Vyanna Hill, Coffy Andrews-Guo

Data 698 003: Analytics Master’s Research Project Date: 11/27/2023

**DATA COLLECTION AND ANALYSIS**

**INTRODUCTION**

For the review of these methods, the team gathered multiple resources for the features mentioned in the literature review. The wildfire likelihood data was retrieved from FEMA, which contained the national risk index of wildfire likelihood for 2022. FEMA’s National Risk Index dataset contains other weather phenomena such as strong winds, heat waves, and lighting with its historical frequencies and annual events. These additional data points are beneficial as strong winds were a risk component measured in Yue’s Journal. The data set had the events at the tract level, which allowed for California-only observations for the split between training and testing. If available, the team aimed for other features at the tract level, which retrieved the tract slope and vegetation data from LandFire.

LandFire contains slope and vegetation data via raster files. It appears lower-level data is shared with the public through geoTiff files as The team used the QGIS platform to retrieve the raster data seen on the Californian tract shapefile and exported it into a CSV file for analysis. The elevation dataset was fetched from the US Geographical Survey; which provided the elevation by county. For the annual precipitation and temperature, the data was found on NOAA’s Climate at a Glance. After the collection, the data sets were combined via Tract and county census ID. One pain point of the data collection process was data availability, as the many data resources were at the county level.

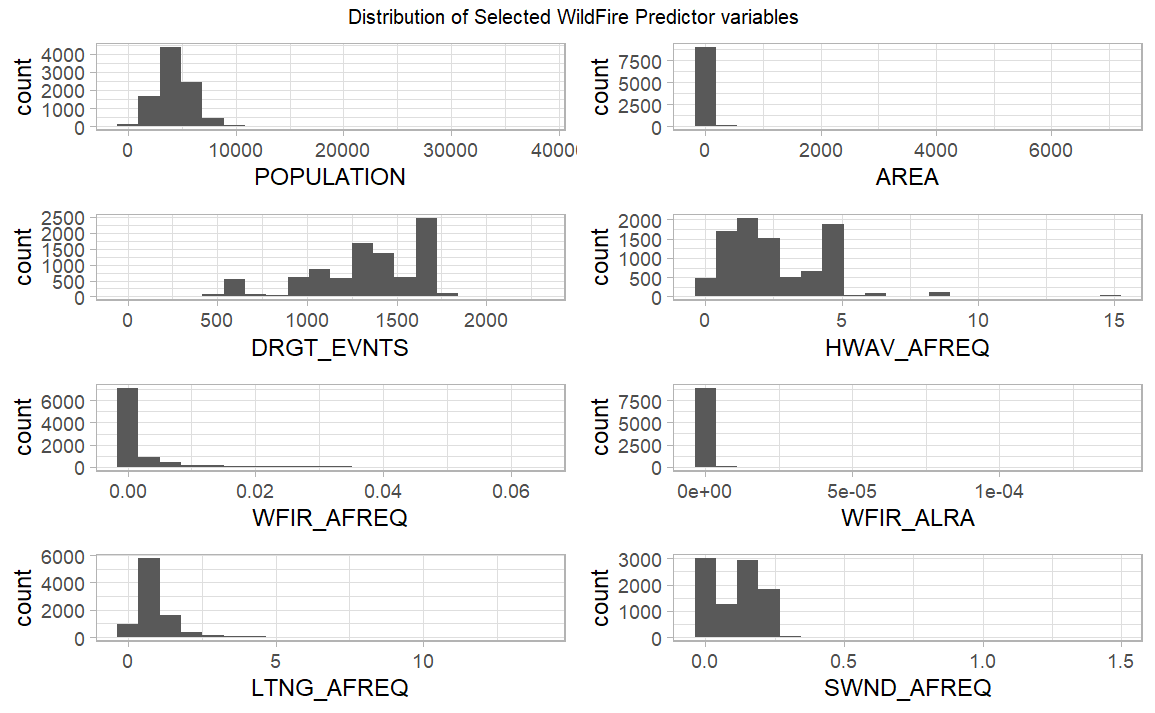
**DATA EXPLORATION**

The data set observed 9,089 Californian tracts with 29 variables in the dataset. The response variable is “WFRI\_R”, which is the binary response of the tract’s likelihood of a wildfire. The predictor variables are seen in the list below.

