPandas())
plt.title('Scatter Plot of ACCESS_ALL_DOWNLOADS vs. ACCESS_CACHE_FILESYSTEM')
plt.xlabel('ACCESS_ALL_DOWNLOADS')
plt.ylabel('ACCESS_CACHE_FILESYSTEM')
plt.show()
```

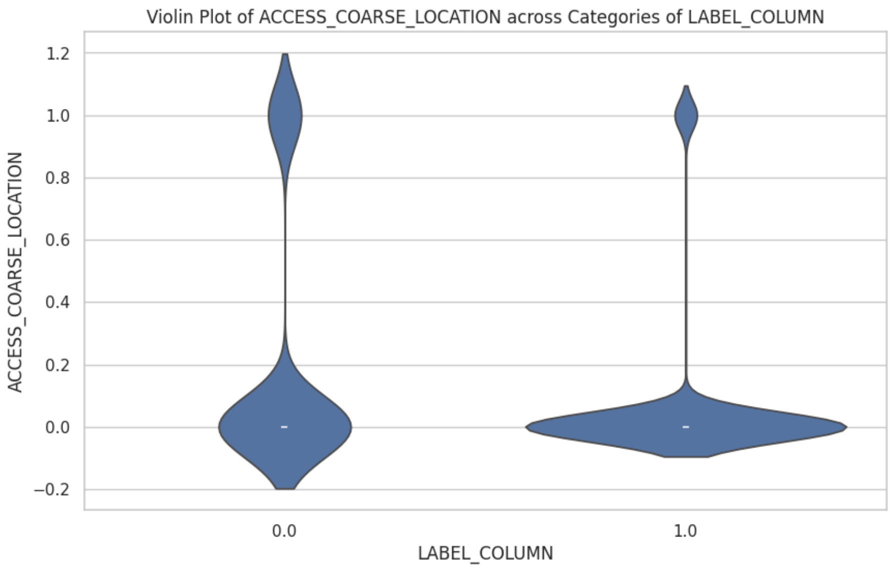


```
# Pair Plots
plt.figure(figsize=(10, 6))
sns.pairplot(df.toPandas()[['ACCESS_ALL_DOWNLOADS', 'ACCESS_CACHE_FILESYSTEM', 'ACCESS_CHECKIN_PROPERTIES']])
plt.suptitle('Pairwise Relationships')
plt.show()
```

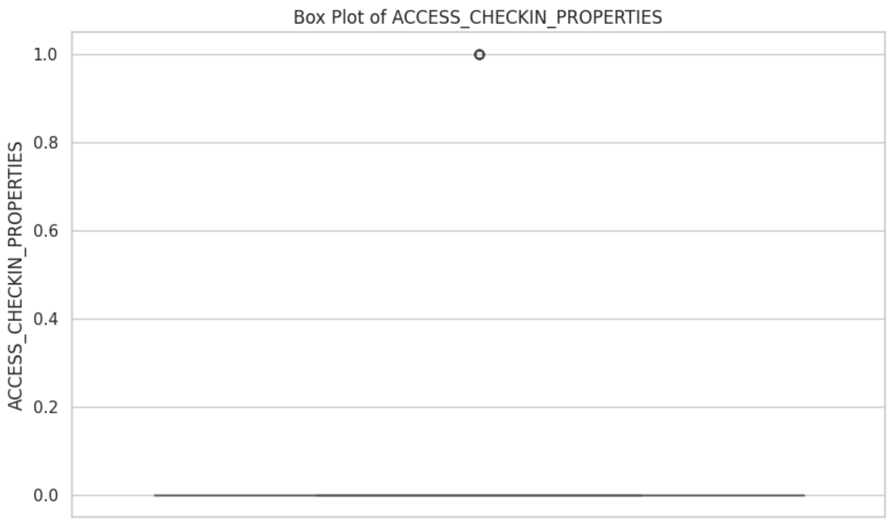
```
<Figure size 1000x600 with 0 Axes>
```



