ENSF 612 - Engineering Large Scale Data Analytics Systems

Semester/Year:	F2023
Title	Final Report

Instructor:	Sarah Shah
Submission Date:	17/12/2023
Due Date:	17/12/2023

Student Name	**Student ID xxxx12345**	**Email**
Aemen Mohsin	30225983	aemen.mohsin@ucalgary.ca
Soumini Mohandas	30105691	soumini.mohandas@ucalgary.ca
Jauhar Fathima	30209352	jauhar.fathima@ucalgary.ca
Shaun Sebastian	30213627	shaun.sebastian1@ucalgary.ca

I. Introduction and Motivation

Among the available datasets, the Global Surface Summary of the Day (GSOD) dataset from the National Oceanic and Atmospheric Administration (NOAA) was chosen for analysis. The data, which was taken from Kaggle, displays how important it is to comprehend climate dynamics. The project uses Apache Spark, a powerful big data processing framework, to handle and analyze large-scale meteorological datasets in an efficient manner. The main goal of the research is to identify past temperature trends in Utah. To process this data in an accurate manner, the data from Salt Lake City International Airport and Bryce Canyon Airport was chosen as one is located in the center of the city, while the other is a mountainous region. Using two locations was necessary for an accurate prediction of average temperature as difference in latitude, longitude and elevation creates a large difference in temperature.

The motivation for analyzing weather data stems from its importance in multiple fields. A detailed understanding of past weather patterns is essential for many reasons, including guiding agricultural practices, supporting infrastructure planning, and offering insightful information on climate change. Moreover, past weather patterns can successfully be used to predict upcoming weather trends and forecasting. With its varied topography, Utah offers a compelling case study for examining regional climate variations. Additionally, using Spark ensures scalability and efficiency in extracting meaningful insights while making handling large datasets easier.

II. Data Collection

The project is built upon an extensive collection of weather data from the NOAA GSOD dataset. This dataset, which covers the years 1929 through 2023, is an extensive global repository of daily surface weather observations collected at 9000 NOAA weather stations. The scope is narrowed by the deliberate focus on Utah through the years 2010 to 2019, making a thorough analysis of temperature trends in this area in recent times possible.

Two specific weather stations were specifically chosen to be the center of the analysis: Bryce Canyon Airport and Salt Lake City International Airport. The necessity to research Utah's diverse climates led to this decision. Similarly, the U.S. dataset was far more robust compared to other parts of the world. Salt Lake City International Airport represented the central city, while Bryce Canyon Airport displayed a higher altitude, revealing the weather patterns associated with both the city and the mountains. This comparison was essential for a diverse estimation of the whole state rather than just a concentrated region. Similarly, this differentiation assisted in discovering the differences between city and mountain weather data which was substantially different comparatively.

Using Google's BigQuery service, the GSOD dataset was queried in order to extract the data. The project's focus on scalability was aligned with the seamless retrieval of large datasets made possible by this cloud-based approach. Large-scale data handling was made even easier with the integration of Spark, allowing for effective processing and analysis.

In order to improve the level of analysis, additional data regarding weather stations was added. By offering contextual information about each station, this supplemental data assisted in creating a more comprehensive picture of the regional variables affecting weather observations.

III. Data Inspection and Validation

To make sure the datasets met the expected standards, a thorough process of data inspection and validation was carried out to find anomalies, irregularities, and missing values. The data was first displayed with the first five rows to ensure it was imported correctly.

index	0	1	2	3	4
station_number	31590	38110	36010	33960	37770
wban_number	99999	99999	99999	99999	99999
year	1929	1929	1929	1929	1929
month	12	11	11	12	8
day	29	25	22	23	25
mean_temp	45.20000076293945	52.20000076293945	53.29999923706055	38	66.19999694824219
num_mean_temp_samples	4	4	4	4	4
mean_dew_point	41.70000076293945	48.099998474121094	51.79999923706055	36.5	56
num_mean_dew_point_samples	4	4	4	4	4
mean_sealevel_pressure	NaN	991	989.5999755859375	991.7999877929688	1022.2000122070312
num_mean_sealevel_pressure_samples	<NA>	4	4	4	4
mean_station_pressure	NaN	NaN	NaN	NaN	NaN
num_mean_station_pressure_samples	<NA>	<NA>	<NA>	<NA>	<NA>
mean_visibility	10.899999618530273	2.5	6.800000190734863	2.200000047683716	8.399999618530273
num_mean_visibility_samples	4	4	4	4	4
mean_wind_speed	27.700000762939453	34.400001525878906	16.799999237060547	15	3.5
num_mean_wind_speed_samples	4	4	4	4	4
max_sustained_wind_speed	36.900001525878906	43.900001525878906	18.100000381469727	23.899999618530273	8.899999618530273
max_gust_wind_speed	NaN	NaN	NaN	NaN	NaN
max_temperature	36	50	52	34	55.900001525878906
max_temperature_explicit	false	true	false	false	false
min_temperature	NaN	NaN	NaN	NaN	NaN
min_temperature_explicit	<NA>	<NA>	<NA>	<NA>	<NA>
total_precipitation	NaN	NaN	NaN	NaN	0
snow_depth	NaN	NaN	NaN	NaN	NaN

