Systematic Analysis of Wildfires in the United States

CMPE 255 - Data Mining

Team 12

Team Members:
Akanksha Joshi - 016690830
Manisha Miriyala - 016522896
Mansi Bhagwat - 016709082
Nandakishor S - 016054753

Github Link: https://github.com/akanksha276/wildfire_analysis

Table Of Contents

1. Introduction

1.1 Objective

This project aims to analyze a dataset of 1.88 million records and 39 features about US wildfires from 1992 to 2015. Using data mining techniques, it seeks to predict wildfires' location, timing, and severity. By considering technical and social factors, this project promotes interdisciplinary thinking to address the complex challenges of society and the environment.

1.2 Motivation

The rise in frequency and intensity of wildfires due to climate change highlights the need to comprehend their underlying patterns and causes. This project will examine a dataset that can offer valuable insights for policymakers and emergency responders to develop effective strategies to reduce wildfire risks and save lives.

1.3 Literature Review

- The impact of wildfires on society and the environment has led to a growing interest in developing accurate and efficient methods for predicting their occurrence and severity.
- Machine learning algorithms have been increasingly used in recent years to predict wildfires due to their ability to handle large datasets and capture complex patterns and interactions between variables.
- A study in Iran [1] used three machine learning models to predict forest fire susceptibility using various predictors. The boosted regression tree (BRT) model outperformed the others with an AUC value of 0.91.
- The authors in [2] utilized a random forest algorithm to predict the likelihood of wildfire, taking into account various meteorological, topographical, and human-related factors.
- The study in [3] tested multiple machine learning algorithms for mapping wildfire susceptibility in the Mediterranean Region of Turkey, with CatBoost achieving the highest accuracy (95.47%).
- Another study used a deep neural network to predict the spread and intensity of wildfires, achieving high accuracy rates and outperforming other machine learning models.
- Overall, the use of machine learning algorithms for wildfire prediction shows promising results and has the potential to aid in the development of effective strategies to mitigate wildfire risks and protect society and the environment.

2. System Design & Implementation details

2.1 Algorithm Selected

We first tried to train simple classification models such as logistic regression and linear SVC, however, they were not able to capture the intrinsic relationship between features and were difficult to support computationally. Thus, we tried larger and more complex algorithms such as
Extra Trees, Random Forest, XGBoost, and Neural Networks. These are popular machine learning algorithms that can be well-suited for a tabular dataset. These algorithms are capable of handling sophisticated and heterogeneous data with 1.8 million data points. Random forest, XGBoost, and Extra Trees are powerful ensemble learning algorithms that have a proven track record of success in solving various machine learning problems.

2.2 Technologies & Tools Used

To tackle the US wildfires prediction problem, we utilized a range of technologies including Scikit-learn, Pandas, Matplotlib, NumPy, and XGBoost. These powerful open-source tools provided us with robust data analysis, visualization, and modeling capabilities, enabling us to effectively preprocess the data, select relevant features, and build high-performance machine learning models for wildfire prediction.

2.3 Model Architectures

2.3.1 Random Forest

Random Forest Regressor is an ensemble-based machine learning algorithm that utilizes multiple decision trees to build a robust predictive model. The primary concept behind this approach is to generate a vast number of decision trees, forming a "forest," and then utilize them to produce a prediction by averaging the output of each tree. Random Forest Regressor is specifically designed for regression tasks, where the objective is to forecast a continuous numerical value.

2.3.2 XGBoost

XGBoost (short for Extreme Gradient Boosting) is a machine learning algorithm that belongs to the family of gradient boosting algorithms. It is a powerful and efficient open-source implementation of the gradient boosting algorithm, which uses decision trees as base learners and employs a set of advanced techniques to improve its performance and speed.

2.3.3 Extra Trees

Extra Trees is a machine learning algorithm that, like random forests, generates multiple decision trees. However, Extra Trees samples data for each tree without replacement and randomly selects a splitting value for a feature, instead of computing an optimal value. This leads to more diversified and uncorrelated decision trees, improving the algorithm's performance. It was a good option for this problem as computation cost is a concern and this algorithm works well in such situations.

