Predicting Wildfire Occurrence and Severity in California with Machine Learning on Apache Spark

Authors: Tang, Serena, Sen, Sophathriya, Christopher Lee

Department of Information Systems, California State University

Los Angeles

Tel. 323-343-2916, Fax. 323-343--5209

e-mail:[**xtang13@calstatela.edu**](mailto:xtang13@calstatela.edu)**,** [**ssen5@calstatela.edu**](mailto:ssen5@calstatela.edu)**, clee219@calstatela.edu**

**Abstract:** Wildfires pose a significant threat to California, driven by natural and human factors. Our project uses machine learning techniques to predict wildfire occurrence and severity across the state. We integrate nine datasets (more than 2GB) covering Historical Wildfire Impact Data, Meteorological Data, Vegetation and Land Cover Data, Topography Data, and Human Activity Data, ensuring consistency by aligning their time ranges. Our approach employs regression models to estimate fire severity and classification models to predict fire occurrence. Specifically, we implement Linear Regression, Random Forest Regression, Gradient Boosted Trees (GBT) Regression, and Support Vector Regression (SVR) to evaluate wildfire severity, while classification models will assess the likelihood of wildfire events.

**1. Introduction**

Predicting wildfires is crucial for improving preparedness, response strategies, and resource allocation in California, a state that experiences frequent and devastating wildfires. Early prediction models can help reduce the economic, environmental, and social impacts by enabling timely interventions and minimizing property damage, loss of life, and long-term ecological harm. With our selected algorithms of Random Forest, Linear Rregression, and Gradient Boosted Trees (GBT), we further apply cross-validation and hyperparameter tuning to enhance model performance, ensuring robustness and generalizability. Additionally, a feature importance analysis will be conducted to identify key factors influencing wildfire trends. By leveraging these insights, our study aims to contribute to more effective wildfire prevention and mitigation strategies for California State.

2. Related Work

As of recent news, California is one of the few states facing numerous wildfires, especially in the severe heat of California’s summer. Several studies have been conducted using machine learning to predict wildfire behaviors in California to gain insights into how the state would be able to assess the situation better. For instance, USC’s researchers developed models that use satellite imagery with artificial intelligence; researchers can reverse-engineer wildfire behavior by analyzing historical wildfire data to identify key factors that drive these damages—tracking ignition causes, spread, and containment—resulting in meaningful insights into patterns like weather condition, fuel type, and topography to simulate fire progression [2]. Unlike their approach, which relies heavily on deep learning and real-time static image classification, our work integrates layers of key features from Google Earth Engine—converted from satellite images to pixel—then transferred to Spark to further create pipelines ready for learning. This would further enable large-scale, distributed wildfire classification by using traditional machine learning algorithms such as regression and classification models. Similarly, a study published utilizing machine learning algorithms predicting wildfire growth rate daily, both on regional and global scales, utilizing various factors like meteorological, topography, and fuel loads with numerous machine algorithms that include Random Forest and XGBoost [3]. Although we have similarities, our key difference is that we focus more on the classification of an area likely to burn or not burn instead of growth rate prediction. These steps are conducted on a cloud-based distributed system that offers better scalability and model management for a Big Data Environment. Lastly, another research study was conducted in specific parts of Central Valley, California, that pertains to a significant number of wildfires, utilizing machine learning to predict wildfires in hopes of fire prevention in these regions [1]. While this study is geographically narrow and does not integrate cloud infrastructure, our project leverages pixels as key features from nine datasets and utilizes Spark to compute predictions while ensuring efficient parallel processes across nodes.

     Therefore, as several recent studies have explored wildfire prediction using artificial intelligence and machine learning through various methodologies, we have specifically leveraged Apache Spark on the cloud infrastructure—HDFS—with distributed data engineering processes, incorporating class balancing with weighting, and tuning the models with Cross Validator and Train Validation Split that makes solution inherently scalable aligning with big data principal.

3. Specifications

The table below shows types of dataset gathered including variables that commonly have significant impact on wildfire existen and growth rate.

