**Machine Learning Assignment Spring 2024**

**Project Topic**

Forest Fire Prediction

**Objective**

To develop predictive models that will identify whether a forest fire will happen.

1. **Overview**

The project aims to create predictive models to determine whether a forest fire will happen. This involves using machine learning methods to examine factors like environmental conditions, air humidity, tree density, and time of day. By constructing accurate predictive models, the project seeks to aid in early detection and mitigation strategies, with the overarching goal of lessening the impact of forest fires on both ecosystems and communities.

1. **Methodology**

There are different steps taken into consideration to carry out this project and the steps are listed below.

**Data collection**

Forest fire dataset downloaded containing features such as humidity, tree density, time of day, and the target (‘fire’) variable indicating fire occurrence.

**Data Exploration and Preprocessing (Data Cleaning)**

**Data Exploration:** Reading the dataset (.csv) into panda pd then exploring the dataset size, shape, and column index. The dataset is about 47kb in size and there are 456 rows and 12 columns in the dataset which require data cleansing.

A screenshot of a computer

Description automatically generated

**Preprocessing:** In this process below are considered.

**Missing value** was identified and handled by checking the entire dataset for null value or empty cell, to identify and eliminate this missing value isnull function was applied.

‘missing\_values = ft\_data.isnull().sum()

ft\_data.fillna(ft\_data.mean(), inplace=True)’

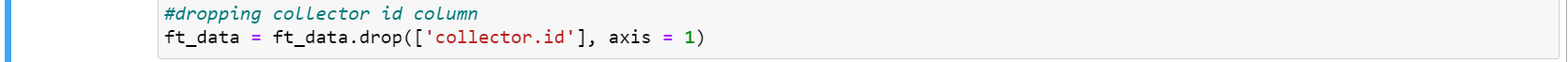
*Snapshot below shows all outputs both the period of null value detection and elimination.*

A screenshot of a computer

Description automatically generated

As part of data cleaning irrelevance column will be dropped.

Here collector id column will be dropped, see python code used below.



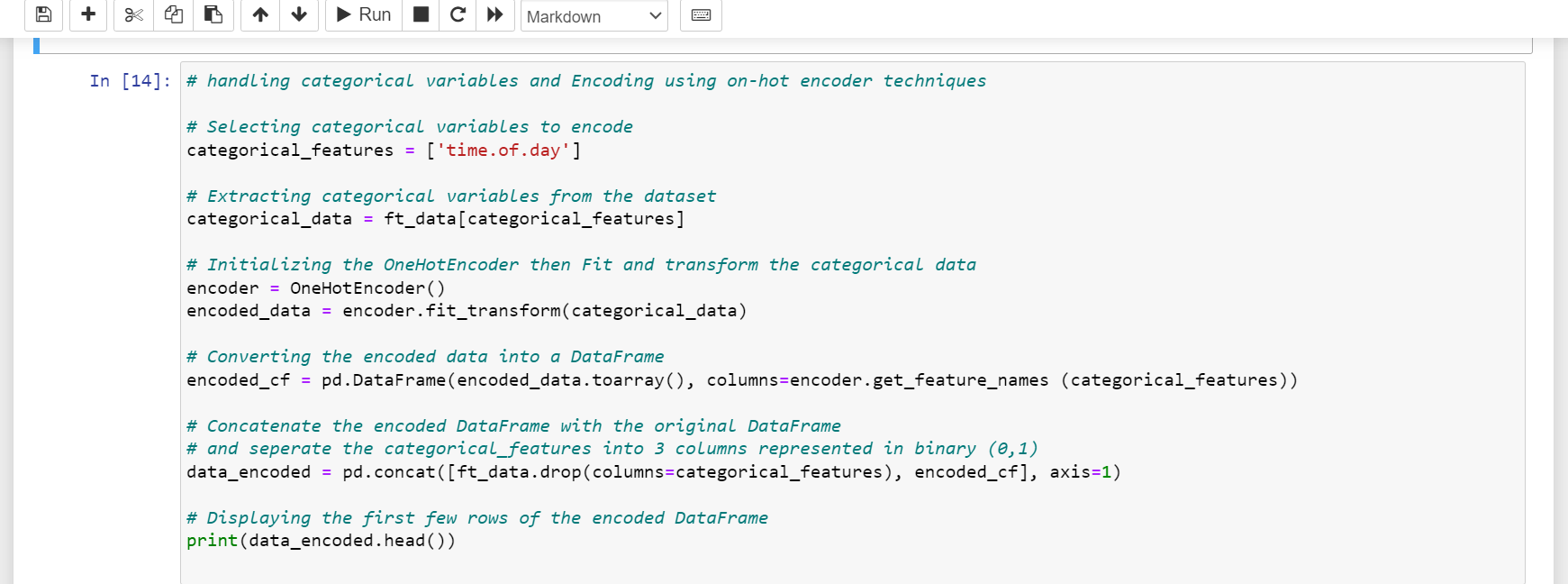
**Note:** From the dataset we are not finding if the forest fire happens or not, we are trying to predict whether forest fire happening and there is no meaningful relationship between the ID and forest fire likelihood.