```
# Violin Plots
plt.figure(figsize=(10, 6))
sns.violinplot(y='ACCESS_COARSE_LOCATION', x='Label', data=df.toPandas())
plt.title('Violin Plot of ACCESS_COARSE_LOCATION across Categories of LABEL_COLUMN')
plt.xlabel('LABEL_COLUMN')
plt.ylabel('ACCESS_COARSE_LOCATION')
plt.show()
```

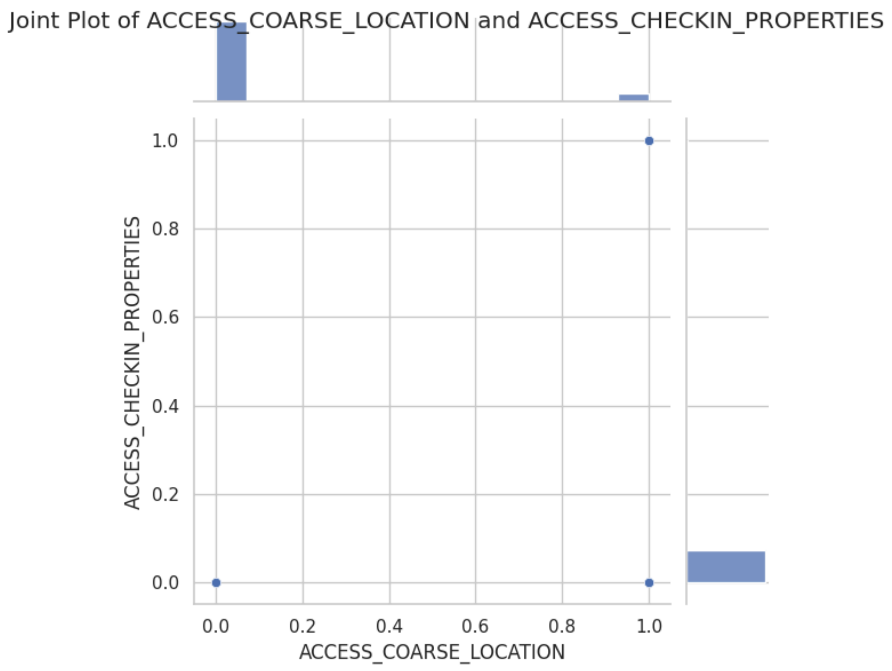


```
# Box Plots
plt.figure(figsize=(10, 6))
sns.boxplot(y=df.select('ACCESS_CHECKIN_PROPERTIES').rdd.flatMap(lambda x: x).collect())
plt.title('Box Plot of ACCESS_CHECKIN_PROPERTIES')
plt.ylabel('ACCESS_CHECKIN_PROPERTIES')
plt.show()
```

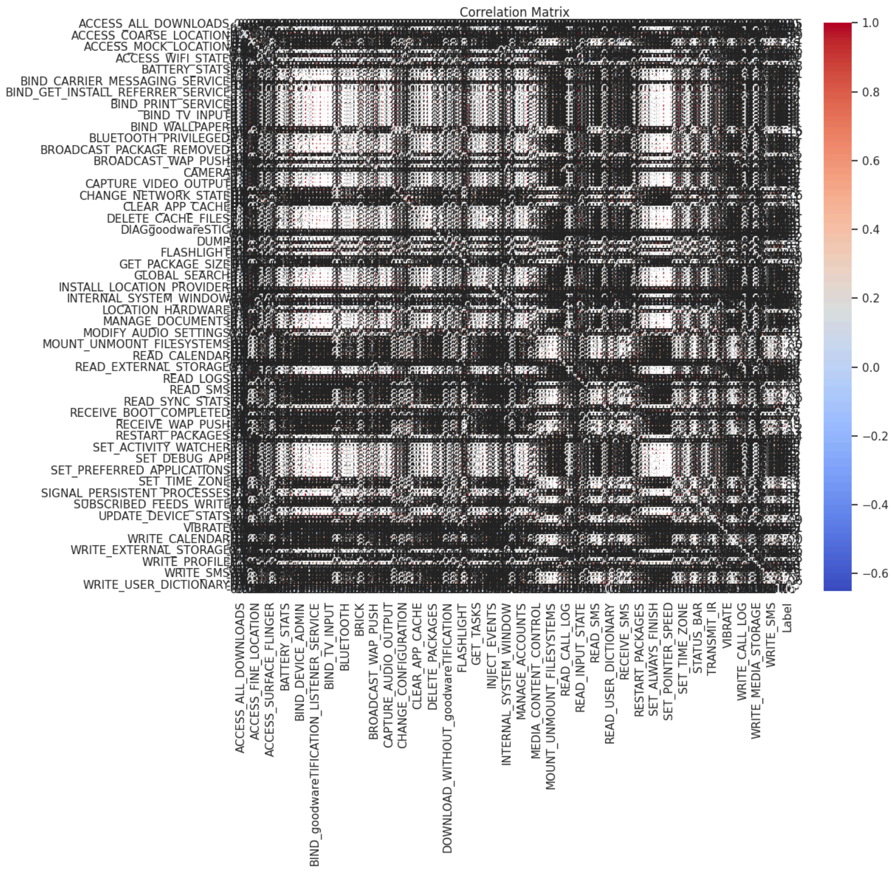