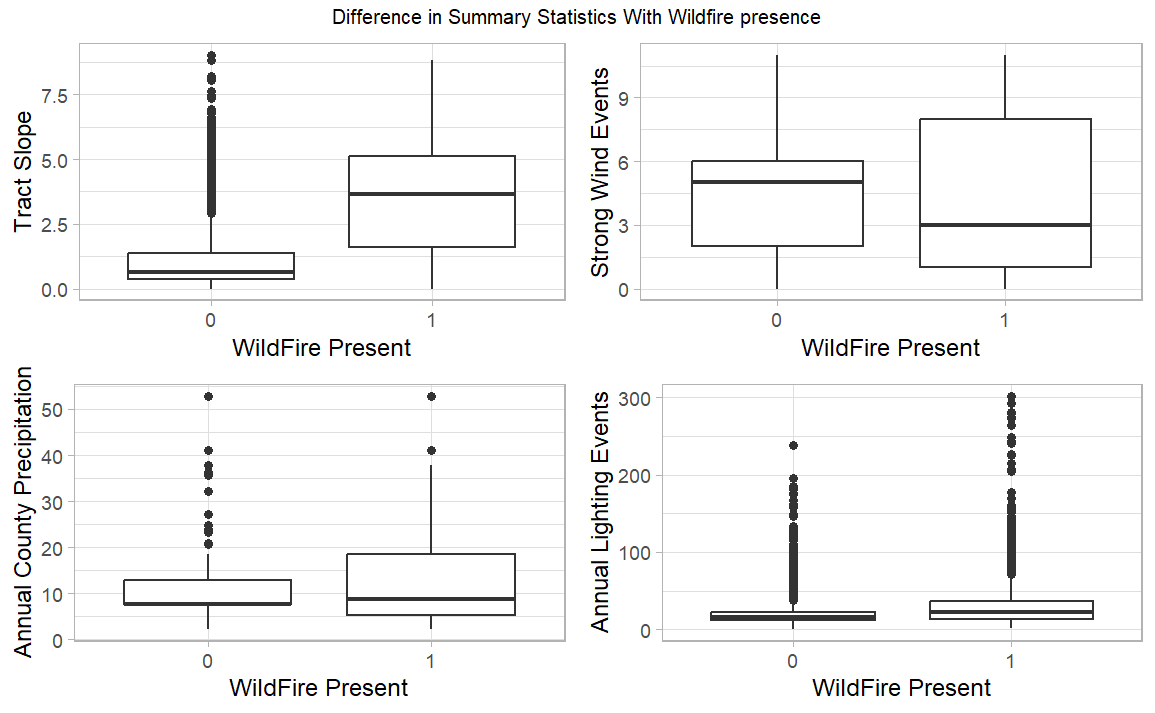
Predictor Variables

* POPULATION: Population (2020)
* AREA: Area (sq mi)
* DRGT\_AFREQ: Drought - Annualized Frequency
* DRGT\_HLRA: Drought - Exposure - Agriculture Value
* HWAV\_EVNTS: Heat Wave - Number of Events
* HWAV\_AFREQ: Heat Wave - Annualized Frequency
* HWAV\_HLRA: Heat Wave - Historic Loss Ratio - Agriculture
* LTNG\_EVNTS: Lightning - Number of Events
* LTNG\_AFREQ: Lightning - Annualized Frequency
* SWND\_EVNTS: Strong Wind - Number of Events
* SWND\_AFREQ: Strong Wind - Annualized Frequency
* SWND\_HLRA: Strong Wind - Historic Loss Ratio - Agriculture
* WFIR\_AFREQ: Wildfire - Annualized Frequency
* WFIR\_HLRP: Wildfire - Historic Loss Ratio - Population
* WFIR\_HLRA: Wildfire - Historic Loss Ratio - Agriculture
* WFIR\_ALRA: Wildfire - Expected Annual Loss Rate - Agriculture
* TRCT\_WAREA: Tract Water Area
* TRCT\_SLOPE: Tract Slope
* CNTY\_ELEV: County Elevation
* CNTY\_TEMP: County Annual Temperature (2022)
* CNTY\_PRECIP: County Annual Precipitation (2022)
* TRCT\_VEGLF: Tract Average Vegetation Lifeform

