Wildfire Detection using Machine Learning

NAME- RISHI KUMAR
CWID- 20015656

Outline

Outline

What is wildfire?

- A wildfire is an uncontrolled fire that burns in the wildland vegetation, often in rural areas.
- Wildfires can burn in forests, grasslands, savannas, and other ecosystems, and have been doing so for hundreds of millions of years
- Wildfires are among the most common forms of natural disaster in some regions, including Siberia, California, and Australia. Areas with Mediterranean climates are particularly susceptible.

How wildfire is affecting California

- In the united regions, wildfires have become a major problem, especially in states like California, where they frequently result in fatalities and significant financial losses.
- As of 21 September 2022, 6,473 fires totaling 365,140 acres (147,770 hectares) had been reported across the state during the wildfire season, with 9 fatalities so far in California.
- To reduce this threat, predicting such an environmental problem becomes essential. Wildfires are caused by a number of variables, including climate change and human activity.
- With the aid of past weather and wildfire data, we can forecast the likelihood of fires to assist the California Department of Forestry and Fire Protection.

Data Sources

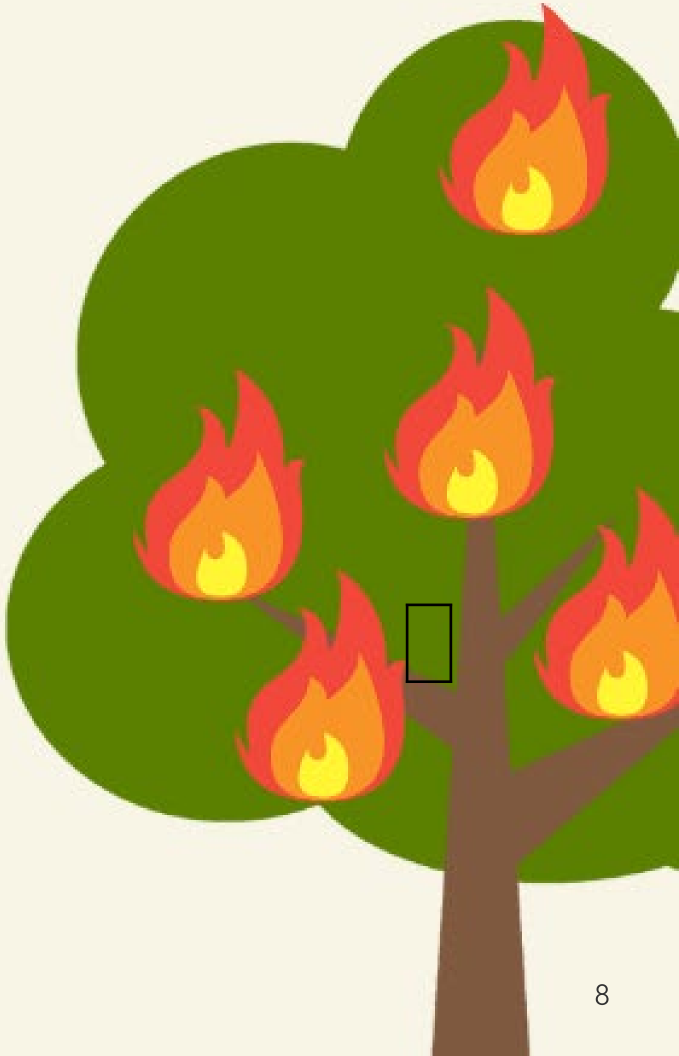
- Geographic and wildfire data from the State of the California
    I. 4,279 fires from July 1, 2008 to December 31, 2020
- Weather data from World Weather Online
    I. Historical weather records from every day in the same range
    II. Aggregated to a monthly basis

About the data set

- **The Dataset contains:**
  a. **10,988 records**
  b. **4,297 of which had fires**
  c. **Additional features such as :**
     - **Monthly weather averages**
     - **Quarterly weather averages**
     - **Quarterly cumulative precipitation**
     - **Acres burned**
     - **Cause of fire**
     - **Geo Location**

