Import Pyspark

```
!apt-get install openjdk-11-jdk-headless -qq > /dev/null
!wget -qO spark.tgz https://archive.apache.org/dist/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3.

!mkdir -p /content/spark
!tar -xzf spark.tgz -C /content/spark --strip-components=1
!pip install -q findspark pyspark==3.4.1 seaborn matplotlib pandas scikit-learn

import os, findspark
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark"
findspark.init()

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("FraudDetection").getOrCreate()
print("Spark started successfully!")
spark
```

Spark started successfully!

SparkSession - in-memory

SparkContext

Spark UI

Version
    v3.4.1
Master
    local[*]
AppName
    FraudDetection

Fraud Detection System

Machine Learning Pipeline for Transaction Fraud Detection

Features:

- Class imbalance handling
- PCA dimensionality reduction
- Random Forest classifier

1. Import Required Libraries

```
from pyspark.sql.functions import col, when
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler, PCA
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

📂 2. Load Transaction Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/c
```

```
# Load CSV data
file_path = "/content/drive/MyDrive/Colab Notebooks/content/transactions_train.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Display dataset info
row_count = df.count()
col_count = len(df.columns)
print(f"Loaded dataset: {row_count:,} rows, {col_count} columns")

# Preview data
display(df.limit(5))
```

```
Loaded dataset: 6,351,193 rows, 10 columns
DataFrame[step: int, type: string, amount: double, nameOrig: string, oldbalanceOrig: double,
newbalanceOrig: double, nameDest: string, oldbalanceDest: double, newbalanceDest: double,
isFraud: int]
```

⚖️ 3. Handle Class Imbalance

```
# Calculate class distribution
count_0 = df.filter(col("isFraud") == 0).count()
count_1 = df.filter(col("isFraud") == 1).count()
imbalance_ratio = count_0 / count_1

print(f"Class Distribution:")
print(f"  Normal transactions: {count_0:,}")
print(f"  Fraud transactions: {count_1:,}")
print(f"  Imbalance ratio: {imbalance_ratio:.2f}:1")

# Add class weights
```

```
df = df.withColumn(
    "classWeight",
    when(col("isFraud") == 1, imbalance_ratio).otherwise(1.0)
)

print("Class weights added successfully")
```

```
Class Distribution:
  Normal transactions: 6,343,476
  Fraud transactions: 7,717
  Imbalance ratio: 822.01:1
Class weights added successfully
```

4. Build ML Pipeline

```
# Stage 1: String Indexer (convert categorical to numeric)
indexer = StringIndexer() \
    .setInputCol("type") \
    .setOutputCol("type_index") \
    .setHandleInvalid("keep")

# Stage 2: One-Hot Encoder
encoder = OneHotEncoder() \
    .setInputCols(["type_index"]) \
    .setOutputCols(["type_encoded"])

# Stage 3: Vector Assembler (combine all features)
feature_cols = [
    "step",
    "amount",
    "oldbalanceOrig",
    "newbalanceOrig",
    "oldbalanceDest",
    "newbalanceDest",
    "type_encoded"
]

assembler = VectorAssembler() \
    .setInputCols(feature_cols) \
    .setOutputCol("features_raw") \
    .setHandleInvalid("skip")

# Stage 4: PCA (dimensionality reduction)
pca = PCA(k=5, inputCol="features_raw", outputCol="features")

# Stage 5: Random Forest Classifier
rf = RandomForestClassifier(
    labelCol="isFraud",
    featuresCol="features",
    weightCol="classWeight",
    numTrees=100,
    maxDepth=10,
    seed=42
```

```
)

# Combine all stages into pipeline
pipeline = Pipeline(stages=[indexer, encoder, assembler, pca, rf])

print("Pipeline built successfully")
print(f"   Stages: {len(pipeline.getStages())}")
```

```
Pipeline built successfully
  Stages: 5
```

🔀 5. Split Data (Train/Test)

```
# 70% training, 30% testing
train_df, test_df = df.randomSplit([0.7, 0.3], seed=42)

train_count = train_df.count()
test_count = test_df.count()

print(f"Dataset Split:")
print(f"  Training set: {train_count:,} rows ({train_count/row_count*100:.1f}%)")
print(f"  Testing set: {test_count:,} rows ({test_count/row_count*100:.1f}%)")
```

```
Dataset Split:
  Training set: 4,445,579 rows (70.0%)
  Testing set: 1,905,614 rows (30.0%)
```