***Variable Statistics***

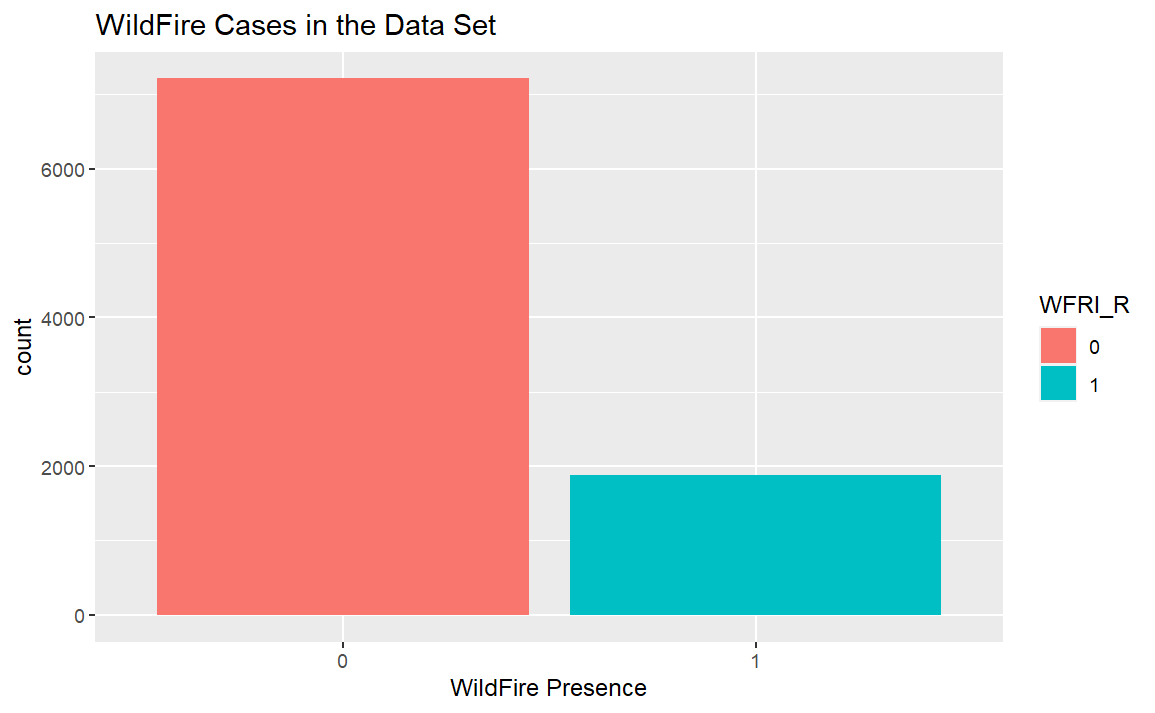
**

The team reviewed a selected group of predictor variables’ density distributions. The distribution. For the non-normal features, frequency variables tend to have right-skewed distributions as the majority of tracts have low frequencies. The predictor variable with a left-skewed distribution was drought events; however, this is expected as droughts are more common in California. The team also reviewed the difference in the distribution by the response variable below.



The box plots provided above display the differences between the likelihood of a wildfire. The team noticed the variables tract slope and strong wind events have the most noticeable difference in the median. It appears that tracts with a higher slope have a higher likelihood of a wildfire than lower slope tracts. It can also be inferred from the graph that tracts with the likelihood of a wildfire see more strong wind events, as the third quantile of its plot hits around seven annual events. The team theorized from the boxplots that the features mentioned above may have a strong influence on the models created. Outside the future feature selection, there are a few cases of outliers seen in the graph above. The exploration of these models will need to handle outliers as removal may not be the best choice, as these outliers are Californian tracts. The team should look into classification methods that can handle outliers.

***Imbalanced data***



The team checked the distribution of the wildfire response cases. The team noticed that the data set is heavily imbalanced as the majority of the cases do not have the likelihood of a wildfire. This imbalance can affect the accuracy of the models, as the models are predominately trained on non-wildfire cases and may falsely predict positive wildfire cases.