Feature Description

Sr.no	Feature	Data type	description
1	date	*object*	The month and year of when the fire took place.
2	county	*object*	The county the fire started in.
3	maxtempF	*float*	The average maximum temperature of that month in °F.
4	mintempF	*float*	The average min temperature of that month in °F.
5	avgtempF	*float*	The average temperature of that month in °F.
6	totalSnow	*float*	The total snow for that month.
7	humid	*float*	The average humidity for that month.
8	wind	*float*	The average wind for that month.
9	precip	*float*	The average precipitation for that month.



Feature Description

Sr.no	Feature	Data type	description
10	q_avgtempF	float	The quarterly average temperature in °F.
11	q_avghumid	float	The quarterly average humidity.
12	q_sumprecip	float	The quarterly average precipitation.
13	sunHour	float	The average hours of sun for that month.
14	FIRE_NAME	object	The name of the fire.
15	CAUSE	float	The cause of the fire.
16	lat	float	The latitude coordinate of the fire's location.
17	long	float	The longitude coordinate of the fire's location.
18	GIS_ACRES	float	The total number of acres burned.

Implementation plan

- First scan the data to identify missing or irregular data as well as the distribution of each feature.
- After screening, data analysis and correlation plotting
- After that, the creation of the training and testing dataset
- After screening, data must be cleaned to create a trainable set. In addition to creating data, we will also create algorithms (SVM, Random forest, decision tree).
- Once we have the model and the data, we will train the model and evaluate its performance on the test set using the metrics of recall, precision, and F1-score

Checking for missing data

- ❖ **As we can see no null values were found in the data set**
- ❖ **If any null value were to be found the non null count would be less than 10988 for that particular feature**

```
In [3]: df.info()

RangeIndex: 10988 entries, 0 to 10987
Data columns (total 18 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   date          10988 non-null  object
 1   county        10988 non-null  object
 2   maxtempF      10988 non-null  float64
 3   mintempF      10988 non-null  float64
 4   avgtempF      10988 non-null  float64
 5   totalSnow     10988 non-null  float64
 6   humid         10988 non-null  float64
 7   wind          10988 non-null  float64
 8   precip        10988 non-null  float64
 9   q_avgtempF    10988 non-null  float64
 10  q_avghumid    10988 non-null  float64
 11  q_sumprecip   10988 non-null  float64
 12  sunHour       10988 non-null  float64
 13  FIRE_NAME     10988 non-null  object
 14  CAUSE         10988 non-null  float64
 15  lat           10988 non-null  float64
 16  long          10988 non-null  float64
 17  GIS_ACRES     10988 non-null  float64
dtypes: float64(15), object(3)
memory usage: 1.5+ MB
```

Data analysis- 1

❖ **To have the basic understanding of independent features which will be used for training the machine learning models**

In [4]:

