AMOD 5440H Final Project

Sophie Ali

Chris de la Rua

Taha Khawaja

Sineth Pathirana

Introduction to the Dataset

Millions of hectares of forests are destroyed each year by forest fires. Forest fires are often unplanned, and can get to a point where they are uncontrollable. Some parts of the world such as California and Australia experience very severe wildfires that can leave behind plenty of destruction. A number of factors contribute to the creation of the conditions necessary for forest fires to occur such as climate change, health of an ecosystem, storms, extreme weather, and even human error. Climate change is resulting in an increase in forest fires, and their severity. Forest fires can result in the displacement of individuals, dangerous levels of pollution, destruction of homes and cities, and tremendous costs to fix and repair the damage. The most efficient way of preventing and stopping forest fires is detecting them quickly. A lot of research has been done to understand the nature of forest fires, the types of conditions that cause forest fires, and the damage that can be expected in terms of the area burnt. Understanding the nature of forest fires may help to prevent more forestation from being destroyed,

especially with increasing concerns about the impact of climate change in the near future.

The dataset chosen for this analysis is the Forest Fires Data Set, which looks at predicting the amount of area burned in forest fires in the northeast region of Portugal. The data was collected between January 2000 and December 2003. The data set consists of both categorical continuous attributes. The data set contains 13 attributes in total, for which there are a total of 517 samples. The variable of importance in this problem is the target variable ‘area’, which is measured in hectares (ha), and ranges from 0.00 to 1090.84 hectares. The attributes ‘X’ and ‘Y’ are the x and y axis spatial coordinates within the Montesinho Park. The coordinates represent the location for where the fires occur, and are mapped between 1-9 for ‘x’, and 2-9 for ‘Y’. The next attribute in the data set is the ‘month’, which is valued for all 12 months of the year. The attribute ‘day’ is the day of the week and contains all 7 days of the week. The attribute ‘FFMC’ is the Fine Fuel Moisture Code from the Fire Weather Index that measures the fuel moisture of forest litter fuels under the shade of a forest canopy (National Wildfire Coordinating Group, 2021). It takes the range 18.7-96.20. The attribute ‘DMC’ is the Duff Moisture Code and measures the fuel moisture of decomposed organic material underneath the litter (National Wildfire Coordinating Group, 2021). The ‘DC’ attribute is the Drought Code from the FWI system and ranges from 7.9 to 860.7. Extreme drought conditions often occur when DC values are close to 800 (National Wildfire Coordinating Group, 2021). The ‘ISI’ index is from the FWI system, and stands for the Initial Spread Index and is a good indication for fire behavior predictions. This attribute ranges from 0.00 to 56.10 (National Wildfire Coordinating Group, 2021). The ‘temp’ attribute is the

temperature in Celsius degrees and ranges from 2.2 to 33.30. The ‘RH’ attribute is the relative humidity in percentages and ranges from 15.0 to 100. The ‘wind’ attribute is measured in kilometers per hour, and ranges from 0.40 to 9.40. Lastly, the ‘rain’ attribute is the amount of rainfall measured in mm/m^2, and ranges from 0.0 to 6.4.

To conduct the analysis, we will start with some preprocessing and visualization techniques that were done in order to understand the data and look into some problems that may exist such as outliers, class imbalances, missing values, and necessary transformations. Next we will explore some methodology to better understand the techniques that will be used in the analysis. The next section will be the results section, which will explain some of the findings, and how well the models performed in predicting forest fires. Some of the models used in the analysis include regression models such as linear regression and random forest regressors, and classification models such as k-NN. Lastly, the conclusion section will wrap things up by reviewing the main findings and takeaways from the analysis as well as some limitations and future directions.

Exploratory Data Analysis

Our goal with exploring the dataset is to develop an understanding of the data using summary statistics and visualizations. This dataset has 517 instances with 13 features describing each instance. In order to get a good understanding of the relationships between the variables, the first 2 questions we’ll aim to answer are:

1. What type of variance occurs within the variables?
2. What type of covariation occurs between variables?

Using the describe() function in pandas we can get some very interesting summary statistics about the data.

```
# summary statistics
df.describe()
```

	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000
mean	4.669246	4.299807	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201	4.017602	0.021663	12.847292
std	2.313778	1.229900	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469	1.791653	0.295959	63.655818
min	1.000000	2.000000	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000	0.400000	0.000000	0.000000
25%	3.000000	4.000000	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000	2.700000	0.000000	0.000000
50%	4.000000	4.000000	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000	4.000000	0.000000	0.520000
75%	7.000000	5.000000	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000	4.900000	0.000000	6.570000
max	9.000000	9.000000	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000	9.400000	6.400000	1090.840000

Table 1. Descriptive summary of dataset

There is a notably large value between the 75th percentile value and the max value for the DMC, ISI, RH and Area columns. This indicates some extreme value and possibly outliers in the dataset.

There are no missing values which is verified using the isnull() function.

```
# checking for missing values (no null values)
df.isnull().sum()
```

```
X        0
Y        0
month    0
day      0
FFMC     0
DMC      0
DC       0
ISI      0
temp     0
RH       0
wind     0
rain     0
area     0
dtype: int64
```

Table 2.. Verification of no null or missing values in dataset

Analysis of Target Variable

We can use some univariate visualizations of the raw data of the target variable along with some summary statistics to better understand the underlying distribution of ‘area’. Below we can see the distribution of the target variable using a histogram and then a box plot.

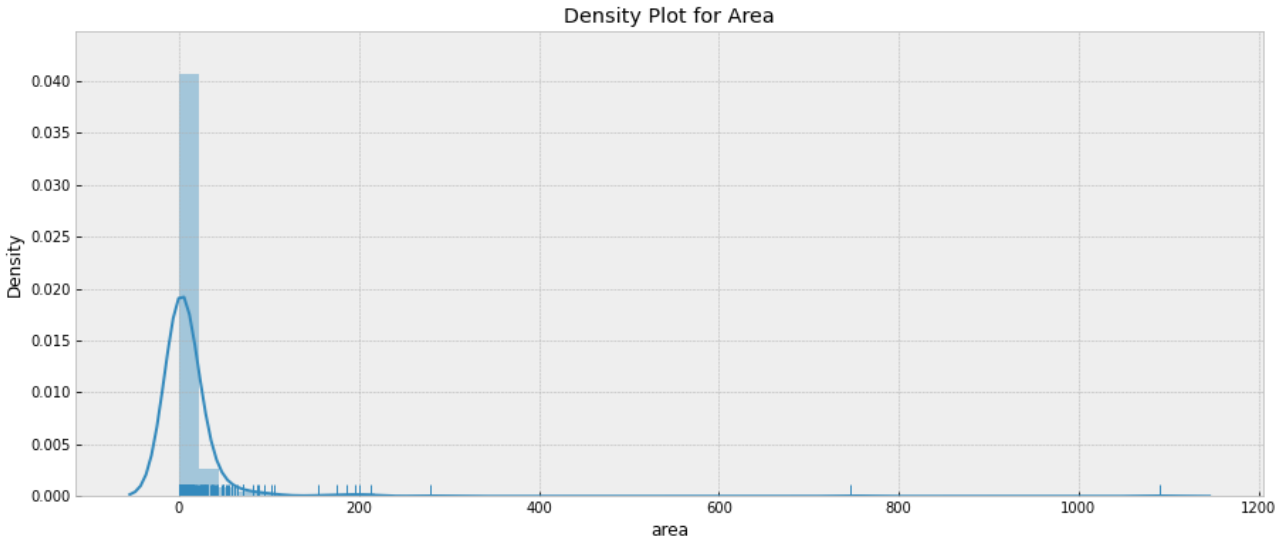

Figure 1. Density Plot of Target Attribute ‘Area’

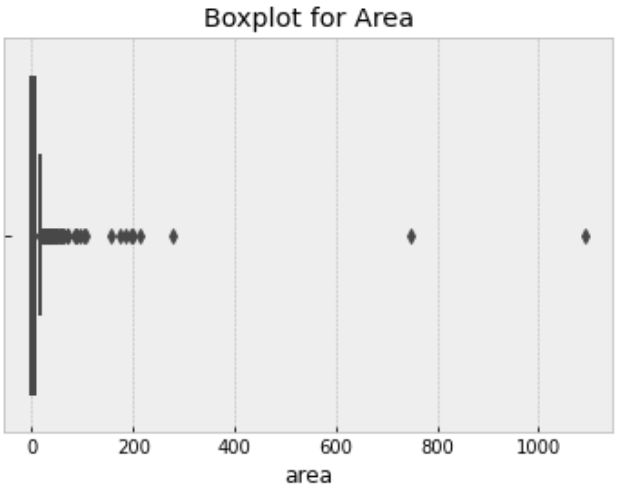

Figure 2. Boxplot of Target Attribute ‘Area’

Both the histogram/density plot and the boxplot will show us the underlying frequency (shape) of the data. The two plots above demonstrate that there are some extreme values in the area variable - we can see how far away the extreme values are from the rest of the distribution of the data.

The density plot shows that the majority of the values are 0, and that the values are highly right-skewed. The data is skewed with a value of +12.8469 and huge kurtosis value of 194.1407. Therefore, we can conclude that most forest fire damages are less than 100 hectares of land. There seems to be a small cluster of points at around 200 hectares and a few extreme outliers at 750 and 1150 hectares. Because the data is so highly skewed, a logarithmic transformation to fix the skewness and kurtosis can be helpful later in our data mining operations.

Analysis of Categorical Data – Covariations Amongst Variables

Next, we can use bivariate and multivariate visualizations to answer the covariation question – what type of covariation occurs between variables? We start off by looking at the distribution of the categorical attributes; day and month.