**DATA PREPARATION**

***Missing Values***

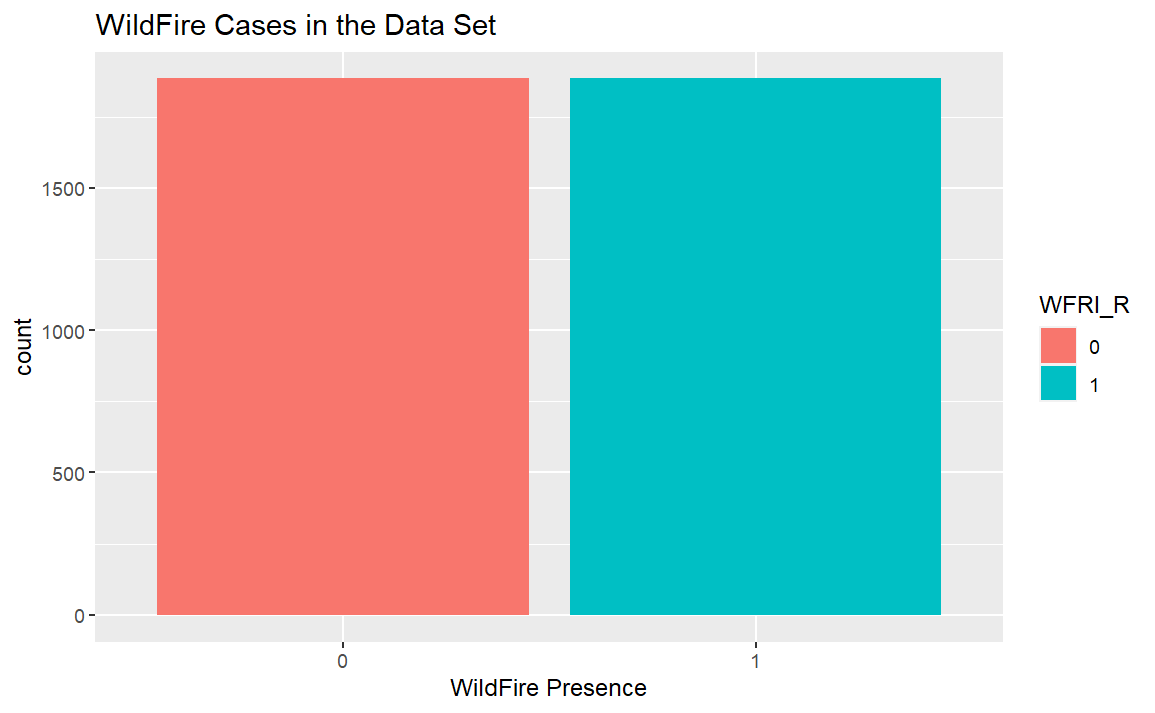
In the combination of the collected data set, there were fourteen tracts with missing values. As these values are not missing with a pattern, the team can note that the missingness is at random. In the preservation of the Californian tracts, the team used MICE imputation for the missing values. MICE uses regression in its prediction of the missing values and fills in the null value. Unlike the median used for imputation on the missing, this imputation offers another way for less biased imputed values in the data set.

***Categorical Variables***

There were a few steps in preparing the data set used for the models. First, wildfire response and vegetation life form variables were categorical variables. Their variables were transformed into dummy binary variables. The wildfire response transformed in the data processing process with the variable *“WFIR\_RISKV”* transformed to its binary variable *“WFRI\_R.”* The vegetation lifeform variable had nine unique values in its feature, so the K-1 approach was used in its dummy variable creation. There were a total of eight new features noted in the specific vegetation life form (i.e. *“.isShrub”*).

***Downsampling***

From the data exploration, the team saw a heavily imbalanced data set. The data set was downsampled for an equal balance between wildfire and non-wildfire cases. Downsampling helped the machine-learning models see more wildfire cases. This change also helps the possible overfitting in the previous process as older models had accuracies of 99% in the training set.



***Multi-collinearity***

The team wanted to leave feature selection to the models as different algorithms may prioritize certain features based on their subsets. The priority was checking if the predictor variables were not correlated with each other, as it can affect the weighting of the coefficients. After the predictor variables ran through the correlation matrix, drought events and drought frequency were flagged as highly correlated. From the correlation matrix, only drought frequency was included in the base models.

**DATA ANALYSIS**

**Wildfire Risk Prediction Models**

We used the H2O.ai open-source learning platform with a user-friendly interface and powerful capabilities. It has a wide range of tools and algorithms that make creating and implementing machine-learning models easy. H2O supports various machine learning techniques, including supervised and unsupervised learning, and can handle large datasets efficiently. Its AutoML functionality is a standout feature, making model selection and hyperparameter tuning accessible to users with limited machine-learning experience. The scalable and distributed architecture of H2O allows users to utilize parallel processing for quicker model training and deployment.

