

CSE 514A Final Report: Algerian Forest Fires

1. Introduction

The Algerian forest fires dataset was published in 2019 in the UC Irvine Machine Learning Repository. Collected between June 2012 and September 2012, the dataset provides date, weather information, fire weather index, and classification label of whether fire occurred or not, and where the data was sampled (Bejaia region in the northeast of Algeria or the Sidi Bel-abbes region in the northwest of Algeria). The dataset has 122 instances each coming from the two regions in Algeria and does not have missing values. In total, it contains a total of 15 features and 244 instances.

Two classification problems were identified. The first classification problem was to identify if data observations come from Bejaia or Sidi-Bel Abbès. The second classification problem was to identify if, given the weather and fire index conditions, wildfire occurred.

The motivation behind the classification problems is to assist government agencies overseeing wildlife and natural habitats in Algeria. Using the classification models, the government agencies can improve their predictions of wildfire occurrences based on weather information and fire index measurements. When the conditions indicate high risk for wildfire, they can act quickly to prevent fires before they start. They can also identify which region is more likely to experience certain weather and fire index conditions, allowing them to create tailored wildfire prevention and control strategies to each region.

2. Results

2.1 Description of Models

For each of the problems, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) models were fitted and tested. KNN is a model that identifies k nearest points from the training dataset by computing the distance between a new data point and all training data points. It predicts the class of the new data point based on the labels of these k nearest points through majority voting. SVM identifies the hyperplane that best separates points belonging to two or more classes by maximizing the margin, the distance between the hyperplane and support vectors, in a multidimensional feature space.

2.2 Model Runtime

KNN model does not train before testing. Instead, during testing, it calculates distances between test points and training data to find the k nearest neighbors. SVM, on the other hand, trains to find the best hyperplane and does not traverse the training dataset during testing.

Because of this, KNN takes a shorter training time but a longer testing time compared to SVM. Table 1 confirms that KNN model was faster to fit but took longer to test than SVM.

Table 1. Runtimes for First Classification Problem and Second Classification Problem

	First Classification Problem		Second Classification Problem	
	Training	Testing	Training	Testing
KNN	0.001974	0.005000	0.003000	0.005021
SVM	0.006180	0.003016	0.004936	0.002000

2.3 Model Performance

A. Before Dimension Reduction

As demonstrated in table 1, there were, in total, four models trained and tested. Table 2 shows the accuracy score of each model for training and testing. The model with the best training accuracy score was the SVM model for the second classification problem. The model with the worst training accuracy score was the KNN model for the first classification problem, which also had the worst testing accuracy score. However, the KNN and SVM models for the second classification problem had the same testing accuracy score. Looking at the overall trends, the two classification problems were not equally predictable. The accuracy scores of both KNN and SVM models for the second classification problem were significantly higher than the accuracy scores for the first classification problem.

Table 2. Accuracy Scores for First Classification Problem and Second Classification Problem
Accuracy Score

	First Classification Problem		Second Classification Problem	
	Training	Testing	Training	Testing
KNN	0.7991	0.64	0.9361	0.92
SVM	0.8174	0.68	0.9817	0.92

B. After Dimension Reduction

Dimension Reduction had different impacts for each classification problem. As shown in table 3, the dimension reduction had a negative impact on the first classification problem. Following the dimension reduction, the training accuracy scores for KNN and SVM models decreased, along with the testing accuracy score for the KNN model. However, as shown in table 4, the dimension reduction had an overall positive impact on the second classification problem. While the testing accuracy scores for KNN and SVM models remained constant, the

training accuracy score for KNN model increased more than the training accuracy score for SVM decreased.

Table 3. Accuracy Scores for First Classification Problem Before and After Dimension Reduction

	Before Dimension Reduction		After Dimension Reduction	
	Training	Testing	Training	Testing
KNN	0.7991	0.64	0.7945	0.52
SVM	0.8174	0.68	0.7808	0.68

Table 4. Accuracy Scores for Second Classification Problem Before and After Dimension Reduction

	Before Dimension Reduction		After Dimension Reduction	
	Training	Testing	Training	Testing
KNN	0.9361	0.92	0.9635	0.92
SVM	0.9817	0.92	0.9772	0.92

