

**COMPLEX ENGINEERING PROBLEM**

**Machine learning**

**(CS-324)**

|  |  |
| --- | --- |
| SUBMITTED BY: | RIMSHA ISHTAQ (CS-20039)  HabIBA ASIF (CS-20045)  AlainA (CS-20056) |
| Course: | machine learning |
| code: | cs-324 |
| batch | 2020 |
| year | third year |
| SUBMITTED TO: | MS. Mehwish RAza |

**DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING**

**BACHELORS IN COMPUTER SYSTEMS ENGINEERING**

**Course Code: CS-324**

**Course Title: Machine Learning**

**Complex Engineering Problem**

**TE Batch 2020, Spring Semester 2023**

#### Grading Rubric

**TERM PROJECT**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Student No.** | **Name** | **Roll No.** |
| S1 | **Rimsha Ishtiaq** | **CS-20039** |
| S2 | **Habiba Asif** | **CS-20045** |
| S3 | **Alaina** | **CS-20056** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CRITERIA AND SCALES** | | | | **Marks Obtained** | | |
| **S1** | **S2** | **S3** |
| Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-2, CPA-3) **[8 marks]** | | | |  |  |  |
| 1 | 2 | 3 | 4 |
| The application does not meet the desired specifications and is producing incorrect outputs. | The application partially meets the desired specifications and is producing incorrect or partially correct outputs. | The application meets the desired specifications but is producing incorrect or partially correct outputs. | The application meets all the desired specifications and is producing correct outputs. |
| Criterion 2: How well is the code organization? [2 marks] | | | |  |  |  |
| 1 | 2 | 3 | 4 |
| The code is poorly organized and very difficult to read. | The code is readable only to someone who knows what it is supposed to be doing. | Some part of the code is well organized, while some part is difficult to follow. | The code is well organized and very easy to follow. |
| Criterion 3: Does the report adhere to the given format and requirements? [6 marks] | | | |  |  |  |
| 1 | 2 | 3 | 4 |
| The report does not contain the required information and is formatted poorly. | The report contains the required information only partially but is formatted well. | The report contains all the required information but is formatted poorly. | The report contains all the required information and completely adheres to the given format. |
| Criterion 4: How does the student performed individually and as a team member?(CPA-1, CPA-2, CPA-3) [4 marks] | | | |  |  |  |
| 1 | 2 | 3 | 4 |
| The student did not work on the assigned task. | The student worked on the assigned task, and accomplished goals partially. | The student worked on the assigned task, and accomplished goals satisfactorily. | The student worked on the assigned task, and accomplished goals beyond expectations. |

Final Score = (Criterial\_1\_score x 2) + (Criteria\_2\_score / 2) + (Criteria\_3\_score x (3/2)) + (Criteria\_4\_score)

= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

[**Preprocessing**: 3](#_Toc140732462)

[**Model Implementation:** 4](#_Toc140732463)

[**Best Model Selection:** 4](#_Toc140732464)

[**KNeighborsClassifier (KNN):** 4](#_Toc140732465)

[**GridSearch** 4](#_Toc140732467)

[**on KNN:** 4](#_Toc140732468)

[**KNN Model 1 (Best model):** 5](#_Toc140732469)

[**KNN Model 2:** 5](#_Toc140732470)

[**KNN Model 3:** 5](#_Toc140732472)

[**Decision Tree:** 5](#_Toc140732473)

[**GridSearch on Decision Tree Classifier** 6](#_Toc140732474)

[**DecisionTreeClassifier Model 1 (Best model):** 6](#_Toc140732475)

[**DecisionTreeClassifier Model 2:** 6](#_Toc140732476)

[**DecisionTreeClassifier Model 3:** 6](#_Toc140732477)

[**Logistic Regression:** 7](#_Toc140732478)

[**GridSearch on Logistic Regression:** 7](#_Toc140732479)

[**Logistic Regression Model 1 (Best model):** 7](#_Toc140732480)

[**Logistic Regression Model 2:** 7](#_Toc140732481)

[8](#_Toc140732482)

[**Logistic Regression Model 3:** 8](#_Toc140732483)