In the previous section, we discussed how we built our risk prediction models using the H2O platform with variables that reflect California's topography, including elevation, temperature, rainfall, vegetation types, and other relevant features to detect wildfire occurrences. We used six models on the platform - Distributed Random Forest (DRF), Automatic Machine Learning (AutoML), Deep Learning, Gradient Boosting Machine (GBM), Naïve-Bayes, and Support Vector Machine (SVM).

*Distributed Random Forest (DRF):*

DRF creates multiple decision trees using a random subset of the dataset and features at each node. In the first version, the team created the maximum amount of trees to a thousand to see where the model produced without the default max of 50 trees. In addition, there was a stopping metric included in the model that checks the AUC score every five iterations. The stopping metric prevents the model from spitting further if the model does not see an improvement in the AUC score. Then, cross-validation was enacted in the training set of the model. The model broke its training set into five folds, which tested the performance of the training set separately. The results of the first model produce a tree of 50 trees with a max depth of 20 branches.

In the final version of the model, the team optimized based on the first model’s performance. From the first model, the top ten features based on their gini coefficient were subseted into a new feature list. The max trees were shortened to 25 trees as fewer trees limit the possibility of overfitting. These improvements to the second model saw a training accuracy of 96% with an average R2 of 85% across the cross-validated sets.

*Automatic Machine Learning (AutoML):*

AutoML simplifies machine learning model building by automating data preprocessing, feature selection, model choice, and hyperparameter tuning. The H2O AutoML framework allows users to integrate specific hyperparameters like *"max\_models = 5"* and *"seed = 1,"* as shown in the example *"aml <- h2o.automl(x = features\_v2, y = response, training\_frame = train.h2o, max\_models = 5, seed = 1),"* to gain more control over the model development process. By setting seed for reproducibility and limiting the number of models generated during the AutoML process, users can fine-tune and customize their machine-learning workflows for specific requirements. AutoML streamlines the machine learning pipeline, allowing for the faster development of effective models without extensive manual intervention.

*Deep Learning:*

The presented code, *"dl <- h2o.deeplearning(x = features\_v2, y = response, distribution = "AUTO", hidden = c(1), epochs = 1000, train\_samples\_per\_iteration = -1, reproducible = TRUE, activation = "Tanh", single\_node\_mode = FALSE, balance\_classes = FALSE, force\_load\_balance = FALSE, seed = 23123, score\_training\_samples = 0, score\_validation\_samples = 0, training\_frame = train.h2o, stopping\_rounds = 0, keep\_cross\_validation\_predictions = TRUE),"* exemplifies implementing a deep learning model using the H2O platform. Deep learning, as a subset of machine learning, employs artificial neural networks with multiple layers to autonomously discern patterns in data. This code specifies various parameters to configure the deep learning model, including the choice of activation function ("Tanh"), the number of hidden layers, and the total number of training epochs. The reproducibility parameter ensures consistency in results across runs. Unlike traditional methods, deep learning avoids the need for manual feature selection by leveraging backpropagation to autonomously adjust internal settings.

*Gradient Boosting Machine (GBM):*

GBM trains new models by iteratively correcting errors in the existing ensemble and uses gradient descent optimization to make classification and regression tasks practical. Version one of the model had the following features: a learning rate of 0.10, a stopping metric at AUC, and a cross-validation fold at five. The learning rate of the model was set to 0.10, as the team noticed with slower learning rates the average R2 was 41%. The team theorized with the slower rate, the model was overfitting the model with more tree subsets.

For the final version of the model, the team included the top ten features of the previous version and increased the stopping interval. The increased stopping interval checks the AUC score more frequently for review, as optimizations are limited on the model. The average R2 from the cross-validated set was 84% and its accuracy at 95%.

*Naïve Bayes:*

A Naive Bayes classification model is constructed using the H2O machine learning framework in the given code snippet. The code, *"pros\_nb <- h2o.naiveBayes(x = features\_v2, y = response, training\_frame = train.h2o, laplace = 0, nfolds = 5, seed = 1234, keep\_cross\_validation\_predictions = TRUE),"* employs explicitly the Naive Bayes algorithm to predict the response variable based on the features provided. The *"laplace = 0"* hyperparameter indicates the absence of Laplace smoothing, while *"nfolds = 5"* specifies the number of cross-validation folds for model evaluation. The seed parameter ensures reproducibility, and the option to retain cross-validation predictions allows for further model performance analysis. Naive Bayes, as described, operates on the assumption of independence between features given the class label, simplifying computations.

