Machine Learning using Apache Spark MLib

Sun Yueqi A0333495X
Qiao Yichen A0330197E
Wu Shuwen A0330073U

September 2025

Abstract

Accurate prediction of sea level pressure is crucial for weather forecasting and climate studies. In this work, we investigate the performance of three regression models—Linear Ridge Regression, Gradient Boosted Trees (GBT) Regression, and Random Forest Regression—on a dataset containing geographical and meteorological features, including latitude, longitude, elevation, wind speed and direction, ceiling height, visibility, air temperature, and dew point temperature.

The dataset was preprocessed by filtering out missing values, extracting relevant features, and scaling them using MinMaxScaler. Models were trained using PySpark on a Google Cloud cluster, and hyperparameter tuning was conducted via grid search with 3-fold cross-validation. Root Mean Squared Error (RMSE) was used to evaluate model performance on both training and test sets.

Among the models, GBT Regression achieved the best performance, with RMSE values of 74.98 on the training set and 74.92 on the test set, outperforming both Random Forest and Ridge Regression. This superior performance is attributed to GBT's ability to model complex, non-linear interactions through an iterative ensemble of decision trees.

The study highlights the effectiveness of ensemble-based methods for predicting meteorological variables and provides insights into hyperparameter optimization and model selection for high-dimensional, non-linear regression tasks. Future work could explore expanded hyperparameter grids, neural networks, and advanced feature engineering techniques to further improve predictive accuracy.

Contents

Chapter 1

Introduction

The objective of this project is to build machine learning models to predict the sea level pressure using a variety of geographical and meteorological features, including:

- Latitude coordinates
- Longitude coordinates
- Elevation
- Wind direction angle
- Wind speed rate
- Ceiling height
- Visibility distance
- Air temperature
- Dew point temperature

Accurate prediction of sea level pressure is crucial for weather forecasting, climate studies, and the analysis of atmospheric phenomena. Traditional numerical methods for predicting atmospheric variables can be computationally intensive and may struggle to capture complex, non-linear relationships among environmental features. Machine learning models, particularly ensemble-based methods, offer an effective alternative by learning patterns from historical data and making predictions without relying on explicit physical equations.

Due to the large size of the dataset, training models on a local machine was impractical. To address this, a Google Cloud Dataproc cluster was utilized for distributed computation. The dataset was first uploaded to a Google Cloud Storage bucket, which served as a staging area. Dataproc employs the Hadoop Distributed File System (HDFS) as its default file system, allowing for efficient distributed storage and retrieval of large datasets. By enabling Spark during the cluster setup, the processing workload could be distributed across multiple nodes, leveraging the cluster's parallel computing capabilities to accelerate data preprocessing and model training.

PySpark was used throughout the project for all stages of the workflow, including data cleaning, feature extraction, transformation, model training, hyperparameter tuning, and evaluation. The use of PySpark not only enabled scalable computation but also ensured reproducibility and flexibility in the experimentation process. This setup allowed for the implementation of multiple machine learning models, including Linear Ridge Regression, Gradient Boosted Trees Regression, and Random Forest Regression, to compare performance and identify the most effective approach for predicting sea level pressure.

The project ultimately aims to demonstrate the effectiveness of distributed machine learning workflows for large-scale environmental data, providing insights into model selection, hyperparameter optimization, and the practical challenges of handling complex meteorological datasets in a cloud computing environment.

Chapter 2

Methodology

2.1 Data Preprocessing

Rows with missing values were filtered out based on the documentation. Some of the columns also contained multiple features. Therefore, the relevant features were extracted and cast to `float` datatype. The names of certain features were also renamed for greater clarity. Missing values were removed as displayed in Table 2.1.

Table 2.1: Feature filtering conditions

Feature	**Filter Condition**
Latitude	Must not equal `+99999`
Longitude	Must not equal `+999999`
Elevation	Must not equal `+9999`
Wind Angle (WND[0])	Must not equal `999`
Wind Speed (WND[3])	Must not equal `9999`
Ceiling Height (CIG[0])	Must not equal `99999`
Visibility Distance (VIS[0])	Must not equal `999999`
Air Temperature (TMP[0])	Must not equal `+9999`
Dew Point Temperature (DEW[0])	Must not equal `+9999`
Sea Level Pressure (SLP[0])	Must not equal `99999`

The `VectorAssembler` class from PySpark was then used to combine the feature columns into a single vector for model training. The `MinMaxScaler` was subsequently applied to the combined feature vectors to obtain a `scaled_features` column. This ensures that features with larger ranges do not disproportionately influence model training compared to those with smaller ranges.