[**Multiple Layer Perceptron (MLP):** 8](#_Toc140732484)

[**GridSearch on MLP Classifier:** 8](#_Toc140732485)

[**MLP Classifier Model 1 (Best model):** 9](#_Toc140732486)

[**MLP Classifier Model 2:** 9](#_Toc140732487)

[**MLP Classifier Model 3:** 9](#_Toc140732488)

[**User Interface:** 10](#_Toc140732489)

**Introduction:**

The problem given on hand is to explore and apply all the machine learning techniques from cleaning data to visualizing the results of multiple models for comparison of best models for the given dataset. The main key features of the dataset given are, the data has more than 30000 examples and the features are multiple also the output is supposed to be generated in multiclass. For this pupose we have applied all the techniques to preprocessed data, scale the data, model training and visualizing the models as following:

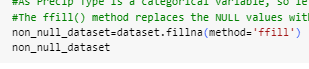
# **Preprocessing**:

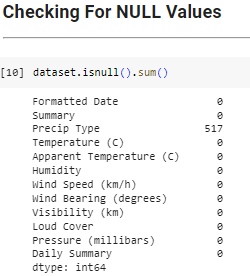
Preprocessing refers to the steps taken to clean, transform, and prepare raw data before it can be used for analysis or modeling. Here are some short steps for preprocessing data:

1. Identifying Features and Target

* Import the necessary libraries: Start by importing the required libraries such as pandas, NumPy, or scikit-learn, which are commonly used for data preprocessing tasks.
* Load the data: Read the raw data into programming environment using the appropriate functions or methods. The data was in CSV format.
* Initially 9 features were identified including:
* Summary was taken as target which consisted of 27 number of classes.

2. Handle missing values:

* There are no missing values for except for ‘Precip Type’ which has 517 missing values.
* As 'Precip Type' is a categorical variable, so we can't fill missing values with mean/med/mode but fill based on the assumption that adjacent observations are similar to one another using ‘ffill’ method.



3. Encoding categorical variables: Convert categorical variables into numerical representations that can be understood by machine learning algorithms.

* This was done on ‘PrecipType’ categorical variable using label encoding.
* Target was remained as categorical.

4. Feature engineering: Created new features or transformed existing ones to enhance the predictive power of the data.

* Collinearity test was carried out on features which resulted in removing ‘Apparent temperature’ feature which was causing collinearity.
* Grouping suitable classes; grouped classes into 4 namely, Partly Cloudy, Mostly Cloudy, Overcast, Non-Cloudy
* Feature Scaling using standardization

5. Split the data: Split the preprocessed data into training and testing sets. The training set is used to build the model, while the testing set is used to evaluate its performance which ensures an unbiased evaluation of the model's capabilities.

# **Model Implementation:**

The models selected for implementation includes:

* Knn (non-parametric)
* Logistic Regression (parametric)
* ANN

## **Best Model Selection:**

As the model are implemented, their hyperparameter selection is not an easy task. For that k-fold cross validation is used to ensure that the best hyperparameters according to the chosen evaluation metric are selected.

## **KNeighborsClassifier (KNN):**

The K-Nearest Neighbors (KNN) algorithm is a non-parametric and instance-based classification algorithm. It does not involve explicit model training but instead stores the entire training dataset in memory for prediction. KNN makes predictions based on the similarity or proximity between instances in the feature space.

KNN classifier with its possible parameters is defined below:

Although, there are many hyperparameters, we choose to tune the following: n\_neighbors, p , weights.

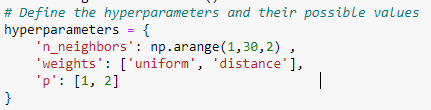
### 

**KNeighborsClassifier**(*n\_neighbors=5*, *\**, *weights='uniform'*, *algorithm='auto'*, *leaf\_size=30*, *p=2*, *metric='minkowski'*, *metric\_params=None*, *n\_jobs=None*)

### **GridSearch**

### **on KNN:**

In order to choose the best value for the hyperparameters, we applied the Grid search technique.



Values of the hyperparameters are defined below:

* n\_neigbors:

It is a hyperparameter that determines the number of neighbors to consider when making predictions for a new instance. Its default value is 5.

