DSA5208 Project 2: Predicting Air Pressure with MLlib (Apache Spark)

S K Ruban A0253837W
Owen Li Dong Lin A0231088H

November 2024

1. Introduction

This project focuses on predicting sea level pressure values using geographical and weather condition data from the National Centers for Environmental Information (NCEI) climate dataset.

Our objective is to implement machine learning models to predict the sea level pressure using latitude coordinates, longitude coordinates, elevation, wind direction angle, wind speed rate, ceiling height, visibility distance, air temperature and dew point temperature data.

Given that we need to predict continuous values in our supervised learning task, we chose to implement three regression models: Ridge Regression, Gradient-Boosted Tree Regression (GBT) and Random Forest Regression. We evaluate and compare their performance using the Root Mean Square Error (RMSE) on both training and test datasets.

2. Preliminary Data Analysis & Pre-processing

Prior to implementing our regression models, we conducted an initial analysis of the climate dataset to understand its characteristics.

Our data analysis primarily relied upon the documentation provided by the NCEI data source site. Based on the documentation and project brief, we filtered out irrelevant columns first. Next, we used the missing data regular regex terms to identify and cast all cells with missing data to None types.

We collated all the missing data filtering conditions in the table below.

Data column name	Missing data filtering condition
Latitude	Discard row if first 6 characters are '+99999'
Longitude	Discard row if first 7 characters are '+999999'
Elevation	Discard row if first 5 characters are '+9999'
Wind direction angle, Wind speed rate (WND)	Discard row if first 3 characters are '999' or characters 6-9 are '9999'
Ceiling height (CIG)	Discard row if first 5 characters are '99999'
Visibility distance (VIS)	Discard row if first 6 characters are '999999'
Air temperature (TMP)	Discard row if first 5 characters are '+9999'
Dew point temperature (DEW)	Discard row if first 5 characters are '+9999'
Sea level pressure (SLP)	Discard row if first 5 characters are '99999'

After filtering and labelling all missing data, we applied max value - min value range filters obtained from the documentation. By filtering the data with max - min values, we ensure that we minimise the possibility of any anomalous data that may influence our model training.

The valid maximum and minimum values for each data column has been tabulated below:

Data Column Name	Range of valid values: [min, max]
Latitude	[-90000, 90000]
Longitude	[-179999, 180000]
Elevation	[-400, 8850]
Wind direction angle, Wind speed rate (WND)	[1, 360], [0,900]
Ceiling height (CIG)	[0, 22000]
Visibility distance (VIS)	[0, 160000]
Air temperature (TMP)	[-932, 618]
Dew point temperature (DEW)	[-982, 368]
Sea level pressure (SLP)	[8600, 10900]

After removing all null values and out-of-range values, our total data size decreased from 130531842 rows to 18471526 rows (85.84% decrease).

Next, as part of our data preprocessing, we applied Min-Max Scaling to each data column (excluding the target 'sea level pressure' column) by using the min-max values from the documentation.

Now that our data had been fully cleaned and processed, we split the data into train and test sets in a ratio of 7:3 using the 'randomSplit' method in PySpark.

Finally, we assembled the feature vector for model training via MLlib's VectorAssembler feature transformer.

3. Model Training

We begin with defining our three regression models, using the LinearRegression, GBTRegressor and RandomForestRegressor classes.

We used Linear Regression with elasticNetParam=0.0 to simulate Ridge Regression as our baseline regression model. We then applied GBTRegressor, an ensemble method, to capture potential non-linear relationships in the data, and RandomForestRegressor, another ensemble method, to evaluate whether a different ensemble learning approach could outperform the other models.

Next, we defined the parameter grids for our three models:

Model	Parameter grid
Ridge Regression	Regularisation parameter: [0.01, 0.1, 1.0]
Gradient Boosted Tree Regressor	Max Depth: [5, 7, 9] Max Iterations: [20, 30, 40]
Random Forest Regressor	Number of trees: [20, 30, 40] Max Depth: [5, 7, 9]

Then, we defined our Regression Evaluator and Cross Validators for each model. For our k-fold cross validation, we decided to use 3 folds. This was intended to keep computation times fast but still maintain a satisfactory level of cross validation.

Finally, we created the pipelines and fit each model. The total training time required on our Google Cloud Cluster was approximately 3 hours.