2.3.4 Neural Network

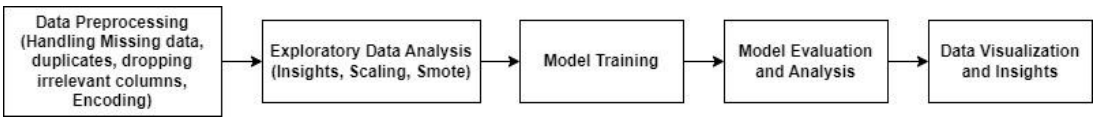
The Neural network used in this project used a kernel size of 180, 120, 60 nodes in each of the hidden layers respectively. Each node uses ReLU(Rectifier Linear Unit) activation function and Adam Optimization algorithm for optimizing the weights. The training is done for 100 epochs with an adaptive learning rate($0 < \text{alpha} \leq 0.001$).

3. Proof of concept evaluation

3.1. Dataset

Dataset name: 1.88 Million US Wildfires
Dataset content: Spatial information about wildfires that occurred in the United States from 1992 to 2015.
Dataset source: Kaggle
Type of data: SQLite database containing int, float, and categorical values
Size of data: 1880465 rows and 39 columns

3.2. Methodology Followed



3.2.1 Preprocessing

The SQLite database was loaded as a dataframe with the help of the pandas read_sql command. The dataset had to be split into train and test sets. Before the data could be split into train and test, we had to handle duplicate data, missing data and encode any categorical variables.

3.2.2 Duplicate Data

Duplicates were found and dropped with the help of .drop_duplicates()

3.2.3 Missing Data

To deal with missing data, we first found columns that had null values with the help of isna().sum(). Some columns such as LOCAL_FIRE_REPORT_ID, LOCAL_INCIDENT_ID, MTBS_ID, ICS_209_INCIDENT_NUMBER, MTBS_FIRE_NAME, and ICS_209_NAME were dropped directly as they would not give us any vital information. COUNTY, FIPS_CODE, and FIPS_NAME had the same information thus we dropped two of the columns and kept only FIPS_NAME. Columns such as FIRE_NAME and FIPS_NAME had missing data and we handled that by replacing the NaN values with an 'Unknown' string keyword. At the end of this step, we had 892007 rows and 29 columns, and zero missing values.

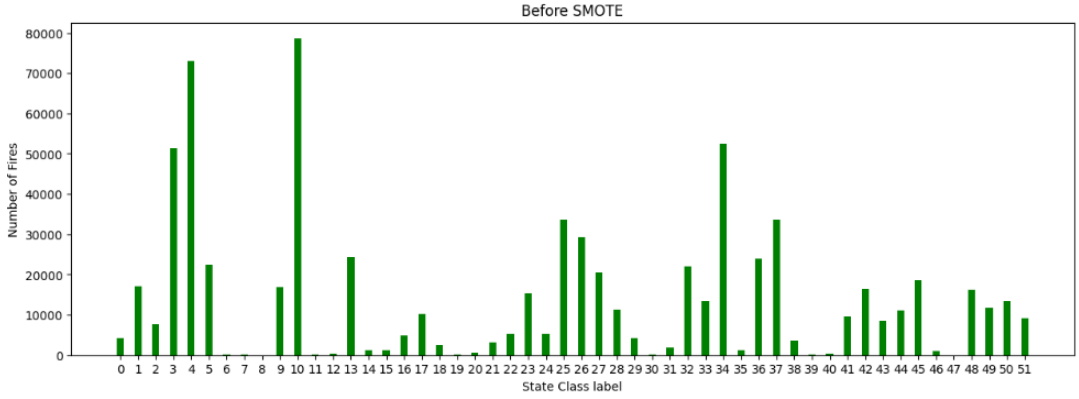
3.2.4 Data Encoding

We have to ensure that the same encoding is applied consistently to both the training and test sets, which is why we need to apply encoding before we split the data. Categorical data has to be encoded. We made use of both label encoding as well as one-hot encoding. Label encoding is useful for ordinal variables (i.e., variables with an inherent order) and when the number of unique categories is large. In our dataset label encoding was done for columns such as FIRE_SIZE_CLASS as this column tells us the severity of the fire size in increasing order. One hot encoding is useful for nominal variables (i.e., variables with no inherent order) and when the number of unique categories is small. Hence, we did one-hot encoding for columns such as STAT_CAUSE_DESCR and FIPS_NAME.

