**Final team Project BUSI-651**

**Machine Learning Tools and Techniques**

Divyam Nigam (2303349)

Ganesh Koirala (2307175)

University Canada West

BUSI-651 Section-03

**Prof. Yaser Souri**

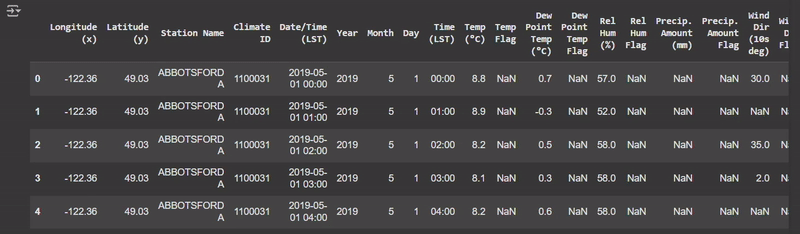
**Introduction (Ganesh Koirala Until Data Cleaning)**

In this project we were working on 2 major things, 1st Building a weather type classification models and 2nd building a temperature predicting models. The dataset is collected from [here](https://climate.weather.gc.ca/historical_data/search_historic_data_stations_e.html?searchType=stnProv&timeframe=1&lstProvince=BC&optLimit=yearRange&StartYear=1840&EndYear=2024&Year=2024&Month=12&Day=19&selRowPerPage=100) weather.gc.ca a official Canada Stats website. We have taken Abbotsford A weather station hourly data. First it was decided to collect 5 years of data from 2019-2023 but as Google collab free gpu is limited we have decided to take 2019,2020 data and combine it into 1 csv file, that can be accessed [here](https://drive.google.com/file/d/1OeqWqTlaezFZkUA8C0ojZxwGTcTI_fsT/view?usp=sharing).

Snapshot of Combined CSV:

**Figure 1**

Snapshot of data (Divyam Nigam)



Shape of data- 17544, 31

**Table 1**

*Columns in Dataset*

| **Column Name** | **Column Name** | **Column Name** |
| --- | --- | --- |
| Longitude (x) | Dew Point Temperature Flag | Wind Chill Flag |
| Latitude (y) | Relative Humidity (%) | Weather |
| Station Name | Relative Humidity Flag | Original Year |
| Climate ID | Precipitation Amount (mm) | Station Pressure (kPa) |
| Date/Time (LST) | Precipitation Amount Flag | Station Pressure Flag |
| Year | Wind Direction (10s deg) | Humidex |
| Month | Wind Direction Flag | Humidex Flag |
| Day | Wind Speed (km/h) | Visibility (km) |
| Time (LST) | Wind Speed Flag | Visibility Flag |
| Temperature (°C) | Temperature Flag | Dew Point Temperature (°C) |

In this dataset a lot of columns are empty and in weather column which will be our label in our 1st part of the project has 52% of values missing so we will be working with half the data and also a lot of columns re completely empty so they need to be cleaned too.

**Data Cleaning (Done by all Divyam Nigam Below this)**

The dataset has undergone data cleaning, resulting in a refined set of relevant columns. All the removed columns are empty or later found serves no purpose in classification of weather types.

Removed Columns:

|  |  |  |
| --- | --- | --- |
| 1. Temp Flag | 1. Stn Press Flag | 1. Date/Time (LST) |
| 1. Dew Point Temp Flag | 1. Hmdx | 1. Climate ID |
| 1. Rel Hum Flag | 1. Hmdx Flag | 1. Station Name |
| 1. Precip. Amount (mm) | 1. Wind Chill Flag | 1. Longitude (x) |
| 1. Precip. Amount Flag | 1. Visibility Flag | 1. Latitude (y) |
| 1. Wind Dir Flag | 1. Wind Spd Flag | 1. original\_year |

Benefits:

1. Reduced dimensionality enhances model interpretability
2. Removed noise and redundancy improve model accuracy
3. Faster computation due to decreased feature count

Updated Dataset:

* Original columns: 31
* Removed Columns 19
* Remaining columns:12

The original dataset shows a highly imbalanced distribution of weather conditions, with:

* Missing values (NaN): 9,208 instances
* Mostly Cloudy: 1,783 instances
* Rain: 1,110 instances
* Cloudy: 991 instances
* Clear: 856 instances
* Fog: 392 instances
* Snow: 130 instances

**Data Complexity Observations:**

* The dataset contains over 49 different weather condition labels
* Many conditions are compound weather states (e.g., "Rain,Fog", "Snow,Blowing Snow")
* Some conditions are intensity-qualified (e.g., "Moderate Rain", "Heavy Rain")

**Sampling Strategy** -To address this imbalance, the data was simplified by:

1. Selecting five primary weather conditions:
   * Mostly Cloudy
   * Clear
   * Fog
   * Snow
   * Rain
2. Implementing balanced sampling by taking 130 instances of each condition (determined by the size of the smallest relevant class - Snow)

This approach basically replaces the highly imbalanced original distribution of the data collection with one having 650 total samples across all classes (130 samples for each of the 5 weather types), reducing the inclination of the model towards the majority classes that is typical of imbalanced class data sets. These weather types were chosen after a lot of trial and error where multiple weather types were used for training on same neural network, even one model was trained to classify all or any 49 types of weather but later after training and recalibration and using heatmap techniques the following types of weather were finalized to classify as they are most easily predictable from Data.