Figure 1: First 5 Data Entries of the Dataset

The above figure (Figure 1), displays the rows of each data with corresponding columns, transposed for easier visibility. Multiple columns were mentioned in the data including station numbers, time series data, average temperatures, dew points, pressure, wind speed, visibility, precipitation and snow depth. Out of all of these columns, only a few were chosen to proceed for an assessment of weather data.

During the inspection phase, one of the modifications made was to look for anomalous temperature values, represented by 9999.9. These numbers were found to be stand-ins for inaccurate or missing data points. An organized method was used to substitute NaN (Not a Number) values for these placeholders, allowing missing data to be represented consistently throughout the datasets (Figure 2). By taking this step, it ensured that the inaccurate values found in different stations wouldn't affect the accuracy of the analyses that would come after. In order to ensure a smooth validation, missing values were dropped.

```
def cleanWeatherData(dataframe):
    # Adapted from # https://www.kaggle.com/tanatiem/eda-bangkok-weather
    weather=dataframe
    weather['temp'] = weather['temp'].replace({ 9999.9 : np.nan })
    weather['min'] = weather['min'].replace({ 9999.9 : np.nan })
    weather['max'] = weather['max'].replace({ 9999.9 : np.nan })
    weather['date'] = weather.apply(lambda x:datetime.datetime(int(x.year), int(x.mo), int(x.da)),axis=1)
    weather = weather.set_index('date')
    start_date = '{}010101'.format(years[0]) #0101
    end_date = weather.index.max().strftime('%Y%m%d')
    missing = weather[weather.isnull().any(axis=1)].index
    weather = weather.interpolate()
    weather['year'] = weather.index.year
    weather['mo'] = weather.index.month
    weather['da'] = weather.index.day
    data = weather[['temp','min','max']]
    data.reset_index(inplace=True)
    data.columns = ['Date','Avg Temp', 'Min Temp', 'Max Temp']
    return(data)
```

Figure 2: Data Cleaning

IV. Data Filtering

The data analysis was focused on a strategic data filtering process that we implemented to extract subsets of data specific to the geographical (Utah weather stations) and contextual (Deriving Average Temperature) requirements of the project. This phase involved constraining the datasets based on select features and feature values (Maximum Temperature, Minimum Temperature or the temperature values, the time values, spatial values which include longitude and latitude), and aligning them with the specific locations of interest and optimizing relevance. To narrow the scope of the analysis to Utah, latitude and longitude filtering criteria were applied. Only records falling within a specific latitude and longitude range corresponding to Utah were retained. This geographical filtering ensured that the subsequent analyses were geographically centered on the region of interest, providing localized insights into weather patterns.

The selection of Salt Lake City International and Bryce Canyon Airport as weather stations allowed for a wide geographic variation because SLC is an urban location and Bryce Canyon is located in a mountainous region. Each location's unique latitude and longitude filters were used. This method made it possible to examine temperature patterns at these two different

Utah locales more closely. The filtered datasets were then the subject of further analysis, now customized to the study's geographic setting.