2.4 Recommendation

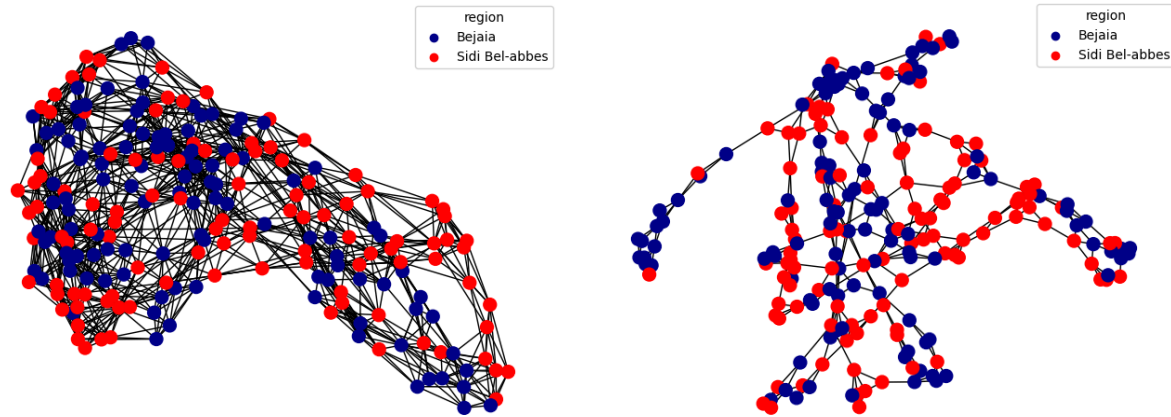
A. First Classification Problem

The first classification problem looked at identifying the region, Bejaia or Sidi Bel-abbes, given a total of 10 weather and fire weather index system variables. In figure 1, I visualized the result for the KNN model as a network graph where each data observation from the training set is represented as a node and each node is connected to its k nearest neighbors. I used the Python network graphing package called 'networkx' and used the spring layout, where edges pull connected nodes together while nodes repel one another, thereby reflecting the relative distances of the data points. The blue nodes indicate data points that come from Bejaia while the red nodes come from Sidi Bel-abbes. The figure on the left was constructed before dimension reduction and the figure on the right was constructed after dimension reduction.

Since the KNN model predicts the labels of new data points by performing a majority voting of the labels of their k nearest neighbors, the model would perform better if the nodes that are connected or closer together are of the same color. However, one can see that the blue and red nodes are not separated from one another, even after dimension reduction. While there are some nodes whose majority of k nearest neighbors are of the same color, most nodes appear to have a mixture of two colors in their pool of k nearest neighbors. This suggests that the weather and fire weather index system variables cannot effectively distinguish each region. From looking at the result of the KNN model for the first classification problem, I would recommend that the government agencies of Algeria treat the two regions as similar in terms of their weather and fire

weather index system conditions. Since wildfire depends on those conditions, it would be best to devise and implement similar fire prevention and control strategies in the two regions.

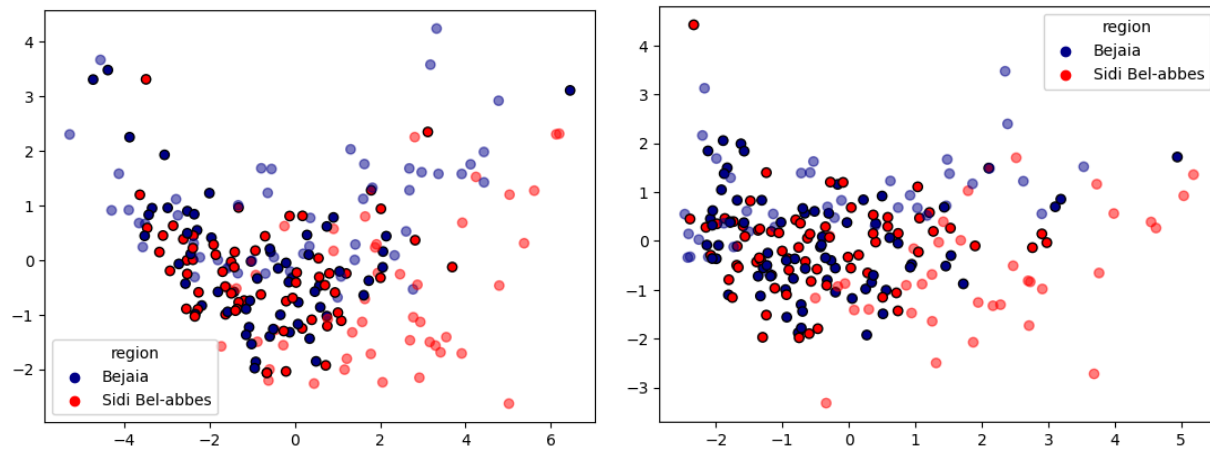
Figure 1. Network Graph of K Nearest Neighbors from the Training Set Before and After Dimension Reduction



In addition to the KNN model, I also visualized the result for the SVM model. Again, due to the high dimension of the dataset, it is difficult to visualize the decision boundary. Therefore, I applied Principal Component Analysis (PCA) to project the training set into a 2-dimensional space. PCA identifies two Principal Components (PC), which are linear combinations of the original features and define the coordinate system of the 2-D space.