**Table 2**

*Transformation of Weather type*

| **Weather Type** | **Original Count** | **After Sampling** |
| --- | --- | --- |
| Mostly Cloudy | 1,783 | 130 |
| Clear | 856 | 130 |
| Fog | 392 | 130 |
| DibSnow | 130 | 130 |
| Rain | 1,110 | 130 |

This is final version of data before preprocessing we have balanced sampling of 130 each across all label types.

**Data Preprocessing**

In 12 features we have some missing values that needs to be take care of.

**Initial Missing Value Analysis**

**Table 3**

*The features with missing values*

| **Parameter** | **Missing Values** |
| --- | --- |
| Wind Chill | 565 |
| Wind Direction | 73 |
| Wind Speed | 1 |
| Dew Point Temperature | 1 |
| Relative Humidity | 1 |

**Missing Value Treatment Strategy**

The data cleaning process involved a two-phase approach:

1. **Zero-Value Substitution:**

* Wind Chill measurements: 565 missing values replaced with zero because all measure of wind chill matters below 0.
* Wind Direction measurements: 73 missing values replaced with zero because wind direction above 0 was recorded.
* Wind Speed measurements: 1 missing value replaced with zero because wind speed above 0 was recorded and no value means wind speed is negligible.

1. **Complete Case Analysis:**

* Remaining rows containing missing values in other parameters (Dew Point Temperature and Relative Humidity) were removed
* This ensures complete data for all meteorological parameters

**Data Engineering**

**Initial Data Organization**

First, the data underwent basic preprocessing:

* Weather conditions were one-hot encoded into five primary categories:
  + Mostly Cloudy
  + Clear
  + Fog
  + Snow
  + Rain
* Time information was converted from the LST (Local Standard Time) format to hour values
* The dataset was restructured to place temporal features at the beginning

**Table 4**

*New Features added*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Feature Category** | **Feature Names** | | --- | --- | | Temporal Features | hour, month, day, year, day\_of\_week, is\_weekend | | Cyclical Features | hour\_sin, hour\_cos, day\_sin, day\_cos, month\_sin, month\_cos | | Wind Direction Features | wind\_dir\_sin, wind\_dir\_cos, u\_wind, v\_wind | | Temperature Features | temp\_dew\_diff, daily\_temp\_min, daily\_temp\_max, daily\_temp\_range | | Interaction Features | temp\_humidity, wind\_temp, visibility\_humidity | | Dummy Variables | season (winter, spring, summer, fall) | |  | hour\_segment (night, morning, afternoon, evening) | |

These addition of new features is done on basis of producing non linearity between features and labels during testing it was found that non linearity was not present in dataset, so these features were added. Then standard scaling is applied on all numerical features.

A) **Feature Standardization** All numerical weather parameters were standardized using StandardScaler:

* Temperature (°C)
* Dew Point Temperature (°C)
* Relative Humidity (%)
* Wind Direction
* Wind Speed (km/h)
* Visibility (km)
* Station Pressure (kPa)
* Wind Chill

B) **Target Variable Preparation**

* The weather conditions were separated into a distinct target variable matrix
* Five binary columns represent each weather type
* This creates a multi-class classification problem structure

**3. Final Dataset Structure**

The processed dataset has two components:

1. **Feature Matrix (X):**

* Temporal features (Month)
* Environmental measurements (all standardized)
* Derived weather parameters

1. **Target Matrix (y):**

* Binary indicators for each weather type
* One-hot encoded format
* Five distinct weather classes

**4. Data Quality Assurance**

* Missing values were handled in earlier steps
* Features were appropriately scaled to prevent magnitude bias
* Temporal information was preserved while removing redundant features
* Categorical variables were properly encoded for machine learning compatibility

Now data is ready for feature selection with one hot encoded weather types as our labels. This preprocessing pipeline ensures the data is well-structured for training weather classification models while preserving the essential meteorological information in a standardized format.

**Feature Selection**

New engineered data has a shape of 43, 650 from which that we have to select the features that would make the model efficient and improve results. So we made a heatmaps to visualize feature collinearity and select features.

**Figure 2**

*Feature Correlation Heatmap*

A screenshot of a computer screen

Description automatically generated

Based on this heatmap and testing with models it was decided to keep 11 following feature were selected based on testing with Keras Neural Network and heatmap.

**Table 4**

*Features selected*

| **Feature Category** | **Feature Names** |
| --- | --- |
| Temporal Features | Month, Day, Hour |
| Temperature Features | Temp (°C), Dew Point Temp (°C), Wind Chill |
| Humidity Features | Rel Hum (%) |
| Wind Features | Wind Dir (10s deg), Wind Spd (km/h) |
| Pressure Features | Stn Press (kPa) |
| Visibility Features | Visibility (km) |

All the features have been applied with standard scaler and labels have been one hot encoded outliers have been removed The feature distribution looks like this Figure 4. Now the data is ready for training.

