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	**Syeda Zobia Irshad**	**CS-20043**
S2	**Vandana**	**CS-20075**

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

Final Score = (Criterial_1_score x 2) + (Criteria_2_score / 2) + (Criteria_3_score x (3/2)) + (Criteria_4_score)

= ______________________

DATA PREPROCESSING STEPS

1. DATA CLEANING:

• By Handling Missing Values:

Here feature precip Type has 517 missing values .We handle by replacing missing values by the **fillna() method** is used to fill or replace missing (NaN) values in a DataFrame or Series with specified values. This method is especially useful during the data preprocessing step to handle missing data before performing data analysis or modeling.

Screenshots For Handling Missing Values Of Precip Type Feature:

```
data = pd.read_csv('weatherHistory.csv')
data.isnull().sum()
```

```
Formatted Date               0
Summary                      0
Precip Type                517
Temperature (C)              0
Apparent Temperature (C)     0
Humidity                     0
Wind Speed (km/h)            0
Wind Bearing (degrees)       0
Visibility (km)              0
Loud Cover                   0
Pressure (millibars)         0
Daily Summary                0
dtype: int64
```

```
data['Precip Type'].fillna(method='ffill',inplace=True,axis=0)
```

```
data.isnull().sum()
```

```
Formatted Date              0
Summary                     0
Precip Type                 0
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
Pressure (millibars)        0
Daily Summary               0
dtype: int64
```

2. DATA TRANSFORMATION:

• Features Scaling:

Now , Scaling numerical features to a similar range, often between 0 and 1, to avoid issues with algorithms sensitive to feature magnitudes

• Encoding Categorical Values:

Using Label Encoder to encode categorical variable (Precip Type) having values of rain and snow converted to "0" and "1" respectively and similarly encoding "Summary" feature

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data['Summary']=le.fit_transform(data['Summary'])
data['Precip Type']=le.fit_transform(data['Precip Type'])
data["Precip Type"].value_counts()
```

```
0    85741
1    10712
Name: Precip Type, dtype: int64
```

Feature Engineering:

Creating new features or extracting relevant information from existing features to improve model performance.

We extracred month and hour features from Formatted Date feature, because prediction of weather depends mostly on month and hour rather than year.

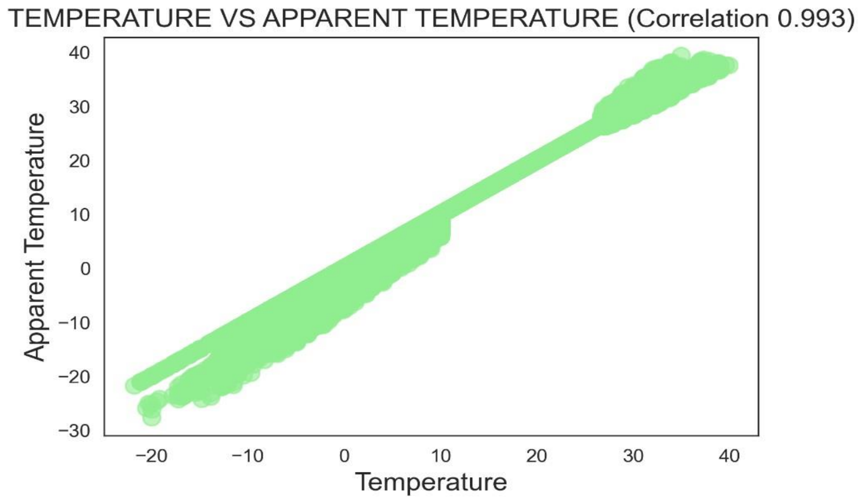
```
In [6]: data['Formatted Date'] = pd.to_datetime(data['Formatted Date'],utc = True)
        data['month'] = data['Formatted Date'].dt.month
        data['hour'] = data['Formatted Date'].dt.hour
```

3. DATA REDUCTION:

- Dimensionality Reduction:

Reducing the number of features to decrease computation time and avoid the curse of dimensionality.

- **Dropping feature "Loud Cover"** because its all values are zero.
- Similarly **dropped feature of "Formatted Date"** as we extracted month and hour from it ..No need of redundant information.
- .Similarly **dropped "Daily Summary "** feature as the :Summary" feature is providing same information .
- **Dropped feature of "Apparent Temperature (C)"** because of strong positive correlation (0.993) between feature "Temperature " as it leads to multicollinearity.



5. DATA SPLITTING

Splittin the dataset into training and testing/validation sets to evaluate model performance accurately.

```
X_train, X_Combine, Y_train, Y_Combine = train_test_split(inp,output,
                                                          train_size=0.8,
                                                          random_state=42)
X_val, X_test, Y_val, Y_test = train_test_split(X_Combine,
                                                Y_Combine,
                                                test_size=0.5,
                                                random_state=42)
```

6. HANDLING IMBALANCED DATA

WeatherHistory data was imbalanced as we had multiple imbalanced classes . So we transformed multiple classes in 3 classes as "Partly Cloudy","Mostly Cloudy","Others" by combining all other classes as new class "Others" and dropping class of "Clear" to make it closely balanced classes

Weather Type	Count
Partly Cloudy	31733
Mostly Cloudy	28094
Overcast	16597
Clear	10890
Foggy	7148
Breezy and Overcast	528
Breezy and Mostly Cloudy	516
Breezy and Partly Cloudy	386
Dry and Partly Cloudy	86
Windy and Partly Cloudy	67
Light Rain	63
Breezy	54
Windy and Overcast	45
Humid and Mostly Cloudy	40
Drizzle	39
Breezy and Foggy	35
Windy and Mostly Cloudy	35
Dry	34
Humid and Partly Cloudy	17

	Weather Type	Count
0	Partly Cloudy	32290
1	Mostly Cloudy	28699
2	Other	24574

7. FEATURE SCALING

Feature scaling is necessary to ensure convergence and improve performance.So scaling train and test sets by standardization

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(X_train)
x_test=sc.transform(X_test)
x_val=sc.transform(X_val)
```