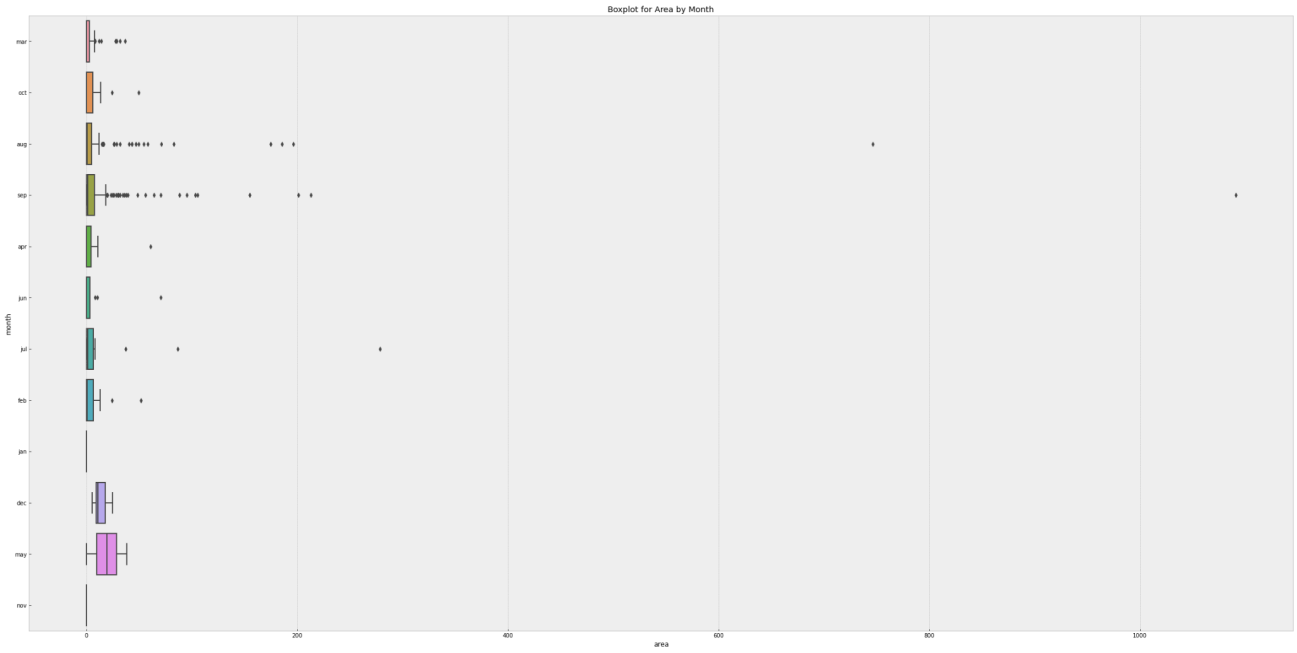

Figure 3. Boxplots of Month vs. Area

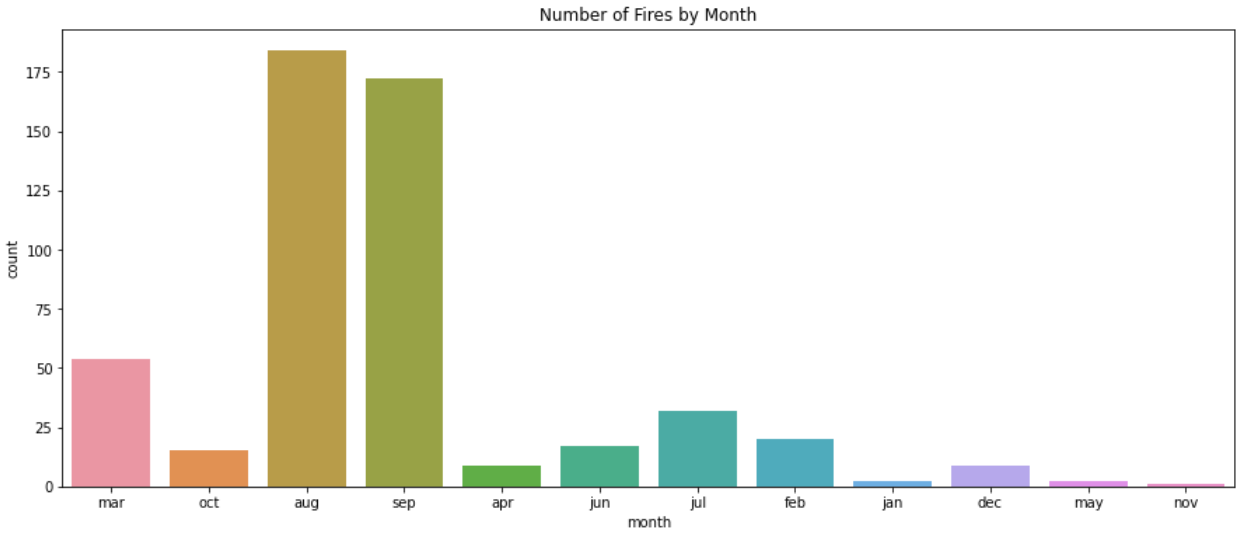

Figure 4. Count of Fires by Month

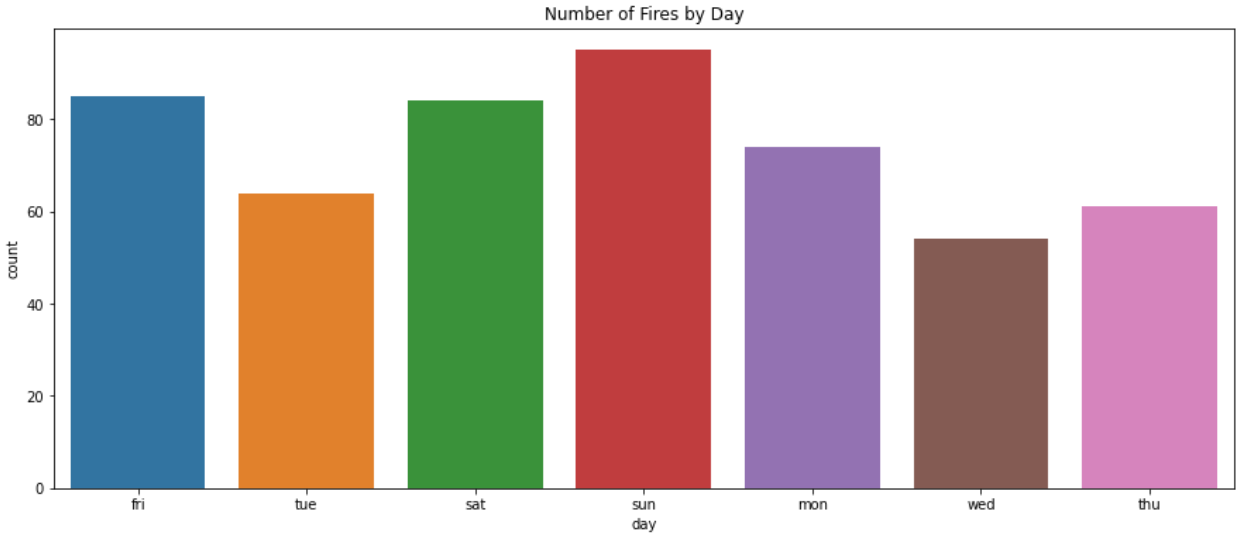

Figure 5. Count of Fires by Day

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables. Here we can see which months had extreme values - July, August and September had fires with the most extreme values. In fact, any fires that destroyed more than 200 hectares of land all happened between the months July to September. There were also an exponentially higher number of fires that happened in August and September so it makes sense that any extreme values for the area would happen in those 2 months. Interestingly, we can also observe that the distribution for the month May is much more spread out than the rest of the months, excluding outliers. This makes the median much higher for May that it is for the rest of the months.

Regarding fire damage per day, there were slightly more fires that happened between Friday to Monday but nothing interesting.

We can further analyze the area damaged by fires by month and day by looking at some summary statistics.

```
# grouping average area of forest fires by month
df.groupby('month').mean()['area']
```

```
month
apr     8.891111
aug    12.489076
dec    13.330000
feb     6.275000
jan     0.000000
jul    14.369687
jun     5.841176
mar     4.356667
may    19.240000
nov     0.000000
oct     6.638000
sep    17.942616
Name: area, dtype: float64
```

Table 3. Average Area damaged by fires by month

```
# grouping average area of forest fires by day
df.groupby('day').mean()['area']
```

```
day
fri     5.261647
mon     9.547703
sat    25.534048
sun    10.104526
thu    16.345902
tue    12.621719
wed    10.714815
Name: area, dtype: float64
```

Table 4. Average Area damaged by fires by day

```
# looking at max area burnt by month
df.groupby('month').max()['area']
```

```
month
apr      61.13
aug     746.28
dec      24.77
feb      51.78
jan       0.00
jul     278.53
jun      70.32
mar      36.85
may      38.48
nov       0.00
oct      49.37
sep    1090.84
Name: area, dtype: float64
```

Table 5. Max Area damaged by fires by month

```
# looking at max area burnt by day
df.groupby('day').max()['area']
```

```
day
fri      43.32
mon     278.53
sat    1090.84
sun     196.48
thu     746.28
tue     212.88
wed     185.76
Name: area, dtype: float64
```

Table 6. Max Area damaged by fires by day

Previously we had observed that August and September had the most number of forest fires. The above summary statistics brings some more interesting conclusions to light. For example, the average hectares of land damaged in fires was highest in May and September. Most land was damaged in July, August, and September.

Analysis of Numerical Data – Covariations Amongst Variables

Next, using a combination of univariate and multivariate visualizations, we can further learn about any potential relationships in the data. A histogram of each numerical attribute is shown below.