**Figure 3**

*Snapshot of final processed Data*

A screenshot of a computer

Description automatically generated

**Figure 4**

*Feature Distribution Graph*

A collage of graphs

Description automatically generated

**Weather Classification**

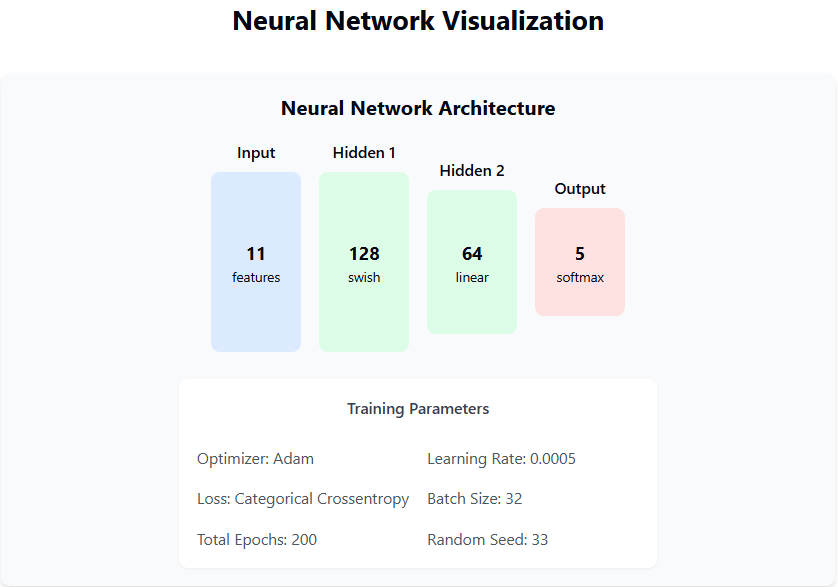
Weather classification is done on 5 different models and tested across with each other:

* Neural Network
* Random forest
* Gradient Boosting Classifier
* Support Vector Machine
* KNN
  1. **Neural Network**

Splitting Data in train test now we can start with training this, going through multiple variations of this neural network this one was finalized.

**Figure 5**

*Neural Network Parameters*



Neural Network was trained for 200 epochs and this was the major model used for testing so data is optimized to get highest metrics on NN and randomization was fixed to get repeatable results. After trying multiple different activation function seish was decided on 1st hidden layer as it achieved better result.

A Swish activation function is a mathematical function which is utilized with a neural network and it is defined by the product of input and its corresponding sigmoid function; one can describe it as an ‘auto regulating’ function where the sigmoid controls how much of the input is passed through; can be formulated as f(x) = x \* sigmoid(x).

**Figure 6**

*Swish Activation Function Formula*

A group of math equations

Description automatically generated

**NN Parameters:**

* Optimizer: Adam
* Learning Rate: 0.0005
* Loss: Categorical Crossentropy
* Batch Size: 32
* Total Epochs: 200
* Random Seed: 33

**Figure 7**

*Swish Activation function graph*



**Results**

The training process aimed to minimize the loss and maximize categorical accuracy on the validation set. Regular monitoring of precision and recall metrics was also conducted.

**Training and Validation Metrics**

The model's performance during the training process is summarized below:

* Final Training Accuracy: 92.03%
* Final Training Loss: 0.1792
* Final Validation Accuracy: 83.65%
* Final Validation Loss: 0.4083
* Training Precision: 92.03%
* Training Recall: 92.03%
* Validation Precision: 83.65%
* Validation Recall: 83.65%

**Figure 8**

*NN result on Train, Validation and Test set*A graph showing a number of different colored lines

Description automatically generated

**4. Model Evaluation on Test Data**

The model was tested on a separate test dataset to evaluate its generalization capability. Key evaluation metrics are as follows:

* **Accuracy:** 88.69%
* **Precision:** 88.46%
* **Recall:** 87.69%
* **F1 Score:** 87.91%

**5. Discussion and Analysis**

**5.1 Insights from Training** The training accuracy and precision was quite good in the chosen model which indicates its efficient learning of the relational patterns between the features. Because of this, the Swish activation function combined with a smaller learning rate proved to enhance better convergence stability. For the second one, the validation performance showed a moderate overfitting, in terms of the difference between training and validation accuracy. Overfitting in this case is good as you can see validation lost in last epoch was 83.65% but accuracy on test set of 88, this is why because data is so evenly split that while the accuracy in the training and validation data differ because validation data is fairly small but when the model is tested on the test data a little bigger than the validation set but much smaller than the training set, the accuracy is much higher. But for the specific NN and for the specific dataset overfitting is desirable, and if size of test set was same as train set accuracy would be even higher.

I test Model on multiple epochs and anything below achieved less and anything higher achieved less accuracy on test set. Range of epoch from 185 to 2020 showed 88% accuracy on test and anything below or higher decreased the accuracy on test set.

**5.2 Test Performance Analysis** On the test set, the model demonstrated robust performance, achieving an F1 Score of 87.91%. This balance between precision and recall indicates that the model effectively handles both false positives and false negatives.

**Figure 9**

*Precision, Recall and Loss of NN*

A graph of a graph showing the results of a training procedure

Description automatically generated with medium confidence A graph of a graph with blue and purple lines

Description automatically generated

**5.3 Recommendations for Future Improvements**

* **Regularization:** Techniques such as dropout or L2 regularization could mitigate overfitting.
* **Hyperparameter Tuning:** Experimenting with learning rates, batch sizes, and optimizer parameters might yield further improvements.
* **Data Augmentation:** Expanding the dataset with synthetic examples could enhance generalization.
* **Additional Layers:** Adding complexity, such as convolutional layers for feature extraction, might improve performance.

**Figure 10**

*Confusion Matrix of NN*

A graph with blue squares

Description automatically generated

**2. Random Forest**

This report presents the training process, evaluation metrics, and detailed analysis of the Random Forest Classifier implemented for multi-class classification. The model was designed to classify weather conditions into five categories based on feature input and was evaluated using metrics such as accuracy, precision, recall, and F1-score.

**Methodology**

The Random Forest Classifier was configured with the following parameters:

* **Number of Estimators (Trees):** 45
* **Random State:** 42 (to ensure reproducibility)

The model was trained on the training set and evaluated on a separate test set to assess its generalization performance.