*Table 1. Data Specification*

|  |  |
| --- | --- |
| Dataset | Size |
| Historical Wildfire: |  |
| Burn Area | 62.95\*2MB |
| Meterological Data: |  |
| Temperature | 728\*0.61MB |
| Humidity | 364\*2.01MB |
| Precipitation | 364\*2.01MB |
| Vegetation and Land Cover Data: |  |
| Landcover | 4,524 MB |
| Topography Data: |  |
| Elevation | 121.46 MB |
| Slope | 242.74 MB |
| Aspect | 242.74 MB |
| Human Activity Data: |  |
| Population Density | 1.97 MB |
| Total | 7.17GB |
| Total Size Afte Simplified: | 1.55GB |

*Table 2. Platform Specification*

|  |  |
| --- | --- |
| Cluster | Version |
| *PySpark* | Spark 3.0.2 |
| *Hadoop* | 3.1.2 |
| *Nodes (masters)* | 2 |
| *Nodes (workers)* | 3 |
| *Memory Size* | 31 GB |
| *Remote Server IP Address* | 144.24.13.0 |

4. Implementation Flowchart

The dataset collected from Google Earth Engine has an original size of 7.17GB and went through a data engineering process through Python to assemble nine features from nine datasets of GeoTIFF formats into one CSV dataset file that contains nine columns—each dataset is extracted and transferred into a single column alongside with another dataset.

A computer screen shot of a diagram

Description automatically generated

*Figure 1. Flowchart of Entire Project-From GeoTIFF Data Processing and Feature Assembly for Wildfire Prediction*

**5. Data Cleaning**

Raw files were downloaded from the Google Earth Engine in GeoTIFF format with nine different datasets, including wildfire perimeters, meteorological variables (temperature, humidity, and precipitation), vegetation and land cover, elevation and slope, and population density. Each dataset was accessed through its respective Earth Engine folder path. A custom JavaScript script was developed and accomplished importing data layers with spatial constraints specific to California, using a shapefile asset (`ca\_boundary`) to define the boundary, merged wildfire burn area records from 2020 and 2021 into a single feature collection, integrated various environmental layers into a unified multiband raster, and exported the processed raster to the user's Google Drive for local use.

     Furthermore, Python script (`main.py`) and raster I/O libraries such as `rasterio` operations were performed, achieving the raster was reprojected to a 200-meter by 200-meter spatial resolution using `calculate\_default\_transform()` and updated metadata parameters, each band within the raster was reshaped into two-dimensional arrays and indexed with corresponding row and column identifiers, and the restructured data were merged into a single DataFrame and saved as a CSV file.

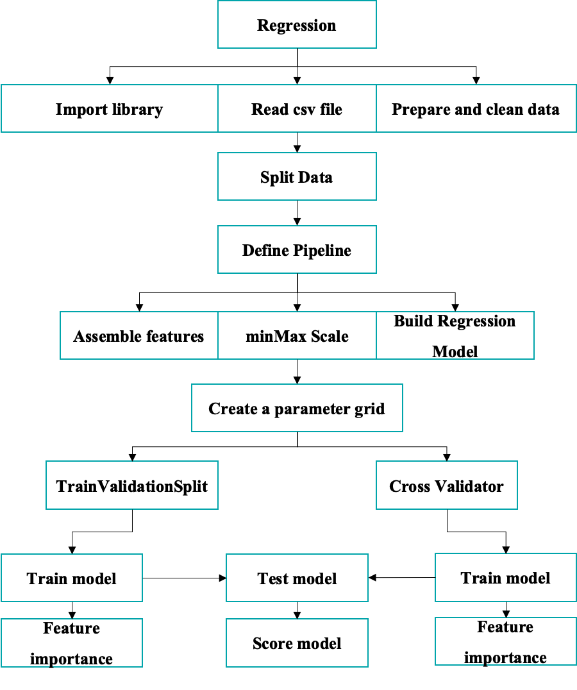
     Therefore, these steps enabled the multiband raster data to be converted into a machine-learning-ready tabular format. Execution time for the full cleaning and transformation process was approximately 60 to 120 minutes.