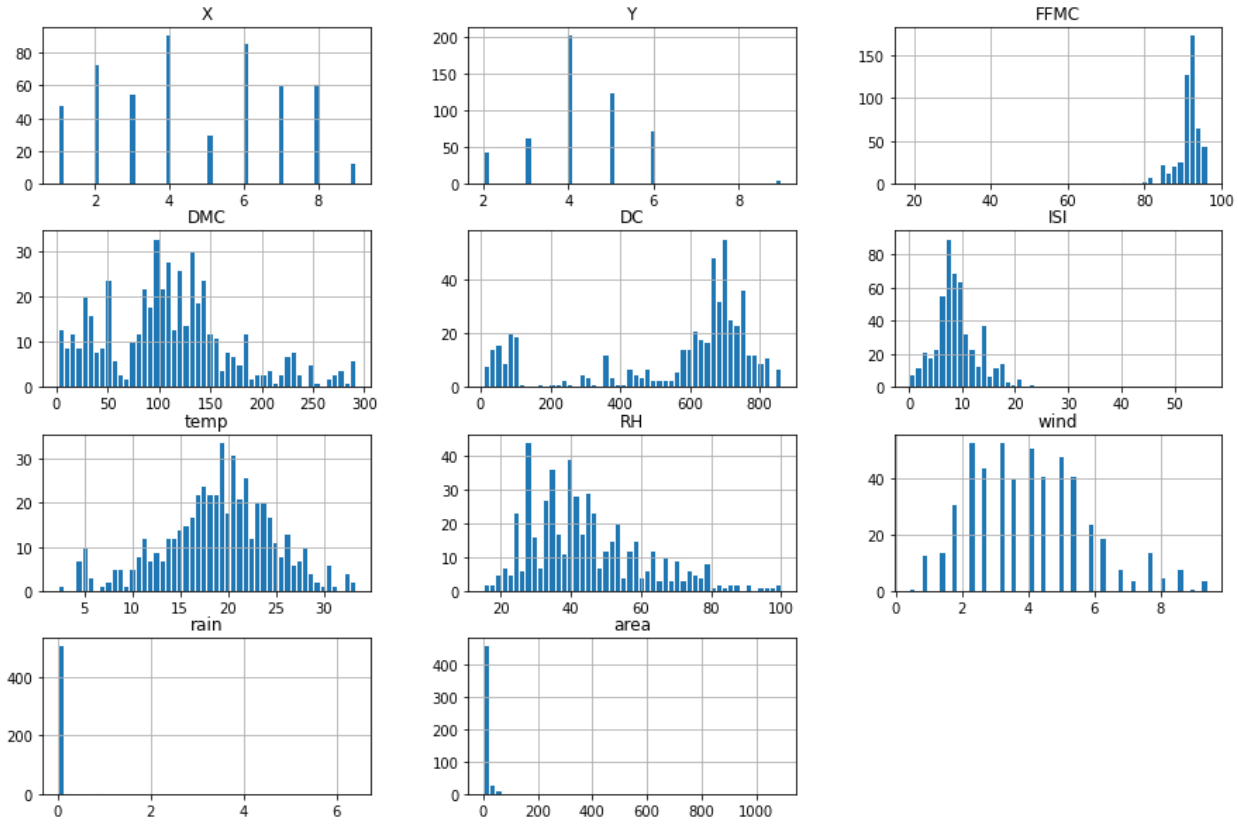

Figure 6. Histogram of each numerical feature in dataset

Temperature has a near Guassian Distribution. There are a mixture of positive skews and negative skews among the other attributes. Rain and ISI match Area's skewness (they are both also largely positively skewed) and FFMC has a large negative skew. DMC and RH have a slight positive skew and wind has a slight negative skew.

Next, we check for any correlations among attributes and each other or the target variable using heatmap as can seen below.

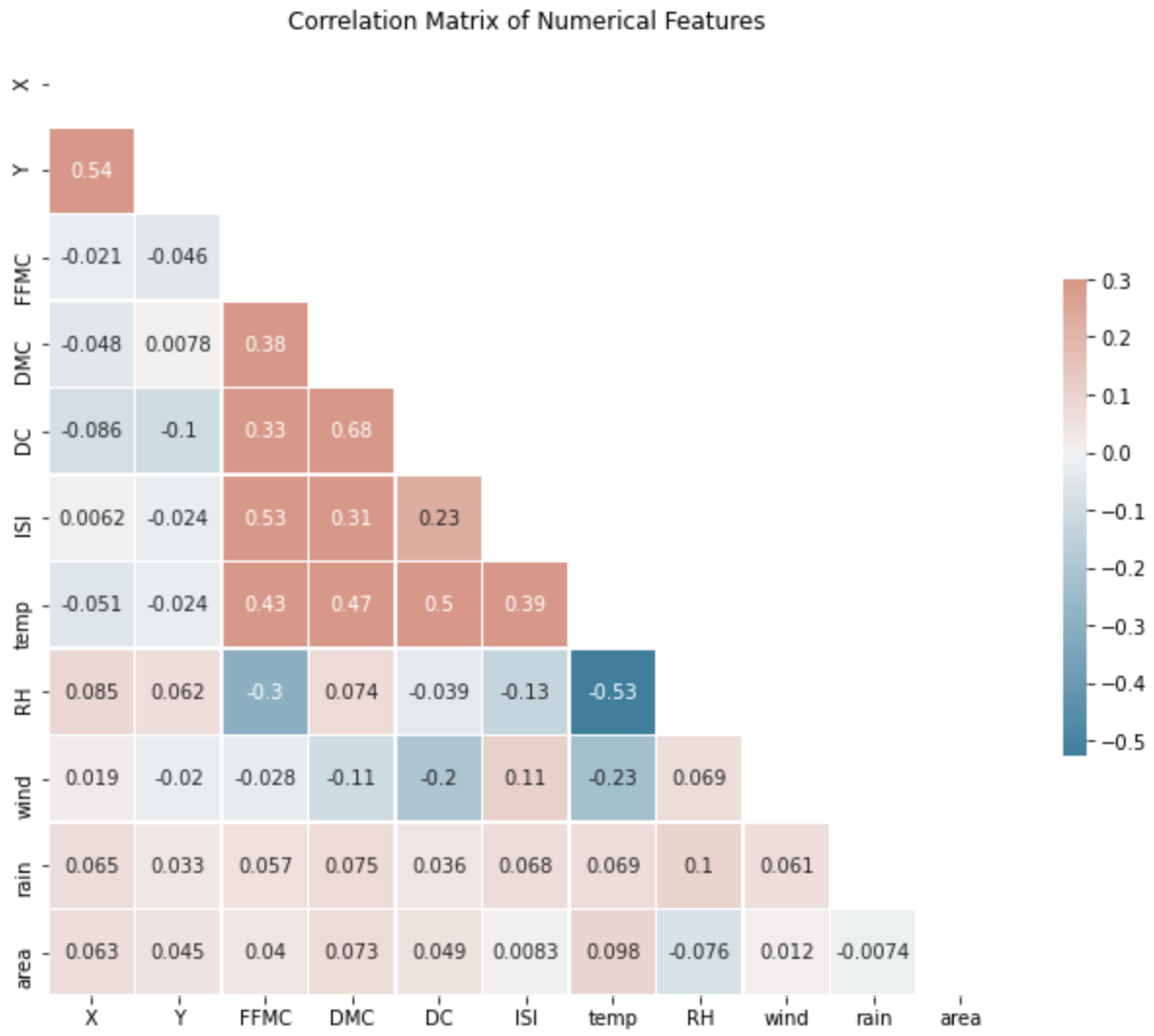

Figure 7. Correlation plot of each numerical feature in dataset

Here we can infer some correlation amongst some of the features against each other.

Any correlations above 0.5 or below -0.5 we'll consider strongly correlated:

- Y and Y are strongly correlated,
- FFMC and ISI are strongly positively correlated,
- DC and DMC are strongly positively correlated,
- Temp and DC are strongly positively correlated, and

- Temp and RH are strongly negatively correlated.

While the correlation matrices above demonstrate that there are some correlations between some of the features, there is no correlation between one feature and the target area.

It should be noted however that correlation is not always a strong indicator on whether a feature has a relationship with the target variable since correlation cannot measure curvilinear relationships. Therefore, here we plot the relationship between all the attributes.