**Identification categorical variables and handled by Encoding.**

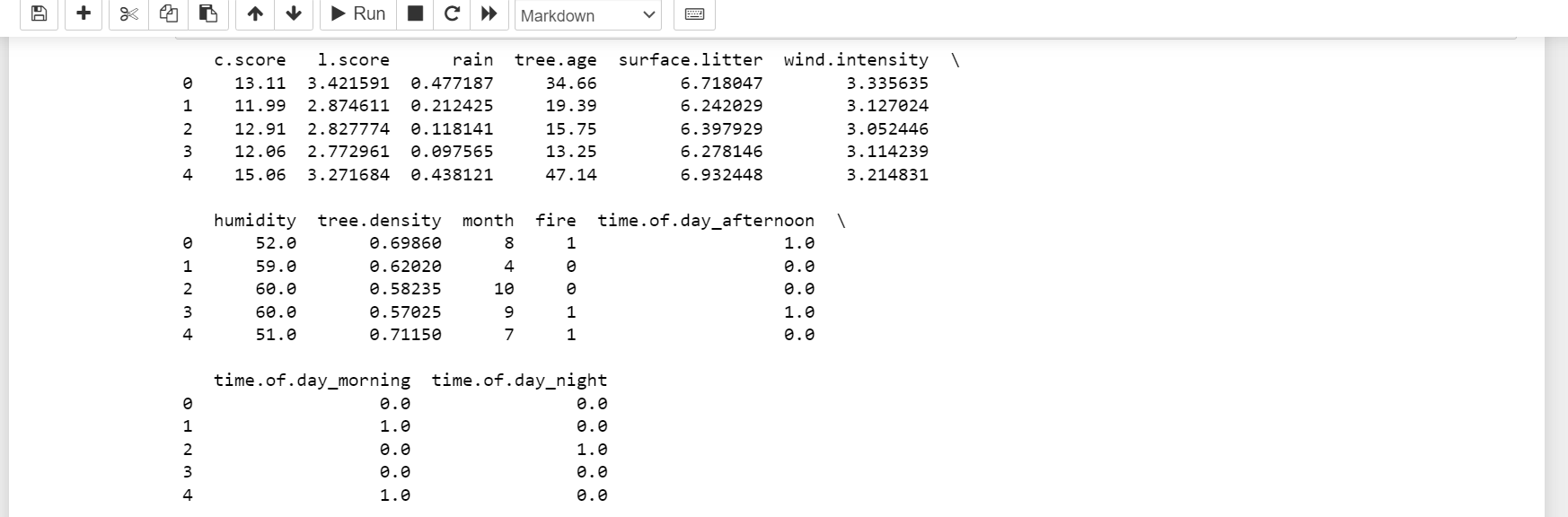
there is categorical variable in the dataset which was identify as “time of day”. So, this was encoded by using one-hot encoding techniques.

After encoding, the “time of day” within the categorical feature will be represented by a binary column (0 or 1), effectively transforming categorical variables into numerical format suitable for machine learning algorithms.

*Snapshot below review the python code and output printed on how the categorical variables was identified and handled using encoding techniques (one-hot encoder).*



Final Output



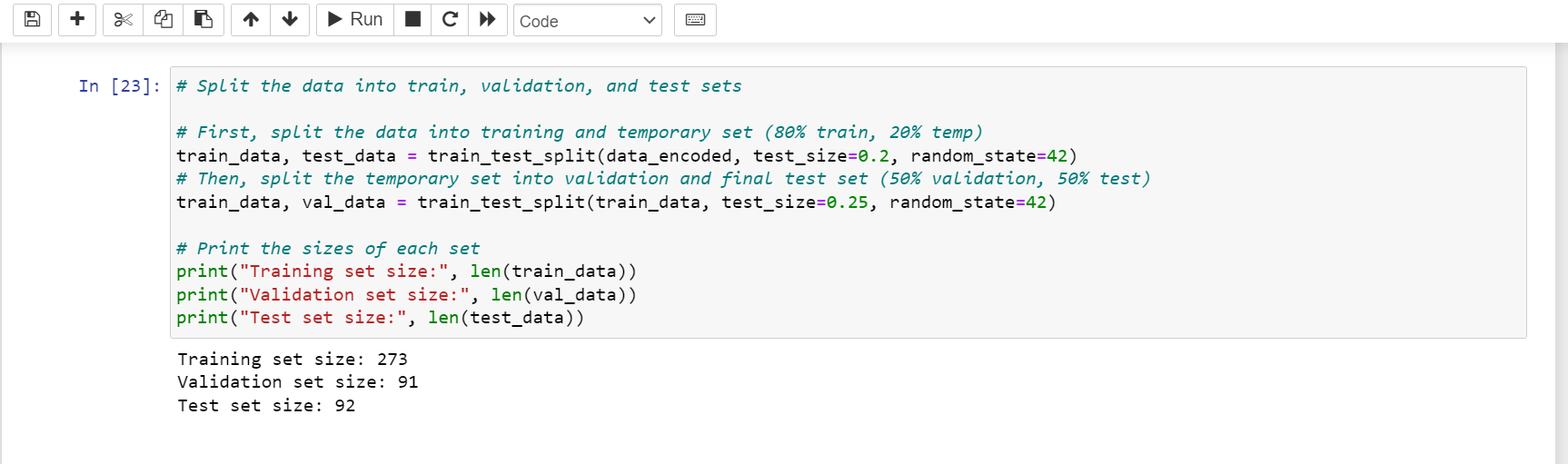
**Splitting the Dataset into Train, Val, and Test**

The concept of splitting the data is to divide the dataset into training, validation, and test sets to prepare for model building and evaluation. However, the basic reason for chosen training, validation and test split is essential for developing and evaluating predictive models effectively.