Figure 2 shows the training set plotted in the 2-D space with support vectors highlighted with greater opacity and black outlines. The SVM model used the 'linear' kernel, I would expect the training data to appear linearly separable if the model performed well in the higher dimensional space. However, the blue and red points, which each represent data observation, are significantly overlapping. There are some regions where the blue and red points are not surrounded by their opposite colors. For example, in the area, defined by positive values of the first PC and negative values of the second PC, mostly red points are found. Nevertheless, this does not demonstrate effective separation between the two regions. Therefore, the recommendation that the government agencies do not distinguish the two regions when it comes to evaluating weather and fire weather index system conditions is supported by the results from both the KNN and SVM models.

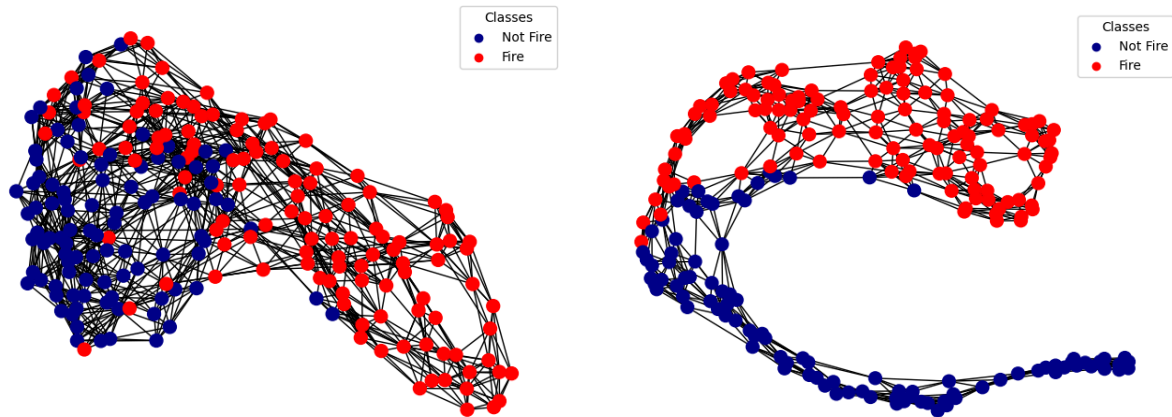
Figure 2. PCA Graph of the Training Set and Support Vectors Before and After Dimension Reduction



B. Second Classification Problem

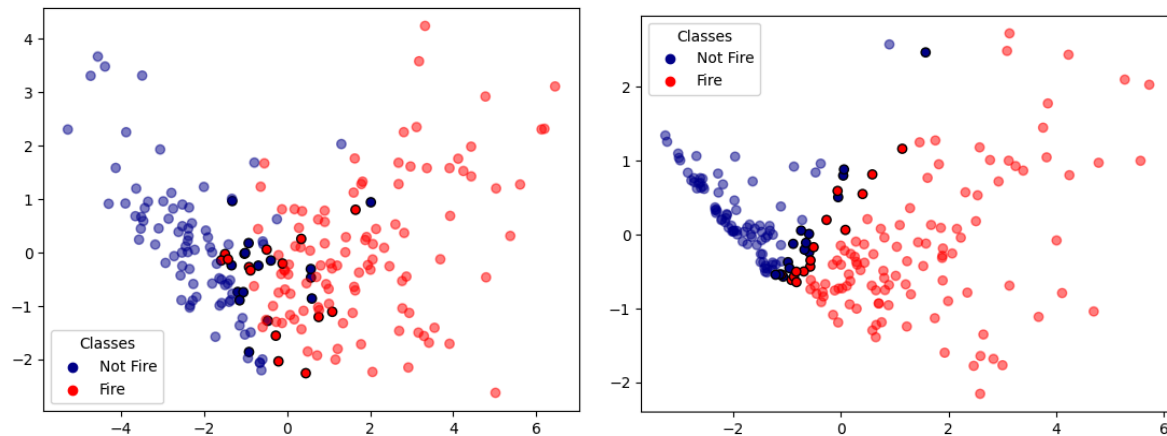
The second classification problem looked at identifying whether fire occurred given the weather and fire weather index system variables. In figure 3, I again visualized the result for the KNN model as a network graph. I used the Python network graphing package called 'networkx' and used the spring layout, like I did in the previous section. The blue nodes indicate data points that come from Bejaia while the red nodes come from Sidi Bel-abbes. The network graphs look very different from the graphs from the first classification problem. The blue and red nodes are relatively well separated before dimension reduction, as shown in the figure on the left, but they are very well separated in the figure on the right, which was constructed after dimension reduction. This suggests that there are clear similarities and differences between the data observations that resulted in fire and not in fire. The five most important features returned by the Random Forest model during dimension reduction for the second classification problem were 'FFMC', 'DMC', 'DC', 'ISI', and 'FWI'. These are all of the fire weather index system variables, except 'BUI'. Therefore, from looking at the result from the KNN model, I would recommend that the government agencies of Algeria closely monitor 'FFMC', 'DMC', 'DC', 'ISI', and 'FWI' conditions to identify and control forest fires in a prompt response.