By hit-and-trial, we take range as (1,30,2)

* weights:

It specifies how the neighbor’s contributions are weighted when making predictions.

Its possible values are: ‘distance’ and ‘uniform’.

* **distance:** weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
* **uniform:** uniform weights. All points in each neighborhood are weighted equally.
* p:

Power parameter for the Minkowski metric.

Its possible values are: 1 and 2

* p=1: Manhattan distance
* p=2: Euclidean distance

This Grid Search selects the following values as the best values for the hyperparameters mentioned above:

Best Hyperparameters:{'n\_neighbors': 29, 'p': 1, 'weights': 'distance'}

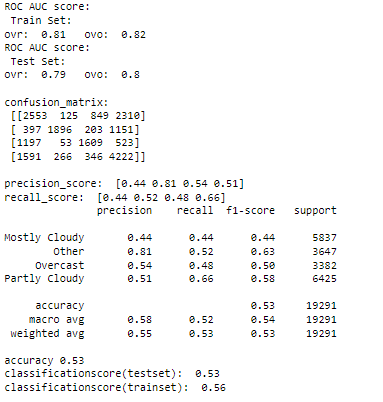
Implementing the model as Function:

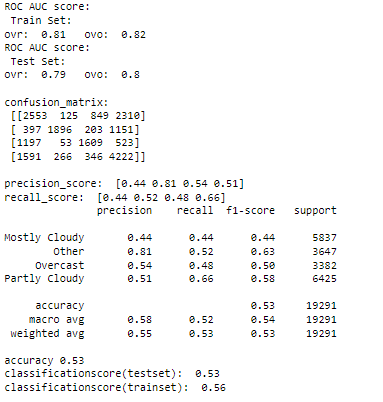
A function is defined as ‘knn\_classifier’, in which KNN model is defined and trained.

This function calculates the roc\_auc\_curves value for one-vs-rest and one-vs-one techniques.

This also gives us the confusion matrix, precision and recall values for the model.

### **KNN Model 1 (Best model):**

 In KNN model 1, we use the best values of the hyperparameters obtained by Grid Search.



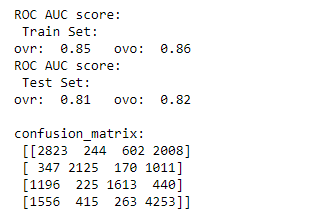
### **KNN Model 2:**

In this the combination of other values for the hyperparameters is used:

### 

### **KNN Model 3:**

In this the combination of other values for the hyperparameters is used:

 A screenshot of a computer program

Description automatically generated

## **Decision Tree:**

**Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and Regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

DecisionTree Classifier with its possible hyperparameters is defined below:

**DecisionTreeClassifier**(*\**, *criterion='gini'*, *splitter='best'*, *max\_depth=None*, *min\_samples\_split=2*, *min\_samples\_leaf=1*, *min\_weight\_fraction\_leaf=0.0*, *max\_features=None*, *random\_state=None*, *max\_leaf\_nodes=None*, *min\_impurity\_decrease=0.0*, *class\_weight=None*, *ccp\_alpha=0.0*)

Although, there are many hyperparameters, we choose to tune the following: max\_depth, min\_samples\_split, min\_samples\_leaf, criterion.

### **GridSearch on Decision Tree Classifier**

In order to choose the best value for the hyperparameters , we applied the Grid search technique.

Values of the hyperparameters are defined below:

* max\_depth:

The maximum depth of the tree. It limits the number of nodes and splits in the tree, helping to control overfitting. Options include are: None, 5 and 10

* min\_samples\_split:

The minimum number of samples required to split an internal node. Options include are: [2,5,10]

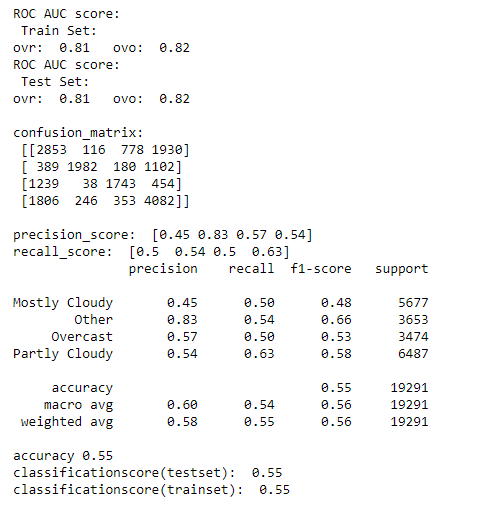
* min\_samples\_leaf:

The minimum number of samples required to be at a leaf node. Options include are: [1,2,4]

* criterion

The function used to measure the quality of a split. Common criteria are "gini" for Gini impurity and "entropy" for information gain.

### **DecisionTreeClassifier Model 1 (Best model):**

 In model 1, we use the best values of the hyperparameters obtained by Grid Search.

### **DecisionTreeClassifier Model 2:**

A screenshot of a computer

Description automatically generated In this the combination of other values for the hyperparameters is used:

### **DecisionTreeClassifier Model 3:**

A screenshot of a computer

Description automatically generated In this the combination of other values for the hyperparameters is used:

## **Logistic Regression:**

Logistic regression is a statistical model used for binary classification problems, where the dependent variable takes two possible values.Logistic regression can be extended to handle multiclass classification problems through various strategies, such as one-vs-rest (also known as one-vs-all).

The goal of logistic regression is to estimate the probability of the dependent variable belonging to a specific class based on the values of the independent variables.

Logistic Regression with its possible hyperparameters is defined below:

Although, there are many hyperparameters, we choose to tune the following: C, penalty, solver.

**LogisticRegression**(*penalty='l2'*, *\**, *dual=False*, *tol=0.0001*, *C=1.0*, *fit\_intercept=True*, *intercept\_scaling=1*, *class\_weight=None*, *random\_state=None*, *solver='lbfgs'*, *max\_iter=100*, *multi\_class='auto'*, *verbose=0*, *warm\_start=False*, *n\_jobs=None*, *l1\_ratio=None*)

### **GridSearch on Logistic Regression:**

In order to choose the best value for the hyperparameters , we applied the Grid search technique.

A computer code with text

Description automatically generatedValues of the hyperparameters are defined below:

* C:

This is Inverse of regularization strength; must be a positive float and  smaller values specify stronger regularization.

By some research and hit-and-trial, we select 0.1, 1.0 and 10.0 as the options.

* penalty:

The penalty hyperparameter specifies the type of regularization used in multiclass logistic regression. Options include":

* 'l1' (L1 regularization)
* 'l2' (L2 regularization)
* 'elasticnet' (combination of L1 and L2 regularization)
* 'none' (no regularization).

But because of the compatibility issue we select only ‘l2’

* solver:

The solver algorithm determines the optimization algorithm used to estimate the coefficients in multinomial logistic regression.

Common choices include 'newton-cg', 'lbfgs', 'sag', and 'saga'.

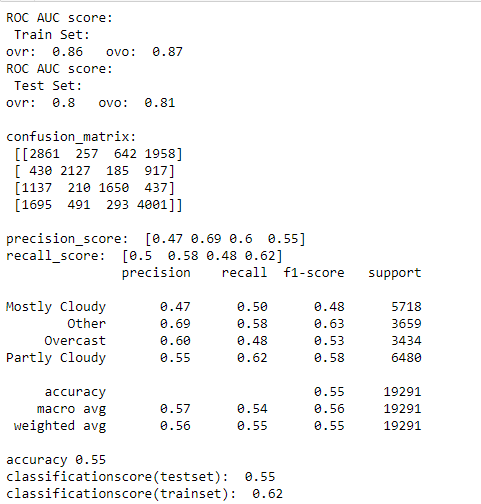
Because l2 is only compatible with ‘lbfgs’ and ‘newton-cg’, we select these as the options.

This Grid Search selects the following values as the best values for the hyperparameters mentioned above: In addition to these hyperparameters, we set:

Best Hyperparameters:{ {'C': 10.0, 'penalty': 'l2', 'solver': 'newton-cg'}

multi-class = auto and max-iter = 1000

### **Logistic Regression Model 1 (Best model):**

 In model 1, we use the best values of the hyperparameters obtained by Grid Search.