The filtering process not only improved the relevance of the datasets but also streamlined computational resources by excluding extraneous records. This optimization was particularly crucial given the scale of the NOAA GSOD dataset.

```
years = range(2010, 2020)
SLCWeather = [helper.query_to_pandas(tempsAtStationName.format(i, "SALT LAKE CITY INTERNATIONAL"))
              for i in years]
SLCWeather = pd.concat(SLCWeather)
cleanSLCWeather = cleanWeatherData(SLCWeather)
mountainWeather = [helper.query_to_pandas(tempsAtStationName.format(i, "BRYCE CANYON AIRPORT"))
                   for i in years]
mountainWeather = pd.concat(mountainWeather)
cleanMountainWeather = cleanWeatherData(mountainWeather)

avgTemps = helper.query_to_pandas_safe(avgTemps)
avgTemps = avgTemps.dropna(axis=0)

lat= 39
lon= -111
ut_latitudeLower = avgTemps['lat'] > lat-3
ut_latitudeUpper = avgTemps['lat'] < lat+3
ut_longitudeLower = avgTemps['lon'] > lon-3
ut_longitudeUpper = avgTemps['lon'] < lon+3
ut_only = avgTemps[ut_latitudeLower & ut_latitudeUpper & ut_longitudeLower & ut_longitudeUpper]
ut_only.head(10)
```

Figure 3: Data Separation and Extraction.

V. Exploratory Data Analysis

An essential stage was Exploratory Data Analysis (EDA), which provided a thorough examination of the enriched datasets in order to identify patterns, trends, and correlations. This phase produced the statistical summaries and visualizations.

To display the average daily temperature over the course of the months, line plots were used. Temperature variations and seasonal patterns were discovered through this temporal investigation. This also assisted in visualizing the peaks and valleys of the temperature along with the minimum, maximum and average temperatures. The temporal analysis was enhanced by heatmaps, which gave a thorough summary of the average daily temperatures by month and year. Further research into climatic changes was guided by the ability to identify trends and variations through the use of this visual representation. This visual representation was key in determining the absolute maximum temperature based on the year and month along with the absolute minimum temperature based on year and month. Lastly, the temperature distributions of maximum, minimum and average temperatures with month and year were displayed in box plots.

These plots also displayed standard deviation, which played an essential role in determining range of temperature per month. These plots are visible below in (Figure 4) and (Figure 5).