Figure 3. Network Graph of K Nearest Neighbors from the Training Set Before and After Dimension Reduction



In addition to the KNN model, I also visualized the result for the SVM model. Again, due to the high dimension of the dataset, I applied Principal Component Analysis (PCA) to project the training set into a 2-dimensional space. Figure 4 shows the training set plotted in the 2-D space with support vectors highlighted with greater opacity and black outlines. Similar to the observations made above, before dimension reduction, there are slight overlaps between the blue and red nodes, but after dimension reduction, the nodes are perfectly separated. The support vectors are exactly where the points of two colors meet each other, which again supports that there exists a clear linear relationship between data observations that witnessed fire and those that did not.

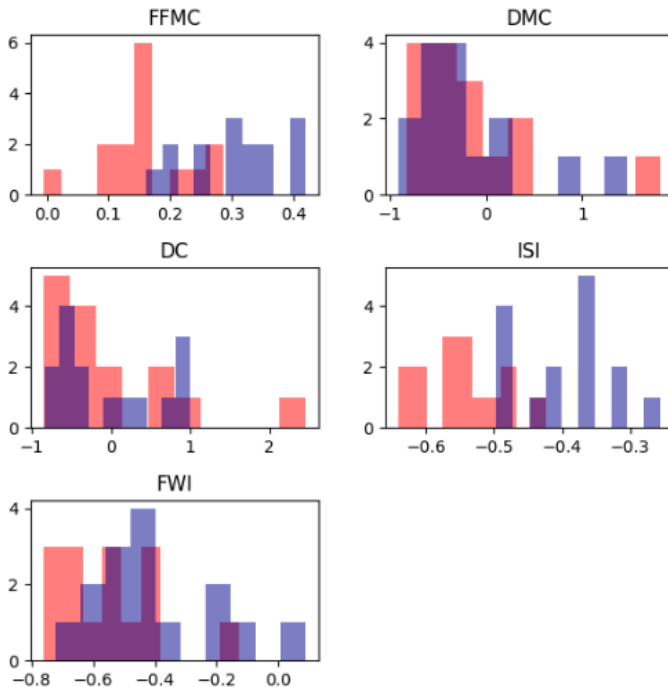
Figure 4. PCA Graph of the Training Set and Support Vectors Before and After Dimension Reduction



To further examine this relationship, I visualized the histogram of variables for the support vectors after dimension reduction in figure 5. The most clear relationship was seen in 'FFMC' and 'ISI' variables. Therefore, I would recommend the government agencies to closely

monitor the conditions of the five variables, 'FFMC', 'DMC', 'DC', 'ISI', and 'FWI', but especially 'FFMC' and 'ISI' since their differences were the most prominent.

Figure 5. Histogram of Variables for Support Vectors After Dimension Reduction



In conclusion, I would recommend the government agencies of Algeria in control of monitoring, preventing, and controlling forest fires to apply similar strategies to both regions, Bejaia and Sidi Bel-abbes and look out for changes in 'FFMC', 'DMC', 'DC', 'ISI', and 'FWI', but especially 'FFMC' and 'ISI' to predict forest fires in advance and take prompt responses when they occur.

3. Methods

3.1 Description of Features

As previously mentioned, there were a total of 15 features in the dataset. There was a 'region' variable that indicates whether data was comes from Bejaia or Sidi Bel-abbes, three date variables ('day', 'month', and 'year'), four weather variables ('Temperature', 'RH' (relative humidity), 'Ws' (wind speed), and 'Rain'), six variables from the fire weather index system ('FFMC' (fine fuel moisture code), 'DMC' (duff moisture code), 'DC' (drought code), 'ISI' (initial spread index), 'BUI' (buildup index), 'FWI' (fire weather index)), and lastly 'Classes' variable that indicates whether wildfire took place or not.