**Note**

The training set acquired the largest portion of the forest fire dataset. Training set to 80% and the 20% temporary set into validation and final test which both set acquired 10%, 10% i.e shared the 20% equally (50%-50%). All sets have their various importance in this model, this will be explained and review in the project.

*View snapshots below for python code used and final output of their sizes after split.*



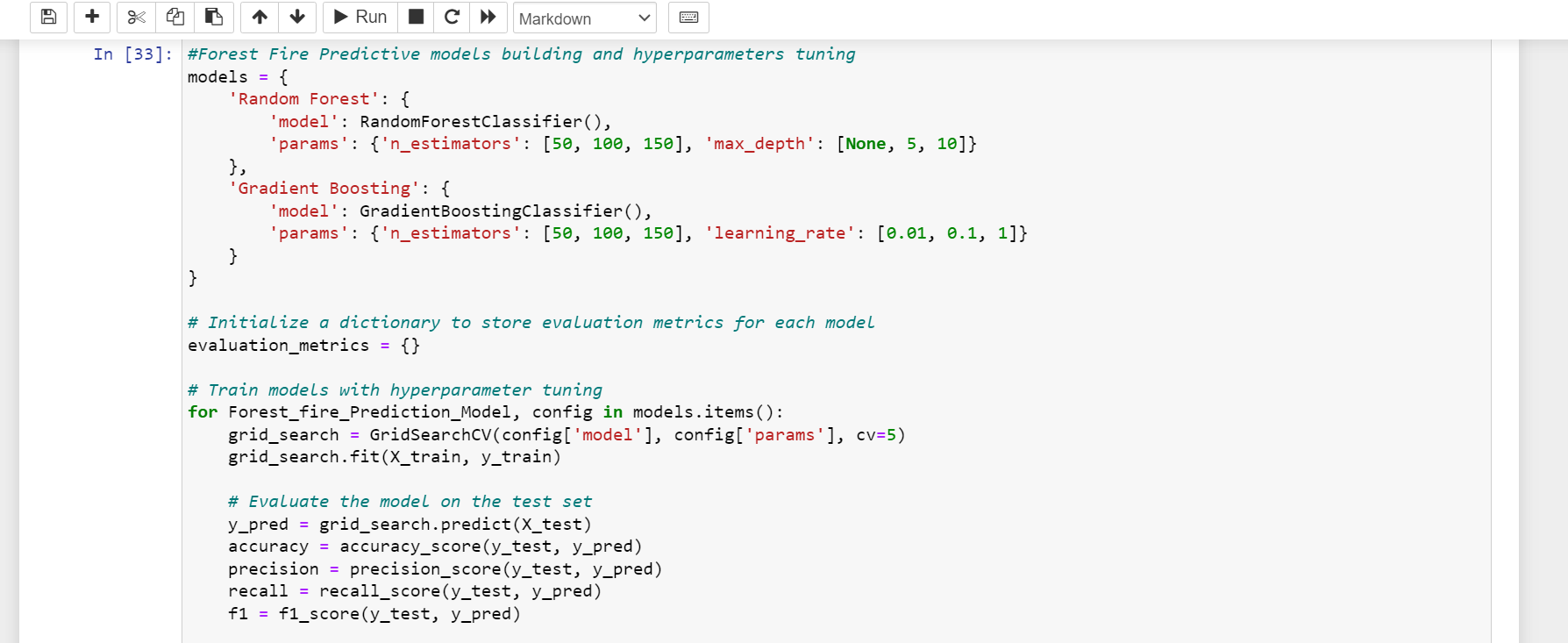
**Model Building**

In the process of model building “Random Forest” and “Gradient Boosting” were the appropriate machine learning models used for the task classification tasks.

Model train by using the training set and hyperparameters tuning by using techniques the Grid Search and Random Search.

Then, further evaluate each model's performance using the validation set and select the optimal performing model based on evaluation metrics.

See below python code used.



**Model Evaluation**

The selected final model was trained again using both the training and validation sets to make the most of the available data.

Next, the performance of the final model was evaluated using the test set to acquire an impartial estimate of its performance on unseen data.

The model's performance was assessed using a range of evaluation metrics including accuracy, precision, recall, and F1-score.

Subsequently, the results were analyzed and interpreted to gain insights into the strengths and weaknesses of the model.

*View the python code used and the result printed on python.*

A screenshot of a computer

Description automatically generated

**Conclusion and Reference**

At this stage the model’s final justification will be concluded and share necessary references while modelling.