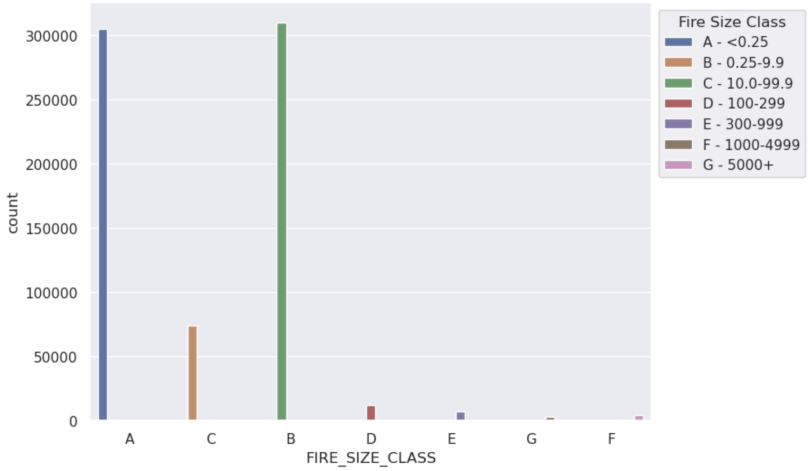
After these three data preparation steps, our dataset was split into train and test csv's with the help of sklearn's train-test split function in the ratio 80:20. The train csv contained 713605 rows × 1584 columns and the test csv contained 178402 rows × 1584 columns.

3.2.5 EDA

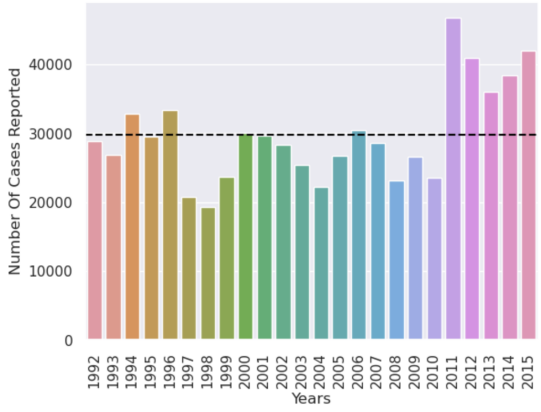
Splitting data into training and testing sets is a common machine learning practice, with a typical ratio of 80:20. However, the optimal ratio varies depending on dataset size and task. A larger training set leads to a well-trained model, but may result in a higher variance when evaluating on a smaller test set. The split ratio should be based on the specific problem, with testing on unseen data essential for generalization. Imbalanced classification is challenging due to a scarcity of minority class samples, hindering decision boundary learning. A graph displays sample classification per state.



To solve uneven class distribution, we have used SMOTE (Synthetic Minority Oversampling Technique) which is an oversampling technique where the synthetic samples are generated for the minority class. Below graph shows that after applying SMOTE, there are 78608 samples classified for each state.



The countplot displays the frequency of each category in the "FIRE_SIZE_CLASS" column on the x-axis, and the number of occurrences on the y-axis. The hue parameter is used to group the data by the same"FIRE_SIZE_CLASS" category, and display each category in a different color.



The resulting bar chart shows the count of each category of "FIRE_SIZE_CLASS" along the x-axis and the number of occurrences of each category along the y-axis. The plot includes a horizontal dashed line that represents the mean value of the data.

