Steps to implement the Project and overview of it

Project title: **Using Big Data for Predictive Policing in Rwanda**.

🔹 Step 1: Data Acquisition

✅ What to Do:

- Download crime-related datasets: CSV, JSON, or direct API pull.
- Datasets should include: `Location`, `Time`, `Date`, `Type of Crime`, and maybe `Demographics`.

✅ Tools & Libraries:

```
import pandas as pd
import requests  # For APIs
```

✅ Example:

```
# If you have a CSV file
df = pd.read_csv("rwanda_crime_data.csv")

# From a URL or API (if available)
# response = requests.get("https://api.rwandapolice.gov/crime-stats")
# data = response.json()
# df = pd.DataFrame(data)
```

🔹 Step 2: Data Preprocessing

✅ What to Do:

- Handle missing values, date parsing, remove duplicates.
- Encode categorical features.
- Normalize or scale data if necessary.
- Feature engineering (e.g., extract `hour`, `weekday`, etc. from date).

✅ Libraries:

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
import numpy as np
```

✅ Example:

```
# Handling missing values
df.dropna(inplace=True)

# Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Feature Engineering
df['Hour'] = df['Date'].dt.hour
df['Weekday'] = df['Date'].dt.day_name()

# Encoding crime types
le = LabelEncoder()
df['Crime_Type_Encoded'] = le.fit_transform(df['Crime_Type'])

# Scaling features
scaler = StandardScaler()
df[['Hour', 'Latitude', 'Longitude']] = scaler.fit_transform(df[['Hour',
'Latitude', 'Longitude']])
```

🔹 Step 3: Model Creation

✅ What to Do:

- Split data into training/testing sets.
- Use models like `Random Forest`, `Logistic Regression`, or `SVM`.

✅ Libraries:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

✅ Example:

```
# Define features and labels
X = df[['Hour', 'Weekday', 'Latitude', 'Longitude']]
y = df['Crime_Type_Encoded']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Model Training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

🔹 Step 4: Model Evaluation

✅ What to Do:

- Use metrics like Accuracy, Precision, Recall, F1-Score.
- Use confusion matrix and ROC curves to visualize performance.

✅ Libraries:

```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
```

✅ Example:

```
# Prediction
y_pred = model.predict(X_test)

# Evaluation Metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

🔹 Step 5: Result Visualization

✅ What to Do:

- Create visual plots to show crime trends by area, type, or time.
- Use `Seaborn`, `Matplotlib`, or even `Plotly` for interactive plots.

✅ Libraries:

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

✅ Example:

```
# Crime frequency by type
sns.countplot(data=df, x='Crime_Type')
plt.title('Crime Frequency by Type')
plt.xticks(rotation=45)
```

```
plt.show()

# Heatmap by time
hourly_crimes = df.groupby('Hour').size()
hourly_crimes.plot(kind='bar', title='Crimes by Hour')
plt.xlabel("Hour")
plt.ylabel("Number of Crimes")
plt.show()
```

🔹 Step 6: Develop an Interface / Dashboard

✅ What to Do:

- Create a simple interactive dashboard for non-technical users.
- Use **Streamlit** (easy for Python users) or **Dash** (more customizable).
- Show predictive results + crime heatmaps or trends.

✅ Option 1: Using Streamlit

Install:

```
pip install streamlit
```

✅ Streamlit Code Example:

```
import streamlit as st

st.title("Predictive Policing Dashboard for Rwanda")

crime_type = st.selectbox("Select Crime Type", df['Crime_Type'].unique())
hour = st.slider("Hour of Day", 0, 23, 12)

# Dummy prediction (replace with model.predict)
st.write(f"Prediction: High crime risk at {hour}:00 for {crime_type}")
```

Run:

```
streamlit run your_script.py
```

🔹 Step 7: Export and Save Model (Optional)

We can save the trained model for later use.

✅ Libraries:

```
import joblib
```

✅ Example:

```
joblib.dump(model, 'crime_predictor_model.pkl')
```

And load it back:

```
model = joblib.load('crime_predictor_model.pkl')
```

✅ Summary

Step	Purpose	Tools/Libraries
1. Data Collection	Get structured crime data	`pandas`, `requests`
2. Preprocessing	Clean, transform data	`pandas`, `sklearn`, `LabelEncoder`
3. Modeling	Train predictive algorithms	`RandomForest`, `LogisticRegression`, etc.
4. Evaluation	Measure performance	`classification_report`, `confusion_matrix`
5. Visualization	Display trends	`matplotlib`, `seaborn`
6. Dashboard	Make results interactive	`streamlit`, `dash`
7. Exporting	Save model	`joblib`