1. **Data Overview**

Below is a tabulated overview showcasing the variables intended for model utilization, along with their designated types. Additionally, discussing the consequences of these choices on model performance.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Description | Consequence |
| c.score | Numeric | A score measuring the relative  carbohydrate makeup of the tree species  present | Considering c.score as a numeric variable enables the model to encompass the continuous variations in carbohydrate makeup, potentially enhancing prediction accuracy. Nonetheless, this treatment presupposes a linear relationship, which may not always be accurate. |
| l.score | Numeric | A score measuring the relative mass of  wood to leaves in the tree species present | Treating l.score as numeric permits the model to grasp the quantitative association between wood mass and leaves, potentially impacting fire likelihood. Nonetheless, this approach assumes a linear relationship, which may not consistently hold true. |
| Rain | Numeric | The amount of rain over the previous three  days, normalized. | Regarding rain as numeric enables the model to account for the quantitative influence of rainfall on fire likelihood. Nevertheless, this treatment assumes a linear relationship, which may not always be applicable. |
| tree age | Numeric | Average age of trees in the forest, normalized. | Treating tree age as numeric empowers the model to capture the quantitative correlation between tree age and fire likelihood. However, this methodology assumes a linear relationship, which may not invariably be accurate. |
| surface litter | Numeric | An index for the amount of litter/trash  found in the forest (potential sources of  ignition). | Considering surface litter as numeric empowers the model to comprehend the quantitative impact of litter accumulation on fire likelihood. Nonetheless, this treatment assumes a linear relationship, which may not always be valid. |
| wind intensity | Numeric | A measure of average wind speeds over the  past 24 hours | Regarding wind intensity as numeric enables the model to assess the magnitude of wind effects on fire likelihood. However, this treatment presupposes a linear relationship, which may not consistently hold true. |
| Humidity | Numeric | Air humidity at time of measurement. | Treating humidity as numeric enables the model to embrace the continuous nature of humidity levels, potentially enhancing its ability to generalize across different humidity ranges. Nonetheless, this approach assumes a linear relationship, which may not always be accurate. |
| tree density | Numeric | Number of trees per square meter, normalized. | Considering tree density as numeric facilitates the model's understanding of the quantitative relationship between tree density and fire likelihood. Nevertheless, this treatment assumes a linear relationship, which may not always hold true. |
| Month | Categorical | Month in the year (1=January, 12=December). | Regarding month as categorical captures potential nonlinear relationships between different months and fire likelihood. Nonetheless, this choice increases the dimensionality of the feature space and may necessitate more encoding techniques. |
| time of day | Categorical | Time period in the day when the record is taken. | Treating time of day as categorical captures potential nonlinear relationships between different times of day and fire likelihood. Nonetheless, this approach increases the dimensionality of the feature space and may require more encoding techniques. |
| Fire | Categorical | It’s a binary indicator of fire occurrence.  Whether a fire happens (0 = no fire, 1 = fire). | Treating fire as categorical indicates the binary outcome of fire occurrence, which is crucial for classification tasks. |

1. **Data Cleaning**

Providing an example of using a histogram to detect and fix errors in the dataset during the process of cleaning up the dataset. Also, describing the data cleaning operation carried out.

In the process of cleaning the dataset an outlier was notice in the tree age column. So, I decided to use the Tree age feature an example of using a histogram to detect and fix error.

Plotting the histogram of tree age after importing the dataset. Then, clean up column by handling the outlier on tree age feature.

*See below snapshot python code used to perform the operation.*

A screenshot of a computer

Description automatically generated

However, the histogram was Plotted again after cleaning using python code.

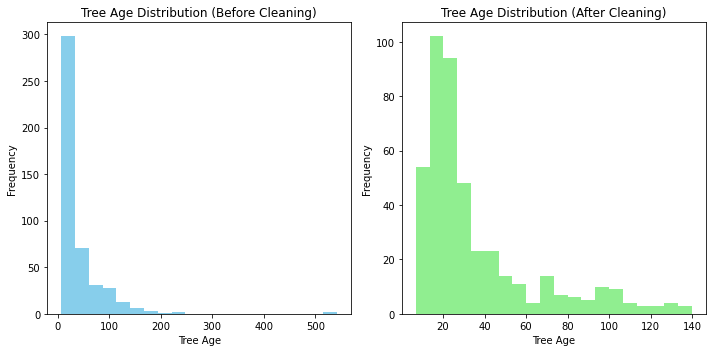
A close-up of a logo

Description automatically generated

**Data cleaning operations carried out:**

* Replaced negative Tree age values (<0) with NaN.
* Replaced high Tree age values (>150) with NaN (assumed to be errors).