**6. Building Machine Learning Algorithms**

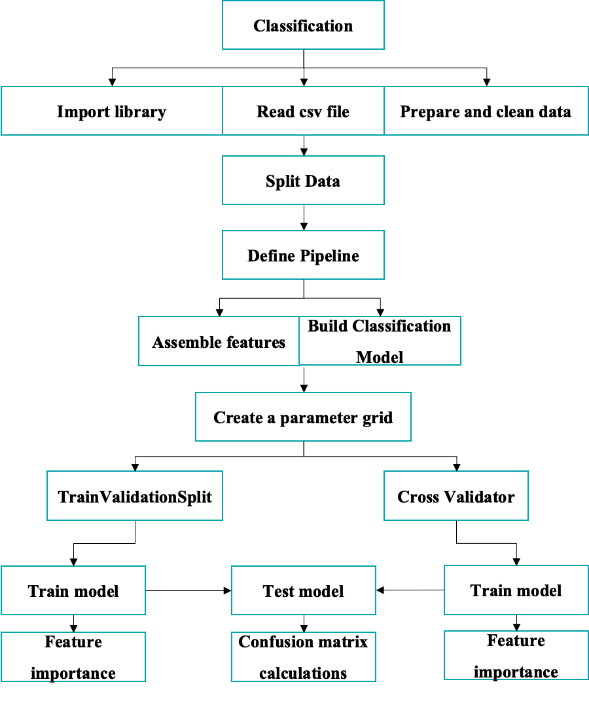
After the data engineering process in Python, a single CSV file featuring nine columns was downloaded and uploaded to HDFS in preparation for building algorithms. Because we simplified the dataset to 1.55GB, we discovered that some records may not be distributed equally; hence, results may potentially be skewed. To correct this, we implemented a class weighting strategy in Pyspark, assigning higher weights to minority class instances to prevent biased learning. This technique would allow equal distribution of data and is ready for feeding into machine learning algorithms.

     The final dataset included the following features: “lst,”  “humidity,” “precip,” “landcover,” “elevation,” “slope,” “aspect,” and “pop\_density,” along with binary target “burned.” Each model’s result was evaluated using both a manual confusion matrix and a classification evaluation for its performance accuracy.

     The two figures summarized processing steps for Regression and Classification integrated in our Pyspark Machine Learning.



*Figure 2. Regression Machine Learning Model*

**

*Figure 3. Classification Machine Learning Model*

**6.1 Linear Regression**

Through HDFS, we used Pyspark to run the linear regression we built and test the model's accuracy to predict the occurrence of wildfires. In this specific model, we have calculated the model through cross-through Train Validation Split and Cross Validator —with 70 percent training and 30 percent testing—resulting in a Root Square Mean Error and R2 that would indicate the model performance and how accurate the model fitting has developed, respectively. This Linear Regression model for both the Train Validator Split and Cross Validator has been shown to have an RMSE of 0.2361 and R2 of 0.0841. The variable "precip" has impacted this model the most, with the highest importance of 0.347238.

We extracts and compares the values from feature importance, two different model tuning approaches: Train Validation Split and Cross Validator. First, we retrieve the best-performing Linear Regression Model from each tuning method by accessing the final stage of their respective pipelines. We then extract the model coefficients, which represent the importance of each feature in predicting the target variable using the assembler. In the getInputCols() method, we collect the original feature names used in the model. The coefficients are paired with their corresponding feature names and converted into pandas DataFrames. To better understand the influence of each feature, we calculate the absolute value of each coefficient (representing the magnitude of importance regardless of direction) and sort the features in descending order of this value. Finally, it prints out the ranked feature importances as below, for both the Train Validation Split and Cross Validator models, enabling a side-by-side comparison of which features were most influential in each model’s predictions.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Feature | Coefficient | Importance |
| 2 | precip | 0.347238 | 0.347238 |
| 7 | pop\_density | 0.323382 | 0.323382 |
| 3 | landcover | 0.309030 | 0.309030 |
| 5 | slop | 0.198257 | 0.198257 |
| 4 | elevation | 0.171926 | 0.171926 |
| 0 | aspect | -0.011826 | 0.011826 |
| 1 | lst | 0.167877 | 0.167877 |
| 6 | humidit | 0.146250 | 0.146250 |