MODELS AND MACHINE LEARNING ALGORITHM CHOSEN

1) LOGISTIC REGRESSION (PARAMETRIC ALGORITHM)

Logistic regression is a statistical model used for binary classification tasks, where the goal is to predict the probability of an event or the likelihood of belonging to a certain class based on input variables.

- **MODEL #1**

Hyperparameters : max_iter =2000,solver=newton-cg , random_state=0

```
clf1 = LogisticRegression(random_state=0,solver='newton-cg',max_iter=2000).fit(x_train, Y_train)
y_pred=clf1.predict(x_val)
print("accuracy on train set",clf1.score(x_train, Y_train))
print("accuracy on validation set",clf1.score(x_val, Y_val))
```

```
accuracy on train set 0.5618407596785975
accuracy on validation set 0.565684899485741
```

- **MODEL #2**

Hyperparameters : max_iter =1000,solver=newton-cholesky, random_state=42

```
clf2 = LogisticRegression(random_state=42,solver='newton-cholesky',max_iter=1000).fit(x_train, Y_train)
y_pred=clf2.predict(x_val)

print("accuracy on train set",clf2.score(x_train, Y_train))
print("accuracy on test set",clf2.score(x_val, Y_val))
```

```
accuracy on train set 0.5560847333820307
accuracy on test set 0.5596072931276297
```

- **MODEL #3**

Hyperparameters : max_iter =2000,solver=newton-cg , random_state=0

```
]: clf3 = LogisticRegression(random_state=30,solver='saga',max_iter=3000).fit(x_train, Y_train)
   y_pred=clf3.predict(x_val)

   print("accuracy on train set",clf3.score(x_train, Y_train))
   print("accuracy on test set",clf3.score(x_val, Y_val))
```

```
accuracy on train set 0.5618115412710007
accuracy on test set 0.565684899485741
```

2) RANDOM FOREST (NON PARAMETRIC ALGORITHM)

Random Forest combines multiple decision trees for making predictions.It uses randomness by considering only a random subset of features for each tree.The predictions of all the trees are combined to make the final prediction.Random Forest is robust against overfitting.It can handle large datasets and high-dimensional feature spaces.Random Forest provides insights into feature importance.It is used for classification and regression tasks in machine learning.

- **MODEL #1**

Hyperparameters : max_depth =32,criterion=’log_loss”,n_estimators=120 , random_state=42

```
4]: from sklearn.ensemble import RandomForestClassifier

    rf1=RandomForestClassifier(max_depth=32,n_estimators=120,random_state=42,criterion='log_loss')
    rf1.fit(x_train,Y_train)
    y_pred=rf1.predict(x_val)
    metrics.accuracy_score(Y_val,y_pred)
```

```
4]: 0.6921458625525947
```

- **MODEL #2**

Hyperparameters : max_depth =20,criterion=’gini”,n_estimators=100 , random_state=42

```
: rf2=RandomForestClassifier(max_depth=20,n_estimators=100,random_state=42,criterion='gini')
  rf2.fit(x_train,Y_train)
  y_pred=rf2.predict(x_val)
  metrics.accuracy_score(Y_val,y_pred)
```

```
: 0.6826788218793829
```

- **MODEL #3**

Hyperparameters: max_depth =12,criterion=’entropy”,n_estimators=300 , random_state=0

```
|: rf3=RandomForestClassifier(max_depth=12,n_estimators=300,random_state=0,criterion='entropy')
   rf3.fit(x_train,Y_train)
   y_pred=rf3.predict(x_val)
   metrics.accuracy_score(Y_val,y_pred)
```

```
|: 0.6466806919121084
```

3) **ARTIFICIAL NEURAL NETWORK**

ANN consists of interconnected nodes (neurons) organized in layers: an input layer, one or more hidden layers, and an output layer.

The neurons receive input data, apply weights to the inputs, and pass the result through an activation function. During training, the network adjusts the weights to minimize the difference between predicted and actual outputs. ANNs are used for various tasks, such as classification, regression, pattern recognition, and function approximation, making them a fundamental tool in modern machine learning.

USING EARLY STOPPING IN TRAINING

- **MODEL #1**

Hyperparameters : 2 hidden layers each of 64 and 32 neurons having activation function “RELU” respectively ,epochs=50, batch-size=32

```
# Define your model using Keras
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(9,)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(3, activation='softmax')
])


# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model with early stopping
history = model.fit(x_train, Y_train, epochs=50, batch_size=32, validation_data=(x_val, Y_val), callbacks=[early_stopping])
test_loss, test_accuracy = model.evaluate(x_val, Y_val)

print(" Validation Set accuracy:", test_accuracy)
```

```
Epoch 32/50
2140/2140 [==============================] - 5s 2ms/step - loss: 0.7604 - accuracy: 0.6332 - v
0.6252
268/268 [==============================] - 1s 2ms/step - loss: 0.7750 - accuracy: 0.6218
 Validation Set accuracy: 0.6217858791351318
```