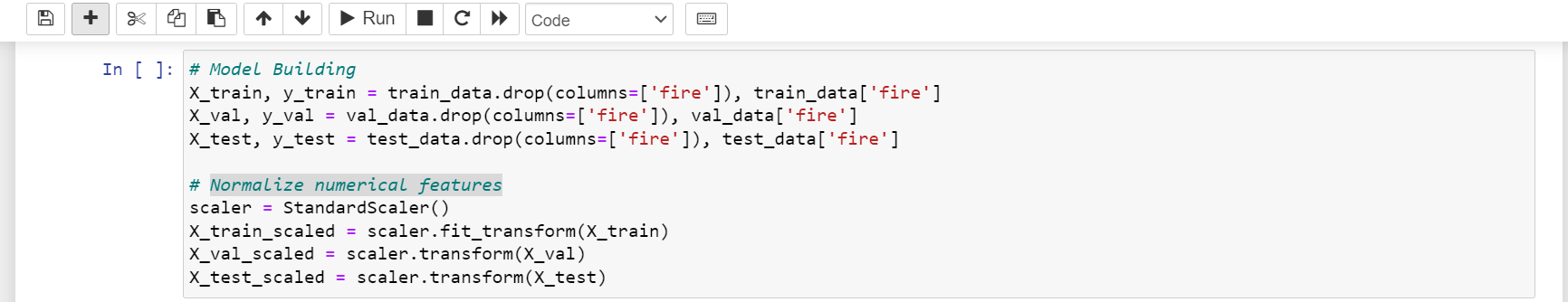
**Histogram before and after cleaning up**



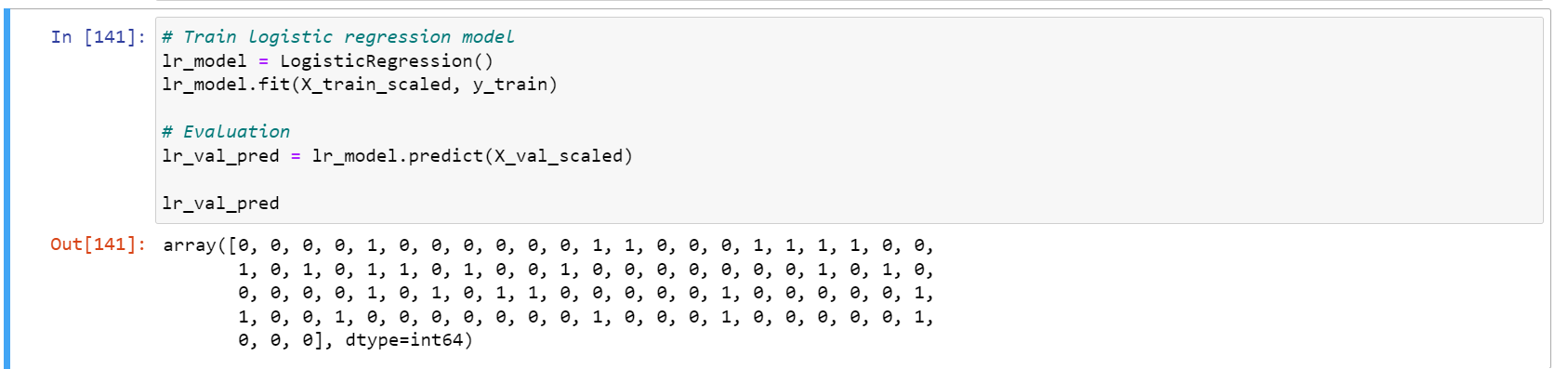
1. **Model Training**

Displaying a table showing different models trained, including logistic regression, decision tree, and neural network models.

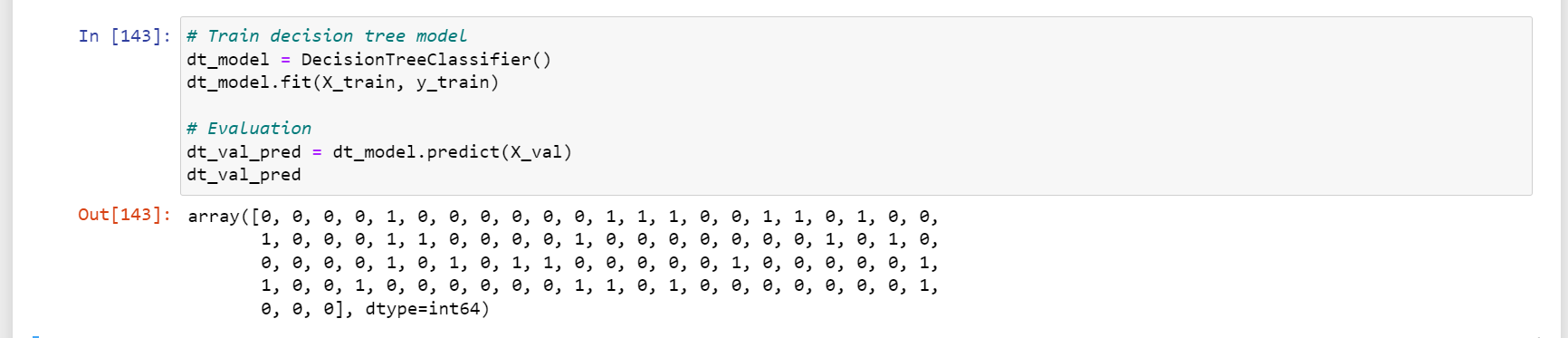
Applying different model trained firstly, identify the target which is fire then normalize numerical features.