### **Logistic Regression Model 2:**

In this the combination of other values for the hyperparameters is used:

### A screenshot of a computer Description automatically generated

### **Logistic Regression Model 3:**

A screenshot of a computer

Description automatically generated In this the combination of other values for the hyperparameters is used:

## **Multiple Layer Perceptron (MLP):**

The Multilayer Perceptron (MLP) is a feedforward artificial neural network that consists of multiple layers of interconnected neurons. It is a versatile model widely used for both regression and classification tasks.

MLP with its possible hyperparameters is defined below:

**MLPClassifier**(*hidden\_layer\_sizes=(100,)*, *activation='relu'*, *\**, *solver='adam'*, *alpha=0.0001*, *batch\_size='auto'*, *learning\_rate='constant'*, *learning\_rate\_init=0.001*, *power\_t=0.5*, *max\_iter=200*, *shuffle=True*, *random\_state=None*, *tol=0.0001*, *verbose=False*, *warm\_start=False*, *momentum=0.9*, *nesterovs\_momentum=True*, *early\_stopping=False*, *validation\_fraction=0.1*, *beta\_1=0.9*, *beta\_2=0.999*, *epsilon=1e-08*, *n\_iter\_no\_change=10*, *max\_fun=15000*)

As we can see there are many hyperparameters, we choose to tune the following: ‘hidden\_layer’, ‘activation’, ‘solver’, ‘alpha’, ‘learning\_rate’

### **GridSearch on MLP Classifier:**

In order to choose the best value for the hyperparameters , we applied the Grid search technique.

Values of the hyperparameters are defined below:

* hidden\_layer:

MLP can have one or more hidden layers between the input and output layers. Each hidden layer consists of multiple neurons that perform nonlinear transformations on the input data.

Here we added two hidden layers, and by hit-and-trial method take the options for the values as:

(100,) , (100,50), (50,50)

* activation:

Activation functions introduce nonlinearity to the MLP, enabling it to learn complex patterns and relationships in the data. Following activation functions are taken for the model

* relu: the rectified linear unit function, returns f(x) = max(0, x)
* tanh:  the hyperbolic tan function, returns f(x) = tanh(x).
* solver:

This hyperparameter refers to the optimization algorithm used to train the neural network by minimizing the loss function. The solver determines how the weights are updated during the learning process.

The solver hyperparameter options we used are:

* sgd (Stochastic Gradient Descent): This solver performs updates to the model's weights based on a randomly selected subset of training samples (mini-batches).
* Adam (Adaptive Moment Estimation): It adapts the learning rates for each parameter based on estimates of the first and second moments of the gradients.
* alpha:

It controls the regularization strength applied to the neural network model.

Values selected are: [0.0001, 0.001, 0.01]

All these values are selected by some research and hit-and-trial method.

* learning\_rate:

It determines the step size taken during the optimization process to update the weights of the neural network.

The different options are as follows:

* constant: this is a constant learning rate given by ‘learning\_rate\_init’.
* invscaling: It gradually decreases the learning rate at each time step ‘t’ using an inverse scaling exponent of ‘power\_t’.

effective\_learning\_rate = learning\_rate\_init / pow(t, power\_t)

* adaptive: It keeps the learning rate constant to ‘learning\_rate\_init’ as long as training loss keeps decreasing.

### **MLP Classifier Model 1 (Best model):**

A screenshot of a computer

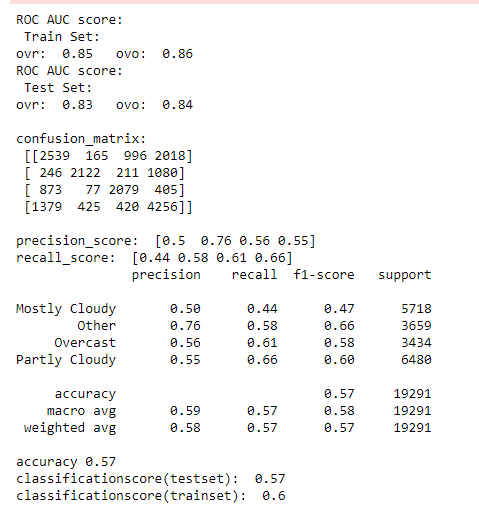
Description automatically generated In model 1, we use the best values of the hyperparameters obtained by Grid Search.