- **MODEL #2**

Hyperparameters : 2 hidden layers each of 64 and 32 neurons having activation function “SOFTMAX” respectively ,epochs=80, batch-size=25

```python
# Define your model using Keras
model1 = keras.Sequential([
    keras.layers.Dense(64, activation='softmax', input_shape=(9,)),
    keras.layers.Dense(32, activation='softmax'),
    keras.layers.Dense(3, activation='softmax')
])


# Compile the model
model1.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model with early stopping
history = model1.fit(x_train, Y_train, epochs=80, batch_size=25, validation_data=(x_val, Y_val), callbacks=[early_stopping])
test_loss, test_accuracy = model.evaluate(x_val, Y_val)

print(" Validation Set accuracy:", test_accuracy)
```

```
0.6081
Epoch 37/80
2738/2738 [==============================] - 6s 2ms/step - loss: 0.8012 - accuracy: 0.6067
0.6080
268/268 [==============================] - 0s 2ms/step - loss: 0.7750 - accuracy: 0.6218
 Validation Set accuracy: 0.6217858791351318
```

- **MODEL #3**

Hyperparameters : 2 hidden layers each of 80 and 80 neurons having activation function “RELU” respectively ,epochs=40, batch-size=25

```python
# Define your model using Keras
model2 = keras.Sequential([
    keras.layers.Dense(80, activation='relu', input_shape=(9,)),
    keras.layers.Dense(80, activation='relu'),
    keras.layers.Dense(3, activation='softmax')
])


# Compile the model
model2.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model with early stopping
history = model2.fit(x_train, Y_train, epochs=40, batch_size=25, validation_data=(x_val, Y_val), callbacks=[early_stopping])
test_loss, test_accuracy = model.evaluate(x_val, Y_val)

print(" Validation Set accuracy:", test_accuracy)
```

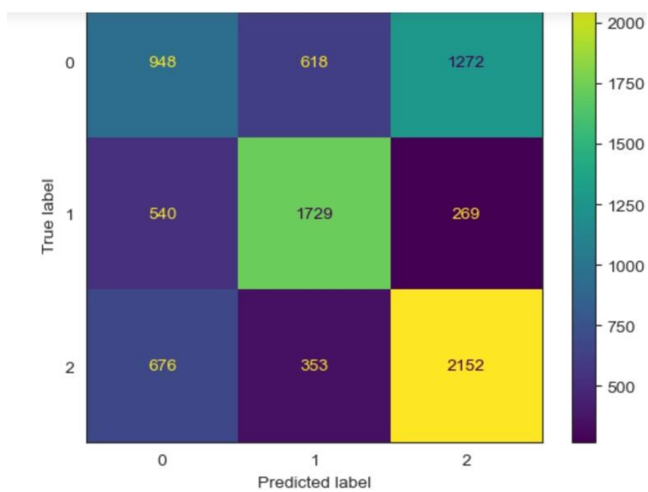
```
2738/2738 [==============================] - 6s 2ms/step - loss: 0.7338 - accuracy: 0.6471 - va
0.6311
268/268 [==============================] - 0s 1ms/step - loss: 0.7750 - accuracy: 0.6218
 Validation Set accuracy: 0.6217858791351318
```

TABULAR AND GRAPHICAL REPRESENTATION

LOGISTIC REGRESSION MODEL1

PERFORMANCE METRICS	**LOGISTIC REGRESSION** (MODEL1)
Training set accuracy	56.1%
Valdation set accuracy	56.5%
Precsion	[0.43807763 0.64037037 0.58272407]
Recall	[0.33403805 0.68124507 0.67651682]
F1-score	[0.37904838 0.66017564 0.62612744]

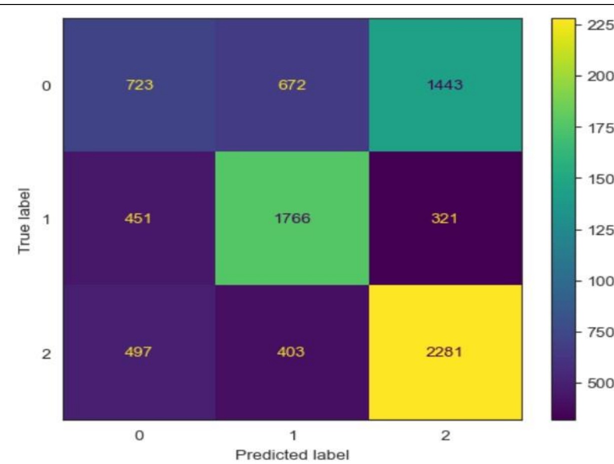
```
The accuracy is 56.43332943788712%
[[ 948  618 1272]
 [ 540 1729  269]
 [ 676  353 2152]]
Precision:  [0.43807763 0.64037037 0.58272407]
Micro Precision:  0.5643332943788711
Macro Precision:  0.5537240256503958
Weighted Precision:  0.551848732051861

Recall score:  [0.33403805 0.68124507 0.67651682]
Micro Recall score:  0.5643332943788711
Macro Recall score:  0.5639333161469612
Weighted Recall score:  0.5643332943788711

F1 score:  [0.37904838 0.66017564 0.62612744]
Micro F1 score:  0.5643332943788711
Macro F1 score:  0.5551171523076306
Weighted F1 score:  0.5542802914192286
```



