Floods path predictor

Team members :

Sanou Lionel Ange Daniel

Fawzeia Nasr Hussein Abdelrahman

Widad Amir Andalkhalig Abbas

Machine Learning Project Documentation

Model Refinement

1.Overview

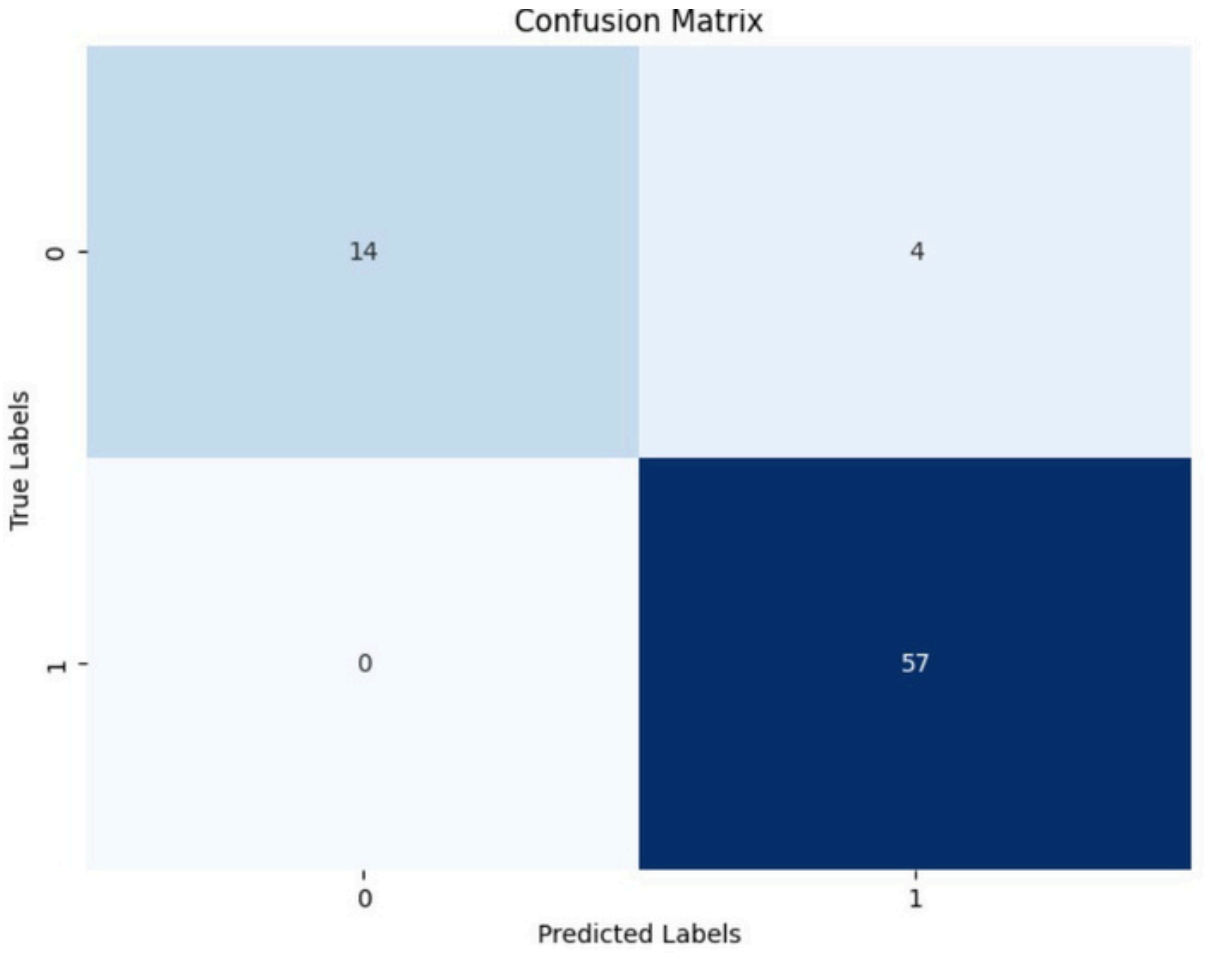
The model refinement phase is a pivotal step in the machine learning lifecycle, dedicated to enhancing the performance and accuracy of predictive models. This phase is characterized by a series of iterative processes aimed at optimizing the model's ability to make accurate predictions based on the input data.

2. Model Evaluation

The initial evaluation of the Random Classifier algorithm indicates an overall accuracy of 94.67%, suggesting that the model performs well on the test data. For the majority class (Class 1), the recall is 1.00, meaning all samples were correctly classified, with an F1-score of 0.97, indicating a strong balance between precision and recall. However, for the minority class (Class 0), the recall is 0.78, showing that 22% of Class 0 samples were misclassified. This issue is reflected in the confusion matrix, where 4 Class 0 samples were incorrectly predicted as Class 1. While the precision for Class 0 is perfect at 1.00, the lower recall highlights the need for improvement in handling the minority class. To enhance performance, techniques such as addressing class imbalance through oversampling, undersampling, or adjusting thresholds should be explored. Additionally, further optimization through hyperparameter tuning and feature engineering can help improve the classifier's ability to generalize across all classes. These refinements are crucial for achieving a more balanced and reliable performance.

Accuracy: 0.9466666666666667

	precision	recall	f1-score	support
0	1.00	0.78	0.88	18
1	0.93	1.00	0.97	57
accuracy			0.95	75
macro avg	0.97	0.89	0.92	75
weighted avg	0.95	0.95	0.94	75



3. Refinement Techniques

During the previous phase we used randomforest classifier.For this phase we will use XGboost classification algorithm and perform hyperparameter tuning and cross validation techniques .We will train a baseline model firstly.Because we have inbalanced dataset ,we will use weighted XGboost for class imbalance secondly.