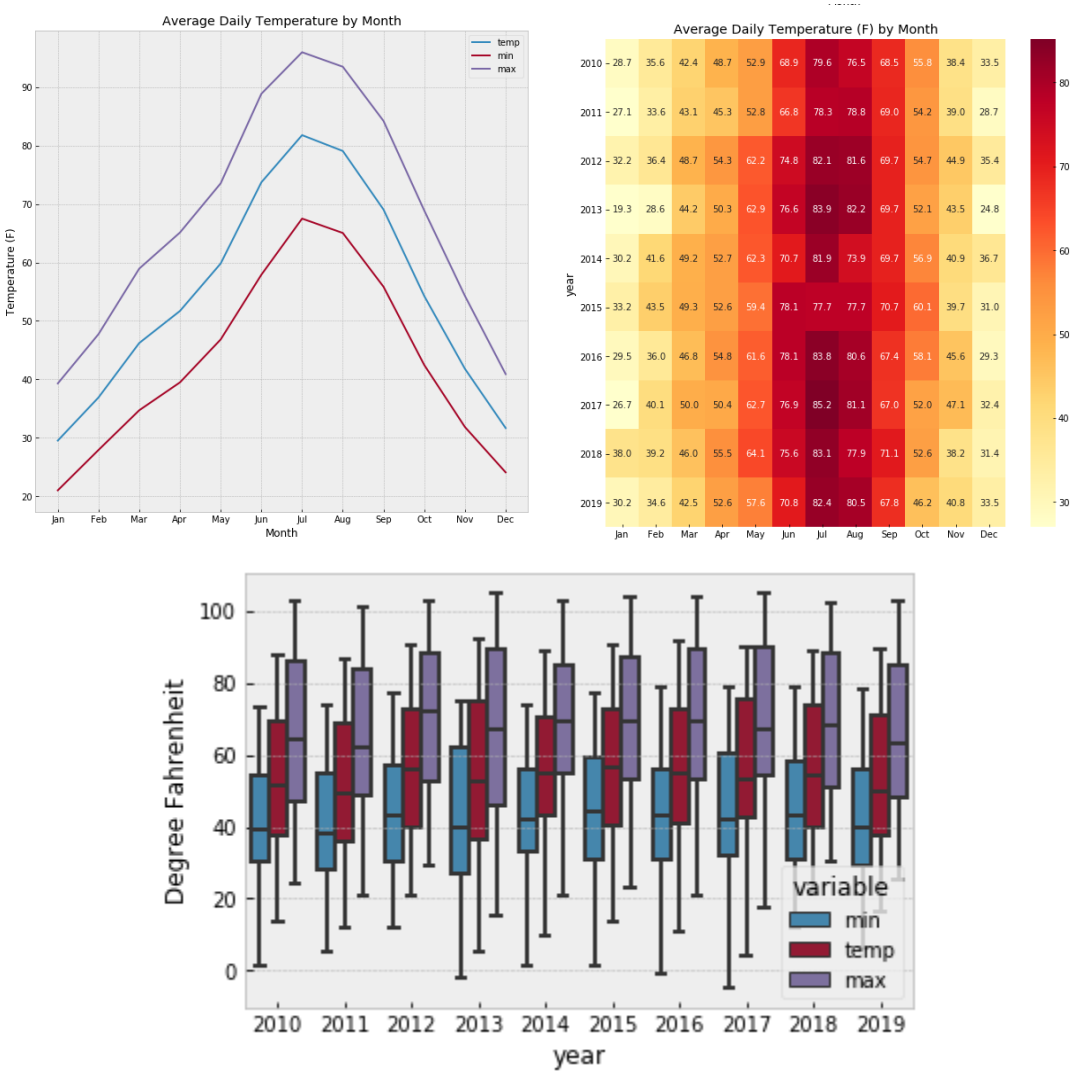

Figure 4: Average Daily Temperatures in Salt Lake City (2010-2020).

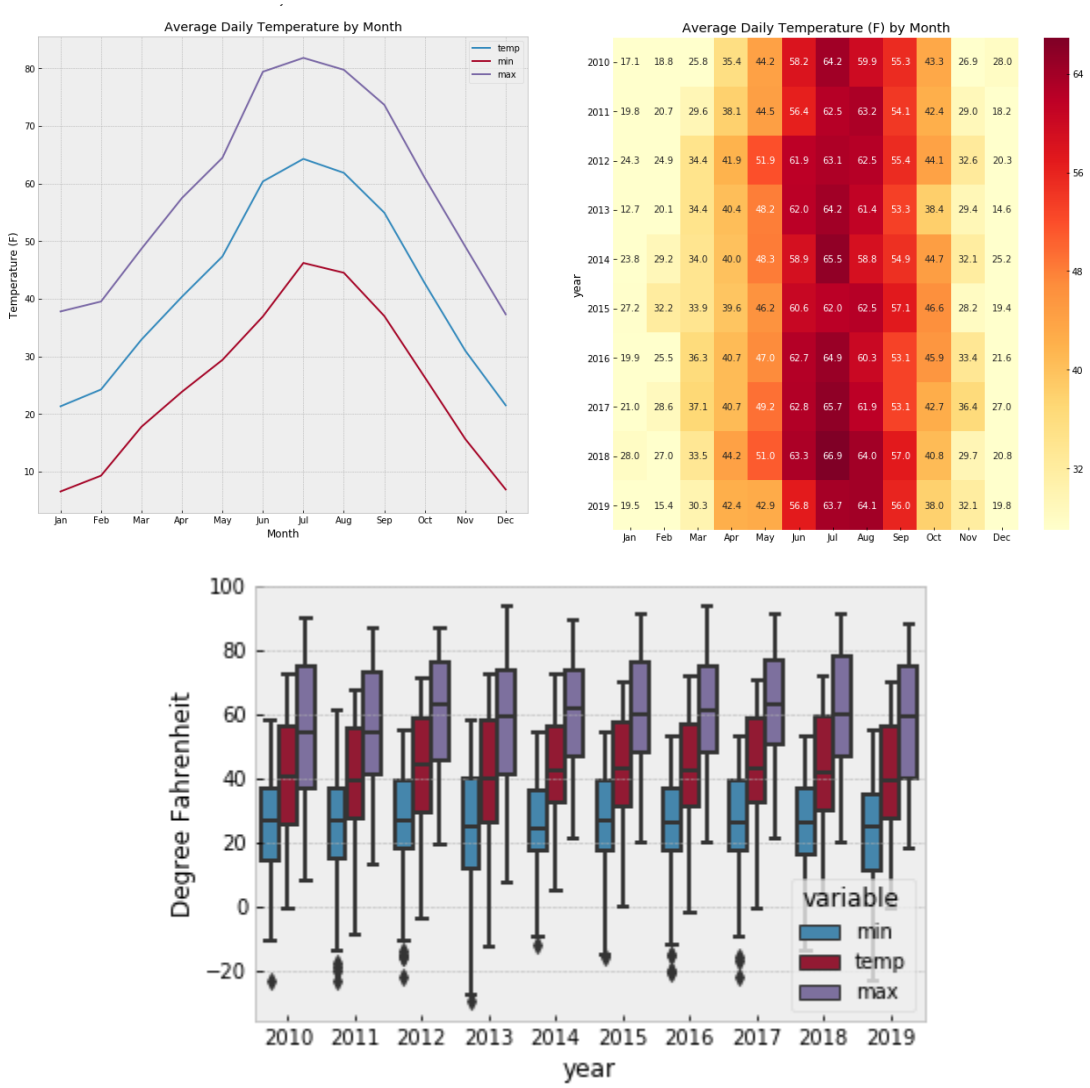

Figure 5: Average Daily Temperatures in Bryce Canyon Airport (2010-2020).