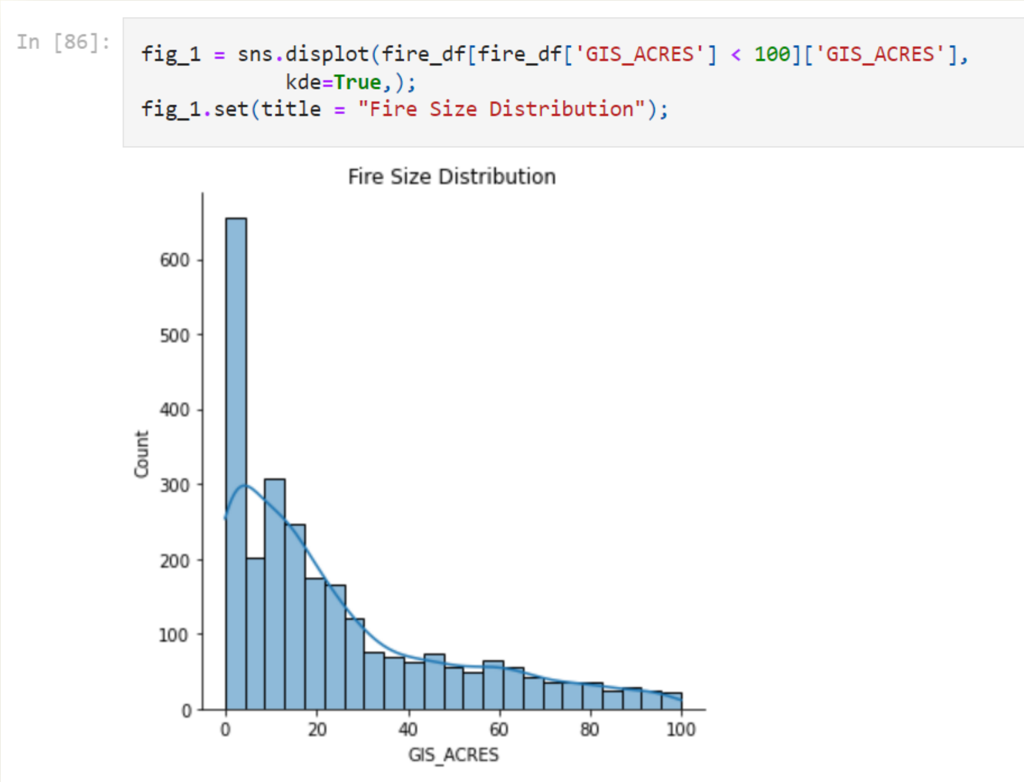
```
df.describe()
```

Out[4]:

	maxtempF	mintempF	avgtempF	totalSnow	humid	wind	precip
count	10988.000000	10988.000000	10988.000000	10988.000000	10988.000000	10988.000000	10988.000000
mean	72.789618	49.036346	64.676692	0.087918	54.408352	5.583294	0.072370
std	15.735935	11.425170	14.635490	0.420449	16.926551	1.514516	0.133537
min	26.214286	0.642857	19.483871	0.000000	10.466667	2.354839	0.000000
25%	61.056452	42.123560	53.870968	0.000000	41.165323	4.533333	0.003226
50%	72.951075	49.633333	65.290323	0.000000	54.098387	5.354839	0.020000
75%	85.032258	56.677419	75.900806	0.000000	67.903226	6.354839	0.080645
max	110.935484	88.935484	102.612903	9.229108	95.935484	14.129032	1.748387

Data analysis- 2

- Plotting number of fires vs number of acres burned
- As we can see, most fires that occurred in our data burned less than 100 acres