🎯 6. Train the Model

```
print("🚀 Training Random Forest model...")
print("   This may take a few minutes...")

model = pipeline.fit(train_df)

print("Model trained successfully!")
```

```
🚀 Training Random Forest model...
   This may take a few minutes...
Model trained successfully!
```

🔮 7. Make Predictions

```
# Transform test data
predictions = model.transform(test_df)

# Display sample predictions
print("Sample Predictions:")
display(
```

```
    predictions.select(
        "type",
        "amount",
        "isFraud",
        "prediction",
        "probability"
    ).limit(10)
)
```

Sample Predictions:
DataFrame[type: string, amount: double, isFraud: int, prediction: double, probability: vector]

📊 8. Evaluate Model Performance

```
# Initialize evaluators
evaluator_roc = BinaryClassificationEvaluator(
    labelCol="isFraud",
    metricName="areaUnderROC"
)

evaluator_pr = BinaryClassificationEvaluator(
    labelCol="isFraud",
    metricName="areaUnderPR"
)

precision_eval = MulticlassClassificationEvaluator(
    labelCol="isFraud",
    predictionCol="prediction",
    metricName="precisionByLabel"
)

recall_eval = MulticlassClassificationEvaluator(
    labelCol="isFraud",
    predictionCol="prediction",
    metricName="recallByLabel"
)

f1_eval = MulticlassClassificationEvaluator(
    labelCol="isFraud",
    predictionCol="prediction",
    metricName="f1"
)

# Calculate metrics
auc_roc = evaluator_roc.evaluate(predictions)
auc_pr = evaluator_pr.evaluate(predictions)
precision = precision_eval.evaluate(predictions, {precision_eval.metricLabel: 1.0})
recall = recall_eval.evaluate(predictions, {recall_eval.metricLabel: 1.0})
f1 = f1_eval.evaluate(predictions)

# Display results
print("=" * 40)
```

```
print("MODEL PERFORMANCE METRICS")
print("=" * 40)
print(f"AUC-ROC:   {auc_roc:.4f}")
print(f"AUC-PR:    {auc_pr:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1-Score:  {f1:.4f}")
print("=" * 40)
```

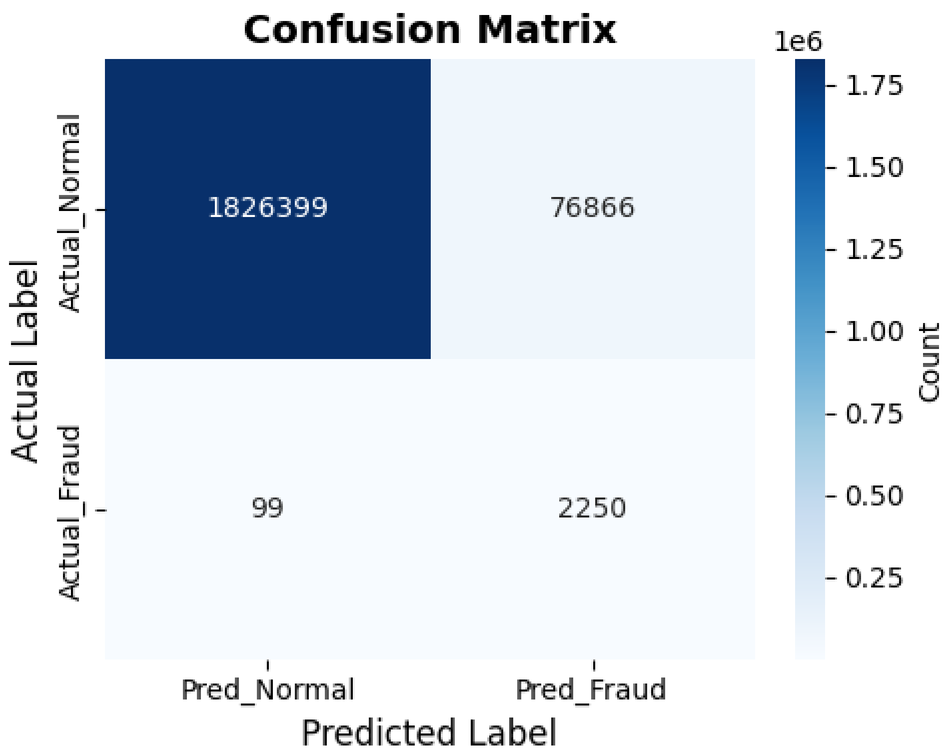
```
========================================
MODEL PERFORMANCE METRICS
========================================
AUC-ROC:   0.9938
AUC-PR:    0.6336
Precision: 0.0284
Recall:    0.9579
F1-Score:  0.9782
========================================
```