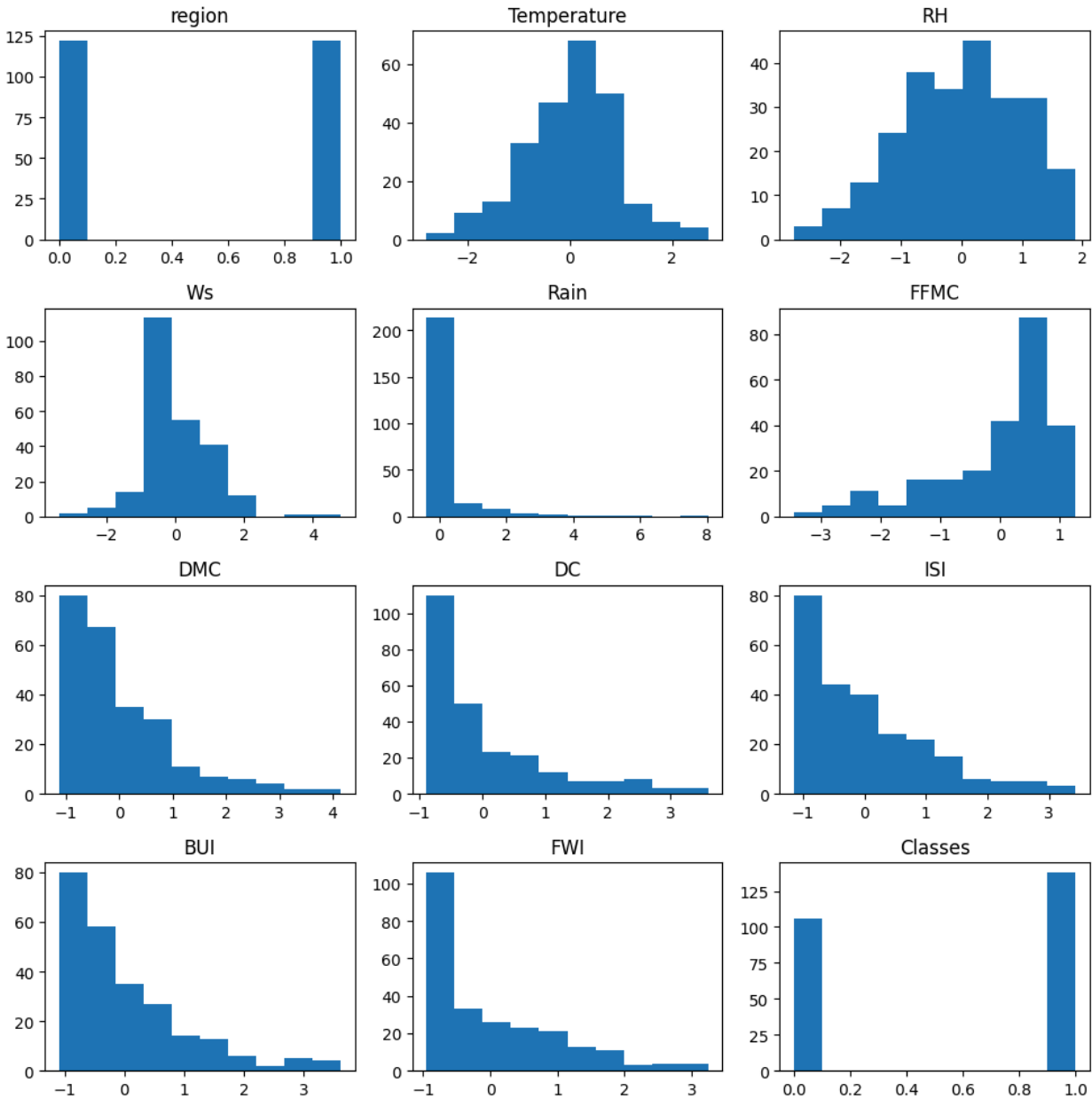
The fire weather index system quantifies the risk of a wildfire occurring. Three of the six variables from the fire weather index system ('FFMC', 'DMC', and 'DC') respectively measure

moisture contents of fine fuels, loosely compacted organic layers of moderate depth, and deep compact organic layers. Their numerical values increase as moisture content of each material decreases. The remaining three variables from the fire weather index system ('ISI', 'BUI', and 'FWI') provide a numerical score of the expected rate of fire spread, the total amount of fuel available for combustion on the landscape, and fire intensity.

3.2 Preprocessing

In constructing a dataset for both classification problems, I dropped the date features because the data was collected from a small time frame of summer (2012/6 ~ 2012/9). I also applied binary encoding to the two categorical variables ('region' and 'Classes') and applied z-score normalization to all numerical features. After the preprocessing steps, the distribution of the remaining variables are shown in figure 6.