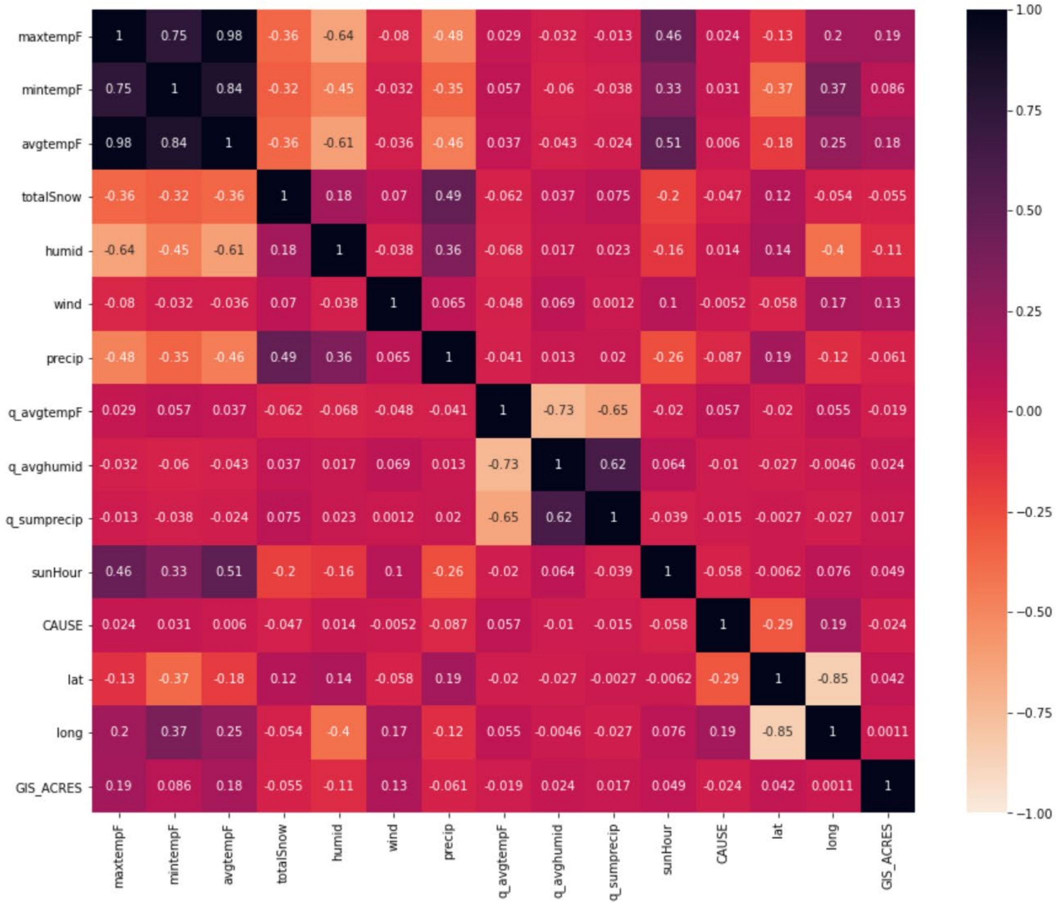
Data analysis-3

- Plotting correlation heatmap using seaborn
- We can see that there are no real strong correlations or patterns between variables that are independent from each other
- There are some strong correlations in the heatmaps, but that is because the variables are depended (i.E. The total amount of snow and the precipitation levels)
- Heatmap on the next slide

Correlation heatmap

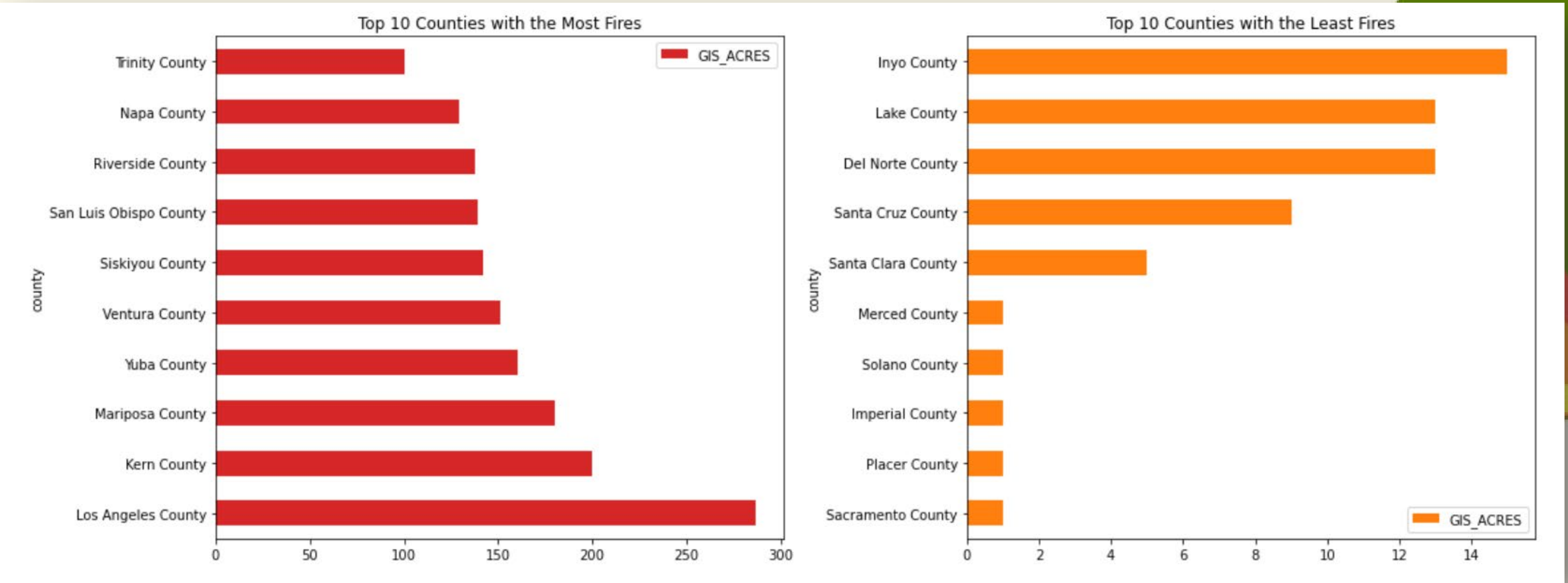
In [74]:

```python
plt.figure(figsize=(15, 12))
sns.heatmap(fire_df.corr(), vmin=-1, vmax=1, annot=True, cmap='rocket_r');
```



