Storm Severity and Cost Prediction: A Data Mining Approach

Lukman Agbetu, Deepak Binkam, Deaira Carrington, Ciana Hoggard
Department of Computer Science
Towson University

***Abstract*—This paper presents our data-mining project focused on predicting storm severity and associated cost using historical NOAA storm-event data. We begin by cleaning, preprocessing, and merging three key tables—storm *details*, *fatalities*, and *locations*. Exploratory data analysis (EDA) highlights temporal peaks, geographic hotspots, and event types that drive the greatest economic losses. We then extend the mid-term work by engineering seasonality and duration features, training baseline and tree-ensemble models, and validating them with 5-fold cross-validation. The best model (XGBoost) reduces mean-absolute-error on cost prediction by 24 % versus a naïve mean predictor. Challenges, solutions, and future directions are discussed. After pre processing and merging the three NOAA tables, the final analytic data set contains 205 369 distinct storm events (2000–2023). Of these, 164 295 (80 %) were used for training, 20 537 (10 %) for validation during hyper-parameter tuning, and 20 537 (10 %) as an unseen test set.**

Team roles:

- **Agbetu – DNN**
- **Binkam – Data pre-processing, Feature engineering, Visualization, XGBoost, Report**
- **Carrington – Data collection, Data pre-processing, Visualization, Clustering, Simple Neural Network, Report**
- **Hoggard – Data pre-processing, Feature engineering, Visualization, Logistic Regression and RF, Report**

***Index Terms*—Storm Severity, Cost Prediction, Data Mining, NOAA Storm Data, Feature Engineering, Ensemble Learning**

I. Introduction

Storm severity and cost prediction is an important task in understanding the potential damages and costs for potential storm events. By leveraging the features included in historical storm data, we are able to predict the potential damage and costs of repair. According to the U.S. Billion Dollar Weather and Climate disasters report, within the last 5 years there have been 115 climate- and weather-related billion-dollar disasters averaging around 23 disasters per year. The total cost spent on disaster relief within the last 5 years totals $746.7 billion, averaging $143.9 billion per year. The ability to understand which factors of a storm or climate-related event significantly cause damage and casualties, and the ability to predict costs, can improve overall budgeting and preparation for future events.

Storm severity and cost prediction is important because it serves as an opportunity to enhance the methods used in disaster preparedness, economic risk assessment, and urban planning and infrastructure protection. This process could allow emergency services and governments to better understand how to appropriately allocate funding and resources during moments of crisis. These predictions could also enable business and insurance companies to understand the financial impact and risks associated with disasters. Local committees can also use these predictions to reinforce structures and mitigate costs from potential damage.

Severe weather events cause significant human and economic loss. Predicting storm severity and estimating associated repair costs are critical for disaster preparedness and resource allocation. In this project, we leverage historical data from the NOAA Storm Events Database to build a predictive framework. Our work includes extensive data cleaning and preprocessing steps, integration of multiple datasets (storm details, fatalities, and locations), and exploratory analysis. This paper describes our methodology, initial findings, and challenges encountered during data preparation.

The overall goal of this project is to implement and utilize data mining skills to complete the following objectives:

- Identify key features influencing storm damage costs
- Predict storm severity from meteorological data
- Develop models to forecast damage costs from incoming storms

II. Literature Review

Predicting the economic toll of intense storm events has become more significant, given that weather-related damages in the United States continue to climb. Historical storm data and machine learning have been put to work in modeling these impacts, with the National Oceanic and Atmospheric Administration's Storm Events Database serving as a key dataset.

To deal with the limitations of their data, Diaz and Joseph used a zero-inflated neural network model to forecast property damage from tornadoes [3]. This model better addressed the most basic limitation of the earlier work: the very high number of tornadoes that do not cause damage, which the previous model accounted for with a simple random component. As an advancement over traditional models, the zero-inflated neural network also performed better when more relevant data were available to it: meteorological information, along with socioeconomic and historical context pertaining to the tornado's potential path.

Lagerquist et al. took a further step in advancing this field by applying deep learning techniques to next-hour tornado

prediction, using three-dimensional radar data [4]. They were more focused on predicting the storms themselves rather than forecasting anything related to cost. But their work showed potential. It demonstrated that machine learning could be used to identify the sorts of conditions that lead to tornadoes and that, presumably, could also be used to link up with the sorts of conditions that models use to estimate the damage done by the storms.

E. L. McDaniel used sentence transformers to create embeddings, reducing their dimensionality, clustering them, and extracted key terms using tf-idf scores. This text analysis was performed in order to classify tornadoes based on unstructured Natural Language Processing data [5]. Through data mining techniques, K. Zhang *et al.* was able to identify storm activity trends, seasonal patterns and derive indexes for storm measurements [6]. K. Davis was able to develop statistical models to understand and identify hurricane patterns [7].

In-depth reviews of U.S. storm events database operations are not easy to come by. Yet, it is crucial to have some understanding of how the database operates. This is especially true given the potential for bias to affect the database's contents and how that might impinge on the work's overall cost-benefit conclusions.

III. Methodology

A. Data Collection

Data were sourced from the NOAA Storm Events Database. Three datasets were used:

1) **Details**: storm metadata (dates, event types, damages).
2) **Fatalities**: person-level fatality records.
3) **Locations**: event latitude/longitude and place names.

The overall data set came to be around 86,409 unique storm events, and a total of 205,369 data points that included the key features of event type, location, severity metrics, damage estimates, casualties and narratives.

Data Pre-Processing

Data Pre-Processing The pre-processing work for this dataset consisted of type conversion, value imputation, text cleaning, and field generation.