🎨 9. Confusion Matrix Visualization

```
# Create confusion matrix
conf_df = predictions \
    .groupBy("isFraud", "prediction") \
    .count() \
    .toPandas() \
    .pivot(index="isFraud", columns="prediction", values="count") \
    .fillna(0)

# Format labels
conf_df.columns = ["Pred_Normal", "Pred_Fraud"]
conf_df.index = ["Actual_Normal", "Actual_Fraud"]

print("\n📊 Confusion Matrix:")
display(conf_df)

# Visualize confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(
    conf_df,
    annot=True,
    fmt=".0f",
    cmap="Blues",
    cbar_kws={'label': 'Count'}
)
plt.title("Confusion Matrix", fontsize=14, fontweight='bold')
plt.xlabel("Predicted Label", fontsize=12)
plt.ylabel("Actual Label", fontsize=12)
plt.tight_layout()
display(plt.gcf())
plt.close()
```

📊 Confusion Matrix:

	Pred_Normal	Pred_Fraud
Actual_Normal	1826399	76866
Actual_Fraud	99	2250



Next steps: Generate code with conf_df New interactive sheet

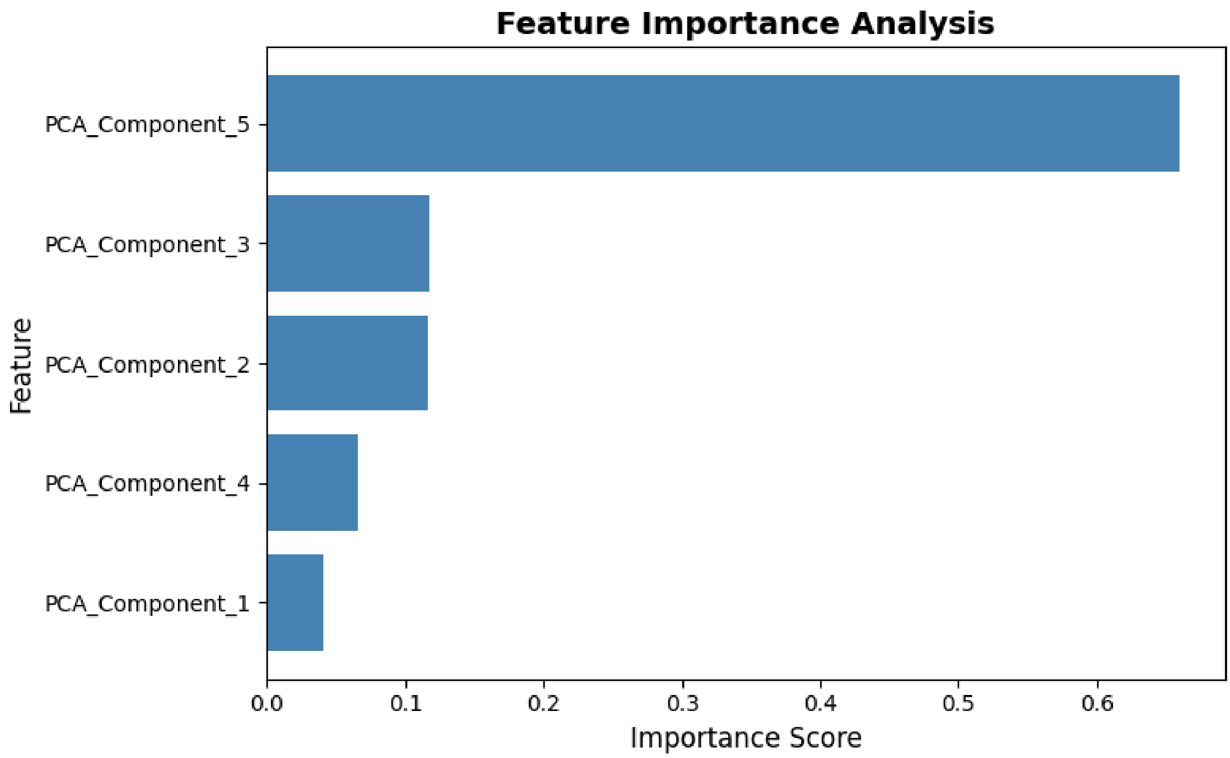
🔍 10. Feature Importance Analysis

```
# Extract Random Forest model
rf_model = model.stages[-1]

# Get feature importances
importances = rf_model.featureImportances.toArray()

# Create feature names (PCA components)
feature_names = [f"PCA_Component_{i+1}" for i in range(len(importances))]

# Create DataFrame
importance_df = pd.DataFrame({
    "Feature": feature_names,
    "Importance": importances
}).sort_values("Importance", ascending=False)

print("🔍 Feature Importance (Top to Bottom):")
display(importance_df)