Data analysis-4

- **Plotting counties with most and least fires in the past using matplotlib**
- **As we can see Los Angeles County has the most number of fires while Sacramento County has the least. This might be due to Los Angeles being the largest city in California. Because of that there are more people which means there is a higher chance fire will occur**



Machine Learning Algorithms Used

- Now we will split the data set into training set and testing set and will create functions for Machine Learning algorithms
- The following Machine Learning algorithms:

   a. Decision Tree

   b. Multilayer Perceptron

   c. Support Vector Machine

Decision Tree

- **Decision tree :**
  - **a.** **Hierarchical structure that used to classify classes based on a set of rules.**
  - **b.** **Decision tree is also efficient and requires less effort in preprocessing data**
  - **c.** **One of the disadvantages of this method is that it tends to be over-fitting**
- **Random forest:**
  - **a.** **Random forest is an ensemble classifier, that works on the basis of a decision tree algorithm.**
  - **b.** **Based on the concept of randomization, random forest creates a collection of independent and non-identical decision trees. The class with the highest votes determines our model's prediction from each**

Decision Tree Results

IMPLEMENTATION:

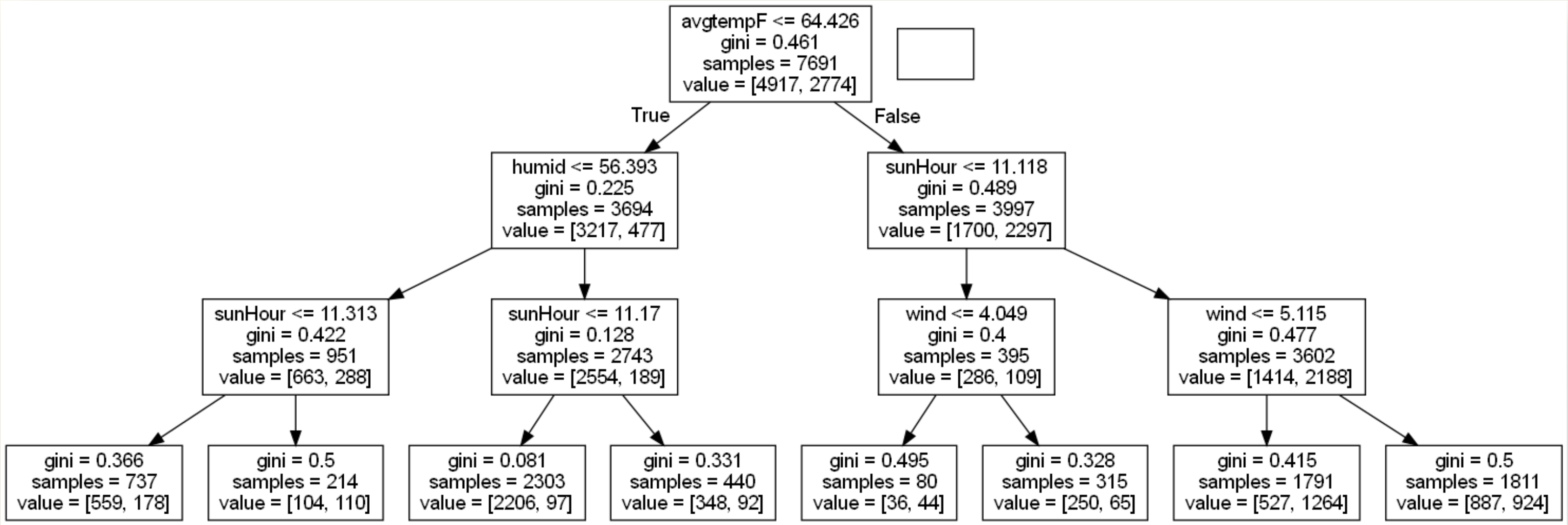
```
# Decision tree classifier function  and classification report and acuracy
print(X_test_sc, y_test)
dec_tree = tree.DecisionTreeClassifier()
trained  = dec_tree.fit(X_train_sc,y_train)
y_pred = dec_tree.predict(X_test_sc)

print("Classification Report: \n\n",classification_report(y_test, y_pred))

print("Accuracy: ",accuracy_score(y_test, y_pred))
```