MODEL 2

PERFORMANCE METRICS	**LOGISTIC REGRESSION** (MODEL2)
Training set accuracy	55.6%
Valdation set accuracy	55.9%
Precsion	[0.43267504 0.62161211 0.56390606]
Recall	[0.25475687 0.69582348 0.7170701]
F1-score	[0.32069195 0.65662763 0.6313313]

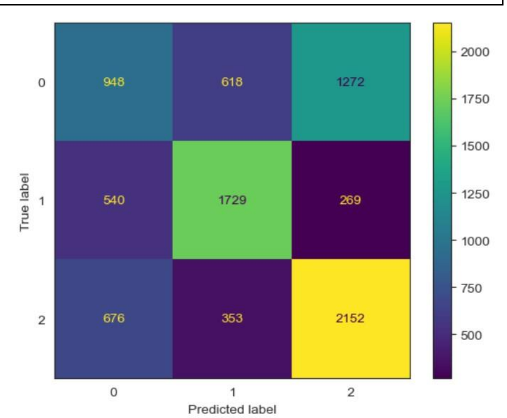
```
The accuracy is 55.74383545635152%
[[ 723  672 1443]
 [ 451 1766  321]
 [ 497  403 2281]]
Precision:  [0.43267504 0.62161211 0.56390606]
Micro Precision:  0.5574383545635152
Macro Precision:  0.5393977367187185
Weighted Precision:  0.5374977767211057

Recall score:  [0.25475687 0.69582348 0.7170701 ]
Micro Recall score:  0.5574383545635152
Macro Recall score:  0.5558834859448094
Weighted Recall score:  0.5574383545635152

F1 score:  [0.32069195 0.65662763 0.6313313 ]
Micro F1 score:  0.5574383545635152
Macro F1 score:  0.5362169596710132
Weighted F1 score:  0.5358080570289612
```

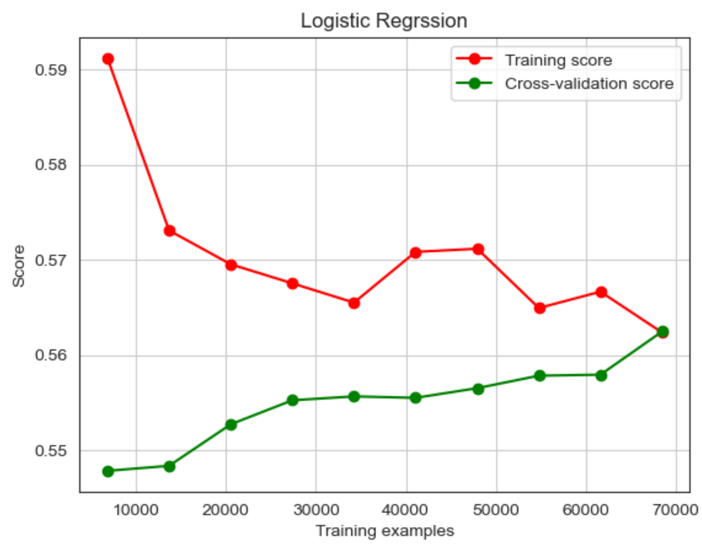


MODEL 3(Best Model so Predict Test Set)

PERFORMANCE METRICS	**LOGISTIC REGRESSION** (MODEL3)(BEST MODEL)
Training set accuracy	56.1%
Valdation set accuracy	56.5%
Precsion	[0.43807763 0.64037037 0.58272407]
Recall	[0.33403805 0.68124507 0.67651682]
F1-score	[0.37904838 0.66017564 0.62612744]
Testing Set Accuracy	56.4%

```
The accuracy is 56.43332943788712%
[[ 948  618 1272]
 [ 540 1729  269]
 [ 676  353 2152]]
Precision:  [0.43807763 0.64037037 0.58272407]
Micro Precision:  0.5643332943788711
Macro Precision:  0.5537240256503958
Weighted Precision:  0.551848732051861

Recall score:  [0.33403805 0.68124507 0.67651682]
Micro Recall score:  0.5643332943788711
Macro Recall score:  0.5639333161469612
Weighted Recall score:  0.5643332943788711

F1 score:  [0.37904838 0.66017564 0.62612744]
Micro F1 score:  0.5643332943788711
Macro F1 score:  0.5551171523076306
Weighted F1 score:  0.5542802914192286
```



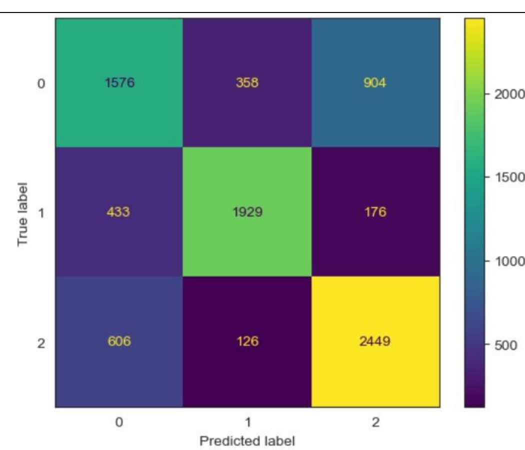


RANDOM FOREST CLASSIFIER

MODEL 1 (Best Model so Predict Test Set)