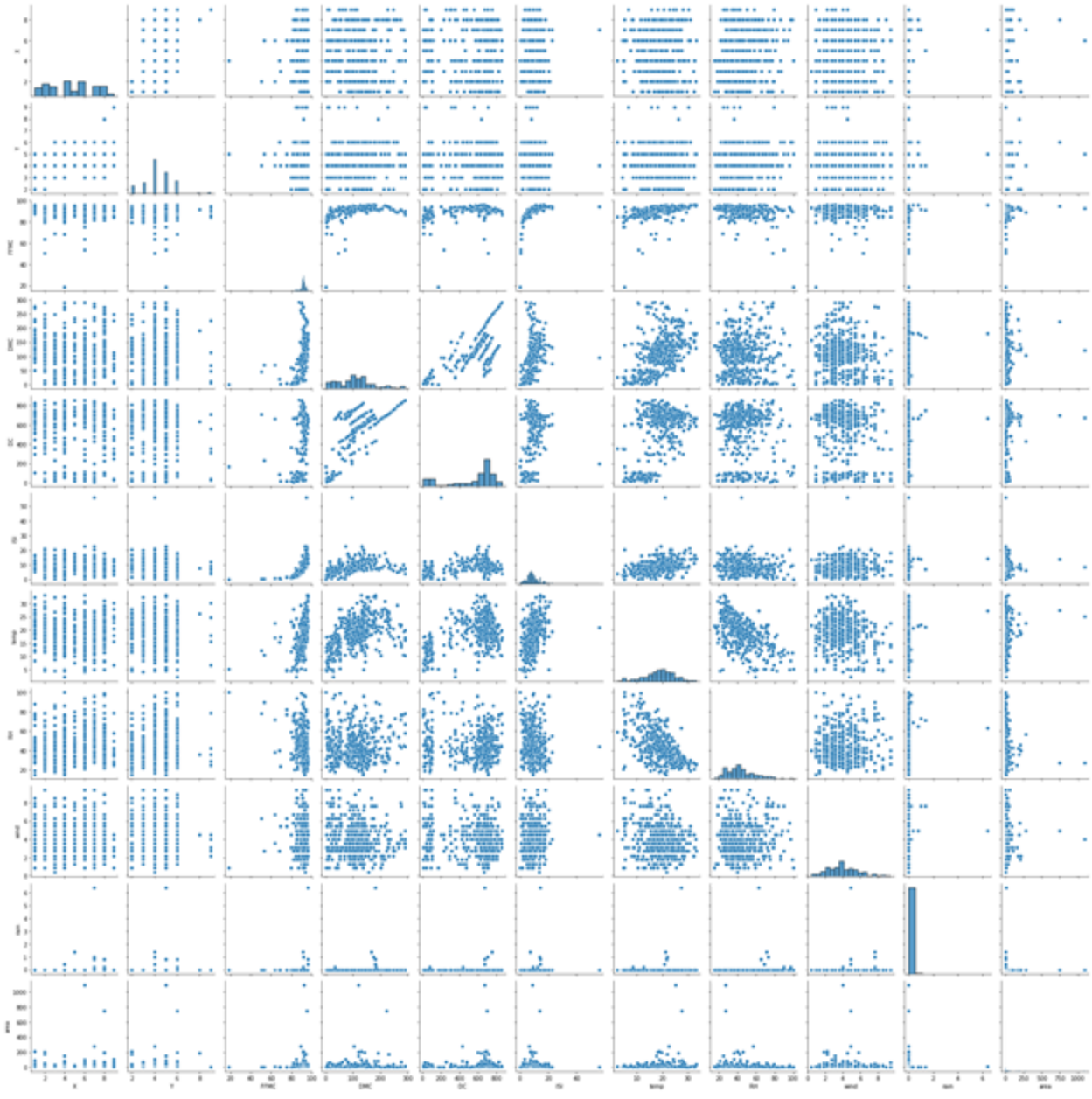

Figure 8. Pairplot of each numerical feature in dataset

Looking specifically at the area variable, there is potentially a curvilinear relationship with FFMC, DC, ISI, temp and RH. We analyze these 5 attributes further below after some preprocessing.

Data Preprocessing

To continue with our exploration of this dataset, we need to transform some of the attributes to aid further analysis and to be in the correct format for the machine learning algorithms.

We already confirmed that there are no missing values in this dataset, so there are 5 main ways we preprocessed this dataset:

1. Binning of the target attribute into discrete classes
2. Transforming categorical features to numerical values
3. Outlier Detection and Removal
4. Oversampling
5. Logarithmic transformation of the data

Binning of Target Attribute into Discrete Classes

To do some further analysis on the attributes, we wanted to look at whether there are any groups in the data by area. To do this, we binned the values in ‘area’ into 5 distinct groups. The discretized bins were then transformed into numerical values.The below bar chart shows that the majority of fires damaged less than 5 hectares of land.

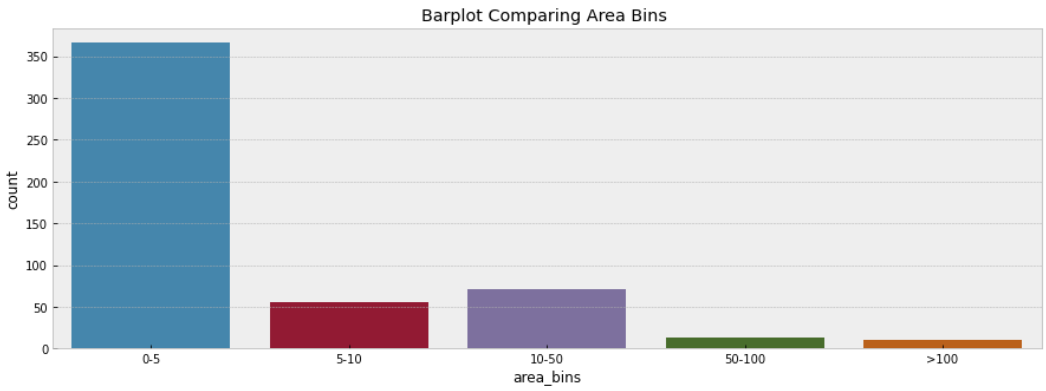

Figure 9. Count of fires by new bin classes

The figure below shows the pair plots color-coded by the new bins for some of the more interesting attributes identified above.

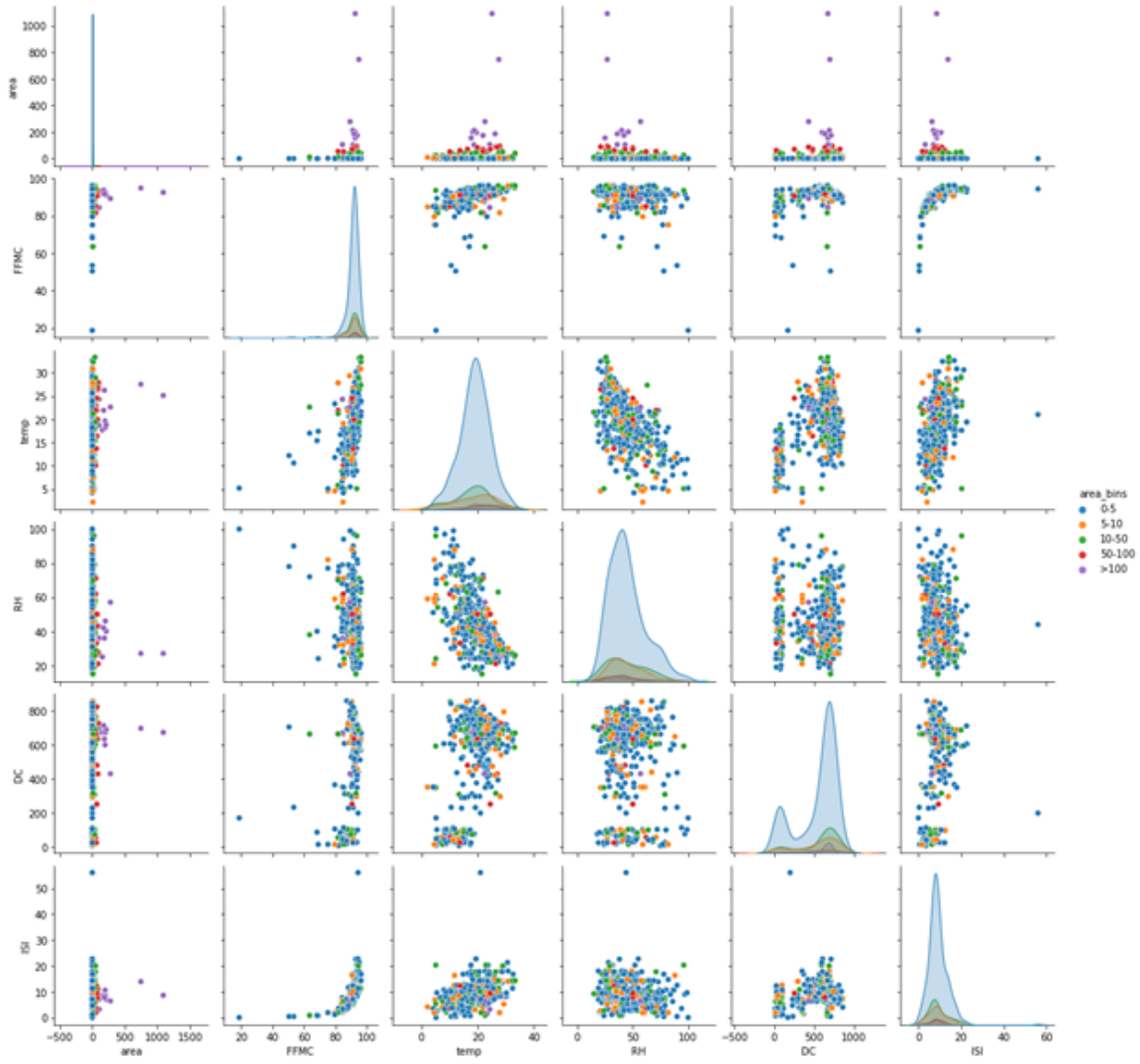

Figure 10. Pairplot of some numerical feature in dataset colour-coded by the new bins

There does not seem to be any immediately clear groupings within the data. Therefore, we conclude that not one specific attribute alone has a significant impact on the target variable ‘area’.