*Table 3. TrainValidationSplit Feature Importances (Linear Regression)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Feature | Coefficient | Importance |
| 2 | precip | 0.347238 | 0.347238 |
| 7 | pop\_density | 0.323382 | 0.323382 |
| 3 | landcover | 0.309030 | 0.309030 |
| 5 | slop | 0.198257 | 0.198257 |
| 4 | elevation | 0.171926 | 0.171926 |
| 0 | aspect | -0.011826 | 0.011826 |
| 1 | lst | 0.167877 | 0.167877 |
| 6 | humidit | 0.146250 | 0.146250 |

*Table 4. CrossValidator Feature Importances (Linear Regression)*

Both the Train Validation Split and Cross Validatior model performed equally well.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | TrainValidationSplit | CrossValidation | Time |
| RMSE | 0.2361 | 0.2361 | 5.980611s |
| R2 | 0.0841 | 0.0841 | 6.166839s |

*Table 5. Evaluation results of Linear Regression*

**6.2 Gradient-Boosted Tree Regression**

Similar to the previous model, Gradient-Boosted Tree Regression was also run through Pyspark to predict the occurrence of the wildfire.

     Looking at the Features Importance table, we can see the features that affected the model's performance the most and those that had the least importance.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Feature | Coefficient | Importance |
| 2 | precip | 0.205551 | 0.205551 |
| 7 | pop\_density | 0.022944 | 0.022944 |
| 3 | landcover | 0.193150 | 0.193150 |
| 5 | slop | 0.074444 | 0.074444 |
| 4 | elevation | 0.177831 | 0.177831 |
| 6 | aspect | 0.011135 | 0.011135 |
| 0 | lst | 0.195445 | 0.195445 |
| 1 | humidit | 0.119499 | 0.119499 |

*Table 6. TrainValidationSplit Feature Importances(GBT)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Feature | Coefficient | Importance |
| 2 | precip | 0.188226 | 0.188226 |
| 7 | pop\_density | 0.034522 | 0.034522 |
| 3 | landcover | 0.388587 | 0.388587 |
| 5 | slop | 0.080365 | 0.080365 |
| 4 | elevation | 0.263370 | 0.263370 |
| 6 | aspect | 0.000000 | 0.000000 |
| 0 | lst | 0.010086 | 0.010086 |
| 1 | humidit | 0.034844 | 0.034844 |

*Table 7. CrossValidator Feature Importances (GBT)*

The TrainValidationSplit performed better than Cross validation, with lower RMSEvalue and higher R2.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | TrainValidationSplit | CrossValidation | Time |
| RMSE | 0.21048852442959 | 0.2166941632 | 14s |
| R2 | 0.27155904371747 | 0.2279739839 | 13s |

*Table 8. Evaluation results of GBT Regression*

**6.3 Logistic Regression**

Logistic regression was used to classify the given column "burned," where the burned area is 1 and the not burned is 0. We tuned the model using both Train Validation Split and Cross Validator optimizing parameters to extract the best performing model. The feature's importance was stronger in the Cross Validator, suggesting that hyperparameter turning improves model confidence and the separation of classes. As for performance, the Cross Validator outperforms the Train Validation Split model, which is marginally similar­.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Feature | Coefficient | Importance |
| 7 | pop\_density | -4.5496 | 4.5496 |
| 3 | landcover | 3.9629 | 3.9629 |
| 2 | precip | 2.8658 | 2.8658 |

*Table 9. The Firt Three Rows of Train Validation Split Feature Importance (Logistic Regression)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Feature | Coefficient | Importance |
| 7 | pop\_density | -12.7606 | 12.7606 |
| 3 | landcover | 5.5140 | 5.5140 |
| 2 | precip | 5.1568 | 5.1568 |