The dataset was split into training and test sets using a 70:30 ratio, with a random seed set to 4 to ensure reproducibility. In addition, the training set was cached for faster access during training without the need to recompute from the source each time.

2.2 Model Selection and Training

Three models were chosen for this study: **Linear Ridge Regression**, **Gradient Boosted Trees (GBT) Regression**, and **Random Forest Regression**.

2.2.1 Linear Ridge Regression

Linear Ridge Regression addresses multicollinearity and captures linear relationships between the features and target variable. It extends the standard linear regression by adding an L_2 regularization term to penalize large coefficient values, thereby improving generalization. The objective function for Ridge Regression is given by:

$$L(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2 \tag{2.1}$$

where $\mathbf{X} \in \mathbb{R}^{n\times p}$ denotes the feature matrix, $\mathbf{y} \in \mathbb{R}^n$ the target vector, $\mathbf{w} \in \mathbb{R}^p$ the model coefficients, and λ the regularization parameter controlling the penalty strength. A higher λ increases the bias but reduces variance.

2.2.2 Gradient Boosted Trees (GBT) Regression

Gradient Boosted Trees (GBT) capture complex, nonlinear interactions through an additive, stage-wise model. It builds an ensemble of weak learners (decision trees) sequentially, where each new tree $h_m(x)$ is trained to minimize the residual errors of the previous model. The general form of a GBT model is:

$$F_M(x) = \sum_{m=1}^{M} \gamma_m h_m(x) \tag{2.2}$$

where M is the number of boosting stages, $h_m(x)$ represents the m-th decision tree, and γ_m is the learning rate or step size. Each iteration aims to minimize a differentiable loss function $L(y, F(x))$ using gradient descent in function space.

2.2.3 Random Forest Regression

Random Forest Regression constructs multiple decision trees in parallel and aggregates their predictions to improve accuracy and reduce overfitting. Each tree is trained on a bootstrap sample of the data, and at each split, a random subset of features is considered. The ensemble prediction is computed as:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(x) \tag{2.3}$$

where T is the total number of trees and $f_t(x)$ represents the prediction of the t-th tree. This bagging approach reduces variance while maintaining low bias.

Chapter 3

Experimental Setup

3.1 Model Implementation and Training Setup

All three models were implemented using PySpark's `LinearRegression()`, `GBTRegressor()`, and `RandomForestRegressor()` classes, respectively. The evaluator used was PySpark's `RegressionEvaluator`, with the default metric being the Root Mean Squared Error (RMSE).

Hyperparameter tuning was performed using Grid Search, defined through parameter grids in PySpark, as summarized in Table 3.1.

Table 3.1: Model hyperparameter grids for tuning

Method	**Parameters**
Linear Ridge Regression	Regularization Parameter: [0.01, 0.1, 1.0]
GBT Regression	Step Size: [0.05, 0.1, 0.2]
Random Forest Regression	Number of Trees: [20, 30, 40]
	Max Depth: [5, 7, 9]

Pipelines were created for each of the three models, comprising three stages: the `VectorAssembler`, `MinMaxScaler`, and the respective model class (`LinearRegression`, `GBTRegressor`, or `RandomForestRegressor`).

Subsequently, 3-fold cross-validation was performed for each model using PySpark's `CrossValidator` class, with the parameter grids defined above and the estimator set to each model's respective pipeline. This process determined the best combination of hyperparameters for each model.

Training was conducted on a Google Cloud Cluster consisting of one master node and two worker nodes, each with 100 GB of storage. Training time took approximately 4.5 hours. Each worker node was an `n2-standard-4` instance with 4 vCPUs and 16 GB of memory.