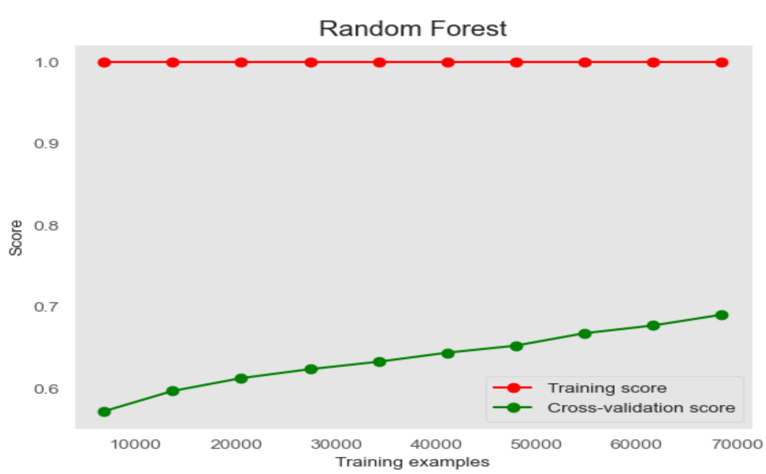
PERFORMANCE METRICS	**RANDOM FOREST** (MODEL1)
Valdation set accuracy	69.2%
Precsion	[0.60267686 0.79941981 0.6939643]
Recall	[0.55532065 0.76004728 0.76988368]
F1-score	[0.57803044 0.77923652 0.72995529]
Testing Set Accuracy	69.5%

```
The accuracy is 69.5804604417436%
[[1576  358  904]
 [ 433 1929  176]
 [ 606  126 2449]]
Precision:  [0.60267686 0.79941981 0.6939643 ]
Micro Precision:  0.695804604417436
Macro Precision:  0.6986869898150635
Weighted Precision:  0.6949660911472372

Recall score:  [0.55532065 0.76004728 0.76988368]
Micro Recall score:  0.695804604417436
Macro Recall score:  0.6950838713479213
Weighted Recall score:  0.695804604417436

F1 score:  [0.57803044 0.77923652 0.72995529]
Micro F1 score:  0.695804604417436
Macro F1 score:  0.6957407501482532
Weighted F1 score:  0.6941849311767276
```

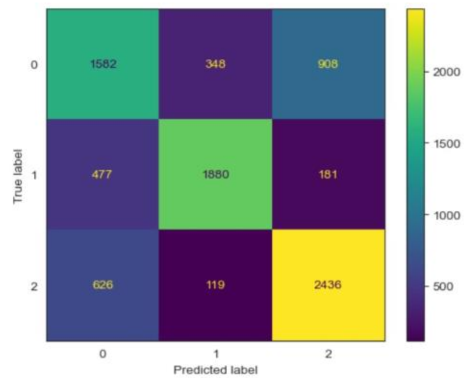


MODEL 2

PERFORMANCE METRICS	RANDOM FOREST (MODEL2)
Valdation set accuracy	69.5%
Precsion	[0.60267686 0.79941981 0.6939643]
Recall	[0.55532065 0.76004728 0.76988368]
F1-score	[0.57803044 0.77923652 0.72995529]



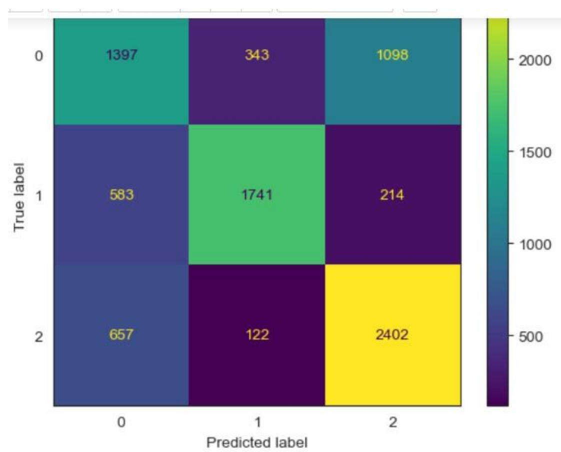
```
The accuracy is 68.9260254762183%
[[1582  348  908]
 [ 477 1880  181]
 [ 626  119 2436]]
Precision:  [0.58919926 0.80102258 0.69106383]
Micro Precision:  0.689260254762183
Macro Precision:  0.6937618889759589
Weighted Precision:  0.6898932852345979

Recall score:  [0.55743481 0.74074074 0.76579692]
Micro Recall score:  0.689260254762183
Macro Recall score:  0.6879908243991012
Weighted Recall score:  0.689260254762183

F1 score:  [0.57287706 0.76970317 0.72651357]
Micro F1 score:  0.689260254762183
Macro F1 score:  0.68969793416165
Weighted F1 score:  0.6883687523720058
```