*Support Vector Machine (SVM):*

The provided code snippet, *"svm\_model <- h2o.psvm(gamma = 0.01, rank\_ratio = 0.1, y = response, training\_frame = train.h2o, disable\_training\_metrics = FALSE, seed = 1),"* encapsulates the implementation of a Support Vector Machine (SVM) model using the H2O machine learning framework. The parameters employed in this code, such as *"gamma"* and *"rank\_ratio,"* are crucial in shaping the SVM's behavior. The *"gamma"* parameter influences the kernel function, determining the shape of the decision boundary, and *"rank\_ratio"* controls the number of support vectors used in the model, impacting its generalization ability. The code ensures a comprehensive SVM model by specifying the response variable training data and incorporating options like *"disable\_training\_metrics = FALSE"* for tracking training metrics. SVMs excel in handling classification tasks by identifying optimal hyperplanes that effectively separate different class data points while maintaining a margin.

According to Table 1, the Distributed Random Forest and AutoML algorithms provided the most accurate results among all other selected algorithms.

|  |  |  |
| --- | --- | --- |
| *Table 1. Model Accuracy with different ML algorithms* | | |
| **Model Used** | **Accuracy** | **Unique Hyperparameters** |
| Distributed Random Forest (DRF) | 0.9478 | balance\_classes = FALSE |
| AutoML (Automatic Machine Learning) | 0.9478 | max\_models = 5 |
| Deep Learning | 0.9381 | Hidden = c(1), activation = “Tanh”, epochs = 1000 |
| Gradient Boosting Machine (GBM) | 0.9363 | learn\_rate = 0.1, ntrees = 1000 |
| Naïve-Bayes (NB) | 0.8788 |  |
| Support Vector Machine (SVM) | 0.5301 | Gamma = 0.01, rank\_ratio = 0.1 |

**RESULTS AND DISCUSSION**

In our analysis, a range of factors influenced the decision to exclude certain features from the model. One important consideration was the relevance or redundancy of features—those that did not provide meaningful information or duplicated existing ones were left out to prevent overfitting and improve overall model performance. Additionally, we removed features with excessive noise or outliers to ensure the model's ability to generalize effectively to new data, avoiding unnecessary complexity. We also addressed collinearity issues stemming from highly correlated features, opting for exclusion to estimate individual effects accurately. Maintaining data quality was a priority, leading to eliminating features with poor quality, missing values, or inconsistencies to uphold the overall reliability of the model.

The performance metrics in Table 2, reveal that Distributed Random Forest and AutoML are the top performers with consistently high scores across all metrics, ranging from 0.9478 to 0.9486. This indicates their strength in delivering accurate and precise predictions while effectively capturing true positives and minimizing false negatives. Deep Learning follows closely behind with a precision score of 0.9476, while Gradient Boosting Machine exhibits creditable results with a precision score of 0.9410. Naive Bayes performs reasonably with a precision score of 0.8794 but is surpassed by the more advanced algorithms. However, the Support Vector Machine excels in Recall, and reaches a perfect F1 score of 1.0000, although it achieves the lowest Accuracy of 0.5301 and Precision of 0.5155.