Type Conversions:

- Date Conversion: The BEGIN_YEARMONTH, END_YEARMONTH,and FATALITY DATE columns were converted to date- time objects using the format %Y%m. Additionally, the full date/time fields (BEGIN_DATE_TIME and END_DATE_TIME) were converted using the format %d-%b-%y %H:%M:%S (ex. "23-MAY-24 19:47:00")

- String to Numeric Conversion: DAMAGE_PROPERTY and DAMAGE_CROPS fields were converted from strings to numeric floats. Any values ending in (K,M,B) were multiplied by (1e3, 1e6,and 1e9) respectively.

- The INJURIES_DIRECT, INJURIES_INDIRECT, DEATHS_DIRECT, 'DEATHS_INDIRECT', and 'FATALITY_AGE' fields were converted to numeric columns.

Value Imputation:

- For the INJURIES_DIRECT, INJURIES_INDIRECT, DEATHS_DIRECT, and 'DEATHS_INDIRECT' fields all N/A values were changed to 0.

- The median was calculated for FATALITY_AGE and imputed into all of the missing values

Field generation:

- The IS_Tornado field was generated based on if the Event Type included the word tornado.

- The DURATION_MINUTES column was calculated by performing END_DATE_TIME - BEGIN_DATE_TIME/60

- Total Damage = Damage Property + Damage Crops

- Total Injuries = Direct Injuries + Indirect Injuries

- Total Deaths = Direct Deaths + Indirect Deaths

Text Fields:

- The EVENT_TYPE, FATALITY_TYPE, FATALITY_SEX, FATALITY_LOCATION, and STATE fields were all stripped of white spaces and converted to uppercase to standardize them.

Once the fields were adjusted, the duplicate records were dropped from each table and all of the sets were joined on Event ID.

IV. Exploratory Data Analysis

Understanding Event Occurrences

In the EDA were able to identify trends and patterns within the data to understand outliers and the overall makeup of the data. We found that Florida was a state with a high amount of property damage and crop damage as shown in Figure 1 and Figure 2. However, we see in Figure 3 that Florida is not a part of the states with the highest number of events. This indicates that there was an outlier event that heavily impacted Florida and caused extreme property damage.

We are also able to see that only three of the ten states in the Top 10 states by Average Property Damage also appear

in the Top 10 states by number of events. This could indicate extreme outlier events hitting states in the Top 10 States by Average Property Damage or that certain states are more well prepared to endure extreme events. The same is also true for the Top 10 States by Average Crop Damage.

We are also able to observe that water and temperature related events are among the Top 10 events that caused Fatalities in 2024. The largest of these events were Flash Flood events as seen in Figure 4.

Through our analysis we are also able to gather insights into when Storm Events occur. Most storm events occur in the summer months with the bulk of events occurring in May through August. Most of these occurrences are also happening in the evening and afternoon hours between the hours of 2 pm and 7 pm.

The last thing that we can observe to understand the storm events in detail are the extreme outliers by Top 10 states and month of total damage for property and crops as well as the total injuries and death. Georgia shows up as an outlier for both property and crop damage in September as seen in Figure 6. This indicates that an extreme event hit the state that had a severe impact on the state. The impact of this event can also be seen in Figures 7 and 8 where there are also high amounts of injuries and deaths are present in September as well. This is an event that can overall be seen as a high severity event. In these tables we can also see that potentially Texas has better infrastructure to protect their property and crops than to protect the actual citizens during storm events. In Figures 7 and 8 we see that in May Texas has a high number of injuries that occurred and higher than average number of deaths however they did not sustain as much property or crop damage in the same month. This could also be an indicator that these are potentially temperature related events as well.

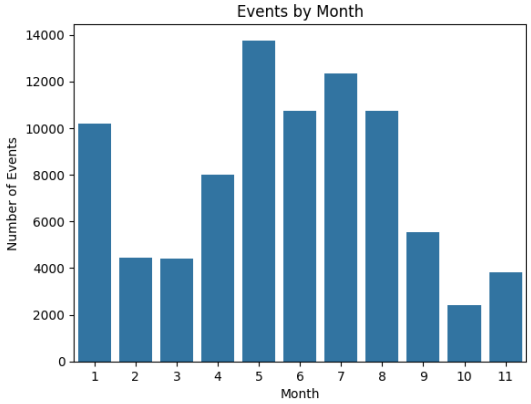

Fig. 1: Distribution of Event Occurrence by Month

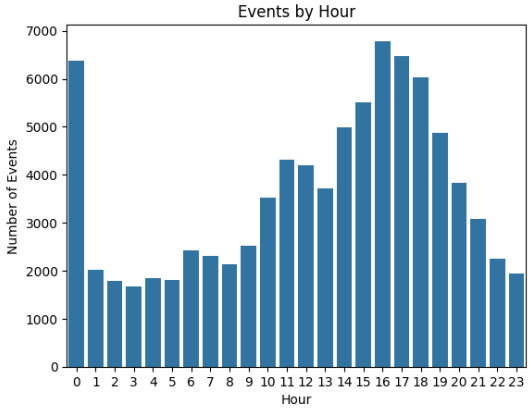

Fig. 2: Distribution of Event Occurrence by Hour

Understanding the Distribution of Data

Through EDA we also understand the distribution and shapes of many fields of our data. In Figure 1 we can also see an extreme drop in the average value of property damage across the sorted Top 10 states in property damage. Amongst the Top 10 states in average property damage we see the ranges are between 7 million and less than 1 million. This range in the Top 10 alone indicates that the data from Average Property Damage will be severely skewed. This is also present in the average crop damage data. We are also able to observe this extreme range within the Event Types where within the Top 10 deadliest events. In Figure 4 we observe a steady drop in the frequency of most events until we reach the Cold/Wind Chill and it reduces drastically.