Figure 6. Histogram of Remaining Variables after Preprocessing



Two of the variables are categorical: 'region' and 'Classes'. As mentioned previously, the 'region' variable indicates whether the sample is from Bejaia or Sidi-Bel Abbès and the 'Classes' variable indicates whether fire occurred or not. 'Temperature' is the only variable that appears to be normally distributed. 'RH' variable appears to be slightly left-skewed. 'FFMFC' variable appears to be left-skewed, while the rest of the fire weather index system variables (DMC, DC, ISI, BUI, FWI) appear right-skewed. 'Ws' (wind speed) variable appears to be short tailed. There are more instances recorded where fire did happen than the instances where fire did not happen. Due to z-score normalization, all numerical variables are centered at 0 and have standard deviation of 1.

The dataset for the first classification problem contains 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI', and 'FWI' with 'region' as the response variable. The dataset for the second classification problem contains 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI', and 'FWI' with 'Classes' as the response variable.

3.3 Hyperparameter Tuning

A. Description of Hyperparameters

KNN and SVM models have hyperparameters, which control how the models learn and make predictions. For the KNN model, I tuned the hyperparameter 'k' which determines how many nearest neighbors from the training dataset are considered when classifying a test point. For the SVM model, I tuned the kernel type, which defines how training data is transformed before classification.

B. Leave-One-Out Cross-Validation (LOOCV)

For hyperparameter tuning, I performed Leave-One-Out Cross-Validation (LOOCV). LOOCV identifies a single data point from the data as a test set while serving the rest as the training set and repeats until all training data points have been set aside as a test set once. LOOCV averages the model performance metric computed n times on the single test set, where n is the size of the data, for each hyperparameter value being tested. For example, for the KNN model, I tested values between 3 and 15 for the hyperparameter value 'k', while for the SVM model, I tested three kernels: 'linear', 'poly', 'rbf'. Out of those tested hyperparameter values, the one with the highest average accuracy score is chosen.

LOOCV is computationally heavy as it requires fitting and testing the model n times. However, it is also more reliable since more values are averaged together and every data point gets the chance to be in the test set at least once. LOOCV worked well in this case because the data was relatively small. There were only 244 instances in total, which kept the computation demands manageable.

C. Selected Hyperparameters

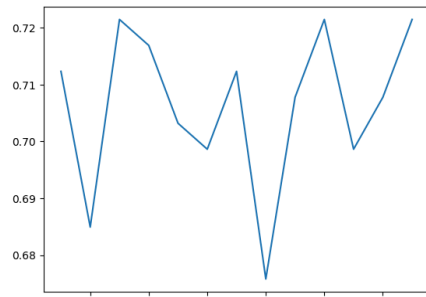
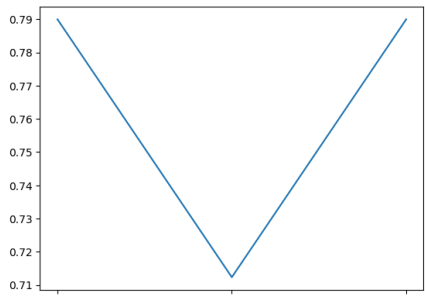
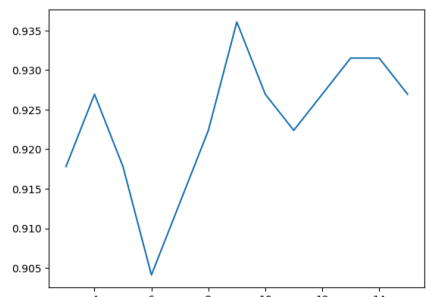
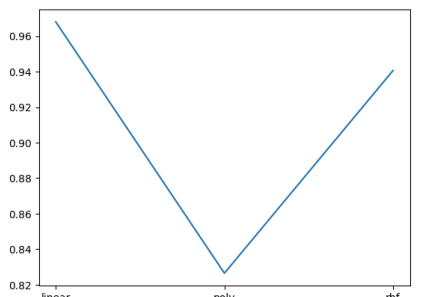
Table 5 provides the plots of average accuracy scores across tested hyperparameter values for all models. The selected hyperparameter values were those that corresponded to the highest average accuracy score across tested hyperparameter values from LOOCV, which are summarized in table 4. In sum, while the hyperparameter 'k' value for the KNN models changed depending on the classification problem and dimension reduction status, the 'linear' kernel was consistently chosen as the kernel type for all SVM models.