RESULT:

```
Classification Report:

              precision    recall  f1-score   support

           0       0.90      0.84      0.87      2098
           1       0.75      0.83      0.79      1199

    accuracy                           0.84      3297
   macro avg       0.82      0.83      0.83      3297
weighted avg       0.84      0.84      0.84      3297

Accuracy:  0.8362147406733395
```

Pruned Decision Tree

Pruned decision tree to identify the most effective parameters



Random Forest Results

IMPLEMENTATION:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 50,max_leaf_nodes=10,n_jobs=-1)
clf.fit(X_train_sc, y_train)
rf_pred=clf.predict(X_test_sc)
conf_matrix = confusion_matrix(y_test, rf_pred)
TN = conf_matrix[0][0]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
FP = conf_matrix[0][1]
sensitivity_test = TP/(TP+FN)
specificity_test = TN/(FP+TN)
print("Accuracy: ",accuracy_score(y_test, rf_pred))
print("clasification report: \n ",classification_report(y_test,rf_pred))
print("percent wildfires correctly predicted (on testing set) : {0:.2f}%\n".format(sensitivity_test*100))
print("percent non-wildfires correctly predicted (on testing set) : {0:.2f}%\n".format(specificity_test*100))
```

RESULT:

```
Accuracy:  0.778586593873218
clasification report:
               precision    recall  f1-score   support

           0       0.84      0.81      0.82      2098
           1       0.68      0.73      0.71      1199

    accuracy                           0.78      3297
   macro avg       0.76      0.77      0.76      3297
weighted avg       0.78      0.78      0.78      3297
```

Multi Layer Perceptron

- The regression Multi-layer Perceptron was chosen due to the binary nature of the output that was being predicted whether there was a fire at a location within the geographic bounds of California
- An input layer, hidden layers, and an output layer make up the MLP.
- With the MLP, three distinct activation functions—Sigmoid, Relu, and Hyperbolic Tangent—were available for training the model within the hidden layers.

MLP Classifier 2 Hidden Layers Results

IMPLEMENTATION:

```
#STEP 2 NN with 2 Layers
classifier = MLPClassifier(solver='adam',hidden_layer_sizes=(10,15),activation='logistic',learning_rate='constant',
                           learning_rate_init=0.12,alpha=0.00000003, momentum=0.54)
classifier = classifier.fit(X_train_sc, y_train)
pred_2 = classifier.predict(X_test_sc)