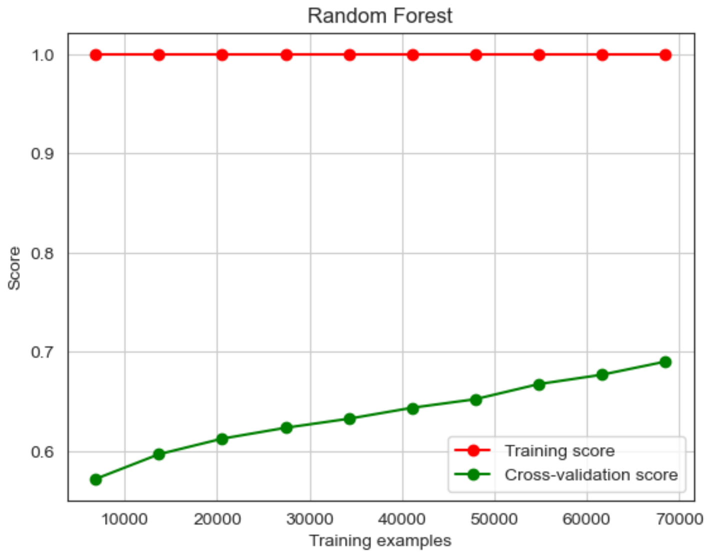


MODEL 3

PERFORMANCE METRICS	**RANDOM FOREST** (MODEL3)
Valdation set accuracy	64.6%
Precsion	[0.52976868 0.78921124 0.64674206]
Recall	[0.49224806 0.68597321 0.75510846]
F1-score	[0.51031963 0.73397976 0.69673677]

```
The accuracy is 64.74231623232441%
[[1397  343 1098]
 [ 583 1741  214]
 [ 657  122 2402]]
Precision:  [0.52976868 0.78921124 0.64674206]
Micro Precision:  0.6474231623232442
Macro Precision:  0.6552406585582532
Weighted Precision:  0.6502031225807796

Recall score:  [0.49224806 0.68597321 0.75510846]
Micro Recall score:  0.6474231623232442
Macro Recall score:  0.6444432419085132
Weighted Recall score:  0.6474231623232442

F1 score:  [0.51031963 0.73397976 0.69673677]
Micro F1 score:  0.6474231623232442
Macro F1 score:  0.6470120547959352
Weighted F1 score:  0.6459562248473526
```

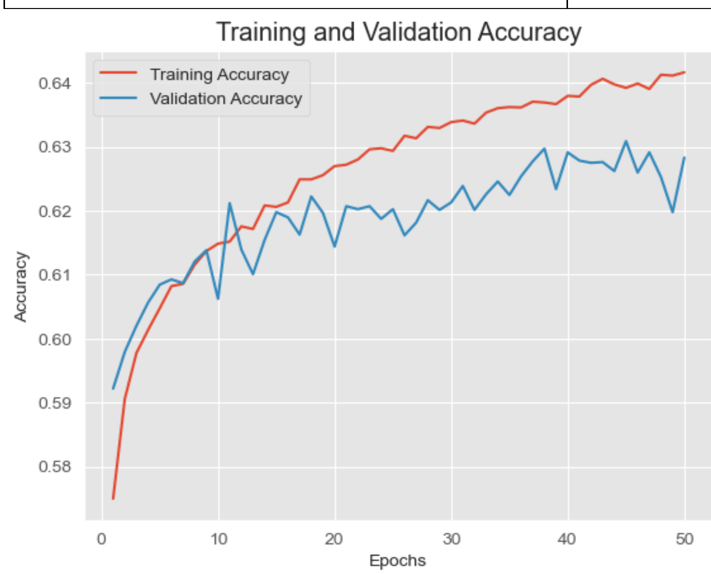




ANN

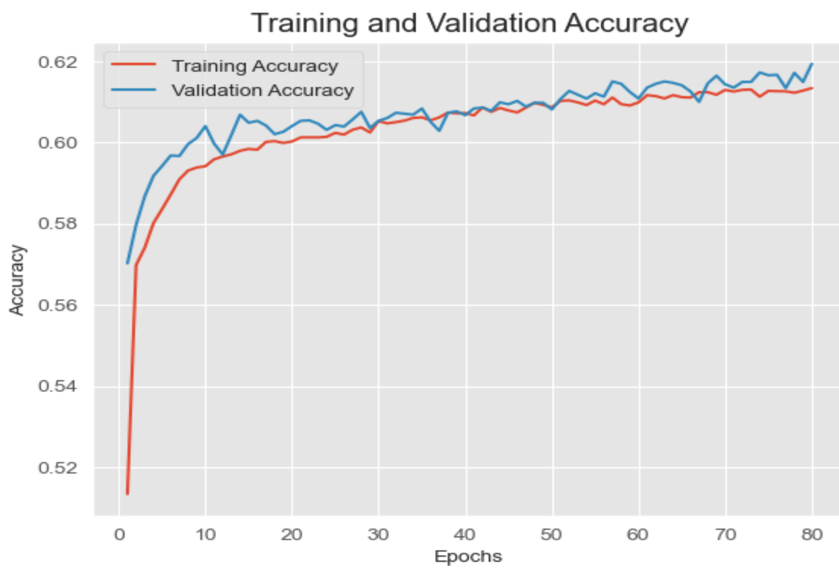
MODEL 1 (Choosing Model 1 as Best Model to predict test set)

PERFORMANCE METRICS	**ANN (**MODEL1)
Valdation set accuracy	64.6%
Precsion	68.2%
Recall	68.9%
F1-score	68.15%
Testing Set Accuracy	61.0%



