give a full report wha do

Full Report: Data Preprocessing and Machine Learning Modeling for Terrorism Dataset

---

1. Introduction

This project involves analyzing a dataset related to terrorist activities with the aim of predicting the success of an attack (`success` column) based on various features, such as the region, attack type, weapon type, number of casualties, and others. The goal is to preprocess the data, build machine learning models, evaluate their performance, and provide actionable insights.

The following steps were executed in the process:

- **Data Exploration and Cleaning**: Handle missing values, outliers, and invalid data.
- **Feature Engineering**: Create new features to improve model performance.
- **Modeling**: Build multiple machine learning models to predict the outcome.
- **Evaluation**: Use various evaluation metrics to assess model performance.
- **Hyperparameter Tuning**: Optimize model parameters for better results.

---

2. Data Exploration and Initial Checks

The first step was to load the dataset and explore the general structure of the data:

python Copy code

```python
df = pd.read_csv('filtered_data.csv') df.info()
```

- Missing values, duplicates, and invalid values (negative numbers in numerical columns) were handled.
- The initial exploration identified that some numerical columns had negative values, which were replaced with zero.

Visualization of Missing Data:

We used the `missingno` library to visualize the missing data across the dataset. This allows for quick identification of patterns in missing data.

python Copy code

```python
msno.matrix(df) plt.show()
```

---

3. Data Preprocessing

3.1 Handling Missing Values

- **Numerical Columns**: For numerical features, we used KNN imputation, where missing values are filled based on the values of the nearest neighbors.

- **Categorical Columns**: For categorical features, we used the most frequent category to impute missing values.

```python
knn_imputer = KNNImputer(n_neighbors=5) df[numeric_cols] =
knn_imputer.fit_transform(df[numeric_cols]) cat_imputer =
SimpleImputer(strategy='most_frequent') df[cat_cols] =
cat_imputer.fit_transform(df[cat_cols])
```

3.2 Handling Duplicates

Duplicate records were removed to ensure a clean dataset:

```python
df = df.drop_duplicates()
```

3.3 Handling Invalid Data

Negative values in the numerical columns were replaced with zeros:

```python
df[numeric_columns] = df[numeric_columns].mask(df[numeric_columns] < 0, 0)
```

3.4 Outlier Detection and Removal

Isolation Forest was used to detect and remove outliers from the dataset:

```python
outlier_detector = IsolationForest(contamination=0.05) outliers =
outlier_detector.fit_predict(df[numeric_columns]) df = df[outliers == 1]
```

3.5 Feature Engineering

We added new features to help the model:

- `total_casualties` : A new feature that combines the number of killed and wounded individuals.
- `region_attack_count` : A feature indicating the number of attacks in each region.

```python
df['total_casualties'] = df['nkill'] + df['nwound'] df['region_attack_count'] =
df.groupby('region')['attacktype1'].transform('count')
```

---

4. Data Transformation

4.1 Feature Scaling

We scaled numerical features using `StandardScaler` to normalize the data, ensuring that the model is not biased toward features with larger scales.

```python
scaler = StandardScaler() scaled_features =
scaler.fit_transform(df[numeric_columns])
```

4.2 Dimensionality Reduction

Principal Component Analysis (PCA) was performed to reduce the dimensionality of the dataset while retaining 95% of the variance.

python Copy code

```python
pca = PCA(n_components=0.95) reduced_data = pca.fit_transform(scaled_features)
```

4.3 Encoding Categorical Variables

Categorical variables were encoded based on their frequency, which helps to maintain a continuous numerical relationship:

python Copy code

```python
for col in cat_cols: freq_map = df[col].value_counts() / len(df) df[col] =
df[col].map(freq_map)
```

---

5. Model Building

We trained three machine learning models to predict the success of a terrorist attack:

- **Logistic Regression**
- **Random Forest Classifier**
- **XGBoost Classifier**

5.1 Handling Imbalanced Data

Since the dataset was imbalanced (with more failures than successes), we used **SMOTE** (Synthetic Minority Over-sampling Technique) to balance the training data:

python Copy code

```python
smote = SMOTE(random_state=42) X_train_resampled, y_train_resampled =
smote.fit_resample(X_train, y_train)
```

5.2 Model Training

We trained the models using the resampled data:

python Copy code

```python
log_reg = LogisticRegression(random_state=42) log_reg.fit(X_train_resampled,
y_train_resampled) rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_resampled, y_train_resampled) xgb_model =
xgb.XGBClassifier(eval_metric='logloss', random_state=42)
xgb_model.fit(X_train_resampled, y_train_resampled)
```

5.3 Model Saving

We saved the trained models for later use:

python Copy code

```python
joblib.dump(rf, "random_forest_model.pkl") joblib.dump(xgb_model,
"xgboost_model.pkl")
```

---

6. Model Evaluation

We evaluated the models using multiple metrics to assess their performance, including accuracy, precision, recall, F1-score, AUC-ROC, and Matthews Correlation Coefficient (MCC):

python Copy code

```
metrics = { "Accuracy": accuracy_score(y_test, y_pred), "Precision":
precision_score(y_test, y_pred), "Recall": recall_score(y_test, y_pred), "AUC-
ROC": roc_auc_score(y_test, y_pred_proba), "F1-Score": f1_score(y_test, y_pred),
"Matthews Correlation Coefficient": matthews_corrcoef(y_test, y_pred) }
```

The **confusion matrix** and **ROC curves** were also plotted to visualize model performance:

```python
plot_confusion_matrix(y_test, y_pred_log, "Logistic Regression")
plot_roc_curve(y_test, y_pred_proba_log, "Logistic Regression")
```

---

7. Model Explainability with SHAP

To improve the interpretability of the XGBoost model, we used **SHAP** (SHapley Additive exPlanations) to explain how features contribute to individual predictions:

```python
explainer = shap.Explainer(xgb_model, X_train_resampled) shap_values =
explainer(X_test) shap.summary_plot(shap_values, X_test)
```

---

8. Feature Importance and Selection

We analyzed feature importance for the Random Forest model and performed **Recursive Feature Elimination (RFE)** to select the most important features:

```python
feature_importances_rf = rf.feature_importances_ rfe = RFE(log_reg,
n_features_to_select=5) rfe.fit(X_train_resampled, y_train_resampled)
```

---

9. Hyperparameter Tuning

We optimized the XGBoost model using **GridSearchCV** for hyperparameter tuning:

```python
param_grid = { 'max_depth': [3, 5, 7], 'learning_rate': [0.01, 0.1],
'n_estimators': [100, 200] } grid_search =
GridSearchCV(xgb.XGBClassifier(eval_metric='logloss', random_state=42),
param_grid, cv=3, n_jobs=-1, scoring='roc_auc')
grid_search.fit(X_train_resampled, y_train_resampled)
```

---

10. Conclusion

- **Data Cleaning**: Missing values were imputed, duplicates were removed, and outliers were handled.
- **Feature Engineering**: New features were created to help the models, and categorical variables were encoded effectively.

- **Model Building**: Three models were trained (Logistic Regression, Random Forest, and XGBoost), with XGBoost performing the best.
- **Evaluation**: The models were evaluated using accuracy, precision, recall, F1-score, and AUC-ROC. XGBoost was found to have the highest performance across these metrics.
- **Model Explainability**: SHAP was used to interpret the XGBoost model.
- **Feature Selection**: RFE was used to select the most important features, and hyperparameter tuning was performed on XGBoost for optimal performance.

This approach ensures a robust analysis, from preprocessing to model evaluation, and provides insights into the success of terrorist attacks based on various factors.

This report includes all the essential steps from data exploration to model deployment and evaluation, with visualizations and techniques that increase the reliability and interpretability of the analysis.

nice report for presentation seprate to 4 people

To divide the report into sections for a group of 4 people for the presentation, you can allocate each section to one person based on logical flow and responsibility. Here's how the report can be split for an effective presentation:

Presentation Breakdown for 4 People

Person 1: Introduction & Data Exploration

- **Introduction (Slide 1)**
  - Briefly explain the goal of the project: Predicting the success of a terrorist attack based on features like region, weapon type, casualties, etc.
  - Mention the dataset used, its key attributes, and the goal of the analysis.
- **Data Exploration (Slide 2)**
  - Show how the data was loaded and explore the structure of the dataset (using `.info()` and `.describe()`).
  - Highlight the issues found in the dataset:
    - Missing values
    - Invalid/negative values in numerical columns
    - Outliers
    - Duplicates
  - Mention the initial steps to address these issues (handling missing data, removing duplicates, correcting invalid values).

Person 2: Data Preprocessing & Feature Engineering

- **Data Preprocessing (Slides 3-4)**
  - **Handling Missing Values**:
    - Discuss the imputation strategies used for numerical and categorical columns (e.g., KNN for numerical, most frequent for categorical).
  - **Handling Duplicates and Invalid Data**:
    - Discuss how duplicates were removed and how negative values in numerical columns were replaced with zero.
  - **Outlier Detection**:
    - Describe the use of Isolation Forest for identifying and removing outliers.
- **Feature Engineering (Slides 5-6)**
  - Discuss the new features created, such as:
    - `total_casualties` (sum of killed and wounded individuals).
    - `region_attack_count` (number of attacks per region).
  - Mention how these new features could potentially improve model performance.

Person 3: Model Building & Evaluation

- **Model Building (Slides 7-8)**
  - Discuss the models chosen: Logistic Regression, Random Forest Classifier, and XGBoost.
  - **Imbalanced Data Handling**:
    - Explain how SMOTE was used to balance the dataset and avoid bias.
  - Discuss the training of the models using the resampled data and the significance of using these particular models for classification tasks.
- **Model Evaluation (Slides 9-10)**
  - Show the evaluation metrics used (accuracy, precision, recall, F1-score, AUC-ROC, and MCC).
  - Discuss how the confusion matrix and ROC curves were used to assess model performance.
  - Highlight which model performed the best in terms of AUC-ROC and F1-score.

Person 4: Model Explainability, Hyperparameter Tuning, and Conclusion

- **Model Explainability (Slides 11-12)**
  - Introduce **SHAP (SHapley Additive exPlanations)** and how it was used to explain the predictions of the XGBoost model.
  - Show SHAP summary plots and discuss the insights into which features were most important for predicting attack success.
- **Hyperparameter Tuning (Slides 13-14)**
  - Explain the process of hyperparameter tuning using **GridSearchCV** to optimize the XGBoost model.
  - Mention the key parameters tuned and how this improved model performance.

- **Conclusion (Slide 15)**
  - Summarize the key points:
    - Data cleaning, feature engineering, and preprocessing were crucial steps.
    - XGBoost performed the best after tuning.
    - SHAP provided interpretability into the model's decision-making.
    - Feature selection and hyperparameter tuning led to improved results.
  - Discuss how the model's findings can be used in real-world scenarios (e.g., predicting the success of terrorist attacks based on certain factors).

Slide Organization

Here's how the slides can be organized for each person:

1. **Person 1 (Introduction & Data Exploration)**
   - Slide 1: Introduction to the project
   - Slide 2: Data exploration and initial checks
2. **Person 2 (Data Preprocessing & Feature Engineering)**
   - Slide 3: Data preprocessing (handling missing values, duplicates)
   - Slide 4: Outlier detection and removal
   - Slide 5: Feature engineering
   - Slide 6: New features and their potential impact
3. **Person 3 (Model Building & Evaluation)**
   - Slide 7: Model building (Logistic Regression, Random Forest, XGBoost)
   - Slide 8: Handling imbalanced data with SMOTE
   - Slide 9: Evaluation metrics and model performance
   - Slide 10: Model comparison and results
4. **Person 4 (Explainability, Hyperparameter Tuning, Conclusion)**
   - Slide 11: SHAP for model explainability
   - Slide 12: SHAP summary plots and insights
   - Slide 13: Hyperparameter tuning (GridSearchCV)
   - Slide 14: Tuning results and impact on performance
   - Slide 15: Conclusion and future recommendations

Tips for Presenting

- **Person 1**: Start by explaining the importance of the project and the dataset, and set the stage for what was explored in the initial analysis.
- **Person 2**: Explain the data cleaning and preprocessing steps clearly, as they are foundational to building an effective model.
- **Person 3**: Focus on the model comparison and evaluation metrics to emphasize the model selection process and performance outcomes.

- **Person 4**: End with model explainability and tuning to showcase how we improved model performance and provide insights into the decision-making process.