*Table 2. Performance Metrics using modified values dataset.*

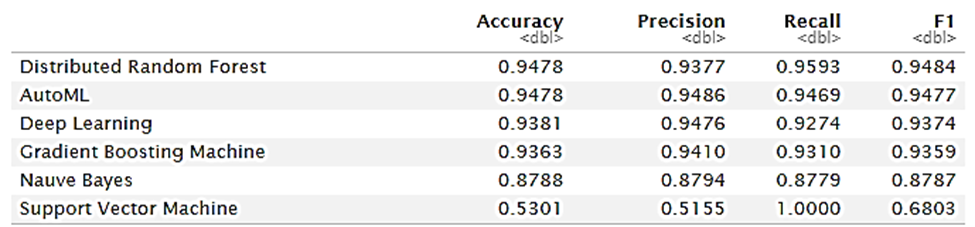
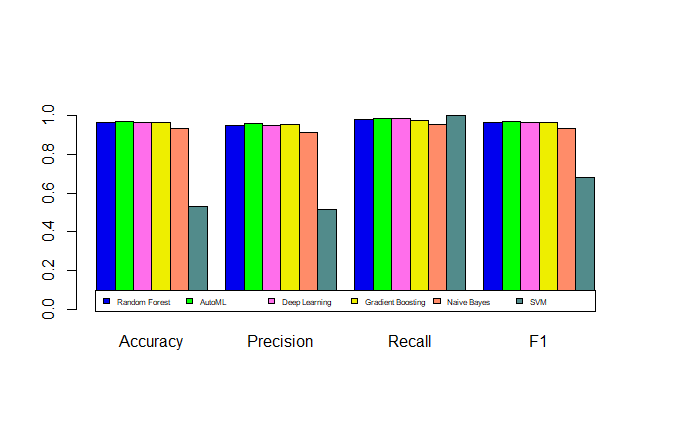
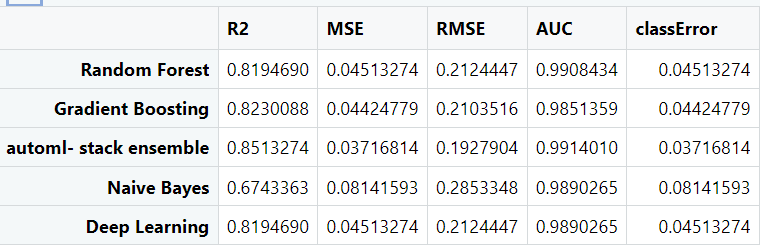
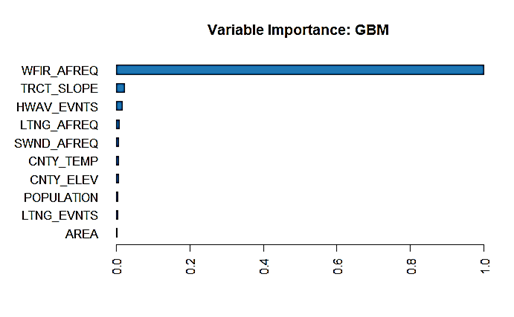
****

Figure 1 presents a graphical representation of the performance of the selected machine learning models on the modified values dataset.

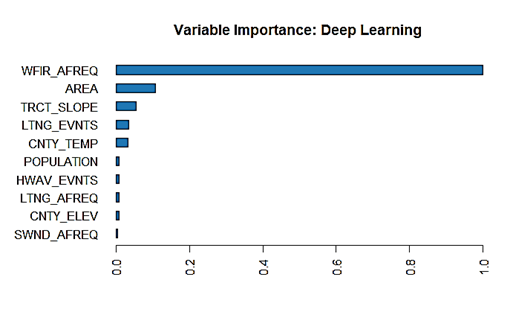
*Figure 1 - Visualization of Model Performance Metrics*

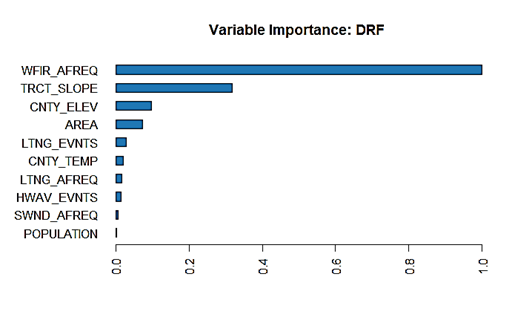
This analysis evaluated various machine learning models based on multiple performance metrics. Random Forest (RF) showed strong performance with an R2 of 0.819, low values for MSE and RMSE, and a high AUC, indicating accurate predictions and robust classification ability. Gradient Boosting (GB) showed similar excellence with a slightly higher R2 of 0.823 and comparable metrics. The AutoML Stack Ensemble outperformed both RF and GB, achieving an impressive R2 of 0.851, low MSE and RMSE, and superior predictive accuracy. Naive Bayes lagged behind RF, GB, and the AutoML ensemble. Deep learning performed on par with random forest. Overall, the AutoML Stack Ensemble emerged as the top performer in predictive accuracy and robustness across the metrics evaluated.

**

To gain insights into decision-making processes, we evaluated the interpretability of models. Different approaches use unique methodologies to assess feature importance across three models - Gradient Boosting Machine (GBM), Deep Learning, and Distributed Random Forest (DRF).