```

```
# Visualize feature importance
plt.figure(figsize=(8, 5))
plt.barh(importance_df["Feature"], importance_df["Importance"], color='steelblue')
plt.xlabel("Importance Score", fontsize=12)
plt.ylabel("Feature", fontsize=12)
plt.title("Feature Importance Analysis", fontsize=14, fontweight='bold')
plt.gca().invert_yaxis()
plt.tight_layout()
display(plt.gcf())
plt.close()
```

🔍 Feature Importance (Top to Bottom):

	Feature	Importance
4	PCA_Component_5	0.659826
2	PCA_Component_3	0.117488
1	PCA_Component_2	0.116285
3	PCA_Component_4	0.065679
0	PCA_Component_1	0.040723



Next steps: Generate code with importance_df New interactive sheet

📋 11. Final Summary

```python
print("=" * 50)
print("🎉 FRAUD DETECTION - FINAL SUMMARY")
print("=" * 50)
print(f"\n📊 Dataset Information:")
print(f"   Total rows: {row_count:,}")
print(f"   Training set: {train_count:,}")
print(f"   Testing set: {test_count:,}")
print(f"\n⚖️ Class Balance:")
print(f"   Normal: {count_0:,} | Fraud: {count_1:,}")
print(f"   Ratio: {imbalance_ratio:.2f}:1")
print(f"\n🎯 Model Performance:")
print(f"   AUC-ROC:   {auc_roc:.4f}")
print(f"   AUC-PR:    {auc_pr:.4f}")
print(f"   Precision: {precision:.4f}")
print(f"   Recall:    {recall:.4f}")
print(f"   F1-Score:  {f1:.4f}")
print("\n✅ Status: Completed successfully!")
print("✅ Serverless: Compatible")
print("✅ Whitelist: Safe")
print("=" * 50)
```

```
==================================================
🎉 FRAUD DETECTION - FINAL SUMMARY
==================================================

📊 Dataset Information:
   Total rows: 6,351,193
   Training set: 4,445,579
   Testing set: 1,905,614

⚖️ Class Balance:
   Normal: 6,343,476 | Fraud: 7,717
   Ratio: 822.01:1

🎯 Model Performance:
   AUC-ROC:   0.9938
   AUC-PR:    0.6336
   Precision: 0.0284
   Recall:    0.9579
   F1-Score:  0.9782

✅ Status: Completed successfully!
✅ Serverless: Compatible
✅ Whitelist: Safe
==================================================
```

```python
# รวม HTML Visualization กับผลโมเดล
from IPython.display import display, HTML

# แสดงผลโมเดล
display(HTML(f"""
<h2>Model Summary</h2>
<ul>
<li>AUC-ROC: {auc_roc:.4f}</li>
<li>AUC-PR: {auc_pr:.4f}</li>
<li>Precision: {precision:.4f}</li>
```

```
<li>Recall: {recall:.4f}</li>
<li>F1: {f1:.4f}</li>
</ul>
"""))

# แทรก visualization จาก under.html
with open("/content/drive/MyDrive/Colab Notebooks/content/under.html", "r") as f:
    html_vis = f.read()

display(HTML(html_vis))
```