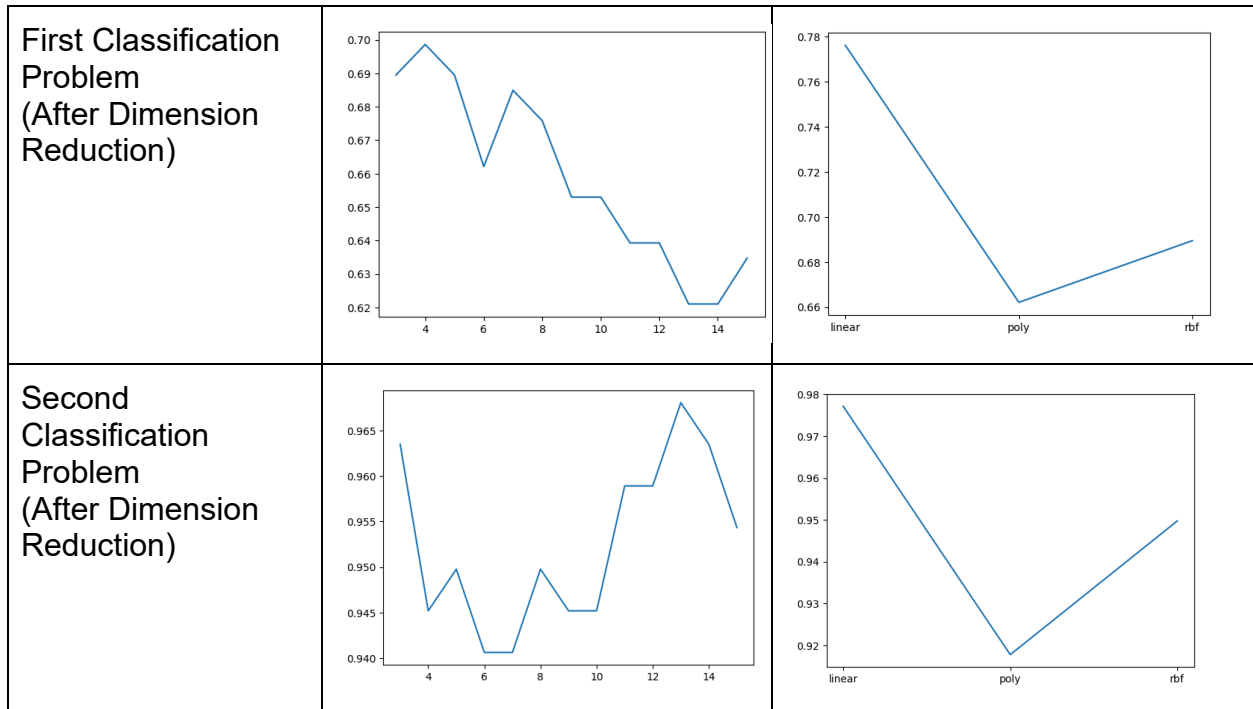
Table 4. Selected Hyperparameter Values for All Models

	KNN	SVM

First Classification Problem (Before Dimension Reduction)	k = 5	Kernel Type = 'linear'
Second Classification Problem (Before Dimension Reduction)	k = 9	Kernel Type = 'linear'
First Classification Problem (After Dimension Reduction)	k = 4	Kernel Type = 'linear'
Second Classification Problem (After Dimension Reduction)	k = 13	Kernel Type = 'linear'

Table 5. Plots of Average Accuracy Scores across Tested Hyperparameter Values for All Models