### **MLP Classifier Model 2:**

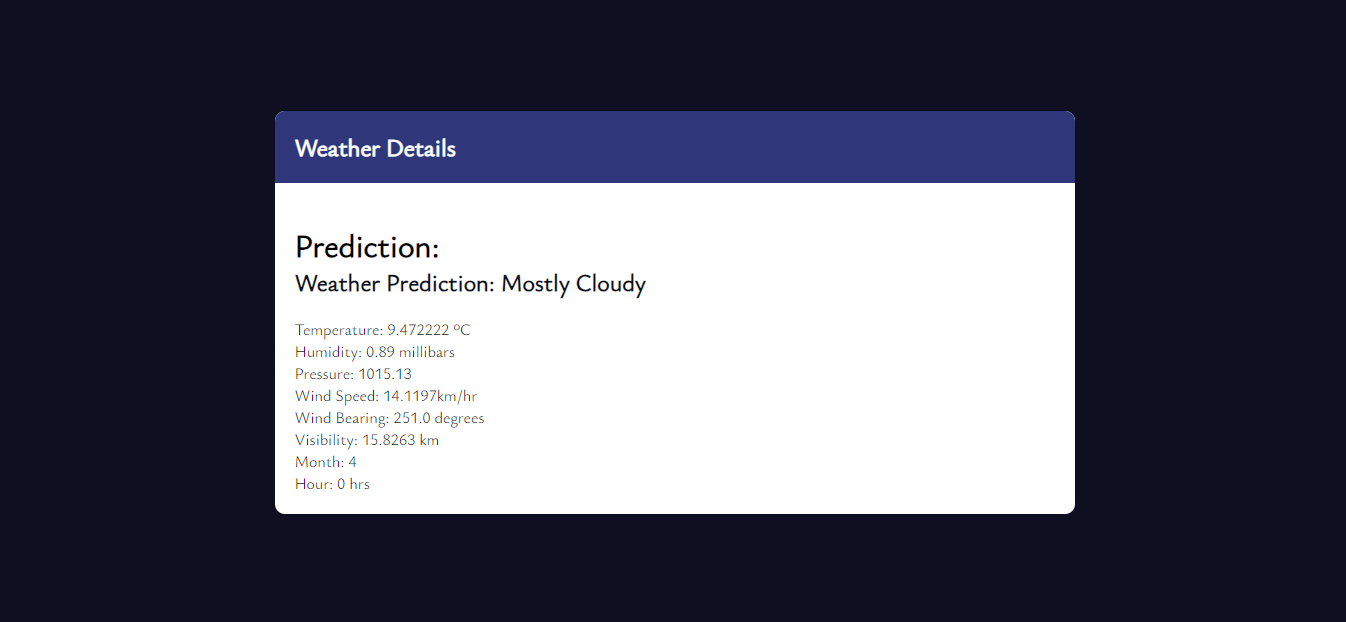
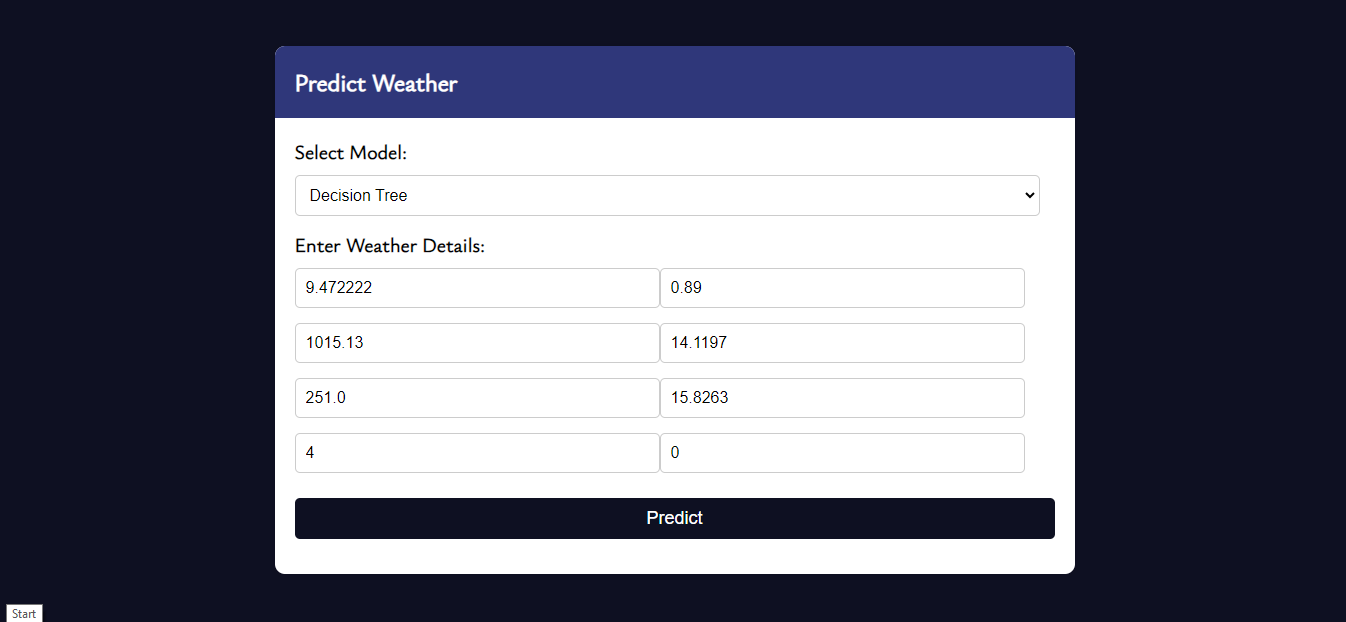
A screenshot of a computer