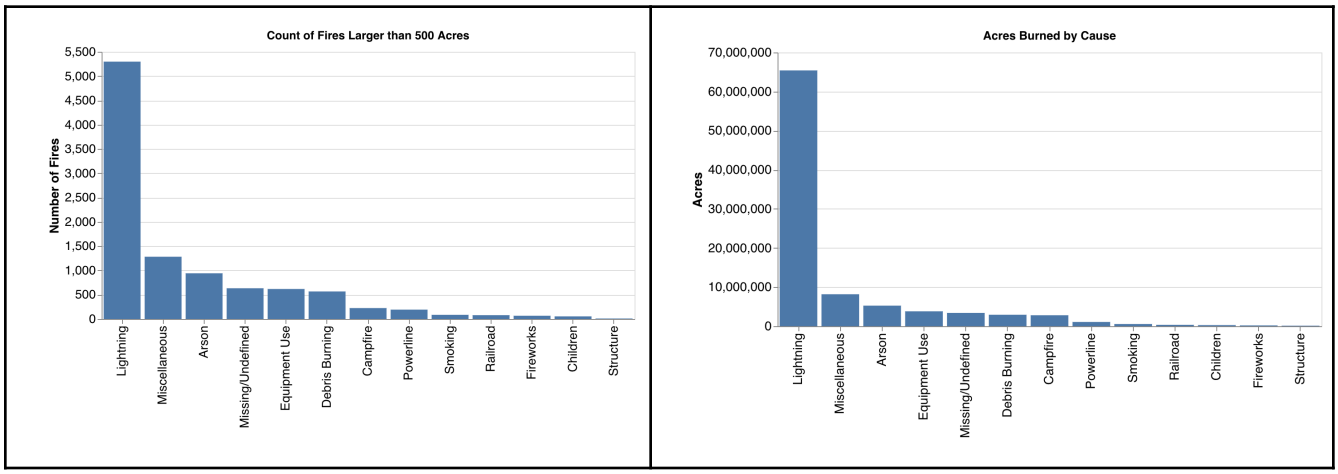


Fig:

a. Altair library is used to create the bar chart, which displays the count of fires by the "STAT_CAUSE_DESCR" column along the x-axis and the number of fires along the y-axis. The chart is given a title that includes the value of the threshold.

b. Bar chart displays the total acres burned by the cause of each fire in the "fires" dataset. Calculates the sum of the "FIRE_SIZE" column for each group and sorts the results in descending order. The chart shows the cause of each fire along the x-axis and the total acres burned along the y-axis.

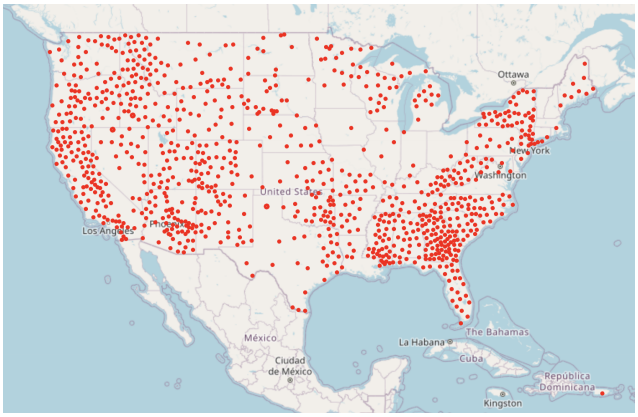


Fig: The center of each cluster is determined by the mean latitude and longitude of the data points in the cluster. The size of each circle marker is determined by the sum of the 'FIRE_SIZE_SCALED' values for the data points in the cluster. The resulting map displays the location and relative size of each cluster.