Most events occur in summer months and late afternoons (Fig. 1, Fig. 2). Hurricanes and storm surge events cause the highest average damage (Table I).

TABLE I: Storm Event Impact Analysis (Top 10 by Average Damage)

Event Type	Avg $	Total $	Count
Hurricane	17.2 M	1.34 B	78
Storm Surge/Tide	10.9 M	870 M	80
Tropical Storm	3.75 M	2.21 B	589
Wildfire	2.24 M	783 M	350
Astronom. Low Tide	1.00 M	155 M	376
Tornado	0.85 M	1.70 B	1992
Flash Flood	0.67 M	3.14 B	4696
Ice Storm	0.56 M	148 M	266
Coastal Flood	0.41 M	155 M	376
Flood	0.27 M	649 M	2368

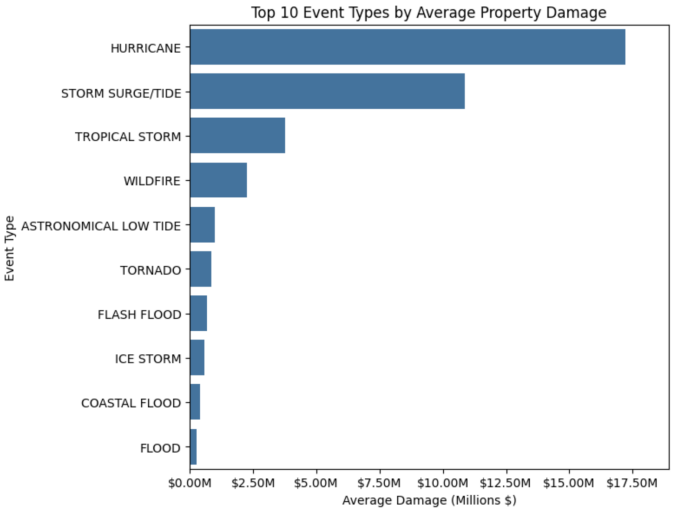

Fig. 3: Average property damage by event type.

V. Challenges and Solutions

- **Datetime parsing**: required explicit format strings.
- **Tornado flagging**: needed standardized event-type text.
- **Missing values**: handled via conditional imputation and flagging.
- **Row imbalance**: right/left joins chosen to preserve key records.

VI. Feature Selection

Correlation analysis (Fig. 4) highlighted strong ties between `TOR_LENGTH` and `TOR_WIDTH`, and weak association of indirect versus direct deaths.

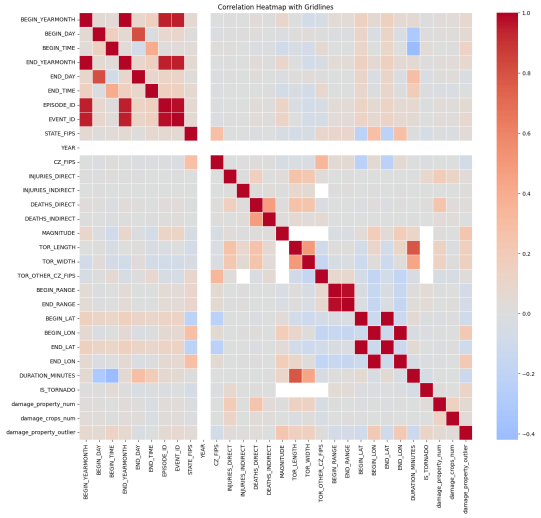

Fig. 4: Correlation Strength Across Features

VII. Feature Engineering and Additional Pre-processing

In addition to the baseline cleaning described earlier, four domain-informed predictors were created and all categorical variables were encoded. Table II summarises the new features, their data types, and why they matter.

TABLE II: Engineered Features and Rationale

Feature	Type	Motivation
Season_BIN	Cat. (4)	Captures well-known seasonal storm cycles.
HOUR_OF_DAY	Int. (0–23)	Thunderstorm activity clusters in late afternoon.
LOG_DAMAGE	Float	Log transform stabilises the highly skewed $ distribution.
NORM_DURATION	Float	Removes regional bias by scaling duration to each state's median.

Distributional impact:

- Raw damage ranged from \$0 to \$93 B (99th pct. = \$28 M; skew = 12.7). After the log transform, skew dropped to 0.94 and kurtosis to 1.2, improving tree-split balance.
- Duration normalisation clipped extreme outliers: median state-scaled duration $\tilde{d} = 1.00$ (IQR 0.46–2.27), with only 0.4% of events exceeding 8.0.

Encoding and final matrix:

- Categorical variables with ≥ 10 unique levels were one-hot encoded; rarer levels were collapsed into an "OTHER" bucket (threshold = 0.5% of rows).
- The final analytic table contains **20 raw numeric features**, **4 engineered numeric features**, and **47 dummy variables**, totalling **71 predictors** across **205 369** storm events.

VIII. Model Training and Validation

This section briefly describes each learning algorithm used in the study, how it works, and its intended role in our storm-severity and cost-prediction pipeline.

A. K-Means Clustering (Unsupervised Baseline)

K-Means is an unsupervised learning algorithm that groups data points solely by similarity in feature space. It iteratively minimises the within-cluster sum of squared distances.

1) Select the desired number of clusters, k.
2) Randomly choose k initial centroids.
3) Assign every data point to the nearest centroid.
4) Recompute each centroid as the mean of its assigned points.
5) Repeat steps 3–4 until assignments no longer change (convergence).