Transforming Categorical Features to Numerical Values

In order to get the data to a state where the algorithms can process the data, we need to encode the categorical attributes. There are only 2 categorical attributes; month and day. Due to the cyclical nature of these values, 1 of n or m of n mapping isn't going to work without introducing bias. Instead, month and day are mapped to numerical values using sin and cos, which maintains their cyclical significance. The code for this can be seen below.

```
import math

df["month_norm"] = 2 * math.pi * df["month"] / df["month"].max()

df["month_cos"] = np.cos(df["month_norm"])
df["month_sin"] = np.sin(df["month_norm"])

df["day_norm"] = 2 * math.pi * df["day"] / df["day"].max()

df["day_cos"] = np.cos(df["day_norm"])
df["day_sin"] = np.sin(df["day_norm"])

del df['month_norm']
del df['day_norm']
df.head()
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	month_cos	month_sin	day_cos	day_sin
0	7	5	2	4	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	0.415415	0.909632	-0.5	-8.660254e-01
1	7	4	9	1	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	0.415415	-0.909632	0.5	8.660254e-01
2	7	4	9	5	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	0.415415	-0.909632	0.5	-8.660254e-01
3	8	6	2	4	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	0.415415	0.909632	-0.5	-8.660254e-01
4	8	6	2	6	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	0.415415	0.909632	1.0	-2.449294e-16

Table 7. Feature encoding of categorical attributes

Outlier Detection and Removal

As was determined earlier in our exploratory analysis, there are a few extreme values in our dataset. Three different methods were compared to decide on our final outliers for the dating mining algorithms. As these outliers could potentially have a significant impact on the final results of our algorithms, we decided to run the algorithms with and

without outliers to see what difference they made. The three outlier detection methods we use are:

1. Z-score
2. Isolation forest
3. Elliptic Envelope

Z-score method was used on individual attributes to identify those instances with extreme values in any one feature. Instances that have a value more than 3 standard deviations away from the mean were considered an outlier.

Isolation forest is an unsupervised learning algorithm from the scikit-learn python library that returns an anomaly score for each instance. This method is based on the concept of decision trees and makes the assumption that outliers are few and far from the ‘normal’ instances (or different). Isolation forests are an ensemble of isolation trees for all the instances in the data set and anomalous points are the shortest branches of the tree. It isolates outlier observations by creating branches for each observation and those observations on the shortest branches are anomalous.
Running the Isolation Forest algorithm on our dataset found 52 outliers. After running a Principal Component Analysis on the dataset, we plotted the points on a scatterplot which displays instances that were classified as outliers.

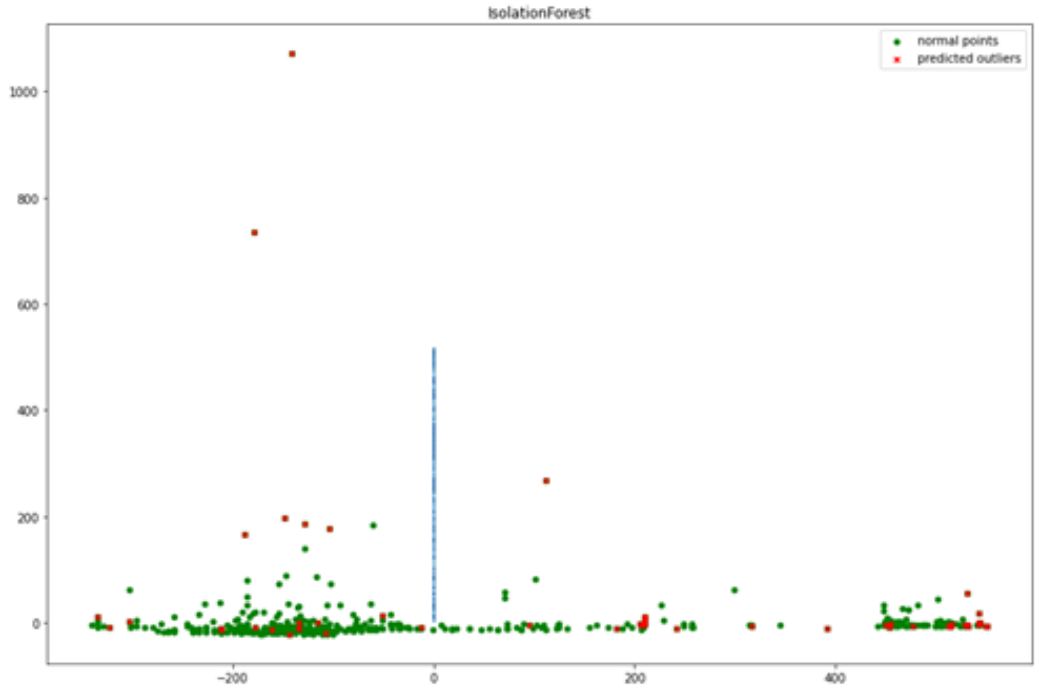

Figure 11. Outliers detected by Isolation Forests algorithm

The elliptic envelope algorithm in python's scikit-learn library uses the Minimum Covariance Determinant (MCD) equation. MCD is a highly robust estimator of multivariate location and scatter. Outliers are detected as those instances that are far from the distribution. The elliptic envelope works by:

1. Fitting a robust covariance estimate to the data.
2. Defining a hypershere/ellipsoid around the perimeter of the majority of the points.
3. Points that fall outside of this ellipsoid are then classified as the outliers.

Similar to isolation forests, the elliptic envelope algorithm will give each observation a value, 1 for inliers and -1 for outliers. This algorithm also found 52 outliers, but not the same instances were classified as outliers between the 2 methods. After running a Principal Component Analysis on the dataset, we plotted the points on a scatterplot which displays instances that were classified as outliers.

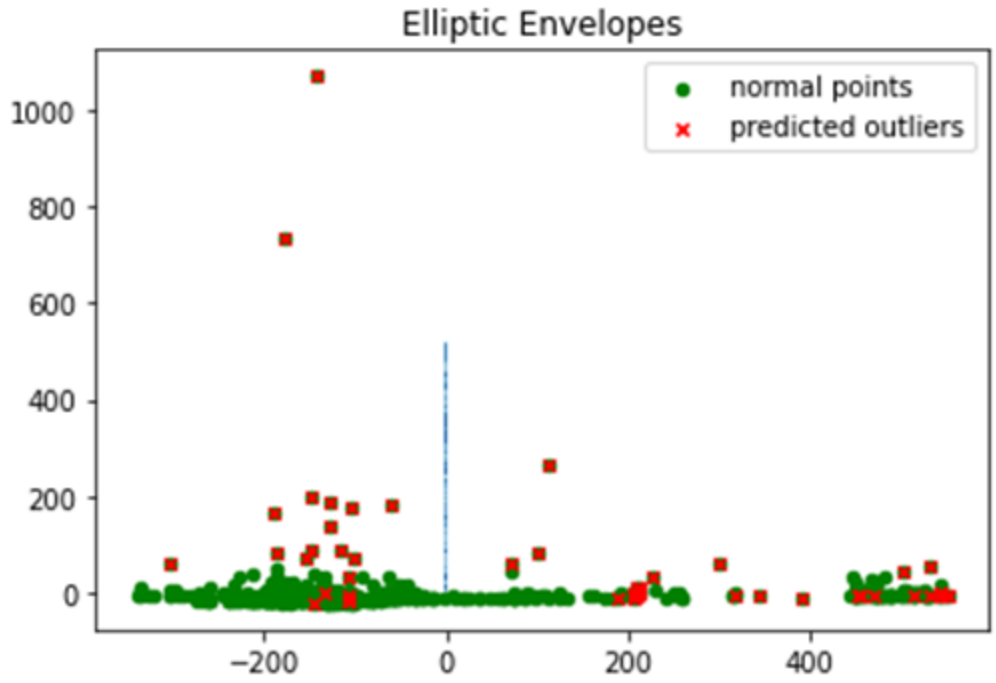

Figure 12:. Outliers detected by Elliptic Envelope algorithm

Since these 3 methods gave us different instances as classifiers, we decided to use a majority vote method wherein if both algorithms classified an instance as an outlier, it would be an outlier. Using this rule, we ended up with 33 outliers in the dataset.

Oversampling

To enhance the precision, recall and f1-score values of classification models that were affected by class imbalance, a technique known as oversampling was carried out. The oversampling technique utilized in this analysis was the Synthetic Minority Oversampling Technique (SMOTE). By default, oversampling replicates the examples of the low abundant classes until they match the majority class.

Logarithmic Transformation of the Data

Due to the high positive skew of the target attribute, we can log transform this data to make it as “normal” as possible so that the statistical analysis results from this data become more valid. The log transformation reduces or removes the skewness of our original data. The log transformation is performed on those variables that contain outliers: 'area','FFMC','ISI','rain' as well as the target attribute ‘area’. We would limit the values of variables to 3 standard deviations if the kurtosis remains high after logarithmic transformations.

Methods

Linear Regression

In linear regression, the target variable that we wish to predict is numeric while the predictor variables can be either categorical or numeric. For a given observation i (where $i = 1, 2, \ldots, n$) with p predictor variables, the multiple linear regression can be modelled as $Y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \varepsilon_i$. In this model, Y_i is the true value of the target variable for observation i while β_0 is the y intercept and $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficient values for each of the p predictor variables. Each of the $X_{i1}, X_{i2}, \ldots, X_{ip}$ are the values of the p predictor variables for observation i. ε_i is the error of the model, which is normally distributed with mean 0 and variance σ^2. In effect, this means that the errors are centered at 0 and have a constant variance. (Helwig, N., 2017)

Our fitted model is then $\widehat{Y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 X_{i1} + \ldots + \widehat{\beta}_p X_{ip}$ where $\widehat{Y}_i$ is the predicted value of the target variable for observation i and $\widehat{\beta}_0$, $\widehat{\beta}_1$, ..., $\widehat{\beta}_p$ are our coefficient estimates. In order to fit our model, we need to minimize the error term, $\varepsilon_i = Y_i - \widehat{Y}_i$, that is, minimize $\varepsilon_i = Y_i - \widehat{\beta}_0 - \widehat{\beta}_1 X_{i1} - \ldots - \widehat{\beta}_p X_{ip}$. For this, we use maximum likelihood estimation (MLE) to solve for the $p + 1$ different $\widehat{\beta}$ estimates using the training data. Once we have found these estimates, we can predict use them to predict the numeric target variable, $\widehat{Y}_i$, using the training data.

It is important to note that for each categorical predictor variable, we need to estimate multiple $\widehat{\beta}$ estimates. Since the data values for this variable are not numeric and therefore cannot be represented by just a single coefficient estimate, we must represent each categorical level with its own binary variable so that the data values are equal 1 if the observation is a part of a particular category and 0 if it is not. In this way, each coefficient estimate for these variables will represent the effect of the observation having that category and not any of the others for that particular categorical variable. For a variable with k levels, we would only have to estimate k-1 different coefficients as one of the levels will be the baseline case and therefore its effect is already taken into consideration if the observation falls into that category.

Tree-based Models

Tree-based models are very common in data analysis and machine learning because they are visually appealing and easy for non-statisticians to understand.

In order to construct an unpruned classification tree, for each of $j = 1, 2,..., p$ predictors, we randomly consider a finite set of n values of $X_j = s$, that is, consider $X_{1j}, X_{2j},..., X_{nj}$. At each of these n values, perform a binary split of the training observations so that you have two regions $R_1(j, s) = [X|X_j < s]\ and\ R(j, s) = [X|X_j \geq s]$. Then for each of the j,s pairings, compute the class probabilities $\widehat{p_{mk}}$ in each of the regions Rm, that is, the proportion of training observations in the mth region that are of each of the k classes. (Mubayi, A., 2017)

Then compute the Gini index as

$$G = \sum_{k=1}^{K} \widehat{p_{mk}}(1 - \widehat{p_{mk}})$$

for each of the two regions resulting from each j, s binary split. We select the j, s pairing that minimizes the sum of the Gini indexes in each of the two regions, that is,

$$argmin_{j,s}[G_1 + G_2]$$

This essentially selects the j, s pair that splits the observations into the purest regions possible (i.e. very high proportions of a class in one region and very low proportions in the other). (Quinlan, J.R., 1986)

Each of the newly split regions are called nodes, as is the location of the initial splitting. This procedure is called recursive partitioning. Each newly formed terminal node will continue attempting to split until it has less than some previously defined number of observations in it. Additionally, if any terminal node in the entire tree has less than another (smaller) previously defined number of observations in it then the entire partitioning process will stop.

Once the tree is finished growing, the predicted class for each terminal node, $\widehat{Y}_m$, is then the most commonly occurring class of all training observations in that region. This model is called an unpruned tree.

Very large trees often result in overfitting of the training data which can result in varying degrees of predictive ability for new test observations as the tree is fit very specifically to the training data. To solve this issue, it is often beneficial to decrease the size of the tree, called pruning. Pruning is essentially the reversing of one or more binary splits, starting from the terminal nodes and working it's way back up (or down) the tree. (Hastie, T., Tibshirani, R., & Friedman, J. H., 2009)

To determine which smaller-sized tree is optimal, we define the cost function as

$$C = \sum_{X_i \in R_m} I(Y_i \neq \widehat{Y}_m) + \alpha|T|$$

that is, the sum of all misclassified training observations with a cross-validated penalty term α multiplied by the number of binary splits, $|T|$. The optimal tree is then the one that minimizes the cost function by reducing expendable binary splits while maintaining the lowest misclassification rate possible. The pruning procedure therefore reduces the overfitting of the training data while being careful not to underfit the data (i.e. make the tree too small).

Bagging is a method that involves the bootstrap aggregation of many different tree models (James, G., Witten, D., Hastie, T., & Tibshirani, R., 2013). In order to create this aggregation, we draw N different random samples from the data, each of which have the same sample size but are drawn with replacement from the dataset. Each of these N random samples is then used to construct an unpruned classification tree. These N different trees make up the bagging model.

When testing a bagging model, we run each observation from the testing data through each of the N trees in the model, resulting in N predicted classes for each observation. The overall predicted class, $\widehat{Y}$, for a given observation is then the class that is predicted the greatest number of times throughout all of the N trees in the model for that observation.

This method does not involve the pruning of trees as the variance in the predictions is decreased when the N trees are averaged out so that if any particular tree overfits the training data, it's effect will be averaged out in the overall model.

Another common tree-based model is called a random forest model. A random forest model is created identically to a bagging model except for one key difference. The

difference is that for each of the N trees in a random forest, we only consider a subset of the total number of p predictors for each binary split. The most common amount of predictors used at each split is $\sqrt{p}$ predictors.

During the construction of each of the N trees, a random sample of size $\sqrt{p}$ is chosen from the total number of predictors and only these selected predictors are considered when determining the best j, s pairing to be used to split each node. A new randomization of predictors is conducted for each subsequent binary split. Due to these randomizations, N significantly different unpruned trees will be created and used in a random forest model. This should go even further in the prevention of overfitting than the bagging model but may or may not improve the accuracy of the overall predictions, depending on the particular trees that are used in each model.

Regression for Tree-based Methods

In addition to classification, tree-based methods can also be used for regression. The primary difference between the use of decision trees for classification and regression is in the metric that needs to be minimized in order to determine the most effective split. For classification, this metric is the sum of the Gini indexes of the two regions resulting from the splitting of a node. When trying to predict the value of a numeric variable, we need to minimize the residual sum of squares (RSS) in each of the two regions, Rm, for each split (Vala, K., 2019). That is, we need to select the j,s pair that results in the smallest total deviation between the true and predicted target variables for all training

observations undergoing that particular split. The predicted target value, $\widehat{C}_m$, for each of the observations in a particular region, m, is equal to the sample mean of the target variable for all of those observations. The RSS is then

$$\sum_{X_i \in R_m(j,s)} (Y_i - \widehat{C}_m)^2$$

for each of the two regions resulting from each j, s binary split. We select the j, s pairing that minimizes the sum of the RSS's in each of the two regions, that is,

$$argmin_{j,s}(RSS_1 + RSS_2)$$

This will then select the j,s pairing that results in predicted target values that most accurately represent the observations in its resulting regions. Of course, the fewer binary splits that a tree has, the more observations that will be present in the resulting terminal nodes and therefore each individual observation will have a lesser effect in the determination of its predicted target value (since the target will be averaged among more points), resulting in less accurate predictions and greater training error. For this reason, it is more effective to initially build a larger tree before eventually pruning it and/or growing a forest of trees, as discussed above.

K-nearest Neighbours

The k-nearest neighbours algorithm assembles each of the training instances in a list (Harrison, O., 2018). A particular value of K is chosen. A specific metric to measure the

distance between any two points by using the values of each instances' attributes is also chosen. The numeric attributes must be scaled so as not to bias the distance calculation toward any particular attribute. To classify a new observation, the distances between this point and all other points in the training set must be calculated and the K closest training instances to the new point are considered its nearest neighbours. The class of the observation is then the class that occurs most often among its K nearest neighbours. Alternatively, we can weigh the votes of the neighbours using the inverse of their distances to the test point so that the class labels of the neighbours that are closer to the test observation are weighed more heavily in determining the class of the test observation. (McCaffrey, J., 2019)

Results