#evaluating the algorithm
from sklearn.metrics import classification_report, confusion_matrix
print ("accuracy score: ",accuracy_score(y_test, pred_2))
print("clasification report: \n ",classification_report(y_test,pred_2))
print("confusion matrix: \n ",confusion_matrix(y_test,pred_2))
```

RESULT:

```
accuracy score:  0.7743403093721565
clasification report:
                precision    recall  f1-score   support

           0       0.82      0.83      0.82      2098
           1       0.70      0.67      0.68      1199

    accuracy                           0.77      3297
   macro avg       0.76      0.75      0.75      3297
weighted avg       0.77      0.77      0.77      3297
```

MLP Classifier 4 Hidden Layers Results

IMPLEMENTATION:

```python
#STEP 4 NN with 4 Layers
classifier = MLPClassifier(solver='adam',hidden_layer_sizes=(10,15,15,10),activation='logistic',
                           learning_rate='constant',learning_rate_init=0.12,alpha=0.00000003, momentum=0.54)
classifier = classifier.fit(X_train_sc, y_train)
pred_4 = classifier.predict(X_test_sc)


#evaluating the algorithm
from sklearn.metrics import classification_report, confusion_matrix
print ("accuracy score: ",accuracy_score(y_test, pred_4))
print("clasification report: \n ",classification_report(y_test,pred_4))
print("confusion matrix: \n ",confusion_matrix(y_test,pred_4))
```

RESULT:

```
accuracy score:  0.7609948437973916
clasification report:
                precision    recall  f1-score   support

           0       0.78      0.87      0.82      2098
           1       0.72      0.57      0.63      1199

    accuracy                           0.76      3297
   macro avg       0.75      0.72      0.73      3297
weighted avg       0.76      0.76      0.75      3297
```

MLP Classifier 5 Hidden Layers Results

IMPLEMENTATION:

```
#STEP 3 NN with 5 Layers
classifier = MLPClassifier(solver='adam',hidden_layer_sizes=(10,15,15,15,10),activation='logistic',
                           learning_rate='constant',learning_rate_init=0.12,alpha=0.00000003, momentum=0.54)
classifier = classifier.fit(X_train_sc, y_train)
pred_5 = classifier.predict(X_test_sc)


#evaluating the algorithm
from sklearn.metrics import classification_report, confusion_matrix
print ("accuracy score: ",accuracy_score(y_test, pred_5))
print("clasification report: \n ",classification_report(y_test,pred_5))
print("confusion matrix: \n ",confusion_matrix(y_test,pred_5))
```

RESULT:

```
accuracy score:  0.7555353351531695
clasification report:
                precision    recall  f1-score   support

           0       0.78      0.86      0.82      2098
           1       0.70      0.58      0.63      1199

    accuracy                           0.76      3297
   macro avg       0.74      0.72      0.72      3297
weighted avg       0.75      0.76      0.75      3297
```

Support Vector Machine

- A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems
- SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.
- A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane
- Following this, characteristics of new data can be used to predict the occurrence of wildfire

Support Vector Machine Results

IMPLEMENTATION:

```python
from sklearn import svm
#classifier = svm.SVC(kernel='linear', gamma='auto', C=2)
classifier = svm.SVC(C=100.0, kernel='poly', degree=3, gamma='auto', coef0=0.0, shrinking=True,
                     probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=0,
                     max_iter=-1, decision_function_shape="ovr", random_state = 0)
classifier.fit(X_train_sc,y_train)
Y_predict_svm = classifier.predict(X_test_sc)
```