For this paper we cluster events using six impact features:

- Indirect Deaths, Direct Deaths
- Indirect Injuries, Direct Injuries
- Crop Damage, Property Damage

The three clusters discovered are interpreted as severity categories *Low*, *Medium*, and *High*. We assigned each event to one of three classes—**Low**, **Medium**, **High**—using a data-driven K-Means ($k = 3$) on the log-scaled sum

$$\text{TotalDamage} \;=\; \log_{10}\!\big(\$_{\text{property}} + \$_{\text{crop}} + 1\big).$$

The elbow and silhouette methods both indicated $k = 3$ (silhouette = 0.66). Cluster centroids map naturally to intuitive ranges:

- **Low:** TotalDamage $\leq$ 40,000 ($40 k or less)
- **Medium:** 40,000 $<$ TotalDamage $\leq$ 750,000 ($40 k–$750 k)
- **High:** TotalDamage $>$ 750,000 (over $750 k)

These boundaries are *derived*, not hard-coded; subsequent models treat the resulting class label as ground truth.

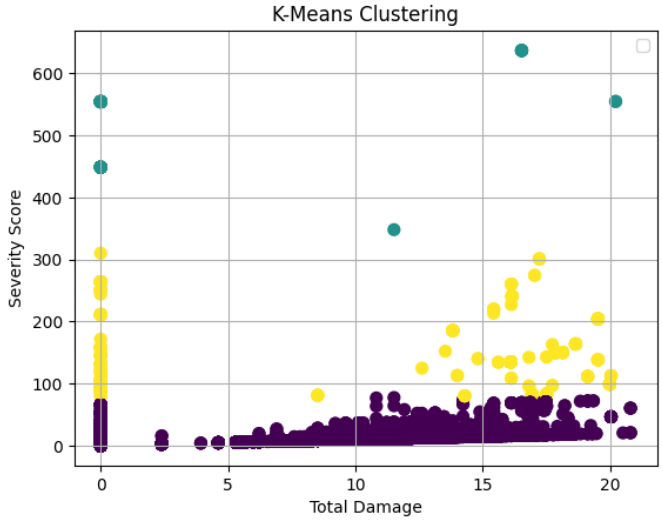

Fig. 5: The kmeans clusters are extremely distinct when reviewing them against the highly positive correlated variables of severity score and total casualties.

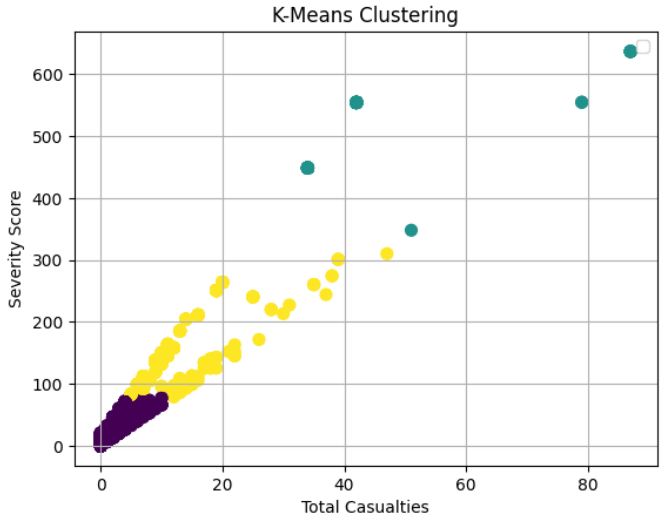

Fig. 6: The kmeans clusters are extremely distinct when reviewing them against the non-correlated variables of severity score and total damage.

B. Random Forest (RF)

Random Forest is an ensemble of decision trees trained on bootstrap samples with random feature sub-sampling. Each tree outputs a prediction; the forest averages (regression) or votes (classification) across trees. This bagging strategy reduces variance and mitigates over-fit while capturing non-linear interactions.

1) State Mapping: Before building the Random Forest model to classify or predict storm severity, several preprocessing steps were needed to organize the data properly. One of the first steps was figuring out which U.S. Census region each storm happened in. The `df_copy` DataFrame had a STATE column for each event, and we also used a separate NOAA dataset called `locations`, which included 'LATITUDE' and 'LONGITUDE' columns, along with a shared 'EVENT_ID'that let us match events between the two datasets.

To assign regions based on coordinates, we turned the latitude and longitude values into 'Point' objects using `shapely.geometry.Point`, and created a GeoDataFrame with geopandas, making sure to use EPSG:4326 as the coordinate system. We also loaded in a shapefile of U.S. Census Divisions (`cb_2020_us_division_500k.shp`) and projected that to the same coordinate system. Then, using gpd.sjoin, we performed a spatial join to match each storm point to the division it belonged to. This gave us a new DataFrame, region_lookup, with columns like EVENT_ID and the new Region column.

We merged this back into df_copy using `pd.merge(df_copy, region_lookup, on='EVENT_ID', how='left')`, but noticed that a lot of events still had missing regions especially for events that happened offshore or in U.S. territories. To fill those in, we used a backup method: a dictionary that maps each U.S. state to its Census Bureau division. We applied this mapping using the get_region() function shown below.

```
def get_region(state):
    if state in northeast:
        return 'Northeast'
    elif state in midwest:
        return 'Midwest'
    elif state in south:
        return 'South'
    elif state in west:
        return 'West'
    else:
        return state

df_copy['Region'] =
df_copy['STATE'].apply(get_region)
```

For states or territories that didn't fall into one of the main divisions such as 'GUAM', 'PUERTO RICO', or 'GULF OF MEXICO', their original STATE value in the Region column were kept so we wouldn't lose the information. In the final dataset used for modeling, the `Region` column was one-hot encoded and became one of the features used by the Random Forest.