*Table 10. The First three Rows of Cross Validator Feature Importance (Logistic Regression)*

Models’ perforamnce summary of Logistic Regresion resulted to have similar precision, recall, and accuracy. In comparison, Cross Validaor have better perforamnce than Train Validation Split.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Precision | Recall | F1 | AUC | Time |
| TrainValidation Split | 0.1623 | 0.7418 | 0.2664 | 0.8154 | 3621.46s |
| Cross Validator | 0.1748 | 0.7488 | 0.2835 | 0.8226 |

*Table 11. Logistic Regression Performance Summary*

**6.4 Random Forest Classification**

A Random Forest Classification algorithm was conducted to predict wildfire occurrence with higher interpretability and performance. Hyperparameters and tuning were used to find the optimal forest structure. The tables below indicate that the model's performance can detect fires with both higher accuracy and minimal false positives.

|  |  |  |
| --- | --- | --- |
|  | Feature | Importance |
| 3 | landcover | 0.245198 |
| 2 | precip | 0.209374 |
| 4 | elevation | 0.181650 |

*Table 12. The First Three Rows of Train Validation Split Feature importance (Random Forest Classification)*

|  |  |  |
| --- | --- | --- |
|  | Feature | Importance |
| 2 | precip | 0.197090 |
| 3 | landcover | 0.196882 |
| 4 | elevation | 0.172029 |

*Table 13. The First Three Rows of Cross Validator Feature importance (Random Forest Classification)*

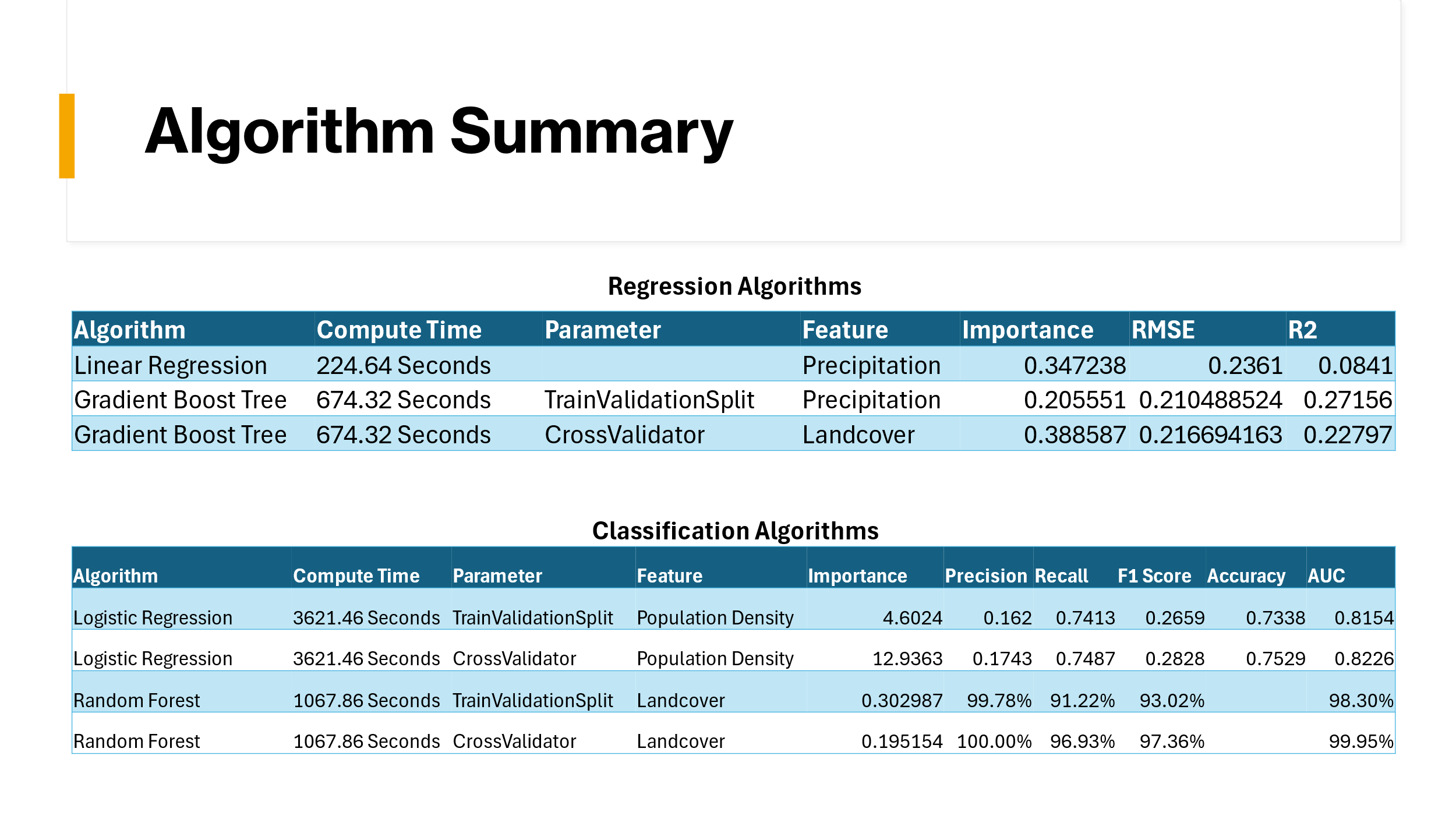
     Models’ performance summary of the Random Forest Classifier resulted in having relatively similar precision, recall, and accuracy. However, in comparison, Cross Validator has the best performance of all models. Unlike Logistic Regression, Random Forest offered a more flexible model capable of capturing nonlinear relationships and feature interaction. This indicates the most effective classifier in our pipeline.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Precision | Recall | F1 | AUC | Time |
| TrainValidation Split | 0.9985 | 0.9139 | 0.9318 | 0.9836 | 1067.86s |
| Cross Validator | 1 | 0.9737 | 0.9772 | 0.9994 |