```
# Joint Plots
plt.figure(figsize=(10, 6))
sns.jointplot(x='ACCESS_COARSE_LOCATION', y='ACCESS_CHECKIN_PROPERTIES', data=df.toPandas(), kind='scatter')
plt.suptitle('Joint Plot of ACCESS_COARSE_LOCATION and ACCESS_CHECKIN_PROPERTIES')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



```
# 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()
```

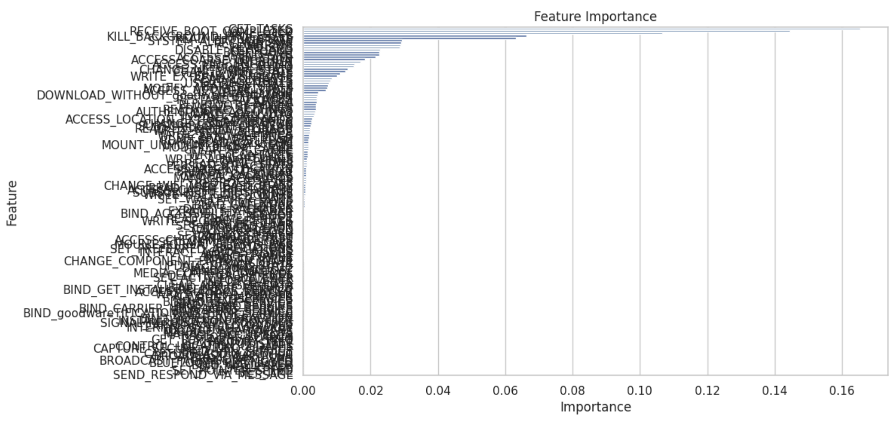


```
# Assemble features
feature_cols = df.columns[1:-1]  # Exclude _c0 and Label columns
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data_assembled = assembler.transform(df)
```

```
# 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_importances})
feature_importance_pd_sorted = feature_importance_pd.sort_values(by="Importance", ascending=False)
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_indices)
data_selected = slicer.transform(data_assembled)

# Train the RandomForestClassifier model using only the top 20 features
rf_selected = RandomForestClassifier(labelCol="Label", featuresCol="selected_features", numTrees=100)
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)
```

```
# Evaluate the model's performance on the testing set
evaluator = BinaryClassificationEvaluator(labelCol="Label")
test_predictions = model_selected.transform(test_data_selected)
auc = evaluator.evaluate(test_predictions)
print("AUC on the testing set:", auc)

# Calculate accuracy
accuracy = test_predictions.filter("prediction == Label").count() / test_predictions.count()
print("Accuracy on the testing set:", accuracy)

# Calculate precision
precision = test_predictions.filter("prediction == 1 AND Label == 1").count() / test_predictions.filter("prediction == 1").count()
print("Precision on the testing set:", precision)

# Calculate recall
recall = test_predictions.filter("prediction == 1 AND Label == 1").count() / test_predictions.filter("Label == 1").count()
print("Recall on the testing set:", recall)
```

```
AUC on the testing set: 0.9897319762728887
Accuracy on the testing set: 0.9688506981740065
```