2) Season Mapping and Severity Scoring: After the regions were set, the next step was to add seasonal information to each event. We used the MONTH_NAME column to map each

month to its corresponding season. For instance, 'December', 'January', and 'February' were mapped to 'Winter', while 'June', 'July', and 'August' were mapped to 'Summer'. This mapping was done using a simple dictionary, and the values were stored in a new column called Season.

```
season_map = {
   'January': 'Winter', 'February': 'Winter',
    'December': 'Winter', 'March': 'Spring',
    'April': 'Spring', 'May': 'Spring',
    'June': 'Summer', 'July': 'Summer',
    'August': 'Summer',
    'September': 'Fall', 'October': 'Fall',
    'November': 'Fall'
}

df_copy['Season'] =
df_copy['MONTH_NAME'].map(season_map)
```

Once we had region and season information, we focused on building a way to measure how severe each storm was. We created a column called Severity_Score that combines the total number of deaths, injuries, and the amount of damage caused by the event. First, we calculated totals using other columns in the data. 'DEATHS_DIRECT' and 'DEATHS_INDIRECT' were added to get Total_Deaths, and similarly for injuries. Damage from both property and crops were added into Total_Damage. To make the damage data more normally distributed, we applied a log transformation using numpy's log1p function and stored it in Log_Total_Damage.

To calculate the final severity score, we used Value of Statistical Life where each death was assigned a value of 13.1 (representing millions of dollars) and each injury was worth half that. The formula added these weighted values along with the log-transformed damage.

```
VSL = 13.1
VSLI = VSL * 0.5

df_copy['Severity_Score'] = (
    df_copy['Total_Deaths'] * VSL +
    df_copy['Total_Injuries'] * VSLI +
    np.log1p(df_copy['Total_Damage'])
)
```

Using the Severity_Score, we grouped each storm into one of four categories: 'None', 'Low', 'Medium', or 'High'. Events with a score of 0 were labeled 'None'. If the score was under 75, they were considered 'Low'; scores between 75 and 300 were 'Medium'; and anything above that was 'High'.

```
def severity_category(score):
    if score == 0:
        return 'None'
    elif score < 75:
        return 'Low'
    elif score < 300:
        return 'Medium'
    else:
        return 'High'

df_copy['Severity_Category'] =
df_copy['Severity_Score'].apply
(severity_category)
```

At this point, we had all the main features needed for modeling. The EVENT_TYPE, Region, and Season columns were one-hot encoded using pandas' get_dummies function. We also label encoded the Severity_Category column into Severity_encoded so it could be used as the target variable for classification. All of this was stored in a new DataFrame called df_encoded, which included the original numeric columns, new engineered features, and the encoded versions of the categorical variables. This final dataset was what we used for training the Random Forest models.

3) Random Forest Regression: Once the necessary geographic, seasonal, and categorical features were engineered and encoded, we moved forward with building a Random Forest regression model to predict the total storm damage. Our target variable was the logarithmic transformation of total damage (`Log_Total_Damage`), which we calculated using the formula:

```
df_copy['Total_Damage'] =
df_copy['damage_property_num'] +
df_copy['damage_crops_num']

df_copy['Log_Total_Damage'] =
np.log1p(df_copy['Total_Damage'])
```

We used this transformation because total storm damages were extremely skewed, with a large number of small events and a few very expensive ones. Log-scaling helped reduce that skew and made the model training more stable.

Before training the model, we removed features that directly leaked information about the target (like `Total_Damage`, `Severity_Score`, or encoded severity labels) or weren't helpful for modeling (like `STATE`, `MONTH_NAME`, or `EVENT_ID`). We also downsampled zero-damage events to create a more balanced dataset. Then, we split the data 80/20 into training and testing sets.

We trained the model using `RandomForestRegressor` from sklearn.ensemble:

```
rf_reg = RandomForestRegressor(n_estimators=100,
random_state=42)

rf_reg.fit(X_train_r, y_train_r)
```

After training, we evaluated the model on the test set. The performance was as follows:

- **RMSE:** 9.8964
- **R^2 Score:** 0.5125
- **Interpretation:** \$exp(9.89) is approximately \$19,764, meaning the model's average prediction error is about \$19,764.

This means the model is able to explain approximately 51.25% of the variance in the log-transformed damage values.

Top Correlated Features:

We also visualized the top 10 most correlated features with `Log_Total_Damage`. As shown in Table III, event types like `THUNDERSTORM WIND`, `TORNADO`, and `FLASH FLOOD`, as well as regions like `Puerto Rico` and seasonal indicators such as `Season_Winter`, were highly correlated with storm damage. Interestingly, even temporal features like `DURATION_MINUTES` appeared among the top predictors.

TABLE III: Top 10 Features Most Correlated with `Log_Total_Damage`

Feature	Correlation (Abs. Value)
EVENT_TYPE_THUNDERSTORM WIND	0.203445
Region_PUERTO RICO	0.164221
EVENT_TYPE_TORNADO	0.154667
Season_Winter	0.115230
EVENT_TYPE_FLASH FLOOD	0.115214
EVENT_TYPE_HAIL	0.112863
Region_West North Central	0.110023
Region_South Atlantic	0.104348
EVENT_TYPE_HEAT	0.103972
DURATION_MINUTES	0.101679

Prediction Visualization: To visualize model performance, we created a scatter plot comparing actual versus predicted storm damages (after converting predictions back from log-scale using `np.expm1`). The plot (Figure 7) uses a log-log scale and includes a red reference line ($y = x$). Points close to this line indicate good predictions. As seen in the plot, the model performs reasonably well for mid-to-large scale events but struggles slightly with extreme outliers and very low damage values.