Description automatically generated In this the combination of other values for the hyperparameters is used:

### **MLP Classifier Model 3:**

 In this the combination of other values for the hyperparameters is used:

# **User Interface:**

For user interface, flask app is used to show the results

**Tabular Comparison:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Knn\_model1 | Knn\_model2 | Knn\_model3 | DecisonTree\_model1 | DecisonTree\_model2 | DecisonTree\_model3 | MLP\_Classifier\_model1 | MLP\_Classifier\_model2 | MLP\_Classifier\_model3 |
| Train roc\_auc score | 0.82 | 0.87 | 0.86 | 0.82 | 0.82 | 0.97 | 0.86 | .86 | 0.86 |
| Test roc\_auc score | 0.82 | 0.81 | 0.82 | 0.82 | 0.80 | 0.75 | 0.84 | 0.84 | 0.84 |
| Precision score | [0.45, 0.83, 0.57, 0.54] | [0.47, 0.69, 0.6, 0.55] | [0.48, 0.71, 0.61, 0.55] | [0.45, 0.83, 0.57, 0.54] | [0.44,0.8, 0.57, 0.51] | [0.44, 0.63, 0.54, 0.54] | [0.5, 0.76, 0.56, 0.55] | [0.5, 0.76, 0.56, 0.55] | [0.5, 0.76, 0.56, 0.55] |
| Recall\_score | [0.5 0.54 0.5 0.63] | [0.47 0.69 0.6 0.55] | [0.5 0.58 0.46 0.66] | [0.5 0.54 0.5 0.63] | [0.45 0.54 0.44 0.66] | [0.5, 0.63, 0.51, 0.5 ] | [0.44, 0.58, 0.61, 0.66] | [0.44, 0.58, 0.61, 0.66] | [0.44, 0.58, 0.61, 0.66] |
| Accuracy | 0.55 | 0.55 | 0.56 | 0.55 | 0.53 | 0.53 | 0.57 | 0.57 | 0.57 |
| Test classification score | 0.55 | 0.55 | 0.56 | 0.55 | 0.53 | 0.53 | 0.57 | 0.57 | 0.57 |
| Train classification score | 0.55 | 0.62 | 0.67 | 0.55 | 0.56 | 0.83 | 0.60 | 0.60 | 0.60 |

**Graphical comparisons:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |