

Data Collection and Preprocessing Phase

Date	15 July 2024
Team ID	739668
Project Title	Octagon Oracle: Machine Learning-Powered UFC FIGHT FORECAST
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description
Data Overview	Basic statistics, dimensions, and structure of the data.
Univariate Analysis	Exploration of individual variables (mean, median, mode, etc.).
Bivariate Analysis	Relationships between two variables (correlation, scatter plots).
Multivariate Analysis	Patterns and relationships involving multiple variables.
Outliers and Anomalies	Identification and treatment of outliers.
Data Preprocessing Code Screenshots	
Loading Data	<pre>[] # Load your dataset df = pd.read_csv('/content/UFC_DATA.csv')</pre>

Handling Missing Data	<pre> # Check for missing values print(df.isnull().sum()) # Impute missing values for numerical columns numerical_columns = df.select_dtypes(include=[np.number]).columns for column in numerical_columns: df[column].fillna(df[column].median(), inplace=True) # Impute missing values for categorical columns (if any) categorical_columns = df.select_dtypes(include=[object]).columns for column in categorical_columns: df[column].fillna(df[column].mode()[0], inplace=True) </pre>
Data Transformation	<pre> # Data Transformation Code # Handling Missing Values (Numerical) numerical_columns = df.select_dtypes(include=[np.number]).columns for column in numerical_columns: df[column].fillna(df[column].median(), inplace=True) # Handling Missing Values (Categorical) categorical_columns = df.select_dtypes(include=[object]).columns for column in categorical_columns: df[column].fillna(df[column].mode()[0], inplace=True) # Capping Outliers def cap_outliers(df, column): Q1 = df[column].quantile(0.25) Q3 = df[column].quantile(0.75) IQR = Q3 - Q1 lower_bound = Q1 - 1.5 * IQR upper_bound = Q3 + 1.5 * IQR df[column] = np.where(df[column] > upper_bound, upper_bound, df[column]) df[column] = np.where(df[column] < lower_bound, lower_bound, df[column]) </pre>
Feature Engineering	<pre> for column in numerical_columns: cap_outliers(df, column) # Encoding Categorical Variables label_encoders = {} for column in categorical_columns: le = LabelEncoder() df[column] = le.fit_transform(df[column]) label_encoders[column] = le # Encoding Target Variable X = df.drop(columns=['winner']) y = df['winner'] if y.dtype == 'object': le = LabelEncoder() y = le.fit_transform(y) label_encoders['winner'] = le </pre>
Save Processed Data	<pre> smote = SMOTE(random_state=42) X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train) print(f"Original dataset shape: {Counter(y_train)}") print(f"Balanced dataset shape: {Counter(y_train_balanced)}") </pre> <div> Original dataset shape: Counter({3: 815, 0: 608, 2: 16, 1: 13}) Balanced dataset shape: Counter({0: 815, 3: 815, 2: 815, 1: 815}) </div>