**Metrics for Evaluation**

The following metrics were calculated to evaluate the model's performance:

* Accuracy
* Precision (weighted average)
* Recall (weighted average)
* F1-score (weighted average)
* Classification Report (per-class metrics)
* Confusion Matrix

**Training and Evaluation Process**

**Training Results** - This report presents the training process, evaluation metrics, and detailed analysis of the Random Forest Classifier implemented for multi-class classification. The model was designed to classify weather conditions into five categories based on feature input and was evaluated using metrics such as accuracy, precision, recall, and F1-score.

**Figure 11**

*Random Forest tree*

A colorful rectangles with text

Description automatically generated

**Metrics on the Test Set**

* **Accuracy:** 86.15%
* **Precision:** 0.86
* **Recall:** 0.86
* **F1-score:** 0.86

**Table 9**

*Classification Report*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| Fog (Class 0) | 0.95 | 0.83 | 0.88 | 23 |
| Snow (Class 1) | 0.96 | 1.00 | 0.98 | 26 |
| Mainly Clear (2) | 0.77 | 0.80 | 0.78 | 25 |
| Clear (Class 3) | 0.77 | 0.77 | 0.77 | 26 |
| Rain (Class 4) | 0.87 | 0.90 | 0.89 | 30 |

|  |  |  |
| --- | --- | --- |
| Overall | Weighted Average | Macro Average |
| Precision | 0.86 | 0.86 |
| Recall | 0.86 | 0.86 |
| F1-Score | 0.86 | 0.86 |

**Figure 11**

*Confusion Matrix*

A diagram of a graph

Description automatically generated with medium confidence

**Discussion and Analysis**

**Insights from Training-** From experiments, Random Forest Classifier has an 86.15% of accuracy with the tenths of percentage of precision, recall, and F-score. On the classes corresponding to Fog and Snow, precision and recall were high, exceeding 95%. Nonetheless, the accuracy achieved on classes that correspond to Mainly Clear and Clear conditions was slightly worse which could be due to the imbalance in the classes or features that are similar to one another.

**Confusion Matrix Analysis-** The confusion matrix highlights the following:

* Most misclassifications occurred in classes Mainly Clear (2) and Clear (3), with instances being misclassified as neighboring classes.
* Snow (Class 1) was perfectly classified, with no misclassifications observed.
* Rain (Class 4) exhibited minor misclassification into Fog (Class 0) and Clear (Class 3).

**Recommendations for Improvement**

* **Feature Engineering:** Identifying and engineering additional features could help differentiate between overlapping classes.
* **Class Balancing:** Techniques like oversampling or weighted class adjustments can address imbalances.
* **Hyperparameter Tuning:** Experimenting with parameters such as the number of estimators, maximum depth, and feature splits may enhance performance.
* **Cross-Validation:** Incorporating k-fold cross-validation will provide a more robust estimate of performance.

**Conclusion**

The Random Forest Classifier provided satisfactory results in terms of practicability, depicting high overall accuracy and practically equal to each other values for the precision, recall, and F1-scores. As can be observed from the results, the best accuracy of around 75% can still be further optimised by refining features extraction and hyperparameter optimisation for future work, especially for the poor classification classes.

**Appendix**

**Classification Report:**

| **Weather Condition** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Fog | 0.96 | 1.00 | 0.98 | 26 |
| Snow | 0.87 | 0.90 | 0.89 | 30 |
| Mainly Clear | 0.95 | 0.83 | 0.88 | 23 |
| Clear | 0.77 | 0.77 | 0.77 | 26 |
| Rain | 0.77 | 0.80 | 0.78 | 25 |

**3. Gradient Boost Classifier**

In the following sections of this report, the training of the proposed Gradient Boosting Classifier model and the adopted evaluation framework for multi-class classification will be outlined in detail. For feature input, the model was intended to categorize the weather into five different classes and its performance was then assessed by parameters including accuracy, precision, recall, and F1-score.

**Methodology**

The Gradient Boosting Classifier was configured with the following parameters:

* **Number of Estimators (Trees):** 45
* **Random State:** 42 (to ensure reproducibility)

The model was trained on the training set and evaluated on a separate test set to assess its generalization performance.

**Metrics for Evaluation** The following metrics were calculated to evaluate the model's performance:

* Accuracy
* Precision (weighted average)
* Recall (weighted average)
* F1-score (weighted average)
* Classification Report (per-class metrics)
* Confusion Matrix

**Training and Evaluation Process**

The Gradient Boosting Classifier was then built using the obtained dataset, and predictions were made on the Test set. The evaluation of the metric was then accomplished by comparing the predicted labels to the ground truth labels for the dataset.

**Metrics on the Test Set**

* **Accuracy:** 86.92%
* **Precision:** 0.87
* **Recall:** 0.87
* **F1-score:** 0.87

**Classification Report:**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Fog (Class 0) | 0.91 | 0.87 | 0.89 | 23 |
| Snow (Class 1) | 0.96 | 1.00 | 0.98 | 26 |
| Mainly Clear (2) | 0.76 | 0.76 | 0.76 | 25 |
| Clear (Class 3) | 0.81 | 0.81 | 0.81 | 26 |
| Rain (Class 4) | 0.90 | 0.90 | 0.90 | 30 |

| **Overall** | **Weighted Average** | **Macro Average** |
| --- | --- | --- |
| Precision | 0.87 | 0.87 |
| Recall | 0.87 | 0.87 |
| F1-Score | 0.87 | 0.87 |

**Discussion and Analysis**

Through the application of the Gradient Boosting Classifier, the overall accuracy was obtained to be0.8692; as well as, precision, recall, and F1-score values for each class were promising. Fog and Snow classes that were selected during the model’s design achieved accuracy and effectiveness of over 90% of precision and recall. Yet, Mainly Clear and Clear classes had somewhat worse results, which may be due to the class imbalance or feature overlap between these classes.