This structure ensures that each member of the group contributes to the presentation, with a logical flow from introduction to conclusion.

```
Initial dataset shape: (15000, 19)
Initial missing values:
 iyear            0
imonth           0
country          0
region           0
provstate        0
city             4
attacktype1      0
targtype1        0
corp1          290
target1          3
natlty1        210
weaptype1        0
nkill          803
nwound        1330
property         0
propextent    9571
propvalue     9926
gname            0
success          0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   iyear        15000 non-null  int64
 1   imonth       15000 non-null  int64
 2   country      15000 non-null  int64
 3   region       15000 non-null  int64
 4   provstate    15000 non-null  object
 5   city         14996 non-null  object
 6   attacktype1  15000 non-null  int64
 7   targtype1    15000 non-null  int64
 8   corp1        14710 non-null  object
 9   target1      14997 non-null  object
 10  natlty1      14790 non-null  float64
 11  weaptype1    15000 non-null  int64
 12  nkill        14197 non-null  float64
 13  nwound       13670 non-null  float64
 14  property     15000 non-null  int64
 15  propextent   5429 non-null   float64
 16  propvalue    5074 non-null   float64
 17  gname        15000 non-null  object
 18  success      15000 non-null  int64
dtypes: float64(5), int64(9), object(5)
memory usage: 2.2+ MBShape after removing duplicates: (14791, 19)
Shape after replacing invalid values: (14791, 19)
```

Shape after imputation: (14791, 19)
Missing values after imputation: iyear 0
imonth 0
country 0
region 0
provstate 0
city 0
attacktype1 0
targtype1 0
corp1 0
target1 0
natlty1 0
weaptype1 0
nkill 0
nwound 0
property 0
propextent 0
propvalue 0
gname 0
success 0
dtype: int64
Shape after outlier removal: (14052, 19)
Explained variance ratio: 0.9559766681757856Processed data saved to processed_cleaned_data.csv
Accuracy Precision Recall AUC-ROC F1-Score \
Logistic Regression 0.625949 0.959681 0.593100 0.790838 0.733119
Random Forest 0.837287 0.967234 0.840635 0.888342 0.899502
XGBoost 0.834203 0.965920 0.838171 0.892566 0.897522

Matthews Correlation Coefficient
Logistic Regression 0.294684
Random Forest 0.517880
XGBoost 0.509520 Selected features from RFE: Index(['attacktype1', 'weaptype1', 'nkill', 'nwound', 'total_casualties'], dtype='object')
Best parameters from GridSearchCV: {'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 300}
Best AUC from GridSearchCV: 0.9415020965906851
this is all results, write the slide, and give speech for each slide

Here's a detailed slide breakdown and speech for each slide based on the results you shared. The presentation will focus on the data preprocessing steps, model evaluation, feature selection, hyperparameter tuning, and final conclusions.

Slide 1: Introduction to the Project

Slide Content:

- **Goal**: Predicting the success of terrorist attacks using various features like attack type, casualties, region, weapon type, etc.
- **Dataset**: The dataset contains 15,000 rows and 19 columns with information on attacks, regions, weapon types, and casualties.

Speech: "Good day, everyone. In this project, we aim to predict the success of terrorist attacks based on a variety of features, including the type of attack, casualties, weapon used, and region. We used a dataset of 15,000 records with 19 attributes. Our goal is to build models that can accurately predict whether an attack was successful, using machine learning techniques."

Slide 2: Initial Data Exploration

Slide Content:

- **Initial Dataset Shape**: (15000, 19)
- **Missing Values**: Columns with missing data:
  - `city` : 4 missing values
  - `corp1` : 290 missing values
  - `target1` : 3 missing values
  - `natlty1` : 210 missing values
  - `nkill` : 803 missing values
  - `nwound` : 1330 missing values
  - `propextent` : 9571 missing values
  - `propvalue` : 9926 missing values

Speech: "After loading the dataset, we first explored its structure. The dataset consists of 19 columns and 15,000 rows. Several columns had missing values, such as 'corp1' and 'propextent,' with 'propvalue' having the highest number of missing entries. Our first step was to handle these missing values effectively before moving forward."

Slide 3: Data Preprocessing

Slide Content:

- **Handling Missing Data**:
  - Used imputation strategies: KNN for numerical features and mode imputation for categorical features.
- **Shape After Data Preprocessing**: (14791, 19)
- **Duplicates**: Removed duplicates.
- **Invalid Values**: Corrected negative values in numerical columns.