*Table 14. Random Forest Classifier Performance Summary*

**7. Conclusion**

This project aimed to predict wildfire occurrence and severity in California using machine learning techniques. The project utilized nine datasets, including historical wildfire impact data, meteorological data, vegetation and land cover data, topography data, and human activity data. These datasets were processed and integrated to train machine learning models. The project employed both regression and classification models. Regression models, specifically Linear Regression and Gradient Boosted Trees (GBT) Regression, were used to predict wildfire severity. Classification models, including Logistic Regression and Random Forest Classification, were used to predict wildfire occurrence. The Random Forest Classification model was identified as the most effective model, demonstrating high AUC, precision, and recall. In contrast, the Logistic Regression model had low precision, and the regression models showed lower predictive performance, with Linear Regression being the least effective. Feature importance analysis revealed that different factors influenced wildfire severity and occurrence. Precipitation and land cover were important features in the regression models, while population density and land cover were key features in the classification models. The project emphasized the importance of recall in wildfire prediction. High recall, the ability of the model to correctly identify actual wildfire events, is crucial to minimize the negative impacts of wildfires, such as property damage, loss of life, and environmental harm. The project highlighted the use of cloud computing and big data technologies, specifically Apache Spark, to handle the large datasets and complex computations involved in wildfire prediction. The data was obtained from Google Earth Engine and processed using Python and raster I/O libraries.



*Table 15. Summary of all algorithms*

### References

[1] Hernandez, K., & Hoskins, A. B. (2024). *Machine learning algorithms applied to wildfire data in California's central valley*. Environmental Challenges, 9, 100273.

[2] Medzerian, D. (2024, July 22). *USC Scientists Use AI to Predict a Wildfire’s Next Move*. USC Today. https://today.usc.edu/using-ai-to-predict-wildfires/

[3] Shmuel, A., & Heifetz, E. (2023). A Machine-Learning approach to predicting daily wildfire expansion rate. *Fire*, *6*(8), 319. <https://doi.org/10.3390/fire6080319>

[4] GitHub Link: <https://github.com/xuewentang/cis5560>