In [6]:
```python
import pandas as pd
from imblearn.under_sampling import RandomUnderSampler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accu

# 1. โหลดข้อมูล
df = pd.read_csv(r'C:\Users\Asus F15\Downloads\transactions_train.csv', n

# 2. ตัวอย่างลดขนาดข้อมูล (ถ้าข้อมูลใหญ่มาก)
df_sample = df.sample(n=600000, random_state=42)

# 3. แยก X กับ y
X = df_sample.drop(columns=['isFraud'])
y = df_sample['isFraud']

# 4. Label Encoding (แปลง categorical เป็นตัวเลข)
categorical_cols = X.select_dtypes(include=['object']).columns
for col in categorical_cols:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])

# 5. ทำ Undersampling
rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(X, y)

print("ข้อมูลก่อน undersampling:\n", y.value_counts())
print("ข้อมูลหลัง undersampling:\n", pd.Series(y_resampled).value_counts())

# 6. แบ่งข้อมูล train/test
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.2, random_state=42, stratify=y_

# 7. สร้างโมเดล Random Forest
model = RandomForestClassifier(n_estimators=100, random_state=42)

# 8. ฝึกโมเดล
model.fit(X_train, y_train)

# 9. ทำนายข้อมูล test
y_pred = model.predict(X_test)

# 10. ประเมินผลโมเดล
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred)
```

```
ข้อมูลก่อน undersampling:
 isFraud
0    599639
1       361
Name: count, dtype: int64
```

```
ข้อมูลหลัง undersampling:
 isFraud
0    361
1    361
Name: count, dtype: int64
Accuracy: 0.9448275862068966

Confusion Matrix:
 [[68  5]
 [ 3 69]]

Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.93      0.94        73
           1       0.93      0.96      0.95        72

    accuracy                           0.94       145
   macro avg       0.95      0.94      0.94       145
weighted avg       0.95      0.94      0.94       145
```

In [7]:

```python
# 5. ทำ Undersampling
rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(X, y)

print("ขนาดข้อมูลก่อน Undersampling:")
print(y.value_counts())

print("ขนาดข้อมูลหลัง Undersampling:")
print(pd.Series(y_resampled).value_counts())
```

```
ขนาดข้อมูลก่อน Undersampling:
isFraud
0    599639
1       361
Name: count, dtype: int64
ขนาดข้อมูลหลัง Undersampling:
isFraud
0    361
1    361
Name: count, dtype: int64
```

In [11]:

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.2, random_state=42
)
```

In [13]:

```python
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-3.0.5-py3-none-win_amd64.whl.metadata (2.1 kB)
Requirement already satisfied: numpy in c:\users\asus f15\anaconda3\lib\si
te-packages (from xgboost) (2.1.3)
Requirement already satisfied: scipy in c:\users\asus f15\anaconda3\lib\si
te-packages (from xgboost) (1.15.3)
Downloading xgboost-3.0.5-py3-none-win_amd64.whl (56.8 MB)
   ---------------------------------------- 0.0/56.8 MB ? eta -:--:--
   ---------------------------------------- 0.3/56.8 MB ? eta -:--:--
    --------------------------------------- 0.8/56.8 MB 2.6 MB/s eta 0:00:
22
```