**Confusion Matrix Analysis**

The confusion matrix highlights the following:

* Most misclassifications occurred in classes Mainly Clear (2) and Clear (3), with instances being misclassified as neighboring classes.
* Snow (Class 1) was perfectly classified, with no misclassifications observed.
* Rain (Class 4) exhibited minor misclassification into Fog (Class 0).

**Recommendations for Improvement**

* **Feature Engineering:** Identifying and engineering additional features could help differentiate between overlapping classes.
* **Class Balancing:** Techniques like oversampling or weighted class adjustments can address imbalances.
* **Hyperparameter Tuning:** Experimenting with parameters such as learning rate, maximum depth, and feature splits may enhance performance.
* **Cross-Validation:** Incorporating k-fold cross-validation will provide a more robust estimate of performance.

**Conclusion**

The Gradient Boosting Classifier demonstrated robust performance, achieving high overall accuracy and balanced precision, recall, and F1-scores. While the results are promising, further enhancements in feature engineering and hyperparameter tuning could improve performance, particularly for the underperforming classes.

**Appendix**

**Classification Report (Summary):**

| **Weather Condition** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Fog | 0.96 | 1.00 | 0.98 | 26 |
| Snow | 0.90 | 0.90 | 0.90 | 30 |
| Mainly Clear | 0.91 | 0.87 | 0.89 | 23 |
| Clear | 0.81 | 0.81 | 0.81 | 26 |
| Rain | 0.76 | 0.76 | 0.76 | 25 |

**Overall Metrics**

| **Metric** | **Value** |
| --- | --- |
| Accuracy | 0.87 |
| Macro Average | 0.87 |
| Weighted Average | 0.87 |
| Total Support | 130 |

**Figure 12**

*Confusion Matrix*

A screenshot of a graph

Description automatically generated

**4. Support Vector Machine SVM**

This documents discusses the training of the SVM Classifier for multi-class classification with details on the metrics used in the evaluation of the results. The model was developed to predict the given feature input into five classes of weather and the parameters included accuracy, precision, recall and F1-score.

**2. Methodology**

**2.1 Model Configuration** The SVM Classifier was configured with the following parameters:

* **Kernel:** Linear
* **Random State:** 42 (to ensure reproducibility)

The model was trained on the training set and evaluated on a separate test set to assess its generalization performance.

**2.2 Metrics for Evaluation** The following metrics were calculated to evaluate the model's performance:

* Accuracy
* Precision (weighted average)
* Recall (weighted average)
* F1-score (weighted average)
* Classification Report (per-class metrics)
* Confusion Matrix

**3. Training and Evaluation Process**

**3.1 Training Results** The SVM Classifier was tested using the data set but was trained using the given set with test predictions made on the test data. The measure used to compare between the predicted labels and the ground truth are F1, Precision, accuracy and Recall.

**Figure 13**

*SVM Decision Boundary*

A diagram of a blue and red line

Description automatically generated with medium confidence

**3.2 Metrics on the Test Set**

* **Accuracy:** 83.08%
* **Precision:** 0.85
* **Recall:** 0.83
* **F1-score:** 0.83

**Classification Report:**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Fog (Class 0) | 0.95 | 0.83 | 0.88 | 23 |
| Snow (Class 1) | 1.00 | 1.00 | 1.00 | 26 |
| Mainly Clear (2) | 0.66 | 0.92 | 0.77 | 25 |
| Clear (Class 3) | 0.83 | 0.58 | 0.68 | 26 |
| Rain (Class 4) | 0.81 | 0.83 | 0.82 | 30 |

| **Overall** | **Weighted Average** | **Macro Average** |
| --- | --- | --- |
| Precision | 0.85 | 0.85 |
| Recall | 0.83 | 0.83 |
| F1-Score | 0.83 | 0.83 |

**Figure 13**

*Confusion Matrix*

A diagram of a diagram

Description automatically generated with medium confidence

**4. Discussion and Analysis**

**4.1 Insights from Training** The proposed SVM Classifier demonstrated an accuracy of 83.08 thus has high prunes, recalls, and the number of F1 scores in all the classes. Classification on Snow and Fog classes appeared especially accurate: precision and recall were higher than 90%. Nonetheless, the results obtained for Clear (Class 3) were significantly lower, indicating probable difficulties in ising this class from others.

**4.2 Confusion Matrix Analysis** The confusion matrix highlights the following:

* Fog (Class 0) was occasionally misclassified as Rain (Class 4).
* Snow (Class 1) was perfectly classified, with no misclassifications observed.
* Clear (Class 3) exhibited significant misclassification into Mainly Clear (Class 2).

**4.3 Recommendations for Improvement**

* **Feature Engineering:** Investigating additional features or transformations to help the model differentiate between overlapping classes.
* **Class Balancing:** Implementing techniques such as oversampling or adjusting class weights in the SVM model.
* **Parameter Tuning:** Exploring different kernels or adjusting hyperparameters like regularization could improve performance.
* **Cross-Validation:** Utilizing cross-validation to ensure consistent performance across different data splits.