3.2 Model Evaluation

Model evaluation was conducted to assess the predictive performance of the trained models. Predictions were generated for both the training and test datasets using each of the three models. The evaluation metric used was the **Root Mean Squared Error (RMSE)**, a common measure of model accuracy in regression tasks. The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.1}$$

where y_i denotes the true value, $\hat{y}_i$ is the predicted value, and n is the total number of samples. A lower RMSE indicates better predictive performance.

Chapter 4

Experimental Results

4.1 Performance Comparison

The RMSE values for both the training and test sets are presented in Table 4.1. It can be observed that the RMSE values for the training and test sets are very similar across all three models, indicating that none of the models show signs of overfitting.

Table 4.1: RMSE performance on training and test datasets

Model	**Training RMSE**	**Test RMSE**
Linear Ridge Regression	83.90	83.85
Gradient Boosted Trees (GBT) Regression	74.98	74.92
Random Forest Regression	76.09	76.06

Among the models, the **Gradient Boosted Trees Regression** achieved the lowest RMSE, demonstrating superior performance in capturing complex, non-linear patterns within the data. This was followed by **Random Forest Regression**, and finally **Linear Ridge Regression**, which, while effective for linear dependencies, was less capable of modeling non-linear relationships.

4.2 Optimal Hyperparameters

The best hyperparameters obtained through cross-validation for each model are summarized in Table 4.2.

Table 4.2: Best hyperparameters obtained from cross-validation

Model	**Best Parameters**
Linear Ridge Regression	Regularization Parameter (λ) = 0.01
Gradient Boosted Trees Regression	Step Size = 0.2
Random Forest Regression	Number of Trees = 20 , Max Depth = 9

These hyperparameters were determined to minimize the RMSE on the validation folds during cross-validation, ensuring that the final models achieve the best generalization performance on unseen data.

Chapter 5

Conclusion

Among the three models evaluated, **Gradient Boosted Trees (GBT) Regression** performed the best, achieving a Root Mean Squared Error (RMSE) of 74.98 on the training set and 74.92 on the test set. The optimal hyperparameter for GBT in this study was a step size of 0.2.

The superior performance of GBT can be attributed to its ability to model complex, non-linear relationships in the data. Unlike linear models such as Ridge Regression, GBT builds an ensemble of decision trees in a sequential manner, where each tree attempts to correct the errors of the previous one. This iterative approach allows the model to capture intricate interactions among features, which is particularly important in this study since the target variable, sea level pressure, depends on a variety of geographical and weather-related factors. GBT's ability to adapt to both linear and non-linear patterns, while simultaneously controlling overfitting through regularization and early stopping techniques, likely contributed to its strong performance on both the training and test datasets.

The results also indicate that **Random Forest Regression** performed reasonably well, with slightly higher RMSE values than GBT. Random Forest's ensemble of trees trained on bootstrap samples provides robustness against variance but lacks the iterative correction mechanism that GBT employs, which could explain its comparatively lower predictive accuracy. **Linear Ridge Regression**, while effective for capturing linear dependencies and handling multicollinearity, was unable to adequately model the complex, non-linear interactions present in the data, resulting in the highest RMSE among the three models.

For future work, several directions can be explored to further improve model performance. First, the hyperparameter grid for GBT can be expanded to include additional parameters such as `maxDepth`, `maxIter`, `minInstancesPerNode`, and `subsamplingRate`, allowing for a more comprehensive search for optimal configurations. Second, other advanced regression methods, such as **neural networks**, **support vector regression (SVR)**, or hybrid ensemble approaches, can be investigated to capture even more complex relationships in the data. Incorporating feature engineering techniques, such as creating interaction terms or polynomial features, may also enhance the predictive capabilities of linear and tree-based models. Additionally, experimenting with feature selection or dimensionality reduction methods like Principal Component Analysis (PCA) could help reduce noise and improve generalization.

Overall, this study demonstrates that ensemble-based methods like GBT are highly effective for modeling complex meteorological phenomena, such as predicting sea level pressure from diverse environmental features. The findings highlight the importance of using models capable of capturing non-linear interactions in high-dimensional datasets, and provide a foundation for further improvements through hyperparameter optimization and exploration of more sophisticated algorithms.