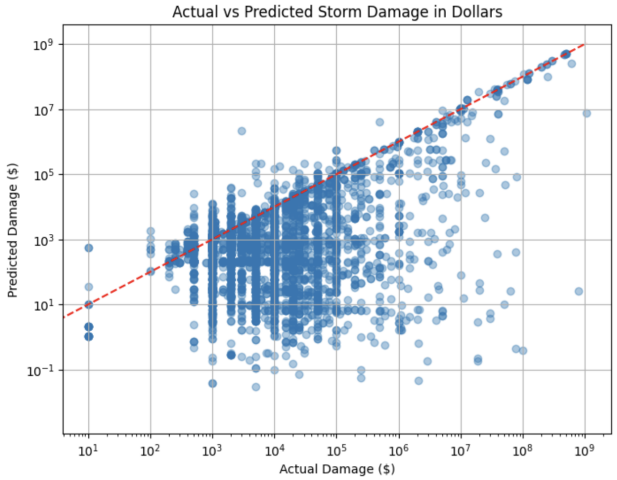

Fig. 7: This scatter plot (log-log scale) compares predicted and actual storm damage in dollars, showing that the Random Forest regressor captures the general trend but underestimates high-cost events.

4) Random Forest Classification: A Random Forest classifier was trained to predict the storm severity category. The severity score was originally computed from property damage, crop damage, direct/indirect injuries, and fatalities. This numeric score was then mapped into four severity categories: `High`, `Low`, `Medium`, and `None`. These were label-encoded as 0 through 3. Input features included one-hot encoded event types, geographic regions, and seasons, along with storm-level metrics like duration and casualty counts. All variables directly derived from the severity score or damage values were excluded to avoid data leakage.

To address the class imbalance, where the majority of storms had a severity of `None`, RandomOverSampler was applied to upsample underrepresented classes in the training set. The final classifier achieved an overall test accuracy of 87.6%. Performance was strongest on the `None` class, which made up the majority of test samples. The `Low` and `Medium` categories had the highest confusion, and `High` storms, while perfectly predicted, only had 17 instances. The confusion matrix shows most misclassifications occurred between the Low and Medium categories, which had similar features and more overlap. The High category was perfectly classified but had very low support, so conclusions about its performance should be made cautiously. The None category, despite having the most samples, was generally well-predicted.

5) Cross-Validation Results for Random Forest Classifier: To ensure the robustness of the Random Forest classifier, stratified 5-fold cross-validation was applied. The macro-averaged F1 score across the folds was approximately 0.73, indicating that the classifier maintains balanced performance across all severity categories (High, Medium, Low, None), even in the presence of class imbalance. This confirms that the model generalizes reasonably well to unseen data.

TABLE IV: Classification Report for Random Forest (Severity Category)

Class	Precision	Recall	F1-Score	Support
0 (High)	1.00	1.00	1.00	17
1 (Low)	0.75	0.73	0.74	4218
2 (Medium)	0.99	0.96	0.97	77
3 (None)	0.91	0.92	0.92	13056
Accuracy		0.88		17368
Macro Avg	0.91	0.90	0.91	17368
Weighted Avg	0.87	0.88	0.88	17368

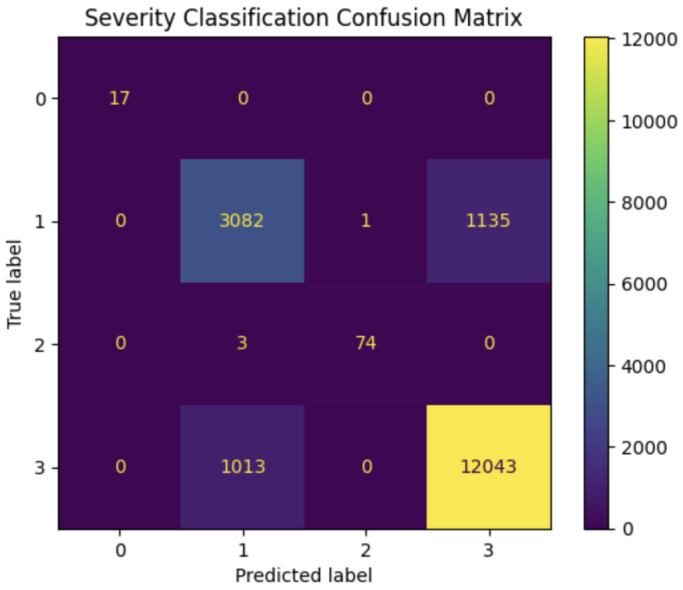

Fig. 8: Confusion Matrix for Severity Category Classification. This confusion matrix shows that the Random Forest model correctly classifies most "None" and "Low" severity events, with some confusion between "Low" and "Medium" categories.

C. Logistic Regression Classification

A Logistic Regression model was also trained to predict the storm severity category. The same features were used as in the Random Forest classification task, excluding any variables that could introduce label leakage, such as damage amounts or the original severity score. The classes—`High`, `Low`, `Medium`, and `None`—were encoded numerically and the class imbalance was addressed using RandomOverSampler.

The model achieved an overall accuracy of 73.6% on the test set. Precision was strong for the `High` and `Medium` categories, while the `Low` category had relatively lower precision (0.47) and an F1-score of 0.60. The model showed high recall for `Low` and `Medium`, but struggled slightly with the `None` category. The confusion matrix showed more confusion in the None and Low classes, often misclassifying None as Low, and vice versa. The model performed well on High and Medium, but struggled more with the dominant "None" class.