```
(area     1.217838
 FFMC   -11.675394
 ISI     -0.937218
 rain    14.173028
 dtype: float64, area      0.945668
 FFMC    185.482383
 ISI       2.584588
 rain    234.240025
 dtype: float64)
```

Table 7. Log transformation to address skew and kurtosis

For the regression analysis, we started off by checking for the skew and kurtosis values of all the columns and noticed that the skew and kurtosis values for ‘area’, ‘FFMC’, ‘ISI’, and ‘rain’ were very large. We decided to apply some log transformations to these columns so that they could conform to normality.

```
Mean Absolute Error: 20.732832685938913
Mean Squared Error: 1074.2383869211876
Root Mean Squared Error: 32.77557607306373
```

Table 8. Linear Regression output

The log transformation helped to bring down the skew and kurtosis values of the selected columns. After applying the log transformation, it was found that the kurtosis value for ‘FFMC’ was still high with a value of 185.48, so the z-score outlier method was used to remove all values above 3 standard deviations of the mean. The log transformations were applied once more, the data was ready for the regression models. For the first regression model, we used data that did not have log transformations done on it, and found that the model did not perform well.

```
Mean Absolute Error: 1.1644711510367398
Mean Squared Error: 1.9298167245114264
Root Mean Squared Error: 1.3891784350872376
```

Table 9. Linear Regression Output with transformed data

The Mean Absolute Error resulted in a value of 20.73, the Mean Squared Error of the model was 1074.23 and the RMSE of 32.78 was high, so we applied the regression model on the data that had log transformations and outlier removal conducted on it, and found that the performance of that model was significantly better. The RMSE value had dropped from 32.78 to 1.39, indicating that the log transformations and outlier removals

```
Mean Absolute Error: 20.894966298840046
Mean Squared Error: 4152.843801477344
Root Mean Squared Error: 64.44256203377815
```

Table 10. Random Forest Regressor Model Output

```
Mean Absolute Error: 1.2840845574773978
Mean Squared Error: 2.6041157746680677
Root Mean Squared Error: 1.6137272925336759
```

Table 11. Random Forest Regressor Model Output with Transformed Data