VI. Data Transformations

The selection of Salt Lake City International and Bryce Canyon Airport as weather stations allowed for a wide geographic variation because SLC is an urban location and Bryce Canyon is located in a mountainous region. Each location's unique latitude and longitude filters were used. This method made it possible to examine temperature patterns at these two different Utah locales more closely. The filtered datasets were then the subject of further analysis, now customized to the study's geographic setting.

The use of PySpark to create Spark DataFrames was one important shift. The dataset was transformed into Spark DataFrames through Apache Spark's distributed processing. This change

made it easier to handle massive amounts of data efficiently and prepared the way to take advantage of Spark's machine learning features.

Temporal features were engineered by specifying the date to extract additional insights from the dataset. Timestamp information was utilized to derive the year, month, and day of each weather observation. This temporal granularity became instrumental in exploring seasonal variations, identifying trends, and aligning the data for subsequent analyses.

The use of Spark's StandardScaler allowed for the standardization of features. By putting all of the features that went into the modeling process on a common scale, this transformation guaranteed that no feature would dominate the model because of variations in its magnitude. The datasets were further improved by removing superfluous columns and organizing them in a more logical order. The removal of redundant data improved the coherence of the datasets and made it easier to evaluate the results of further analysis.

Latitude and longitude data were kept for every weather observation, which added to the analysis' spatial context. These spatial coordinates made it easier to create interactive maps that show the temperature distributions in various Utah sites.

[31]:
```
#Create a Spark Dataframe for SLC Weather dataset
sp1_SLCweather = spark.createDataFrame(cleanSLCWeather)
```

[32]:
```
# Extracting timestamp format
sp1_SLCweather = sp1_SLCweather.withColumn("Year", year(sp1_SLCweather["Date"]))
sp1_SLCweather = sp1_SLCweather.withColumn("Month", month(sp1_SLCweather["Date"]))
sp1_SLCweather = sp1_SLCweather.withColumn("DayOfMonth", dayofmonth(sp1_SLCweather["Date"]))
sp1_SLCweather = sp1_SLCweather.drop('Date')
```

[33]:
```
# Drop rows with any missing values
SLC1_clean = sp1_SLCweather.na.drop()
# Drop duplicates
SLC1_clean = SLC1_clean.dropDuplicates()
```

Figure 6: Data Transformation and Preprocessing.

VII. Model Building and Results

The project's final phase, known as "model building," involved using the enriched datasets to create predictive models that can be used to identify temperature trends in Utah. This

phase's key components were the implementation and optimisation of the Gradient Boosting, Random Forest, and Linear Regression models using Spark's machine learning capabilities.

The baseline model was linear regression and represented the linear relationships between the predictor variables and the average temperature. The interpretability and simplicity of the model allowed for important insights into how features directly affect temperature variations. In order to identify complex patterns and non-linear relationships in the data, Random Forest and Gradient Boosting models were developed. Since these ensemble models are naturally able to manage intricate relationships.

[34]:
```
# First, assemble the features into a vector
assembler = VectorAssembler(inputCols=
        ["Year", "Month", "DayOfMonth", "Min Temp", "Max Temp"], outputCol="features_vec2")
```

[35]:
```
# Standardize features
scaler = StandardScaler(inputCol="features_vec2", outputCol="scaledFeatures2", withStd=True, withMean=False)
```

[36]:
```
# Split the data
(train_data3, test_data3) = SLC1_clean.randomSplit([0.8, 0.2], seed=42)
```

Figure 7: Creation of Assembler and Scaler.