```python
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_predict_svm))
print ("accuracy score: ",accuracy_score(y_test, Y_predict_svm))
```

RESULT:

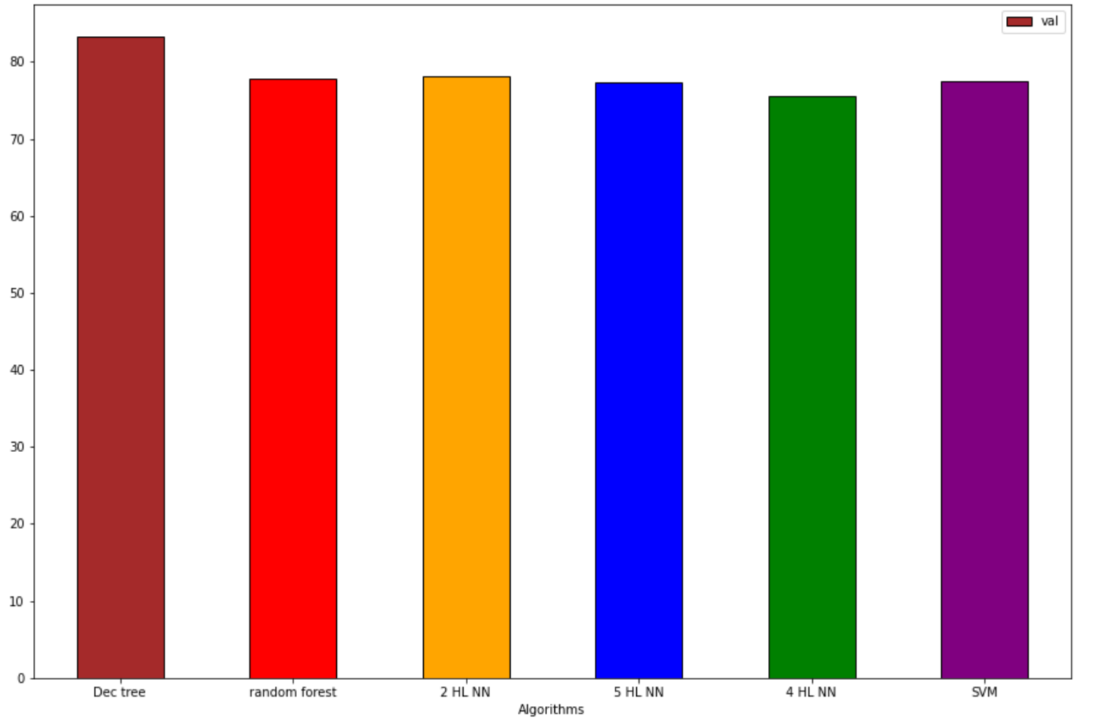
```
              precision    recall  f1-score   support

           0       0.85      0.74      0.79      2098
           1       0.63      0.77      0.69      1199

    accuracy                           0.75      3297
   macro avg       0.74      0.76      0.74      3297
weighted avg       0.77      0.75      0.76      3297

accuracy score:  0.7521989687594783
```

Result Comparison

Accuracy Comparison Dataframe:

	Parameter	Dec tree	Random forest	2 HL NN	5 HL NN	4 HL NN	SVM
0	accuracy score	0.832575	0.778587	0.781013	0.77343	0.755839	0.774947

Accuracy Comparison Bar Graph:



Conclusion

- ❖ **Decision tree came out on top with 83% percent accuracy score**
- ❖ **Random forest has average results due to nonlinear correlation present between features**
- ❖ **Multilayer perceptron and SVM both gave average results due to overfitting caused by high number of features present in the dataset**

References

- "California Department of Forestry and Fire Protection", [online] Available: https://www.fire.ca.gov/stats-events/.
- Ashima Malik et al. "Wildfire Risk Prediction and Detection using Machine Learning in San Diego, California". In: Oct. 2021. Doi: 10.1109/SWC50871.2021.00092.
- https://www.kaggle.com/datasets/ananthu017/california-wildfire-incidents-20132020
- M. E. Alsahaft, M. A. Alharthi and M. Arif, "Role of Machine Learning Algorithms in Forest Fire Management: A Literature Review", J. Robot. Autom., vol. 5, pp. 212-226, February 2018.
- S. Youssef and B. Abdelaziz, "Prediction of Forest Fires Using Artificial Neural Networks", Appl. Math. Sci., vol. 7, pp. 271-286, January 2013