```
The final RMSE on the test set is 1.39
The final MAE on the test set is 1.14
```

Table 12. Random Forest Regressor Model Output with Parameter Tuning

The next model that we used was a Random Forest Regressor. The random forest regressor was applied on both the data without the log transformation, and data with the transformation. The model output shown in Table 10 produced a MAE of 20.89, MSE of 4152.84 and RMSE of 64.44 which indicated poor performance. Next we tried applying the random forest model on the transformed data with log transformations and outlier removal, and found that it produced better results. Looking at the output in table 11, the

MAE was 1.28, MSE was 2.60, and the RMSE was 1.61. All of which were significant improvements over the previous model. In the last model, some parameter tuning was done by using a bootstrapping method both with and without replacement, varying the number of trees in the forest, and varying the maximum number of features included. The results of the random forest after parameter tuning produced an RMSE value of 1.39, and MAE of 1.14.

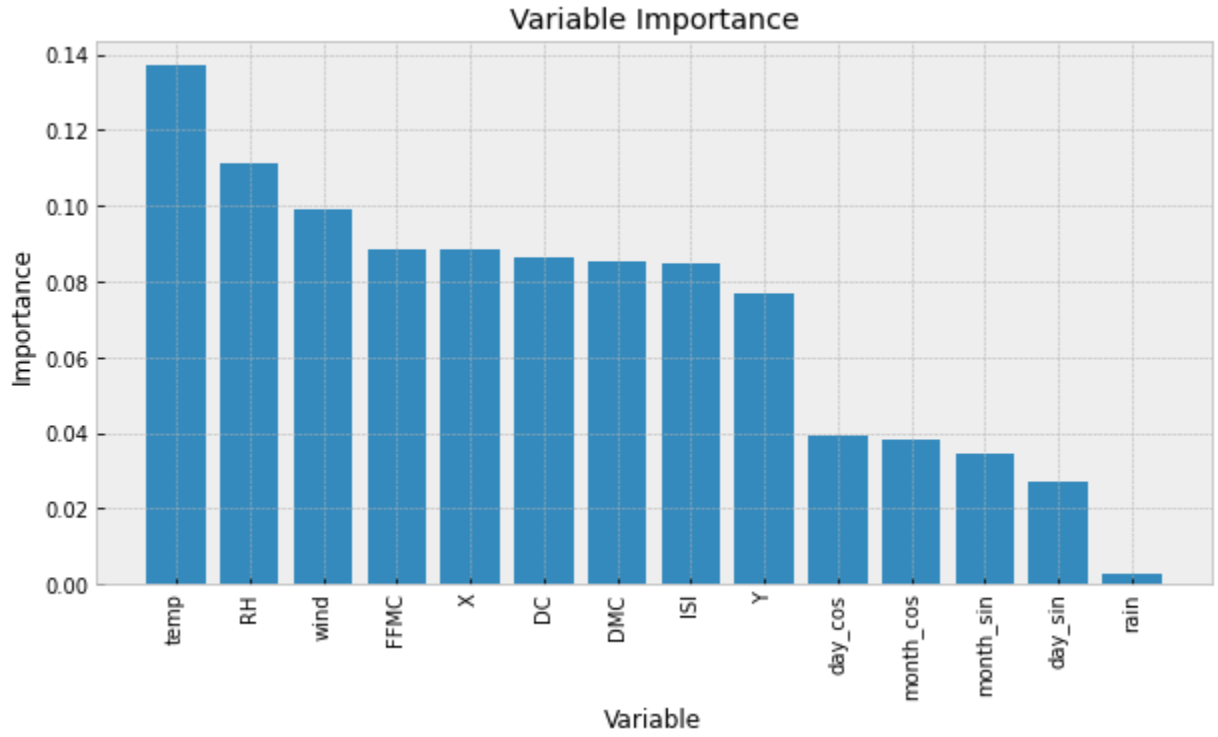

Figure 13. Bar graph for Variable Importance

The results of the regression model were used to create a plot of the features ordered by importance. The importance here is related to the features that reduce the error the most. The plot indicated that the top most important variable linked to forest fires was the temperature, while rain was the lowest.

	precision	recall	f1-score	support
0	0.78	0.97	0.87	119
1	0.20	0.05	0.08	19
2	0.50	0.08	0.14	12
3	0.00	0.00	0.00	4
4	0.00	0.00	0.00	2
accuracy			0.76	156
macro avg	0.30	0.22	0.22	156
weighted avg	0.66	0.76	0.68	156

Table 13. Classification Report of k-NN Model

The classification report displayed above shows results of the k-NN model. The accuracy score of the test set is 76%. The precision, recall and f1-score of the outcome of 0 (‘0-5’) are at 0.78, 0.97 and 0.87. For the outcome of 1 (‘5-10’), precision, recall and f1-score are at 0.20, 0.05, and 0.08. The precision, recall and f1-score of the outcome of 2 (‘10-50’) are at 0.50, 0.08 and 0.14. The precision, recall and f1-score for both the outcome of 3 (‘’50-100) and the outcome of 4 (‘>100’) are at 0.00, 0.00 and 0.00.

	precision	recall	f1-score	support
0	0.77	0.92	0.84	119
1	0.11	0.05	0.07	19
2	0.25	0.08	0.12	12
3	0.00	0.00	0.00	4
4	0.00	0.00	0.00	2
accuracy			0.71	156
macro avg	0.23	0.21	0.21	156
weighted avg	0.62	0.71	0.66	156

Table 14. Classification Report of Random Forest Model

Table 14 presents the results of the Random Forest model. The accuracy score of the test set is 71%. The precision, recall and f1-score of the outcome of 0 (‘0-5’) are at 0.77, 0.92 and 0.84. For the outcome of 1 (‘5-10’), precision, recall and f1-score are at 0.11, 0.05, and 0.07. The precision, recall and f1-score of the outcome of 2 (‘10-50’) are at 0.25, 0.08 and 0.12. The precision, recall and f1-score for both the outcome of 3 (‘’50-100) and the outcome of 4 (‘>100’) are at 0.00, 0.00 and 0.00.

Upon oversampling, the performance of the machine learning models was tested with a double cross-validation or nested cross-validation. This is known to be an ideal procedure to perform hyperparameter tuning while reducing biased evaluations. In the nested cross-validation the outer loop was input a k fold value of 10 while the inner loop had a k fold value of 3. As for hyperparameter tuning, there were 2 parameters with 3 values resulting in (3*3) combinations for each classification algorithm:

Random Forest Classification

- n_estimators: {10, 100, 500}
- max_features:{2,4,6}

KNN Classifier

- neighbors: {3, 5, 10, 20}
- weight: {‘uniform’, ‘distance’}

The iteration performed in the nested cross-validation produced the accuracy scores of the best performing classification models with the corresponding hyperparameters. A classification report and a confusion matrix was generated for both random forest and k-NN classification models.

	precision	recall	f1-score	support
0	0.75	0.44	0.56	34
1	0.69	0.91	0.78	32
2	0.76	0.82	0.79	39
3	0.97	0.95	0.96	37
4	0.91	0.95	0.93	41
accuracy			0.82	183
macro avg	0.82	0.81	0.80	183
weighted avg	0.82	0.82	0.81	183

Table 15. Classification Report of k-NN Model (oversampling)

Table 15 above illustrates the results of the k-NN Model model after conducting the oversampling technique. The accuracy score of the test set is 82 %. The precision, recall and f1-score of the outcome of 0 (‘0-5’) are at 0.75, 0.44 and 0.56. For the outcome of 1 (‘5-10’), precision, recall and f1-score are at 0.69, 0.91, and 0.78. The

precision, recall and f1-score of the outcome of 2 (‘10-50’) are at 0.76, 0.82 and 0.79. The precision, recall and f1-score of the outcome of 3 (‘’50-100) are at 0.97, 0.95 and 0.93. The precision, recall and f1-score of the outcome of 4 (‘>100’) are at 0.98, 0.98 and 0.98.

	precision	recall	f1-score	support
0	0.81	0.65	0.72	34
1	0.71	0.91	0.79	32
2	0.92	0.87	0.89	39
3	1.00	1.00	1.00	37
4	0.98	0.98	0.98	41
accuracy			0.89	183
macro avg	0.88	0.88	0.88	183
weighted avg	0.89	0.89	0.88	183

Table 16. Classification Report of Random Forest Model (oversampling)

Table 16 presents the results of the Random Forest model after conducting the oversampling technique. The accuracy score of the test set is 89%. The precision, recall and f1-score of the outcome of 0 (‘0-5’) are at 0.81, 0.65 and 0.72. For the outcome of 1 (‘5-10’), precision, recall and f1-score are at 0.71, 0.91, and 0.79. The precision, recall and f1-score of the outcome of 2 (‘10-50’) are at 0.92, 0.87 and 0.89. The precision, recall and f1-score of the outcome of 3 (‘’50-100) are at 1.00, 1.00 and 1.00. The precision, recall and f1-score of the outcome of 4 (‘>100’) are at 0.98, 0.98 and 0.98.

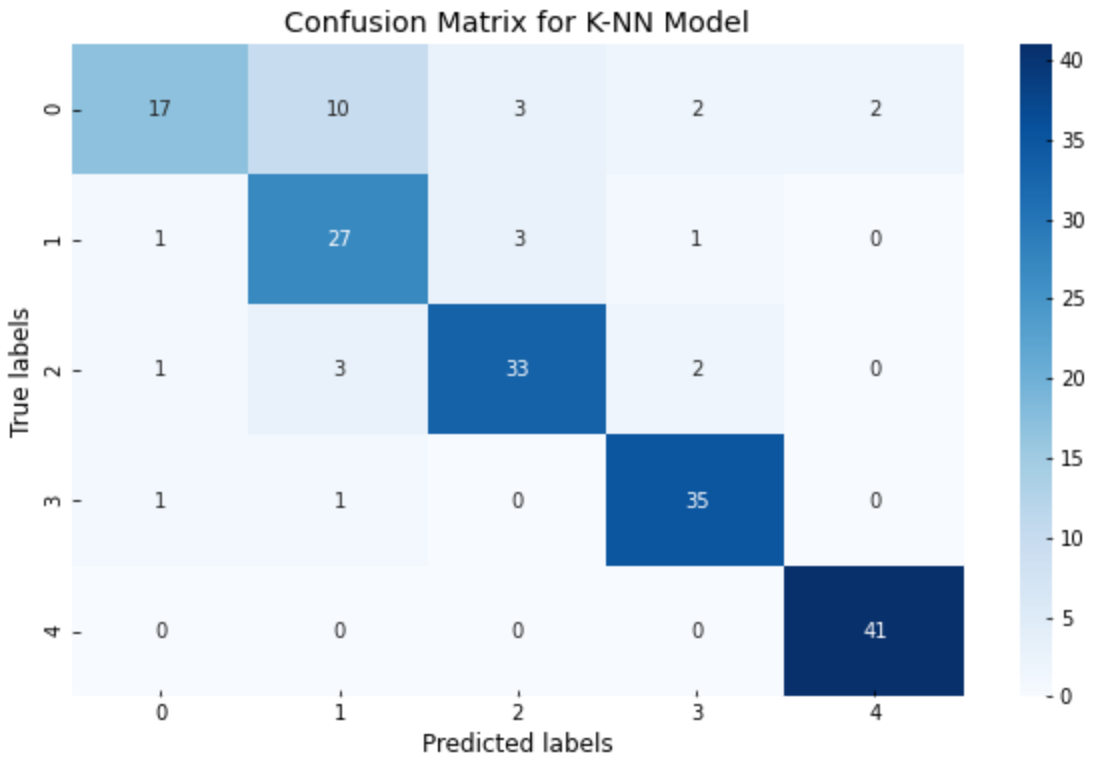

Figure 14. Confusion Matrix of the k-NN model (oversampling)

Figure 14 highlights the True Positive, False Positive, True Negative and False Negatives values of the random forest model. Class 0 (‘0-5’) has the lowest number of True Positives along with the greatest number of False Positives and False Negatives. Class 3 (’’50-100) has more True Positives while having a small number of False Positives and False Negatives.