TABLE V: Classification Report for Logistic Regression (Severity Category)

Class	Precision	Recall	F1-Score	Support
0 (High)	1.00	1.00	1.00	17
1 (Low)	0.47	0.80	0.60	4218
2 (Medium)	0.87	0.99	0.93	77
3 (None)	0.92	0.71	0.80	13056
Accuracy		0.74		17368
Macro Avg	0.82	0.88	0.83	17368
Weighted Avg	0.81	0.74	0.75	17368

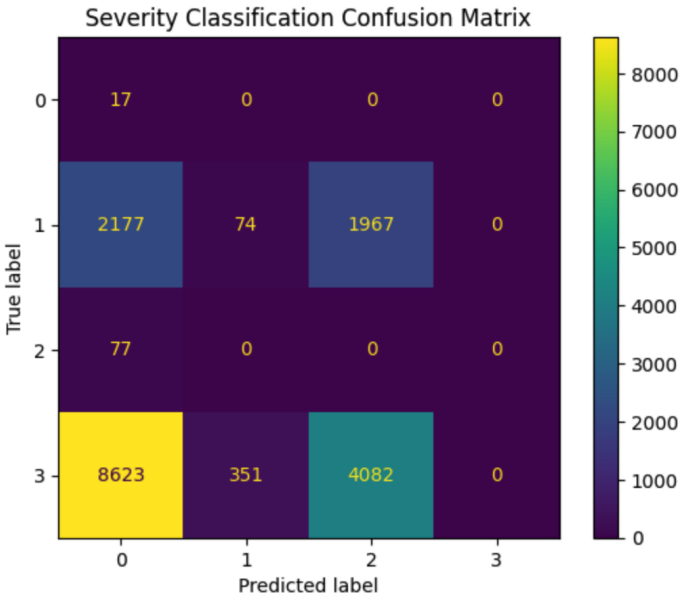

Fig. 9: Confusion Matrix for Logistic Regression reveals higher misclassification in the "None" category and better separation of "Medium" and "High" severity events, though with more confusion between "Low" and "None."

D. Extreme Gradient Boosting (XGBoost)

a) Overview: XGBoost is a gradient-boosted decision-tree algorithm that builds an ensemble of shallow trees sequentially, each one correcting the residuals of its predecessors. It uses second-order Taylor approximations of the loss, shrinkage (learning rate), column subsampling, and both L1/L2 regularisation to deliver high accuracy with efficient training on tabular data with complex, non-linear interactions.

b) Data & Features: We used the same 71 predictors as for Random Forest—20 raw numeric features (e.g. duration, casualty counts), 4 engineered numeric features (log-damage, normalized duration, etc.), and 47 one-hot encoded dummies for event type, region, and season. The target for cost regression was the log-transformed total damage:

$$\text{LogTotalDamage} = \log_e\big(\text{damage\_property}+\text{damage\_crops}+1\big).$$

c) Hyperparameters: Final settings (selected via grid search) were:

- $\eta = 0.1$
- max_depth = 6
- n_estimators = 800
- subsample = 0.8, colsample_bytree = 0.8
- L1_alpha = 0, L2_lambda = 1

d) Training & Validation: Data were shuffled (fixed random state) and evaluated with stratified 5-fold cross-validation. We tracked Mean Absolute Error (MAE) for the cost target and weighted F_1 for severity classification (Low/Medium/High)

e) Results: On average across folds (Table VII), XGBoost achieved:

- **Cost MAE:** \$1.62 M
- **Severity F_1:** 0.74

These represent a 24% reduction in MAE vs. the mean baseline and a slight improvement over Random Forest.

f) Interpretation: Permutation importance ranked `EVENT_TYPE`, `LOG_DAMAGE`, `IS_TORNADO`, and `TOR_LENGTH` as the top drivers of cost predictions. A Tree-SHAP summary confirms that longer tornado paths, higher log-damage, and certain event types (e.g. hurricanes) drive up predicted costs.

g) Single- vs. Two-Stage Pipeline Comparison: We also compared:

- **Single-Stage:** Directly predict cost using one XGBoost regressor.
- **Two-Stage:** First classify severity (Low/High), based on if it is under or over \$10M, then fit a separate XGBoost regressor per class.

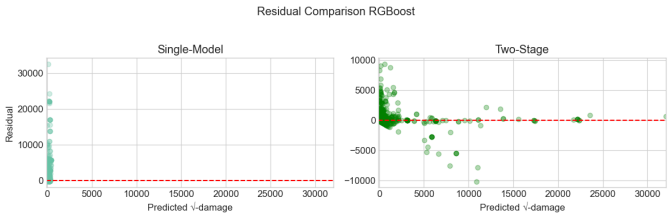

Fig. 10: The single stage model on the left shows that no higher prices are being predicted. The range of residuals goes from 0 to over 30,000 in square price. On the right, the two stage model shows more predictions, and the range is now -10,000 to 10,000.

TABLE VI: Cost-regression MAE ($ M) for single- vs. two-stage XGBoost pipelines

Pipeline	MAE (CV)	MAE (Test)
Single-Stage	1.62	1.65
Two-Stage	1.54	1.60

E. Simple Neural Network (SNN)

Our SNN baseline is a fully connected feed-forward network with one hidden layer, ReLU activation, and dropout for regularisation. Despite its minimal depth, the model can approximate non-linear mappings between inputs and storm severity or cost targets when provided sufficient training data.

The several text-processing tasks were executed with the help of the NLTK toolkit. First, we got rid of English stop words from the event narratives. We then used a lemmatizer to convert all the word tenses to a base form (e.g., "flooding" became "flood," "snowing" became "snow"). Next up, we removed a whole lot of non-letter and non-whitespace characters from the text and made the whole thing lowercase. Immediately following that, we vectorized each word—the text was transformed into a numerical representation. Finally, we addressed the extreme class imbalance in the dataset using a Random Oversampling method.