An assembler was created to assemble the features into a single vector. This vector was then scaled using a StandardScaler and split into a training and testing set (Figure 7).

```
# Initialize Regression Models
lr_1 = LinearRegression(featuresCol="scaledFeatures2", labelCol="Avg Temp")
rf = RandomForestRegressor(featuresCol="scaledFeatures2", labelCol="Avg Temp")
gbt = GBTRegressor(featuresCol="scaledFeatures2", labelCol="Avg Temp")
```

```
# Creating Pipelines
lr_pipeline = Pipeline(stages=[assembler, scaler, lr_1])
rf_pipeline = Pipeline(stages=[assembler, scaler, rf])
gbt_pipeline = Pipeline(stages=[assembler, scaler, gbt])
```

Figure 8: Creation of Pipeline for each model.

Each of the three models were initialized using the scaled features vector and the target column of “Avg Temp”. A pipeline was created for each of the three models. This pipeline consisted of three stages including the assembler, scaler and model (Figure 8).

[41]:
```
# Define parameter grids
param_grid_rf = (ParamGridBuilder()
                 .addGrid(RandomForestRegressor.numTrees, [50, 100, 150])
                 .addGrid(RandomForestRegressor.maxDepth, [3, 5, 7])
                 .build())

param_grid_gbt = (ParamGridBuilder()
                  .addGrid(GBTRegressor.maxIter, [10, 20, 30])
                  .addGrid(GBTRegressor.maxDepth, [3, 5, 7])
                  .build())
```

[42]:
```
# Create evaluator
evaluator = RegressionEvaluator(labelCol=target_column, predictionCol='prediction', metricName='r2')
```

[43]:
```
# Perform grid search for each pipeline
rf_cv = CrossValidator(estimator=rf_pipeline, estimatorParamMaps=param_grid_rf, evaluator=evaluator, numFolds=5)
gbt_cv = CrossValidator(estimator=gbt_pipeline, estimatorParamMaps=param_grid_gbt, evaluator=evaluator, numFolds=5)
```

Figure 9: Parameter Hypertuning and Cross-Validation.

Grid search was used to rigorously optimize the models in order to find the optimal hyperparameters. This was only possible for Random Forest and Gradient Boosting models since Linear Regression does not have any hyperparameters. An evaluator was also created using the "Avg Temp" and 'prediction' along with the 'r2' score. This evaluator was used in cross-validation for each of the models to determine the best parameters and testing scores (Figure 9). For both Random Forest and Gradient Boosting the optimal "numTrees" were 20 and "numFeatures" were 5.

Key metrics such as R-squared, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) were used to assess the performance of the model. These metrics were used as standards to evaluate the precision, accuracy, and generalizability of the models to new data.

The modeling phase produced some notable results. Despite being more straightforward, the Linear Regression model offered a reliable baseline with comprehensible coefficients. Following grid search optimization, the Random Forest and Gradient Boosting models demonstrated good predictive performance, capturing complex patterns; however, they did not outperform the baseline in terms of accuracy. The best model for forecasting temperature trends was found by comparing the results of the final models, which were validated on a test dataset. The best option was the Linear Regression model for both SLC and mountain weather data since it provided a fair trade-off between interpretability and accuracy. This model also had the least mean absolute error compared to the rest of the models. This was a great determiner for the best model in this scenario along with a high R2 score. The scores and their comparisons can be viewed below in (Figure 10) and (Figure 11).

	Model	R2	MSE	RMSE	MAE
0	Linear Regression	0.985859	5.035995	2.244102	1.675267
1	Random Forest	0.979784	7.199690	2.683224	2.018609
2	Gradient Boosting	0.981850	6.463749	2.542390	1.927441

Figure 10: Validation Metrics of Salt Lake City.

	Model	R2	MSE	RMSE	MAE
0	Linear Regression	0.977922	6.041322	2.457910	1.877220
1	Random Forest	0.967351	8.933808	2.988948	2.253013
2	Gradient Boosting	0.975431	6.722833	2.592843	1.987717

Figure 11: Validation Metrics of Bryce Canyon Airport.

VIII. Future Works

After the completion of the model training, both the SLC and Mountain data could be used to create a validation set of the data. This validation set would be used to test the model from the years 2020 to 2021. This would assist in determining whether the model did perform as expected by allowing identification of any potential overfitting or underfitting issues and provides insights into the model's robustness. After validating the dataset it is then possible to use the trained model to predict future weather values. This would be similar to weather forecasting as it would use previously seen data to predict what the weather should be in the years 2022, and 2023. This will then be compared to the actual data for the years 2022 and 2023 from the dataset to determine the accuracy of the results. Weather forecasting is essential in daily life and the trends determined by the Linear Regression model using Spark would be pertinent for this cause.