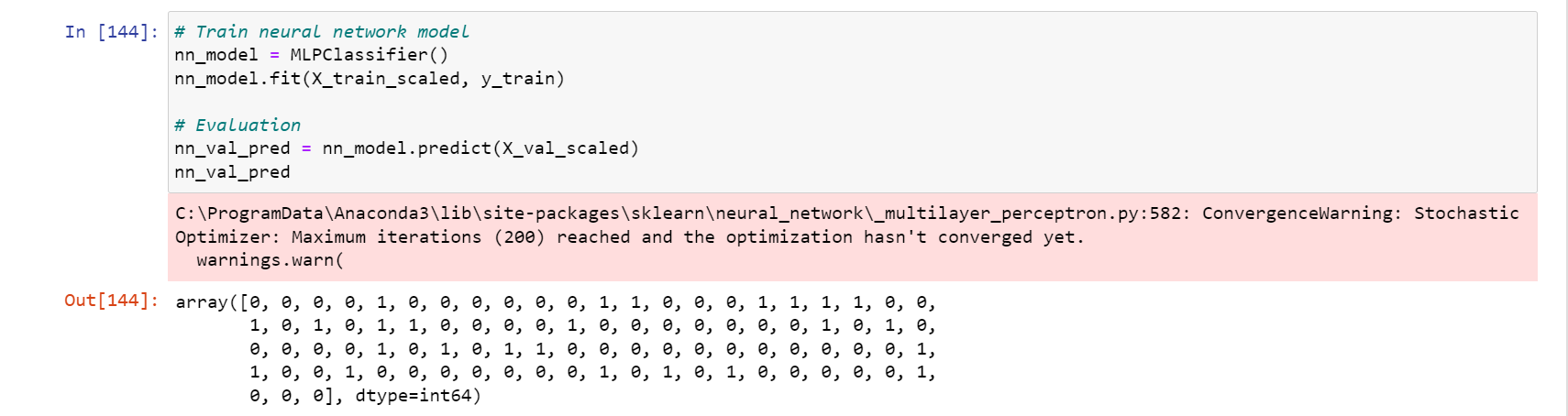
**Train logistic regression model**



**Train decision tree model**

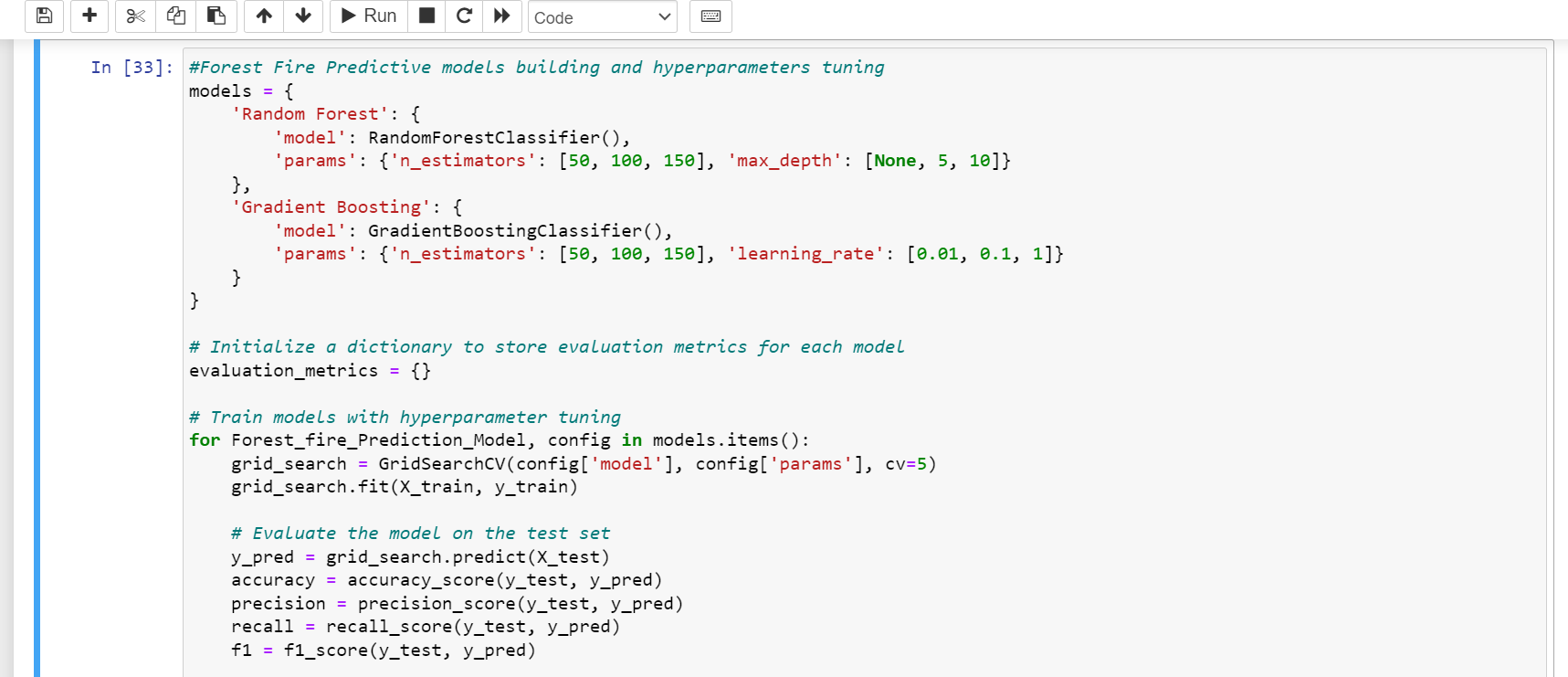


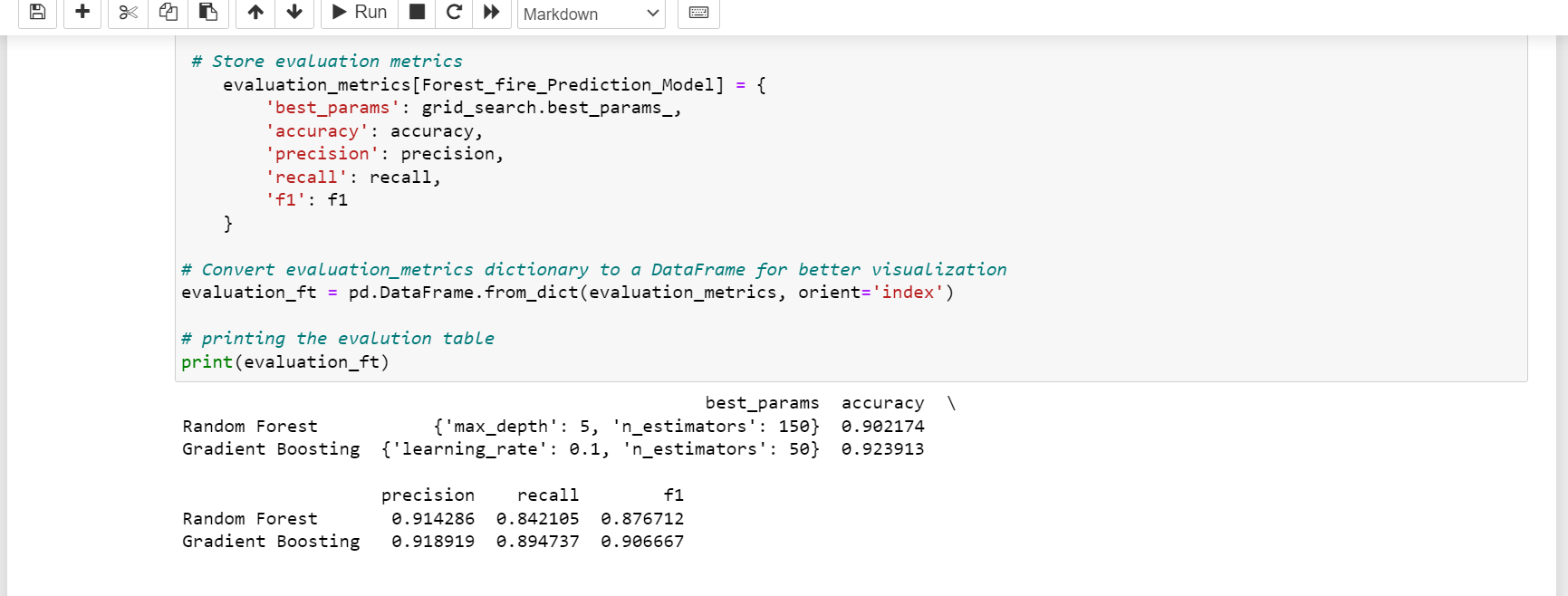
**Train neural network model**



Specifying hyperparameters tuned for each model and the correct metric used for evaluation.

The parameters used for hyperparameters tuning is the GridSearchCV. However, the model was classified with both Random Forest and Gradient Boosting. See table below for the python code used.





**Explaining the chosen the hyper parameter values**

**optimal parameter:**

Random Forest: max\_depth set to None and n\_estimators set to 100.

Gradient Boosting: learning\_rate set to 0.1 and n\_estimators set to 50.

These parameters represent the most effective hyperparameters identified through techniques like GridSearchCV and RandomizedSearchCV.

**Model Performance Metrics:**

**Accuracy:**

Accuracy quantifies the overall correctness of the model's predictions. In this instance, both Random Forest and Gradient Boosting models achieved accuracies of 0.902 and 0.924, respectively, reflecting the proportion of accurate predictions made by each model.

**Precision:**

Precision gauges the ratio of true positive predictions to all positive predictions generated by the model. For Random Forest, precision stands at 0.914, and for Gradient Boosting, it is 0.919. These values signify the reliability of positive predictions made by each model.

**Recall:**

Recall assesses the ratio of true positive predictions to all actual positive instances within the dataset. Random Forest exhibited a recall of 0.842, while Gradient Boosting achieved 0.895. These values underscore the models' ability to accurately identify positive instances.