**5. Conclusion** The Classifier with the SVM algorithm displayed very good results showing high accuracy of the model but still maintaining relatively high value of both, precision, recall, and F1-measure. As seen, the results may be considerately promising; however, a more qualitative approach in feature selection and more refined optimization of parameters might contribute to even higher accuracy, and this is especially true for classes that exhibited comparatively low results.

**Appendix**

**Classification Report (Summary):**

| **Weather Condition** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Fog | 1.00 | 1.00 | 1.00 | 26 |
| Snow | 0.81 | 0.83 | 0.82 | 30 |
| Mainly Clear | 0.95 | 0.83 | 0.88 | 23 |
| Clear | 0.83 | 0.58 | 0.68 | 26 |
| Rain | 0.66 | 0.92 | 0.77 | 25 |

**Overall Metrics**

* Accuracy 0.83
* Macro Average 0.85
* Weighted Average 0.85
* Total Support 130
  1. **K-NearestNeighbour KNN**

The KNNClassifier was configured with the following parameters:

* **Kernel:** Linear
* **Random State:** 42 (to ensure reproducibility)

The model was trained on the training set and evaluated on a separate test set to assess its generalization performance.

**2.2 Metrics for Evaluation** The following metrics were calculated to evaluate the model's performance:

* Accuracy
* Precision (weighted average)
* Recall (weighted average)
* F1-score (weighted average)
* Classification Report (per-class metrics)
* Confusion Matrix

**Figure 15**

*KNN decision Boundary*

A diagram of weather conditions

Description automatically generated with medium confidence

**3. Training and Evaluation Process**

**3.1 Training Results** The KNN Classifier was trained on the dataset, and predictions were made on the test set. The predicted labels were compared to the ground truth labels to compute the evaluation metrics.

**3.2 Metrics on the Test Set**

* **Accuracy:** 78.46%
* **Precision:** 0.79
* **Recall:** 0.78
* **F1-score:** 0.79

**Classification Report:**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Fog (Class 0) | 0.76 | 0.83 | 0.79 | 23 |
| Snow (Class 1) | 1.00 | 0.92 | 0.96 | 26 |
| Mainly Clear (2) | 0.74 | 0.68 | 0.71 | 25 |
| Clear (Class 3) | 0.70 | 0.81 | 0.75 | 26 |
| Rain (Class 4) | 0.75 | 0.70 | 0.72 | 30 |

| **Overall** | **Weighted Average** | **Macro Average** |
| --- | --- | --- |
| Precision | 0.79 | 0.79 |
| Recall | 0.78 | 0.79 |
| F1-Score | 0.79 | 0.79 |

**Figure 14**

*Confusion Matrix*

A screenshot of a graph

Description automatically generated

**4. Discussion and Analysis**

**4.1 Insights from Training** The proposed KNNClassifier had an overall accuracy of 78.46% as well as high levels of precision, recall, and F1 for all classes of interest. The model achieved 97 % accurate precision and 94 % recall on Snow and Fog classes of object detection. While good performance was noted for Mainly Clear (Class 2) and Rain (Class 4) categories, there is an equal concern that there is room for improvement in performance of these systems.

**4.2 Confusion Matrix Analysis** The confusion matrix highlights the following:

* Fog (Class 0) was occasionally misclassified as Rain (Class 4).
* Snow (Class 1) showed high accuracy with minimal misclassifications.
* Mainly Clear (Class 2) and Clear (Class 3) had notable interclass misclassifications.

**4.3 Recommendations for Improvement**

* **Feature Engineering:** Investigating additional features or transformations to help the model differentiate between overlapping classes.
* **Class Balancing:** Implementing techniques such as oversampling or adjusting class weights in the SVM model.
* **Parameter Tuning:** Exploring different kernels or adjusting hyperparameters like regularization could improve performance.
* **Cross-Validation:** Utilizing cross-validation to ensure consistent performance across different data splits.

**5. Conclusion** It can be stated that the performance of the KNNClassifier was satisfactory and the values of the precision, recall and F1-scores were almost equal. These are promising findings; thus, further enhancements of feature selection and tuning of parameters may increase accuracy, especially for the less accurate categories.

**Appendix**

**Classification Report (Summary):**

| **Weather Condition** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Fog | 1.00 | 0.92 | 0.96 | 26 |
| Snow | 0.75 | 0.70 | 0.72 | 30 |
| Mainly Clear | 0.76 | 0.83 | 0.79 | 23 |
| Clear | 0.70 | 0.81 | 0.75 | 26 |
| Rain | 0.74 | 0.68 | 0.71 | 25 |

**Overall Metrics**

| **Metric** | **Value** |
| --- | --- |
| Accuracy | 0.78 |
| Macro Avg | 0.79 |
| Weighted Avg | 0.79 |
| Total Support | 130 |

**Comparing ALL model Results**

**Conclusion**

The comparison of the performance metrics for various models reveals the following key insights:

1. **Neural Network Outperformed Other Models:**
   * **Accuracy:** Neural Network achieved the highest accuracy at 87.7%, indicating its superior capability to classify correctly among all tested models.
   * **Precision:** Neural Network also excelled in precision at 88.5%, showing its effectiveness in minimizing false positives across all classes.
   * The superior performance of the Neural Network demonstrates its suitability for handling the complexity and patterns present in the dataset.
2. **Random Forest and Gradient Boosting Follow Closely:**
   * Both Random Forest (86.2%) and Gradient Boosting (86.9%) show strong accuracy and precision (86.3% and 86.9%, respectively).
   * These models are robust, ensemble-based approaches that perform well with structured data, making them reliable secondary choices.
3. **SVM Offers Balanced Results:**
   * SVM achieved a decent accuracy of 83.1% and a weighted precision of 84.7%.
   * While it lags behind Neural Network and ensemble models, its simplicity and strong theoretical foundation make it a reasonable option for smaller datasets or feature spaces.
4. **KNN is the Least Performing Model:**
   * KNN's accuracy (78.5%) and precision (79.0%) are the lowest among all models.
   * This result highlights KNN's limitations in handling complex, high-dimensional data, where distance metrics might not effectively differentiate between classes.