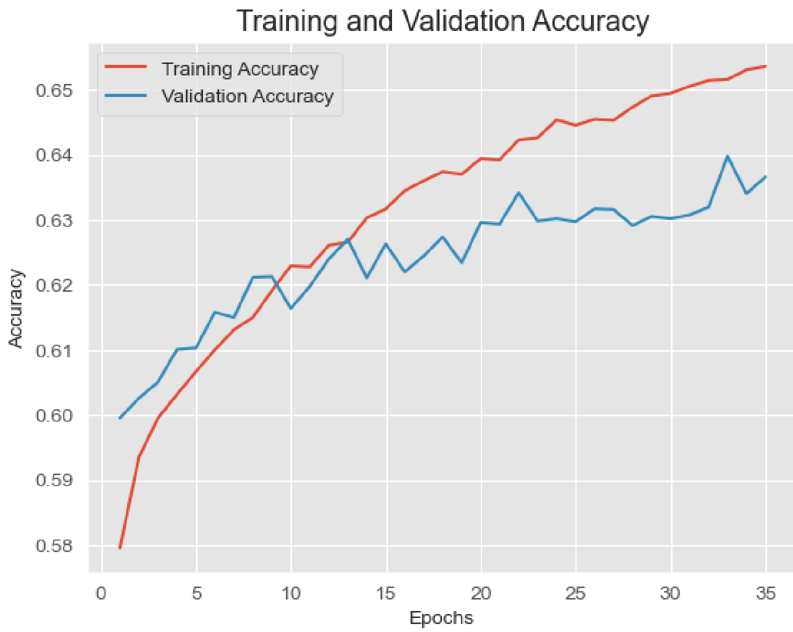
MODEL 2

PERFORMANCE METRICS	**ANN** (MODEL2)
Valdation set accuracy	62.4%
Precsion	68.2%
Recall	68.2%
F1-score	68.15%



MODEL 3

PERFORMANCE METRICS	**ANN** (MODEL3)
Valdation set accuracy	62.48%
Precsion	68.3%
Recall	68.26%
F1-score	68.15%


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epochs
0.65
0.64
0.63
0.62
0.61
0.60
0.59
0.58
0
5
10
15
20
25
30
35

PERFORMANCE OF MACHINE LEARNING ALGORITHMS

Choosing best models among three of each algorithm

PERFORMANCE METRICS	LOGISTIC REGRESSION	RANDOM FOREST	ANN
Testing set accuracy	56.5%	69.5%	61.0%
Valdation set accuracy	56.4%	69.0%	64.6%
Precsion	[0.43807763 0.64037037 0.58272407]	[0.60267686 0.79941981 0.6939643]	68.2%
Recall	[0.33403805 0.68124507 0.67651682]	[0.55532065 0.76004728 0.76988368]	68.9%
F1-score	[0.37904838 0.66017564 0.62612744]	[0.57803044 0.77923652 0.72995529]	68.15%

Comments On Performance:

Random forest is giving more accuracy than the other two because it is an ensemble learning method that combines multiple decision trees.So by aggregating the predictions of individual trees it reduces the overfitting and improves generalization

Issues of Algorithms And How We Resolve:

Overfitting: Overfitting occurs when a model learns to perform exceptionally well on the training data but fails to generalize to new, unseen data. It memorizes the noise and specific patterns of the training set, losing its ability to capture the underlying patterns in the data. To tackle overfitting, you can try the following techniques:

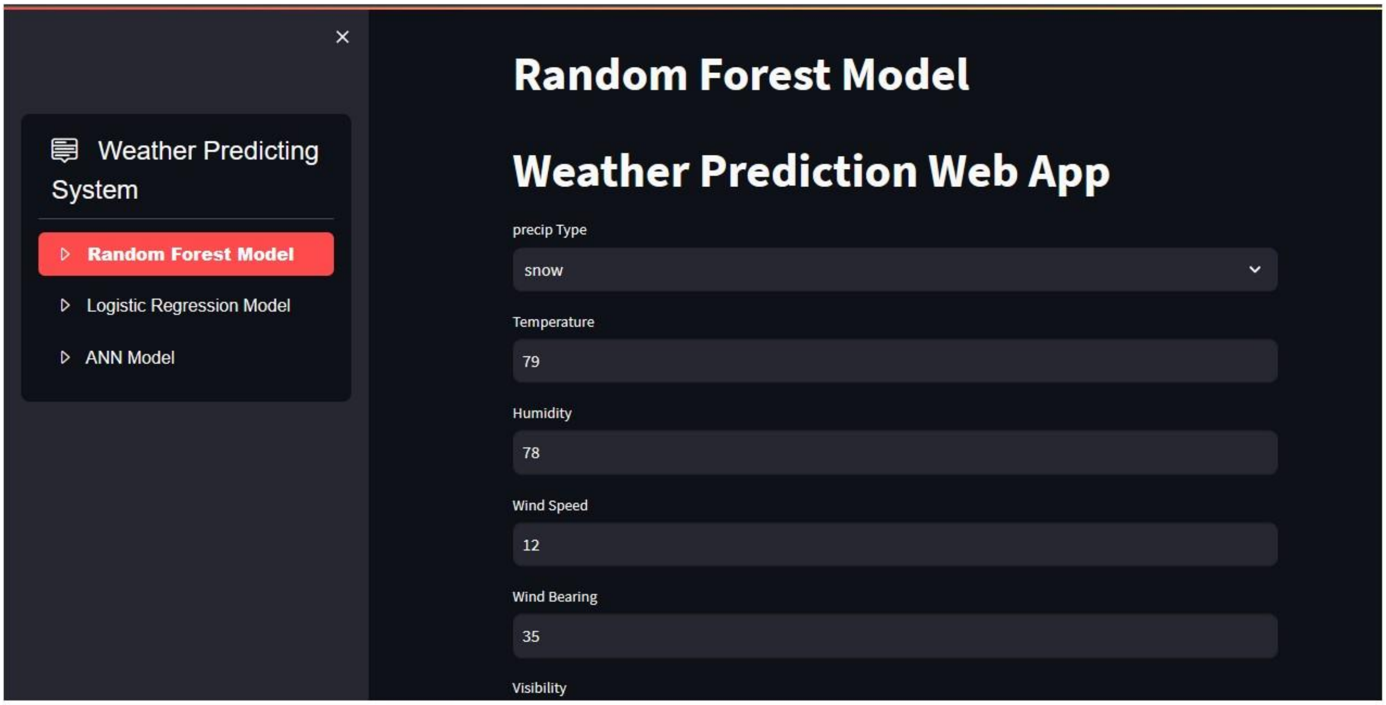
- Feature Selection/Extraction: Careful feature selection is crucial in reducing overfitting. So, we have removed irrelevant or redundant features to focus on the most informative ones.
- Early Stopping: To reduce overfitting we can use Early stopping technique In ANN . Early stopping is a regularization technique commonly used in training Artificial Neural Networks (ANNs) to prevent overfitting and improve generalization. Overfitting occurs when a model performs very well on the training data but fails to generalize well to unseen data. Early stopping helps prevent this by monitoring the model's performance during training and stopping the training process when the performance on a validation dataset starts to degrade.