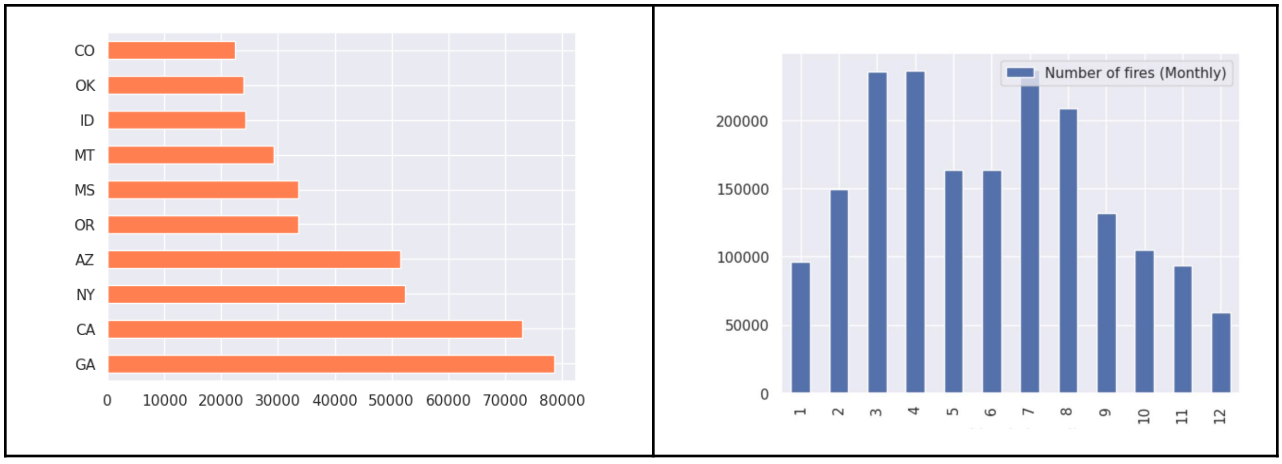


Fig:

a. Horizontal bar chart using Matplotlib library that shows the top 10 states with the most fires in the dataframe 'df'. The code first counts the number of occurrences of each unique state in the "STATE" column of the dataframe, and then selects the top 10 states with the most occurrences.

b. A bar chart showing the number of wildfires that occurred in each month of the year.

3.2.6 Models

Problem 1: Predict the US State in which the wildfire was detected

- **Extra Trees**

	Precision	***Recall***	***f1***
Accuracy			0.73
Macro avg	0.74	0.71	0.72
Weighted avg	0.76	0.73	0.74

- **Random Forest**

	Precision	***Recall***	***f1***
Accuracy			0.63
Macro avg	0.63	0.63	0.63
Weighted avg	0.66	0.63	0.64

- **XGBoost**

	Precision	***Recall***	***f1***
Accuracy			0.71
Macro avg	0.71	0.72	0.71
Weighted avg	0.72	0.72	0.72

- **Neural Network**

	Precision	***Recall***	***f1***
Accuracy			0.83
Macro avg	0.78	0.76	0.75
Weighted avg	0.85	0.83	0.83

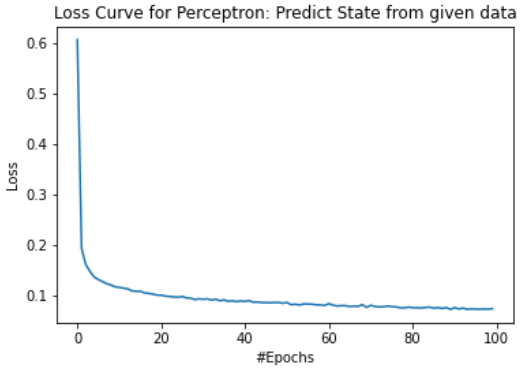


Fig: Loss Curve for Perceptron: Predict the US State

- **Neural Network**

	Precision	***Recall***	***f1***
Accuracy			0.78
Macro avg	0.77	0.78	0.77
Weighted avg	0.77	0.78	0.77

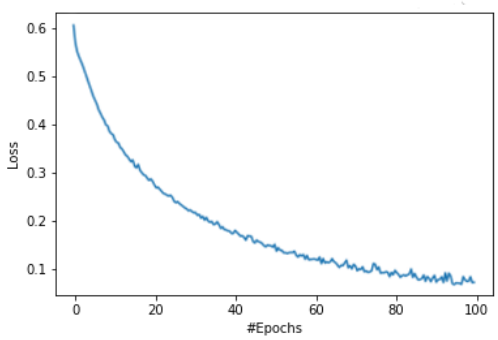


Fig: Loss Curve for Perceptron: Predict the fire severity

3.2.7 Analysis of results

The best results were achieved for predicting the fire class severity class and the US state with the help of neural networks. The neural network got 0.78 accuracy on fire class severity and 0.83 on predicting the state where the fire occurred.