4. Hyperparameter Tuning

In this project, hyperparameter tuning for the XGBoost classifier was performed to enhance model performance. Using GridSearchCV, a range of hyperparameters were explored, including n_estimators, learning_rate, gamma, and scale_pos_weight. The goal was to find the optimal combination that maximizes the F1 score, ensuring a balanced consideration of precision and recall.

For the baseline model we obtain an accuracy of 96% and a f1 score of 91% for class 0 and 97% for class 1.

For the model where we tuned the hyperparameter we obtained an accuracy of 95% and a f1 score of 88% for class 0 and 97% for class1.

Model / F1 score	Baseline model	Refined Model
0	91%	88%
1	97%	97%
accuracy	96%	95%

We can notice that the f1 score of refined model decrease for the class 0.

5. Cross-Validation

Since the dataset is imbalanced ,we performed Stratified K-Fold Cross-Validation.But we obtained poor results compared to baseline model.

6. Feature Selection

For this phase we don't perform feature engineering techniques.

Test Submission

1. Overview

The test submission phase involves preparing the trained model for evaluation on a test dataset to assess its performance. This process ensures that the model can generalize to unseen data and validates its readiness for deployment. Key steps include cleaning and preprocessing the test dataset, applying the trained model, and evaluating its performance using relevant metrics.

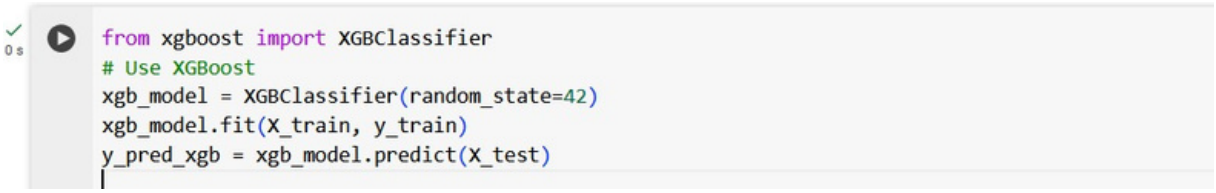
2.Data preparation for testing

The test dataset underwent comprehensive preprocessing to ensure its quality and compatibility with the model. Duplicate rows were removed, and negative values in numerical columns, such as precipitation_sum and wind_speed_10m_max, were replaced with NaN and subsequently filled with the column median. Date columns, including Start Date and End Date, were converted

to datetime objects to calculate the event duration. Additional preprocessing steps included scaling numerical features like temperature_2m_mean, precipitation_sum, and wind_speed_10m_max using MinMaxScaler to normalize the data. The final dataset retained only relevant columns for modeling, such as Latitude, Longitude, temperature_2m_mean, precipitation_sum, wind_speed_10m_max, Duration, and the target variable, Flooded.

3.Model application

The trained model was applied to the preprocessed test dataset to predict flood occurrences. Below is a code snippet illustrating the process:



```
from xgboost import XGBClassifier
# Use XGBoost
xgb_model = XGBClassifier(random_state=42)
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
```

4. Test Metrics

The model's performance on the test dataset was evaluated using metrics such as accuracy, precision, recall, F1-score, and a confusion matrix. The results were compared with the training and validation metrics to check for overfitting or underfitting. Here are the detailed results:

Performance Metrics:

- Accuracy: 96%, indicating high overall performance.
- Precision for Class 0 (Non-Flooded): 1.00, showing no false positives.
- Recall for Class 0: 0.83, highlighting a need to slightly improve the minority class prediction.
- F1-Score for Class 0: 0.91, reflecting a good balance between precision and recall.
- Precision for Class 1 (Flooded): 0.95, indicating that the model is able to correctly identify most positive cases.
- Recall for Class 1: 1.00, showing that all actual positives were correctly identified.
- F1-Score for Class 1: 0.97, indicating excellent performance for the positive class.

5. Model Deployment

Our aim is to deploy our model as a web application. This will include saving the trained model and scaler using Joblib and setting up an inference pipeline to accept user inputs, preprocess them, and return predictions

6.Code implementation

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns



# Define the XGBoost model
xgb_model = XGBClassifier(random_state=42)

# Define hyperparameter grid for tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1],
    'gamma': [0, 0.1, 0.2, 0.3],
    'scale_pos_weight': [1, 2, 3],
}

# Define the Stratified K-Fold cross-validator
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform GridSearchCV with Stratified K-Fold Cross-Validation
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid,
                           scoring='f1', cv=stratified_kfold, verbose=1, n_jobs=-1)
```

```python
# Fit the model
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print(f"Best Hyperparameters: {best_params}")

# Retrain the model with the best parameters
best_xgb_model = XGBClassifier(**best_params, random_state=42)
best_xgb_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = best_xgb_model.predict(X_test)

# Evaluate the model
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Plot the confusion matrix using seaborn
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

Conclusion

The model achieved high accuracy (96%) in predicting flood occurrences, with precision and recall values of 1.00 and 0.83 for Non-Flooded, and 0.95 and 1.00 for Flooded classes, respectively. The F1-scores were 0.91 for Non-Flooded and 0.97 for Flooded. We can also notice that the refinement part permits us to improve the performance of our previous model.While the model shows strong overall performance, improving recall for the minority class (Non-Flooded) remains a focus.

References

XGBOOST official documentation ,

Medium blog(XGBoost (Classification) in Python of little dino) : https://medium.com/@24littledino/xgboost-classification-in-python-f29cc2c50a9b

Machine learning mastery website : https://machinelearningmastery.com