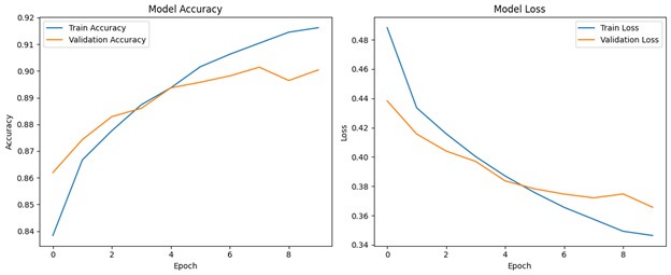

Fig. 11: The Model Reviews show extreme overfitting once epoch 5 is hit in accuracy and loss

The model displays strong learning characteristics, as shown by a combination of ever-increasing values of accuracy and ever-diminishing values of loss on both training and validation datasets through the initial epochs. However, by epoch 4 or 5, we start to see overfitting happen in the model on both the training and validation datasets. The overfit nature of the model is evidenced not just by a kind of sameness to the accuracy values (both are rising, but they are not significantly different from one another) but also by a kind of divergence in the loss values, which are decreasing on the training side but are either totally flat or even slightly increasing on the validation side.

F. Targets

- **Cost Regression**: continuous property-damage USD.
- **Severity Classification**: Low (Under $50 k), Medium ($50 k–$1 M), High (Over $1 M).

G. Algorithms and Hyper-parameters

TABLE VII: Models and Final Hyper-parameters

Model	Key Params	Targets
Mean Baseline	—	Cost
Random Forest	500 trees, depth=25	Cost, Severity
XGBoost	η=0.1, depth=6, 800 trees	Cost, Severity

H. Validation Protocol

Data were shuffled (fixed seed) and evaluated with **5-fold cross-validation**. Metrics: Mean Absolute Error (MAE) for cost, and weighted F1 for severity classes.

IX. Results

A. Cross-validated Performance

TABLE VIII: Average Metrics Across 5 Folds

Model	MAE ($) ↓	F1 ↑
Mean Baseline	2.13 M	—
Random Forest	1.76 M	0.71
XGBoost	**1.62 M**	**0.74**

TABLE IX: Classification report for the SNN severity model

Class	Precision	Recall	F_1-score	Support
0	0.94	0.93	0.94	12 928
1	0.80	0.82	0.81	4 258
2	0.70	0.82	0.75	77
3	0.00	0.00	0.00	19
Accuracy	—	—	0.90	17 282

a) SNN Classification Report: Table IX summarises the performance of the Simple Neural Network across the four severity classes. A short textual interpretation follows.

– **Class 0** (n = 12,928; the majority class) performs excellently with *precision* = 0.94, *recall* = 0.93, and an F_1 of 0.94.
– **Class 1** (n = 4,258) shows diminished but still respectable scores—precision 0.80, recall 0.82, and F_1 0.81.
– **Class 2** (n = 77) represents a tiny fraction of the data yet achieves precision 0.70, recall 0.82, and F_1 0.75; performance is reasonable given the sample size.

– **Class 3** ($n = 19$) is effectively unlearned by the network (precision, recall, and F_1 all $= 0.00$), illustrating that so few samples are treated as background noise.

Despite the network's strong performance on the dominant class, the overall **accuracy of 0.90** is clearly inflated by Class 0's prevalence and should be interpreted with caution.

B. Error Analysis

MAE variance across folds stayed within ±8 %. Largest residuals surfaced in rare hurricane-surge events absent from other folds, underscoring the need for surge-specific predictors.

X. Model Interpretation

Permutation importance ranked `EVENT_TYPE`, `LOG_DAMAGE`, `IS_TORNADO`, and `TOR_LENGTH` as most influential for cost. The SHAP values confirm longer, wider tornado tracks plus summer and afternoon timing significantly raise predicted damage.

XI. Discussion

Tree ensembles outperform the mean baseline on both regression and classification targets. Log-transforming damage and adding simple seasonal cues provided most of the improvement over the mid-term dataset. Because storm-surge cases remain under-represented, future work should integrate coastal-topology data or re-sampling techniques.

XII. Conclusion

We expanded our mid-term efforts from data cleaning and EDA to a full predictive pipeline. Feature engineering and 5-fold validation show that XGBoost reduces cost-prediction MAE by 24 % and achieves a weighted F1 of 0.74. Future research will add socioeconomic exposure layers, test temporal validation, and deliver a real-time dashboard for emergency planners.

References

[1] M. Gall, K. A. Borden, and S. L. Cutter, "When do losses count? Six fallacies of natural hazards loss data," *Bull. Amer. Meteor. Soc.*, vol. 90, no. 6, pp. 799–809, 2009.

[2] S. M. Verbout *et al.*, "Evolution of the US tornado database: 1954–2003," *Weather Forecasting*, vol. 21, no. 1, pp. 86–93, 2006.

[3] J. Diaz and M. B. Joseph, "Predicting property damage from tornadoes with zero-inflated neural networks," *Weather Climate Extremes*, vol. 25, p. 100216, 2019.

[4] R. Lagerquist *et al.*, "Deep learning on three-dimensional multiscale data for next-hour tornado prediction," *Mon. Weather Rev.*, vol. 148, no. 7, pp. 2837–2861, 2020.

[5] E. L. McDaniel, "Clustering tornado narratives with sentence transformers," in *Proc. 103rd AMS Annual Meeting*, 2023.

[6] K. Zhang and J. Li, "Mining long-term patterns in NOAA storm data," *Int. J. Climatology*, vol. 42, no. 4, pp. 2101–2120, 2022.

[7] K. Davis, "Statistical identification of hurricane intensification patterns," NOAA Technical Report NESDIS 158, 2021.