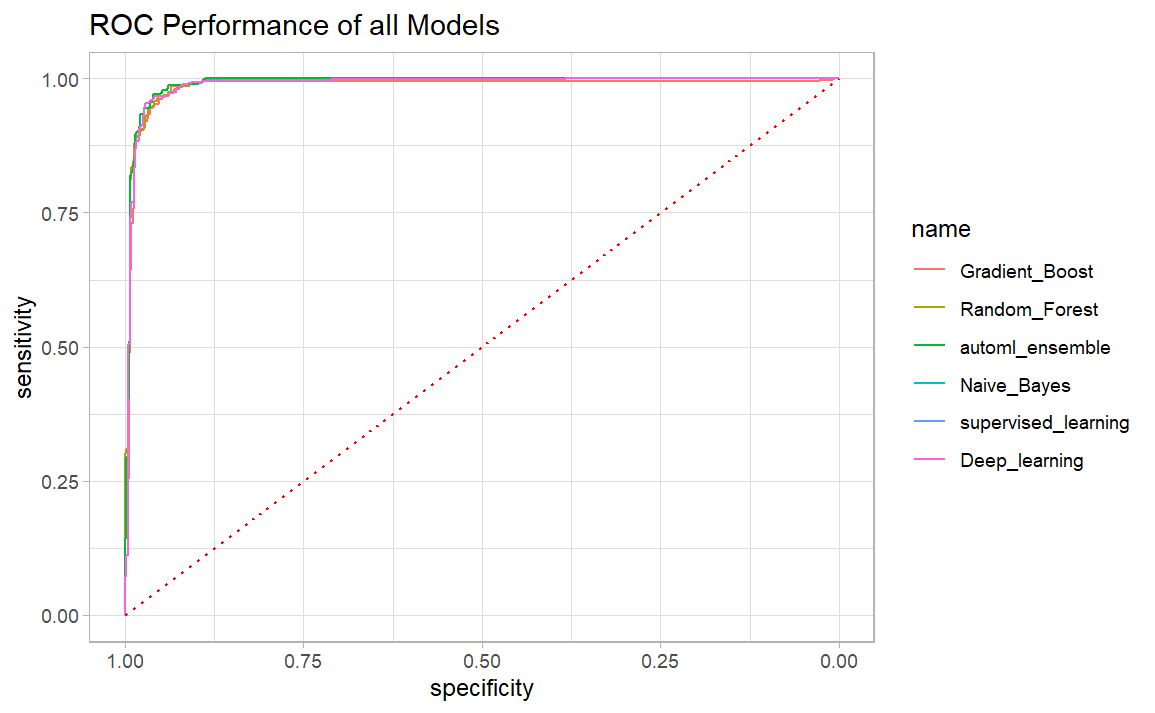
GBM creates a sequence of decision trees, iteratively correcting errors, and assigns feature importance based on the frequency of features used in tree splits that improve overall model performance. A higher importance score in GBM suggests a more significant impact on the model's predictions.

In contrast, with its layered neural network architecture, Deep Learning applies layer-wise relevance propagation or sensitivity analysis techniques to evaluate feature importance. The greater sensitivity or relevance indicates a more significant effect of the feature on the model's output.

On the other hand, DRF constructs an ensemble of decision trees independently and calculates feature importance by measuring the reduction in impurity each feature brings to decision trees. Higher feature importance in DRF implies a more significant influence on decisions across the ensemble.

The GBM, Deep Learning, and DRF models agree that "Wildfire Annualized Frequency" is the most important predictor for predicting wildfires. This shared emphasis indicates that it significantly affects the likelihood of wildfires. Additionally, while GBM and DRF prioritize "Tract Slope" in their predictions, there is a subtle difference between them. On the other hand, Deep Learning differs by emphasizing "Area (sq)" as another critical feature. These differences highlight each model's unique perspective, with GBM and DRF focusing on the timing of wildfires and Deep Learning highlighting the affected area. Understanding these model-specific preferences helps us grasp the factors influencing wildfire predictions and choose models that suit the dataset and analysis goals.

Even when downsampling is employed to address class imbalance in our dataset, the ROC (Receiver Operating Characteristic) curve and AUC (Area Under the Curve) metrics are still important for evaluating a classification model's performance. These metrics consider the trade-offs between true positive and false positive rates at different classification thresholds. While downsampling helps balance the dataset, the ROC and AUC metrics provide valuable insights into the model's generalizability, regardless of class distribution. The selected models in the ROC plot demonstrate a higher AUC, closer to 1, indicating their ability to balance sensitivity and specificity across various thresholds. This highlights the models' proficiency in distinguishing between positive and negative classes, reinforcing their reliability in wildfire prediction scenarios.

****