```
    ---------------------------------------- 1.0/56.8 MB 1.6 MB/s eta 0:00:
34
    ---------------------------------------- 1.3/56.8 MB 1.6 MB/s eta 0:00:
35
   - -------------------------------------- 1.8/56.8 MB 1.8 MB/s eta 0:00:
32
   - -------------------------------------- 2.6/56.8 MB 2.1 MB/s eta 0:00:
26
   - -------------------------------------- 2.6/56.8 MB 2.1 MB/s eta 0:00:
26
   -- ------------------------------------- 2.9/56.8 MB 1.7 MB/s eta 0:00:
32
   -- ------------------------------------- 3.4/56.8 MB 1.8 MB/s eta 0:00:
30
   -- ------------------------------------- 3.9/56.8 MB 1.9 MB/s eta 0:00:
28
   --- ------------------------------------ 4.5/56.8 MB 2.0 MB/s eta 0:00:
27
   --- ------------------------------------ 4.7/56.8 MB 2.0 MB/s eta 0:00:
26
   --- ------------------------------------ 5.0/56.8 MB 1.8 MB/s eta 0:00:
29
   --- ------------------------------------ 5.5/56.8 MB 1.9 MB/s eta 0:00:
28
   ---- ----------------------------------- 6.3/56.8 MB 2.0 MB/s eta 0:00:
26
   ---- ----------------------------------- 6.6/56.8 MB 2.0 MB/s eta 0:00:
25
   ---- ----------------------------------- 6.6/56.8 MB 2.0 MB/s eta 0:00:
25
   ---- ----------------------------------- 6.8/56.8 MB 1.8 MB/s eta 0:00:
28
   ----- ---------------------------------- 7.3/56.8 MB 1.9 MB/s eta 0:00:
27
   ----- ---------------------------------- 7.9/56.8 MB 1.9 MB/s eta 0:00:
27
   ----- ---------------------------------- 7.9/56.8 MB 1.9 MB/s eta 0:00:
27
   ----- ---------------------------------- 8.1/56.8 MB 1.8 MB/s eta 0:00:
27
   ----- ---------------------------------- 8.4/56.8 MB 1.8 MB/s eta 0:00:
28
   ------ --------------------------------- 8.9/56.8 MB 1.8 MB/s eta 0:00:
28
   ------ --------------------------------- 8.9/56.8 MB 1.8 MB/s eta 0:00:
28
   ------ --------------------------------- 9.2/56.8 MB 1.7 MB/s eta 0:00:
29
   ------ --------------------------------- 9.4/56.8 MB 1.7 MB/s eta 0:00:
29
   ------ --------------------------------- 9.4/56.8 MB 1.7 MB/s eta 0:00:
29
   ------- -------------------------------- 10.0/56.8 MB 1.6 MB/s eta 0:0
0:29
   ------- -------------------------------- 10.5/56.8 MB 1.7 MB/s eta 0:0
0:28
   ------- -------------------------------- 10.7/56.8 MB 1.6 MB/s eta 0:0
0:29
   ------- -------------------------------- 11.3/56.8 MB 1.7 MB/s eta 0:0
0:28
   -------- ------------------------------- 11.5/56.8 MB 1.7 MB/s eta 0:0
0:27
```