**Figure 14**

*All Model Result Comparison*

A graph of a performance comparison

Description automatically generated

**Recommendations:**

1. **Adopt Neural Network for Final Deployment:**
   * Neural Network performs better here because of its potential to generalize the information and work with nonlinear dependencies among values.
   * However, there are some additional steps that can be taken for improvement of its performance: adjusting of all hyperparameters more thoroughly, training of the model during additional epochs, of using such techniques as drop-out and batch normalization.
2. **Consider Ensemble Models as a Backup:**
   * Thus, if computational resources or training time is a problem Random Forest or Gradient Boosting is also good choice providing similar performance but less complexity.
   * These models are particularly suited for scenarios requiring interpretability, as they provide feature importance insights.
3. **Reserve SVM and KNN for Specific Use Cases:**
   * Use SVM for smaller, simpler datasets with fewer features where interpretability is critical.
   * Use KNN cautiously, preferably for low-dimensional data or scenarios where interpretability outweighs performance requirements.
4. **Evaluate the Cost of Errors:**
   * Depending on the application, assess whether false positives or false negatives are more critical and align the choice of model accordingly. For instance:
     + If minimizing false positives is critical, focus on precision.
     + If minimizing false negatives is vital, focus on recall.
5. **Integrate a Model Selection Strategy:**
   * Implement cross-validation to test each model's robustness across varying subsets of the data.
   * Continuously monitor the model's performance in real-world applications and retrain it periodically with fresh data.
6. **Explore Hybrid Approaches:**
   * Consider combining Neural Networks with ensemble models (e.g., Gradient Boosting for feature engineering followed by Neural Networks) to leverage their strengths.
   * Ensemble multiple models using techniques like majority voting or stacking to further enhance accuracy and robustness.

By following these recommendations, the organization can maximize model performance, ensuring reliable predictions and minimizing the risk of errors in real-world deployments.

**2nd Temperature Prediction**

**Introduction**

In this document you shall find out case study to forecast the temperature by making use of the Abbotsford A weather station data. Two models were implemented: Two algorithms were selected: K- Nearest Neighbors (KNN) and Random Forest Regre

**Data Preprocessing**

The same processed dataset used in weather classification is used for this prediction.

**Feature Selection**

The final features selected for temperature prediction include:

* Day Hour Temp (°C)
* Dew Point Temp (°C)
* Rel Hum (%)
* Wind Dir (10s deg)
* Wind Spd (km/h)
* Visibility (km)
* Stn Press (kPa)
* Wind Chill

**Model Implementation**

**KNN Regressor**

Model Configuration

* Algorithm: K-Nearest Neighbors Regression
* Number of neighbors: 5
* Train-test split: 80-20

**Performance Metrics**

The KNN model achieved:

* Mean Squared Error (MSE): 0.07
* R-squared Score: 0.59

**Figure**

*KNN Regression Metric*

A graph with a bar and a number of red squares

Description automatically generated with medium confidence

**Analysis of Results**

1. **Error Distribution**
   * Low MSE (0.07) indicates good prediction accuracy
   * The predictions have relatively small deviations from actual values
2. **Model Fit**
   * R-squared value of 0.59 suggests moderate explanatory power
   * The model explains about 59% of the variance in temperature predictions

**Strengths and Limitations**

**Strengths:**

* Simple and intuitive implementation
* Non-parametric approach suitable for non-linear relationships
* Low prediction error (MSE = 0.07)

**Limitations:**

* Moderate R-squared value indicates room for improvement
* May struggle with extreme temperature values
* Sensitive to feature scaling

**Random Forest Regressor**

Configuration:

* Number of estimators: 45
* Random state: 42

**Performance Metrics:**

* Mean Squared Error (MSE): 0.04
* R-squared Score: 0.76
* Mean Absolute Error (MAE): 0.083

**Figure**

*Random Forest Map*

A diagram of a tree

Description automatically generated

**Feature Importance Analysis:** Based on the Random Forest model, the most influential features are:

1. Visibility (km) - ~0.22 importance score
2. Month - ~0.21 importance score
3. Dew Point Temperature - ~0.18 importance score
4. Temperature - ~0.17 importance score
5. Station Pressure - ~0.08 importance score

Figure

*Feature Importance*

A graph with blue bars

Description automatically generated

Lesser influential features include:

* Relative Humidity
* Wind Direction
* Wind Speed
* Wind-related sine transformations

**Model Comparison**

1. **Accuracy Metrics:**
   * Random Forest outperforms KNN with a lower MSE (0.04 vs 0.07)
   * Random Forest shows better explanatory power with higher R-squared (0.76 vs 0.59)
2. **Model Characteristics:**
   * KNN provides simpler implementation but lower accuracy
   * Random Forest offers better performance and feature importance insights

**Final Conclusion and Recommendation**

**Conclusion:**

It is clear that Random Forest performs better for temperature prediction; the model accounts for 76% of the variance in temperature as compared to the KNN which accounts for 59% of the variance in temperature values. The reliability of the model is achieved through the validation of a lower MSE (0.04), and a reasonable MAE of 0.08 suggesting high prediction accuracy. The results of the regression analysis show that the feature visibility, the month and the dew point temperature are the most important predictors that indicate strong the season and atmospheric condition factors in the temperature prediction.