	KNN	SVM																																		
First Classification Problem (Before Dimension Reduction)	 <table><caption>KNN Accuracy (Before Dimension Reduction)</caption><thead><tr><th>k</th><th>Accuracy</th></tr></thead><tbody><tr><td>4</td><td>0.712</td></tr><tr><td>5</td><td>0.685</td></tr><tr><td>6</td><td>0.722</td></tr><tr><td>7</td><td>0.715</td></tr><tr><td>8</td><td>0.702</td></tr><tr><td>9</td><td>0.712</td></tr><tr><td>10</td><td>0.675</td></tr><tr><td>11</td><td>0.708</td></tr><tr><td>12</td><td>0.722</td></tr><tr><td>13</td><td>0.698</td></tr><tr><td>14</td><td>0.710</td></tr><tr><td>15</td><td>0.722</td></tr></tbody></table>	k	Accuracy	4	0.712	5	0.685	6	0.722	7	0.715	8	0.702	9	0.712	10	0.675	11	0.708	12	0.722	13	0.698	14	0.710	15	0.722	 <table><caption>SVM Accuracy (Before Dimension Reduction)</caption><thead><tr><th>Kernel</th><th>Accuracy</th></tr></thead><tbody><tr><td>linear</td><td>0.788</td></tr><tr><td>poly</td><td>0.712</td></tr><tr><td>rbf</td><td>0.788</td></tr></tbody></table>	Kernel	Accuracy	linear	0.788	poly	0.712	rbf	0.788
k	Accuracy																																			
4	0.712																																			
5	0.685																																			
6	0.722																																			
7	0.715																																			
8	0.702																																			
9	0.712																																			
10	0.675																																			
11	0.708																																			
12	0.722																																			
13	0.698																																			
14	0.710																																			
15	0.722																																			
Kernel	Accuracy																																			
linear	0.788																																			
poly	0.712																																			
rbf	0.788																																			
Second Classification Problem (Before Dimension Reduction)	 <table><caption>KNN Accuracy (After Dimension Reduction)</caption><thead><tr><th>k</th><th>Accuracy</th></tr></thead><tbody><tr><td>4</td><td>0.918</td></tr><tr><td>5</td><td>0.927</td></tr><tr><td>6</td><td>0.905</td></tr><tr><td>7</td><td>0.918</td></tr><tr><td>8</td><td>0.925</td></tr><tr><td>9</td><td>0.935</td></tr><tr><td>10</td><td>0.928</td></tr><tr><td>11</td><td>0.922</td></tr><tr><td>12</td><td>0.928</td></tr><tr><td>13</td><td>0.932</td></tr><tr><td>14</td><td>0.932</td></tr><tr><td>15</td><td>0.928</td></tr></tbody></table>	k	Accuracy	4	0.918	5	0.927	6	0.905	7	0.918	8	0.925	9	0.935	10	0.928	11	0.922	12	0.928	13	0.932	14	0.932	15	0.928	 <table><caption>SVM Accuracy (After Dimension Reduction)</caption><thead><tr><th>Kernel</th><th>Accuracy</th></tr></thead><tbody><tr><td>linear</td><td>0.962</td></tr><tr><td>poly</td><td>0.828</td></tr><tr><td>rbf</td><td>0.940</td></tr></tbody></table>	Kernel	Accuracy	linear	0.962	poly	0.828	rbf	0.940
k	Accuracy																																			
4	0.918																																			
5	0.927																																			
6	0.905																																			
7	0.918																																			
8	0.925																																			
9	0.935																																			
10	0.928																																			
11	0.922																																			
12	0.928																																			
13	0.932																																			
14	0.932																																			
15	0.928																																			
Kernel	Accuracy																																			
linear	0.962																																			
poly	0.828																																			
rbf	0.940																																			



3.4 Dimension Reduction

I trained a Random Forest model to reduce the dimension of the dataset to half. Random Forest is an ensemble learning technique for decision trees, where multiple decision trees are fitted on bootstrapped samples of points using a random subset of features. I chose the Random Forest model because it produces feature importance scores by averaging the decrease in gini impurity score for splits using each feature across the trees. It also inherently reduces overfitting, is more intuitive than approaches like Principal Component Analysis (PCA) which selects principal components instead of individual features, and handles non-linear relationships.

For the first classification problem, the five most important features returned by the Random Forest model were 'RH', 'Rain', 'DMC', 'ISI', and 'BUI'. For the second classification problem, the five most important features returned by the Random Forest model were 'FFMC', 'DMC', 'DC', 'ISI', and 'FWI'.

3.5 Order of Procedures

With the addition of hyperparameter tuning and dimension reduction steps, the exact order of procedures I took may be confusing. In this section, I go over the order of steps I took to produce my results.

1. Data exploration and preprocessing (Section 3.1 and 3.2)
2. Train-Test Split

With the constructed datasets for each problem from the previous step, I split each dataset into training and testing sets. I randomly assigned 10% of each dataset to be the testing set.

3. Hyperparameter tuning (Section 3.3)

I performed LOOCV on the training set for each classification problem. I then selected hyperparameter values for each model, following the steps outlined in section 3.3.

4. Model Runtime and Performance (Section 2.2 and 2.3)

I trained each model with the selected hyperparameter values. I computed the accuracy score given the model predictions and the true values of the training set. Then, I got model predictions on the test set. Model performance was measured using the accuracy score between the model prediction and the true values of the test set. The runtime of training measured the time it took to fit the model. The runtime of testing measured the time it took for the model to make predictions on the test set.

5. Dimension Reduction (Section 3.4)

Random Forest model was trained on the dataset for each problem. The five most important features were selected from each dataset. Using the reduced dataset, steps 2 through 4 were repeated—train-test split, hyperparameter tuning, model runtime and performance.