```
   -------- ------------------------------- 11.8/56.8 MB 1.7 MB/s eta 0:0
0:28
   -------- ------------------------------- 11.8/56.8 MB 1.7 MB/s eta 0:0
0:28
   -------- ------------------------------- 12.1/56.8 MB 1.6 MB/s eta 0:0
0:29
   -------- ------------------------------- 12.3/56.8 MB 1.6 MB/s eta 0:0
0:29
   --------- ------------------------------ 12.8/56.8 MB 1.6 MB/s eta 0:0
0:28
   --------- ------------------------------ 13.6/56.8 MB 1.6 MB/s eta 0:0
0:27
   --------- ------------------------------ 13.9/56.8 MB 1.7 MB/s eta 0:0
0:26
   --------- ------------------------------ 13.9/56.8 MB 1.7 MB/s eta 0:0
0:26
   --------- ------------------------------ 14.2/56.8 MB 1.6 MB/s eta 0:0
0:27
   ---------- ----------------------------- 14.9/56.8 MB 1.6 MB/s eta 0:0
0:26
   ---------- ----------------------------- 15.2/56.8 MB 1.7 MB/s eta 0:0
0:26
   ---------- ----------------------------- 15.2/56.8 MB 1.7 MB/s eta 0:0
0:26
   ----------- ---------------------------- 16.0/56.8 MB 1.6 MB/s eta 0:0
0:25
   ----------- ---------------------------- 16.8/56.8 MB 1.7 MB/s eta 0:0
0:24
   ------------ --------------------------- 17.6/56.8 MB 1.7 MB/s eta 0:0
0:23
   ------------ --------------------------- 18.4/56.8 MB 1.8 MB/s eta 0:0
0:22
   ------------- -------------------------- 19.1/56.8 MB 1.8 MB/s eta 0:0
0:21
   ------------- -------------------------- 19.4/56.8 MB 1.8 MB/s eta 0:0
0:21
   -------------- ------------------------- 19.9/56.8 MB 1.8 MB/s eta 0:0
0:21
   -------------- ------------------------- 20.4/56.8 MB 1.8 MB/s eta 0:0
0:20
   -------------- ------------------------- 20.4/56.8 MB 1.8 MB/s eta 0:0
0:20
   -------------- ------------------------- 21.0/56.8 MB 1.8 MB/s eta 0:0
0:20
   -------------- ------------------------- 21.2/56.8 MB 1.8 MB/s eta 0:0
0:20
   --------------- ------------------------ 22.0/56.8 MB 1.8 MB/s eta 0:0
0:20
   --------------- ------------------------ 22.0/56.8 MB 1.8 MB/s eta 0:0
0:20
   --------------- ------------------------ 22.3/56.8 MB 1.8 MB/s eta 0:0
0:20
   --------------- ------------------------ 22.5/56.8 MB 1.8 MB/s eta 0:0
0:20
   ---------------- ----------------------- 22.8/56.8 MB 1.8 MB/s eta 0:0
0:20
   ---------------- ----------------------- 23.1/56.8 MB 1.8 MB/s eta 0:0
0:20
   ---------------- ----------------------- 23.6/56.8 MB 1.8 MB/s eta 0:0
0:19
   ----------------- ---------------------- 24.4/56.8 MB 1.8 MB/s eta 0:0
0:19
```

```
   ----------------- ---------------------- 24.9/56.8 MB 1.8 MB/s eta 0:0
0:18
   ----------------- ---------------------- 25.4/56.8 MB 1.8 MB/s eta 0:0
0:18
   ------------------ --------------------- 25.7/56.8 MB 1.8 MB/s eta 0:0
0:18
   ------------------ --------------------- 26.7/56.8 MB 1.8 MB/s eta 0:0
0:17
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 27.5/56.8 MB 1.9 MB/s eta 0:0
0:16
   ------------------- -------------------- 28.0/56.8 MB 1.7 MB/s eta 0:0
0:17
   -------------------- ------------------- 28.6/56.8 MB 1.7 MB/s eta 0:0
0:17
   -------------------- ------------------- 28.6/56.8 MB 1.7 MB/s eta 0:0
0:17
   -------------------- ------------------- 29.1/56.8 MB 1.7 MB/s eta 0:0
0:17
   -------------------- ------------------- 29.1/56.8 MB 1.7 MB/s eta 0:0
0:17
   -------------------- ------------------- 29.1/56.8 MB 1.7 MB/s eta 0:0
0:17
   -------------------- ------------------- 29.4/56.8 MB 1.7 MB/s eta 0:0
0:17
   -------------------- ------------------- 29.6/56.8 MB 1.7 MB/s eta 0:0
0:17
   --------------------- ------------------ 30.1/56.8 MB 1.7 MB/s eta 0:0
0:17
   --------------------- ------------------ 30.4/56.8 MB 1.7 MB/s eta 0:0
0:16
   --------------------- ------------------ 31.2/56.8 MB 1.7 MB/s eta 0:0
0:16
   ---------------------- ----------------- 31.5/56.8 MB 1.7 MB/s eta 0:0
0:16
   ---------------------- ----------------- 32.0/56.8 MB 1.7 MB/s eta 0:0
0:15
   ---------------------- ----------------- 32.2/56.8 MB 1.7 MB/s eta 0:0
0:15
   ---------------------- ----------------- 32.2/56.8 MB 1.7 MB/s eta 0:0
0:15
   ---------------------- ----------------- 32.5/56.8 MB 1.7 MB/s eta 0:0
0:15
   ----------------------- ---------------- 33.0/56.8 MB 1.7 MB/s eta 0:0
0:15
   ----------------------- ---------------- 33.0/56.8 MB 1.7 MB/s eta 0:0
```