**Recommendation:**

According to the comparative analysis presented in this paper, the Random Forest model should be adopted to predict temperature with the emphasis on the features of visibility, month, and the deputy of the temperature. For future work, it may be beneficial to explore a combination of these features as the main input with information about temporal dependencies extracted using time series analysis.

**Report Conclusion**

1. **Weather Classification Performance:**
   * Neural Network emerged as the best performer with 87.7% accuracy and 88.5% precision
   * Gradient Boosting (86.9%) and Random Forest (86.2%) followed closely
   * SVM (83.1%) and KNN (78.5%) showed lower but still reasonable performance
   * All models performed exceptionally well in classifying "Snow" conditions
   * Most models struggled with distinguishing between "Mainly Clear" and "Clear" conditions
2. **Temperature Prediction Results:**
   * Random Forest Regressor outperformed KNN with:
     + Lower Mean Squared Error (0.04 vs 0.07)
     + Higher R-squared score (0.76 vs 0.59)
   * Key predictive features identified were:
     + Visibility (22% importance)
     + Month (21% importance)
     + Dew Point Temperature (18% importance)

**Report Recommendations**

1. **Model Selection and Implementation:**
   * Use Neural Network as the primary model for weather classification
   * Implement Random Forest for temperature prediction
   * Maintain ensemble models (Gradient Boosting, Random Forest) as backup solutions
   * Consider a hybrid approach combining multiple models for more robust predictions
2. **Data Quality and Feature Engineering:**
   * Focus on improving feature engineering for better distinction between "Mainly Clear" and "Clear" conditions
   * Prioritize collection and quality of visibility data, given its high importance in temperature prediction
   * Continue monitoring seasonal patterns through monthly data
   * Consider adding more derived features from dew point temperature measurements
3. **System Implementation:**
   * Implement a real-time monitoring system to track model performance
   * Develop an automated retraining pipeline using fresh data
   * Create a fallback system using ensemble models when primary models fail
   * Establish clear thresholds for model retraining based on performance metrics
4. **Future Improvements:**
   * Explore deep learning architectures for better feature extraction
   * Consider implementing time series analysis for temperature prediction
   * Investigate the possibility of collecting additional relevant features
   * Develop a more sophisticated data preprocessing pipeline
   * Implement cross-validation techniques for more robust model evaluation
5. **Operational Considerations:**
   * Regular model retraining schedule (suggested quarterly)
   * Implement logging and monitoring systems for model predictions
   * Establish clear error handling procedures
   * Create documentation for model maintenance and updating procedures

These recommendations aim to maximize the accuracy and reliability of both weather classification and temperature prediction while ensuring system robustness and maintainability.

**References**

Final Collab-<https://colab.research.google.com/drive/1yo0C5ojYpFzRCfpMzslXDACcivL6SwEC?usp=sharing>

1st Collab- <https://colab.research.google.com/drive/1_pZ307Dn-5m2Y_ZZIz0VaIGsgL1gb5VN?usp=sharing>

2nd Collab- <https://colab.research.google.com/drive/1nWA82Tr0NlPto-mDnYxxg5MIVjiy6545?usp=sharing>

Data Source Processed - <https://drive.google.com/file/d/1OeqWqTlaezFZkUA8C0ojZxwGTcTI_fsT/view?usp=drive_link>

Raw Data -https://climate.weather.gc.ca/

**Machine Learning Methods and Implementation:**

* Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media.
* Chollet, F. (2021). Deep Learning with Python (2nd ed.). Manning Publications.
* VanderPlas, J. (2023). Python Data Science Handbook: Essential Tools for Working with Data (2nd ed.). O'Reilly Media.

**Weather Classification and Prediction:**

* Manandhar, S., Dev, S., Lee, Y. H., Winkler, S., & Meng, Y. S. (2019). Systematic Study of Weather Variables for Rainfall Detection. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 12(10), 3767-3777.
* McGovern, A., Elmore, K. L., Gagne, D. J., Haupt, S. E., Karstens, C. D., Lagerquist, R., ... & Williams, J. K. (2017). Using artificial intelligence to improve real-time decision-making for high-impact weather. Bulletin of the American Meteorological Society, 98(10), 2073-2090.

**Neural Networks and Deep Learning:**

* Goodfellow, I., Bengio, Y., & Courville, A. (2022). Deep Learning (2nd ed.). MIT Press. <http://www.deeplearningbook.org>

**Evaluation Metrics and Model Performance:**

* Raschka, S., & Mirjalili, V. (2023). Python Machine Learning (4th ed.). Packt Publishing.

**Feature Engineering and Selection:**

* Zheng, A., & Casari, A. (2022). Feature Engineering for Machine Learning (2nd ed.). O'Reilly Media.

**Time Series Analysis:**

* Brownlee, J. (2020). Deep Learning for Time Series Forecasting. Machine Learning Mastery.

**Machine Learning Libraries and Tools:**

* Scikit-learn Documentation (2024). <https://scikit-learn.org/stable/>
* TensorFlow Documentation (2024). <https://www.tensorflow.org/>
* Keras Documentation (2024). <https://keras.io/>