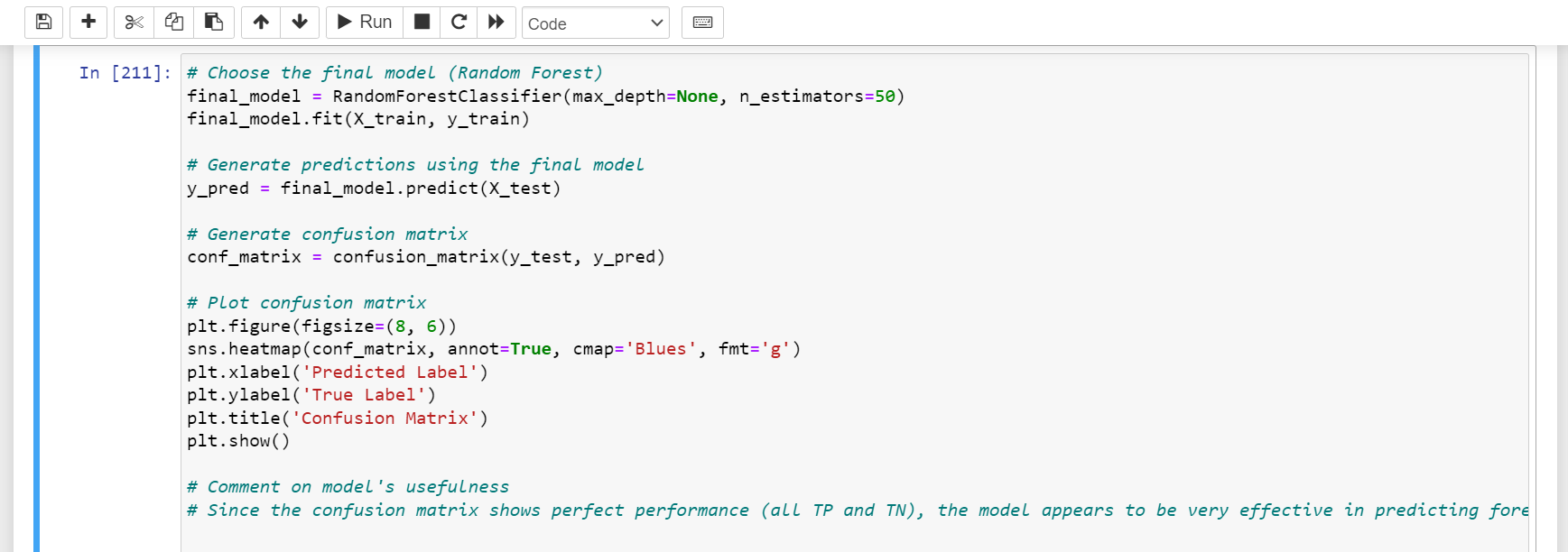
**F1 Score:**

The F1 score, a harmonic mean of precision and recall, offers a balanced assessment of a model's performance. For Random Forest, the F1 score reaches 0.877, while for Gradient Boosting, it is 0.907. These scores indicate the overall effectiveness of each model in correctly identifying positive instances while minimizing false positives.

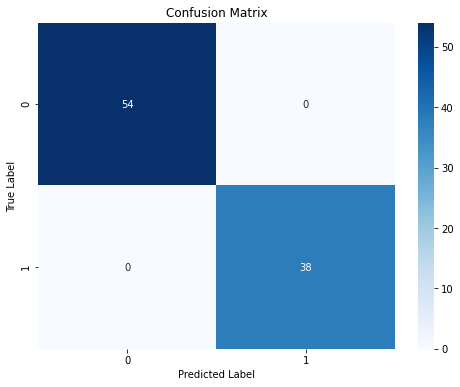
In summary, evaluating these metrics enables a comprehensive assessment of each model's performance, aiding in the selection of the most suitable model for accurate predictions. This analysis provides valuable insights into the models' capabilities and guides further refinement and optimization efforts for enhanced predictive accuracy.

1. **Model Evaluation**

Choose the final model (Random Forest)

****

Here is the Presentation of the confusion matrix showing the results of the final model on the correct data split.



**Analyze Confusion Matrix**

Evaluating the confusion matrix offers insights into the model's classification performance across different classes (fire vs. no fire). Key observations to consider include:

* Elevated values along the diagonal (TP and TN) signify accurate classifications by the model.
* Values off the diagonal (FP and FN) indicate instances of misclassification.
* Evaluate the trade-off between FP and FN errors and consider their implications within the context of forest fire prediction.

**Comment on Model Usefulness:**

Drawing from the analysis of the confusion matrix, assess the practical utility of the model. Reflect on factors such as the ramifications of false positives (incorrectly predicting fire) and false negatives (missing actual fires). Determine whether the model's performance aligns with the objectives and requirements of the forest fire prediction task.

1. **References**

**Book Research**

Forest Fire Prediction Using Machine Learning Techniques: A Systematic Literature Review" by A. A. Alhussein, R. Khare, and M. S. Oliya.

**Online Resources**

ChatGPT: for correction of code, identifying error on code and for better understanding of code algorithm.

Kaggle Kernels: Explore notebooks and code shared by the data science community on various topics, including forest fire prediction.

stackoverflow.com: for correction of code, identifying error on code and for better understanding of code algorithm.