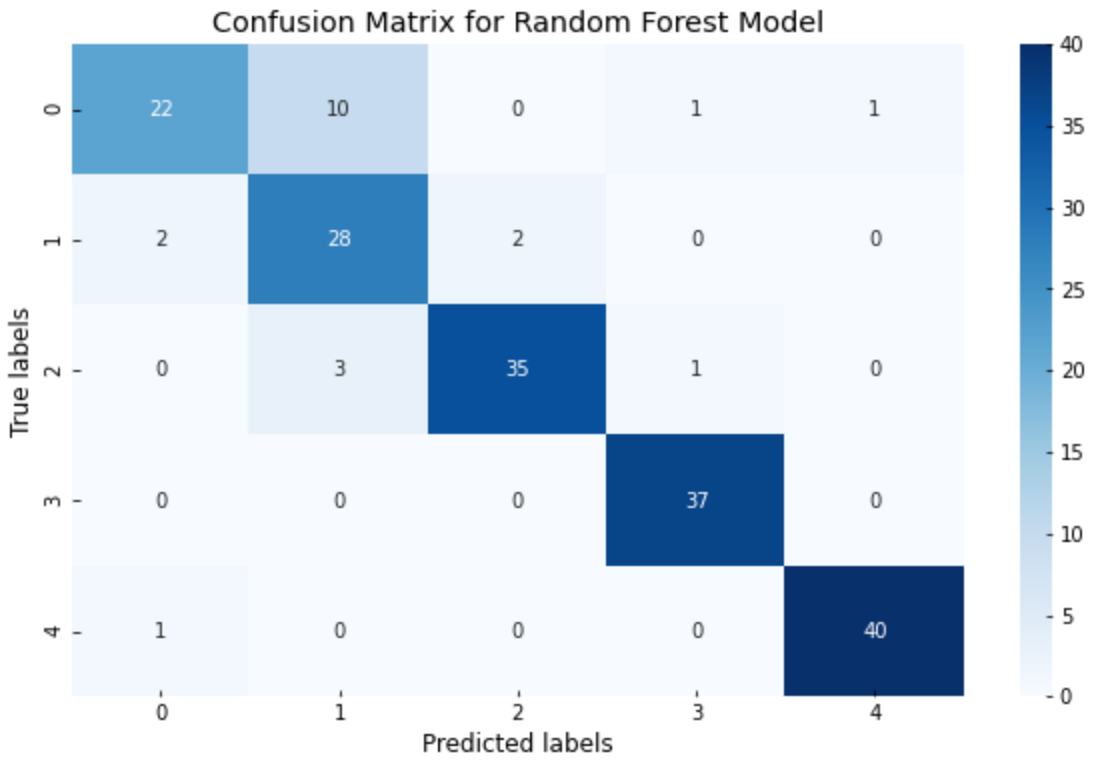

Figure 15. Confusion Matrix of the Random Forest model (oversampling)

Figure 15 highlights the True Positive, False Positive, True Negative and False Negatives values of the random forest model. The most number of True Positives are observed in class 4 (‘>100’). Class 3 has 0 False Negatives and False Positives. Class 0 (‘0-5’) has more False Positives and False Negatives compared to other classes.

The optimal parameters that produced the highest accuracy score for the k-NN model after performing GridSearchCV are 'n_neighbors': 1, 'weights': 'uniform'. The optimal parameters that produced the highest accuracy score for the random forest are 'max_features': 2, 'n_estimators': 500. The most important features according to the random forest model are ‘DC’, temp’ and ‘Y’.

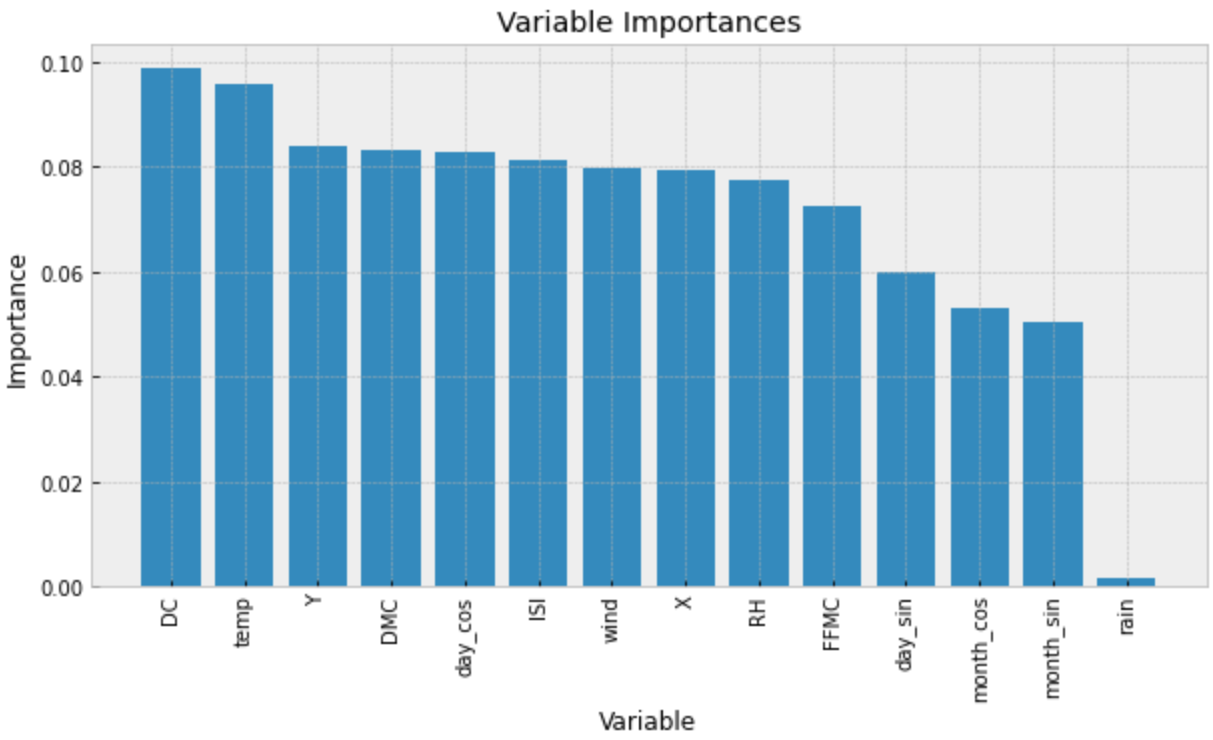

Figure 16. Feature Importance bar graph of the Random Forest model (oversampling)

Conclusions and Limitations

The continuous variable ‘area’ was assigned to be the dependent variable for the linear regression and random forest regression models. The RMSE and MAE metrics returned high values in both linear regression and random forest regression models. Upon further analysis through kurtosis and histogram plots, it was deemed that there were outliers in some features of the dataset. In response to this problem, log transformations were applied to specific features that contained the outliers. The data frame that consisted of transformed features was input to the regression models. The RMSE and MAE values have diminished significantly with log transformed data.

The performance of random forest and kNN classification models were poorly impacted by the problem of class imbalance. Despite having accuracy scores over 70% for both models, the low f1 scores indicated subpar performance. The accuracy scores generated over 70% was controlled by the over abundance of class 0 (‘0-5’). In the case of predicting forest fires, the emphasis should be placed on from classes 1 (‘5-10’) to class 4 (‘>100’), since the cost of not predicting a forest forest fire is more catastrophic than correctly predicting the outcome of 0 (True Negatives). To mitigate this problem, an oversampling technique was implemented. The oversampling technique produced robust results in classification reports and confusions matrices in both random forest and kNN models. In accordance with boxplots on month vs. area, forest authorities should allocate their attention on the months June, July, August, and September, as the average temperature appears to rise in these months. Based on the random forest classification feature importance metric in Figure 16, the three most important features for forest fire prediction are ‘DC’, ‘temp’, ‘ISI’ and ‘DMC’. Therefore, the local fire authorities should be cognizant of the average of moisture content of deep compact organic layers, the temperature, the numerical rate of fire spread and the numerical rating of the average moisture content of loosely compacted organic layer and medium size woody material. Interestingly, the rising temperature values seemed to be correlated with ‘DC’, ‘temp’, ‘ISI’ ,‘DMC’ and ‘FFMC’. However, in the random forest regression model, the variable ‘RH’, relative humidity has been revealed as one of the top three features of importance. Therefore, the fire authorities should also consider ‘RH’ as a significant predictor along with the aforementioned variables.

A prominent limitation of this analysis was the small sample size. Another limitation of our analysis was that all of the forest fires in our dataset occurred in one particular region of Portugal. By using a much more geographically broad dataset or a consolidation of multiple datasets from different localities our conclusions could have been much more applicable to stakeholders from around the world. It would be of interest to examine if the conclusions that we were able to draw from this Portuguese dataset could be applied in other countries where climate and weather conditions may be very different. If there appeared to be a divergence in the results when comparing different regions, determining the cause of these variations could enhance global efforts to fight forest fires. A future direction of this analysis can be the incorporation of other machine learning models such as SVM classification/regression, Boosting, Neural Networks etc.

References

P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9.
http://www.dsi.uminho.pt/~pcortez/fires.pdf

National Wildfire Coordinating Group. (2021, 03, 10). *Fire Weather Index (FWI) System.* Retrieved from NWCG:
https://www.nwcg.gov/publications/pms437/cffdrs/fire-weather-index-system

Helwig, H. (2017). Multiple Linear Regression. *University of Minnesota*.
http://users.stat.umn.edu/~helwig/notes/mvlr-Notes.pdf

Mubayi, Anuj. (2017). *Handbook of Statistics, 249-304.*
https://www.sciencedirect.com/science/article/pii/ S0169716117300172

Quinlan, J.R. (1986). Induction of Decision Trees. *Mach Learn 1, 81–106*.
https://doi.org/10.1007/BF00116251

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R. Corrected edition New York: Springer.*
https://faculty.marshall.usc.edu/gareth-james/ISL/

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York: Springer.*
https://web.stanford.edu/~hastie/ElemStatLearn/

Faraway, J. (2016). *Extending the Linear Model with R: Generalized Linear, Mixed Effects, and Nonparametric Regression Models. 2nd ed. Boca Raton: CRC Press.*

https://englianhu.files.wordpress.com/2016/01/faraway-extending-the-linear-model-with-r-e28093-2006.pdf

Vala, K. (2019). Tree-Based Methods: Regression Trees. *Towards Data Science*. https://towardsdatascience.com/tree-based-methods-regression-trees-4ee5d8db9fe9

Harrison, Onel. (2018). Machine Learning Basics with the K-Nearest Neighbors Algorithm. *Towards Data Science*. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

McCaffrey, James. (2019). Weighted k-NN Classification Using Python. *Visual Studio Magazine*. https://visualstudiomagazine.com/articles/2019/04/01/weighted-k-nn-classification.aspx