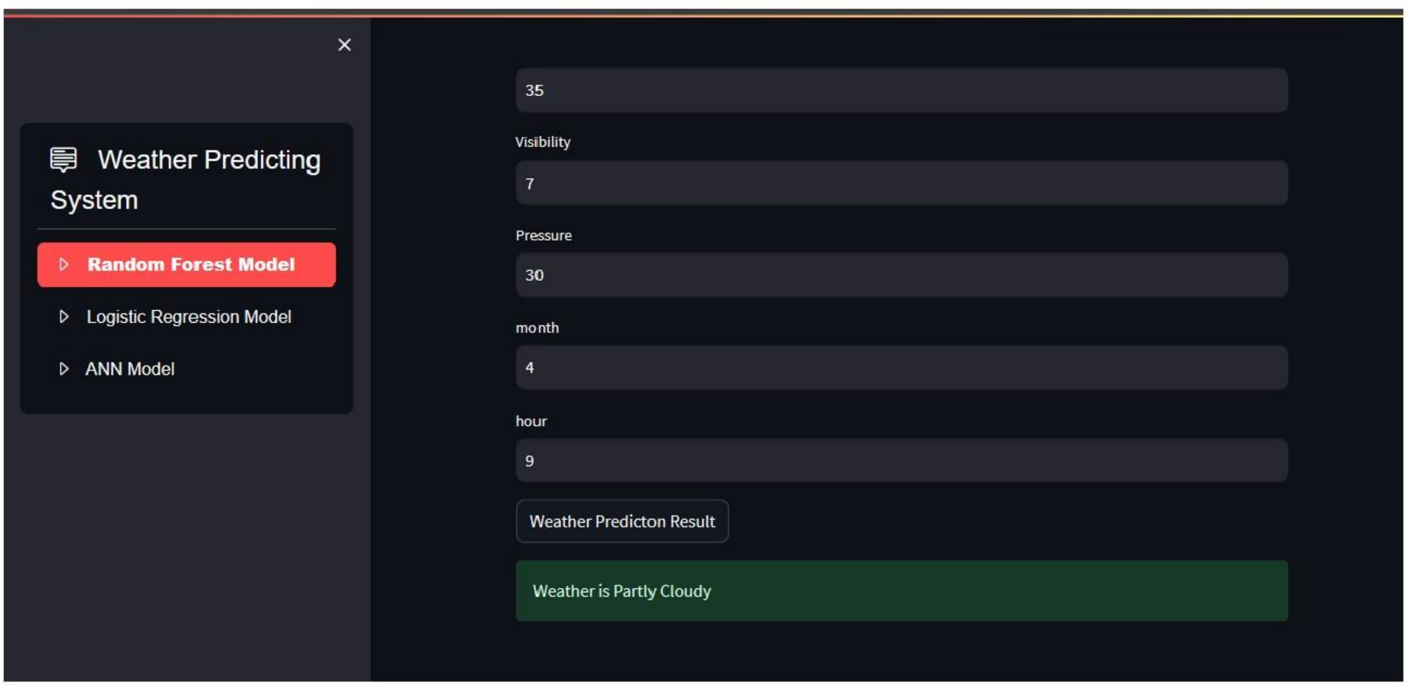
- Regularization: Introduced penalties on complex model parameters during training to prevent them from becoming too large. Common regularization techniques include L1 (Lasso) and L2 (Ridge) regularization.
- Limiting Tree Depth: One way to prevent overfitting is to limit the depth of the individual decision trees in the Random Forest. So we have achieved by setting a maximum depth (max_depth) for each tree during model training.
- Dropout: In neural networks, We apply dropout regularization, where random neurons are temporarily dropped during training, forcing the network to learn more robust features.
- Ensemble Methods: Combine predictions from multiple models (e.g., Random Forest, Gradient Boosting) to reduce overfitting and enhance generalization.

Underfitting:

Underfitting occurs when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both training and test data. To address underfitting, try the following methods:

- Feature Engineering: We make Ensure that we are using relevant features and that the data is appropriately prepared for training. .
- Hyperparameter Tuning: Adjusted hyperparameters like learning rate, number of layers, and units to find the right balance between simplicity and complexity.
- Check for Data Mismatch: WE ensue that training data represents the same distribution as your test data. A data mismatch can lead to underfitting.


Weather Predicting System
Random Forest Model
Logistic Regression Model
ANN Model
Random Forest Model
Weather Prediction Web App
precip Type
snow
Temperature
79
Humidity
78
Wind Speed
12
Wind Bearing
35
Visibility


Weather Predicting System
Random Forest Model
Logistic Regression Model
ANN Model
35
Visibility
7
Pressure
30
month
4
hour
9
Weather Predicton Result
Weather is Partly Cloudy