4. Model Evaluation and Analysis

To evaluate our models, we first made predictions on the test and train sets using each model.

Next, using Root Mean Squared Error (RMSE) as our model performance metric, we used the PySpark Regression Evaluator to obtain the RMSE on both our train and test sets.

The code output is shown here:

5.1 Calculate RMSE of model predictions on train and test sets

In [21]:
```
ridge_train_rmse = evaluator.evaluate(ridge_train_predictions)
gbt_train_rmse = evaluator.evaluate(gbt_train_predictions)
rf_train_rmse = evaluator.evaluate(rf_train_predictions)

ridge_test_rmse = evaluator.evaluate(ridge_test_predictions)
gbt_test_rmse = evaluator.evaluate(gbt_test_predictions)
rf_test_rmse = evaluator.evaluate(rf_test_predictions)
```

In [22]:
```
print(f"\nRidge Regression Train RMSE: {ridge_train_rmse}")
print(f"Gradient-Boosted Tree Train RMSE: {gbt_train_rmse}")
print(f"Random Forest Train RMSE: {rf_train_rmse}\n")

print(f"\nRidge Regression Test RMSE: {ridge_test_rmse}")
print(f"Gradient-Boosted Tree Test RMSE: {gbt_test_rmse}")
print(f"Random Forest Test RMSE: {rf_test_rmse}\n")
```

```
Ridge Regression Train RMSE: 86.09802416032562
Gradient-Boosted Tree Train RMSE: 72.59316352077796
Random Forest Train RMSE: 78.11254781302699


Ridge Regression Test RMSE: 86.09335444183954
Gradient-Boosted Tree Test RMSE: 72.69570278675752
Random Forest Test RMSE: 78.09106589460644
```

We can see that the RMSE for each model is very consistent across both the training and test sets, with no anomalous values, showing that our models are not overfitted.

Comparing each model, we can see that the Gradient Boosted Tree model performed the best, with the lowest RMSE, followed by the Random Forest model, then finally the Ridge Regression model.

Following that, we also obtained the best hyperparameters for each model, by using the .stages attribute to access our model pipeline stages and .bestModel attribute to obtain the hyperparameters that achieved the highest evaluation metric during tuning.

The code output in the next page:

5.2 Identify the optimized hyperparameters for each model

In [25]:
```
ridge_best_model = ridge_model.stages[-1].bestModel
gbt_best_model = gbt_model.stages[-1].bestModel
rf_best_model = rf_model.stages[-1].bestModel

print("\nBest Hyperparameters:")

print(f"Ridge Regression: regParam = {ridge_best_model._java_obj.parent().getRegParam()}")

print(f"GBT: maxDepth = {gbt_best_model._java_obj.parent().getMaxDepth()},\
    maxIter = {gbt_best_model._java_obj.parent().getMaxIter()}")

print(f"Random Forest: numTrees = {rf_best_model._java_obj.parent().getNumTrees()},\
      maxDepth = {rf_best_model._java_obj.parent().getMaxDepth()}")
```

```
Best Hyperparameters:
Ridge Regression: regParam = 0.01
GBT: maxDepth = 9,    maxIter = 40
Random Forest: numTrees = 40,      maxDepth = 9
```

5. Discussion & Conclusion

The best performing model was the Gradient Boosted Tree model, which had a train RMSE of 72.59 and test RMSE of 72.70. The best hyperparameters for our GBT model were Max Depth = 9 and Max Iter = 40.

Our GBT model might have performed the best out of the three, primarily because GBT trees are built sequentially, to correct the error of the previous tree. This allowed our model to capture complex relationships in the data more effectively.

Given the task domain of predicting sea level pressure from multiple geographical and weather parameters, the relationships between variables are likely non-linear and highly complex.

Therefore GBT outshined the simple Ridge Regression model and the generalised, averaged Random Forest model.

Moving forward, there is some further work that can be done to explore this task further.

For example, we could do further optimization of our parameter grids by using techniques like Bayesian Optimization to fine-tune more effectively.

Additionally, revisiting our feature engineering to explore non-linear transformations could help us with optimising our model selection.

Finally, we could explore advanced models like XGBoost or neural networks to capture even more intricate patterns in data.