4. Discussion & Conclusions

4.1 Decisions Made:

1. Identifying datasets that are suitable for techniques learned in class and meet the feature and sample size requirement.
2. Performing EDA (Exploratory data analysis)
    - Analyze columns of the "Fires" table from a SQLite database on Kaggle.
    - Determine correlations between columns via a matrix.
    - Eliminate duplicates and unnecessary ID columns.
    - Rescale numerical data to [0,1] and encode categorical data using one-hot encoding for X and LabelEncoder for y.
    - Split data into Training and Test sets, use Test set to evaluate best model for target prediction.
    - Address class imbalance with SMOTE.
    - Divide Training set into train and validation subsets for each model.
    - Evaluate the best model's performance on the Test set with new data.

4.2 Difficulties faced

We encountered challenges when working with a large dataset of over 8.5 GB, which could not be processed by our local machines. To overcome this, we had to resort to using Google Colab and Cloud servers with adequate memory. In addition, running GridSearchCV for hyperparameter tuning was computationally expensive and we opted to use Bayesian search instead. While encoding non-numeric features, we had to decide between LabelEncoder and OneHotEncoder. LabelEncoder assigns numerical values to each unique class label, which could lead to a model misinterpreting correlations. OneHotEncoder generates new columns for each unique value, solving the correlation problem but increasing the data size and leading to memory errors. We had to drop uninformative columns like "FIPS_NAME" and rely on "LATITUDE" and "LONGITUDE" instead.

4.3 Things that worked

We were able to solve memory issues by finding alternative methods mentioned below:

- Use Bayesian search instead of GridSearchCV for Hyperparameter tuning
- Drop FIPS_NAME which generated 1523 new columns and keep LATITUDE and LONGITUDE columns
- Neural Network model gave us the best F1 score and we used this model to predict unknown test data.

4.4 Things that didn't work well

- The LinearSVC model gave a very low F1 score (0.18) and accuracy score (0.51) on the train and validation set. It was also taking very long to run and was computationally heavy. Hence, we decided to drop this model.

5. Task Distribution

Name	**Tasks**
Akanksha Joshi	Worked on data preparation to create train and test csv, Performed clustering using DBScan to visualize fire size on USA map for EDA, Wrote script for XGBoost, Wrote script for hyper-parameter tuning using grid search, bayesian search
Manisha Miriyala	Worked on EDA, and compiling all EDA results, Worked on Random Forest Regression and Classification
Mansi Bhagwat	Worked on EDA, Worked on encoding data using LabelEncoder and One Hot Encoder, Implemented SMOTE technique, Implemented Linear SVC and ExtraTreesClassifier
Nandakishor S	Worked on EDA, Neural Network implementation

References

[1] B. Kalantar et al., "Forest Fire Susceptibility Prediction Based on Machine Learning Models with Resampling Algorithms on Remote Sensing Data". *Remote Sens.* 2020, *12*, 3682. https://doi.org/10.3390/rs12223682

[2] M. Rodrigues and J. de la Riva, "An insight into machine-learning algorithms to model human-caused wildfire occurrence", *Environmental Modelling & Software*, vol. 57, pp. 192-201, 2014, ISSN: 1364-8152, doi: 10.1016/j.envsoft.2014.03.003.

[3] K. Sohaib et al., "Wildfire Susceptibility Mapping Using Five Boosting Machine Learning Algorithms: The Case Study of the Mediterranean Region of Turkey", *Advances in Civil Engineering*, vol. 2022, Article ID 3959150, 18 pages, 2022. https://doi.org/10.1155/2022/3959150