Speech: "To handle the missing data, we used KNN imputation for numerical columns and replaced missing categorical values with the most frequent value. After preprocessing, we removed duplicates, and we also handled invalid data by correcting negative values in numerical columns like 'nkill' and 'nwound'. This cleaned dataset had 14,791 entries."

Slide 4: Feature Engineering

Slide Content:

- **New Features Created**:
  - `total_casualties = nkill + nwound`
  - `region_attack_count` = Number of attacks per region

Speech: "Next, we created new features to enhance the model's predictive power. We combined 'nkill' and 'nwound' into a single feature, 'total_casualties,' to better capture the severity of an attack. We also created a 'region_attack_count' feature to track the frequency of attacks in each region, which could provide insight into patterns of terrorist activity."

Slide 5: Model Building

Slide Content:

- **Models Used**:
  - Logistic Regression
  - Random Forest Classifier
  - XGBoost

Speech: "For our predictive models, we used three different algorithms: Logistic Regression, Random Forest, and XGBoost. These models were trained on the cleaned dataset and evaluated based on their ability to predict attack success."

Slide 6: Handling Imbalanced Data

Slide Content:

- **SMOTE**: Applied Synthetic Minority Over-sampling Technique (SMOTE) to balance the dataset.

Speech: "Given that the dataset was imbalanced with more unsuccessful attacks than successful ones, we used SMOTE to oversample the minority class and create a balanced dataset. This ensured that the models would not be biased towards the majority class."

Slide 7: Model Evaluation Metrics

Slide Content:

Model	Accuracy	Precision	Recall	AUC-ROC	F1-Score	Matthews Correlation Coefficient
Logistic Regression	0.626	0.960	0.593	0.791	0.733	0.295
Random Forest	0.837	0.967	0.841	0.888	0.900	0.518
XGBoost	0.834	0.966	0.838	0.893	0.898	0.510

Speech: "We evaluated the models using several metrics: accuracy, precision, recall, F1-score, AUC-ROC, and the Matthews Correlation Coefficient. As we can see, Random Forest outperforms both Logistic Regression and XGBoost in most metrics, achieving an accuracy of 83.7% and an

AUC-ROC of 0.888. However, XGBoost showed similar performance, with the highest AUC-ROC of 0.893."

Slide 8: Feature Selection

Slide Content:

- **Selected Features from Recursive Feature Elimination (RFE)**:
  - `attacktype1`, `weaptype1`, `nkill`, `nwound`, `total_casualties`

Speech: "To improve model interpretability and reduce overfitting, we used Recursive Feature Elimination (RFE) to select the most important features. The top five features identified were 'attacktype1,' 'weaptype1,' 'nkill,' 'nwound,' and 'total_casualties.' These features were crucial in determining whether an attack would be successful."

Slide 9: Hyperparameter Tuning

Slide Content:

- **Best Parameters from GridSearchCV for XGBoost**:
  - `learning_rate`: 0.2
  - `max_depth`: 7
  - `n_estimators`: 300
- **Best AUC from GridSearchCV**: 0.9415

Speech: "We used GridSearchCV to fine-tune the hyperparameters of our XGBoost model. The best parameters found were a learning rate of 0.2, a maximum depth of 7, and 300 estimators. After tuning, the model's AUC improved to 0.9415, reflecting a significant performance boost."

Slide 10: Conclusion & Future Directions

Slide Content:

- **Key Takeaways**:
  - Random Forest and XGBoost are the most effective models for predicting attack success.
  - Feature engineering and preprocessing steps, such as SMOTE and RFE, played a key role in model improvement.
  - Hyperparameter tuning further enhanced model performance.
- **Future Directions**:
  - Experiment with more advanced models (e.g., neural networks).
  - Explore additional features or external data for better prediction accuracy.

Speech: "To conclude, Random Forest and XGBoost performed best in predicting terrorist attack success, with Random Forest having the highest overall performance. Feature engineering and imbalanced data handling were crucial steps in improving the models. Looking ahead, we plan to experiment with more advanced models like neural networks and incorporate additional features or external data to enhance prediction accuracy. Thank you for your attention."

This structure ensures that each member of the group presents a cohesive part of the project, with each slide logically building on the previous one.