GEORGETOWN UNIVERSITY

ANLY 502 - Massive Data Fundamentals

Weather Prediction Project

Members:

Name:	Kuiyu Zhu	Guiming Xu	Zihao Zhou	Yuxuan Yao
NetID:	kz175	gx26	zz267	yy560

Introduction

Background

Because of the exponential development in urban departments and increasing number of industries in the Metropolitan area, the change in global surface is turning from good condition to bad. The problems of climate change are increasingly concerned by people and scientists. When it comes to climate and global surface, predicting some surface indicators seems particularly important.

Dataset description

The data we use in this project is Global Surface Summary of Day, which is a collection of daily weather measurements (temperature, wind speed, humidity, pressure, and more) from 9000+ weather stations around the world. The data is highly structured without any missing values. In addition, the data is intended for free and unrestricted use in research, education, and other non-commercial activities.

The data can be accessed through Amazon AWS Resource with name `arn:aws:s3:::aws-gsod`, and the data size is 20.1 GB.

Prediction Models

Data Preprocessing

In the S3 bucket, our data is saved in many folders, each of which represents a year. Within the folder there are multiple CSV files recording the weather measurements for that year. Therefore, we use wildcard to search, read and merge multiple CSV files in different folders.

```
|-- ID: string (nullable = true)
|-- USAF: string (nullable = true)
|-- WBAN: string (nullable = true)
|-- Elevation: string (nullable = true)
|-- Country_Code: string (nullable = true)
|-- Latitude: string (nullable = true)
|-- Longitude: string (nullable = true)
|-- Date: string (nullable = true)
|-- Year: string (nullable = true)
|-- Month: string (nullable = true)
|-- Day: string (nullable = true)
|-- Mean_Temp: string (nullable = true)
|-- Mean_Temp_Count: string (nullable = true)
|-- Mean_Dewpoint: string (nullable = true)
|-- Mean_Dewpoint_Count: string (nullable = true)
|-- Mean_Sea_Level_Pressure: string (nullable = true)
|-- Mean_Sea_Level_Pressure_Count: string (nullable = true)
|-- Mean_Station_Pressure: string (nullable = true)
|-- Mean_Station_Pressure_Count: string (nullable = true)
|-- Mean_Visibility: string (nullable = true)
|-- Mean_Visibility_Count: string (nullable = true)
|-- Mean_Windspeed: string (nullable = true)
|-- Mean_Windspeed_Count: string (nullable = true)
|-- Max_Windspeed: string (nullable = true)
|-- Max_Gust: string (nullable = true)
|-- Max_Temp: string (nullable = true)
|-- Max_Temp_Quality_Flag: string (nullable = true)
|-- Min_Temp: string (nullable = true)
|-- Min_Temp_Quality_Flag: string (nullable = true)
|-- Precipitation: string (nullable = true)
|-- Precip_Flag: string (nullable = true)
|-- Snow_Depth: string (nullable = true)
|-- Fog: string (nullable = true)
|-- Rain_or_Drizzle: string (nullable = true)
|-- Snow_or_Ice: string (nullable = true)
|-- Hail: string (nullable = true)
|-- Thunder: string (nullable = true)
|-- Tornado: string (nullable = true)
```

Figure 1: all predictors

We totally have 38 columns in our data and the data is highly structured, which means they are all pre-processed and they are all numbers. In that case, we are easy to do the processing work and they are good for modeling. The basic method we process the data is, selecting the variables we think are useful in prediction and then transforming their string types into float types. Finally, we drop the rows with null value and this is the advantage of highly structured data while it is the shortage in data exploring. As well as in building the pipeline, we don't have the stage to encode the categorical string columns into integers, so that reduces our burden in pipeline work.

Tornado Prediction

Before modeling, we also do some exploratory data analysis.To begin with, we find the count of tornadoes happened groupby month, and also find the ratio of tornadoes. Through these tables, we can find tornadoes rarely occur.

Month_Tornado	0.0	1.0
5.0	7623008	1811
10.0	7404850	1410
1.0	7311418	891
6.0	7341287	2250
9.0	7228248	1878
2.0	6791996	798
12.0	7208320	871
7.0	7526712	2512
3.0	7603247	1041
11.0	7079264	1010
8.0	7559303	2130
4.0	7359652	1209

Figure 2: Tornado occurred each month

Tornado	count
1.0	25500
0.0	140816194

Figure 3: Count of Tornado

Because this dataset has over 100 million rows, we just select a small portion of it, and draw the plot to further confirm that this dataset is imbalanced.

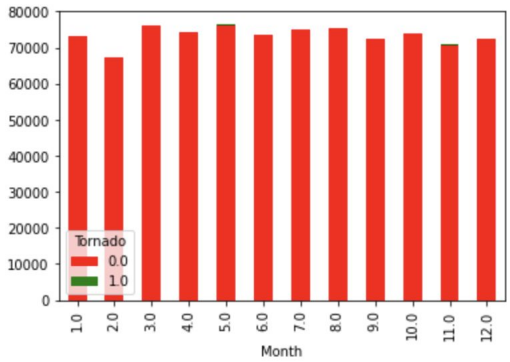

Figure 4: Barplot of Tornado

And then, we picked several important features, and tried to find the correlation between them and Tornado. it is not difficult to see that the distributions are not uniform.

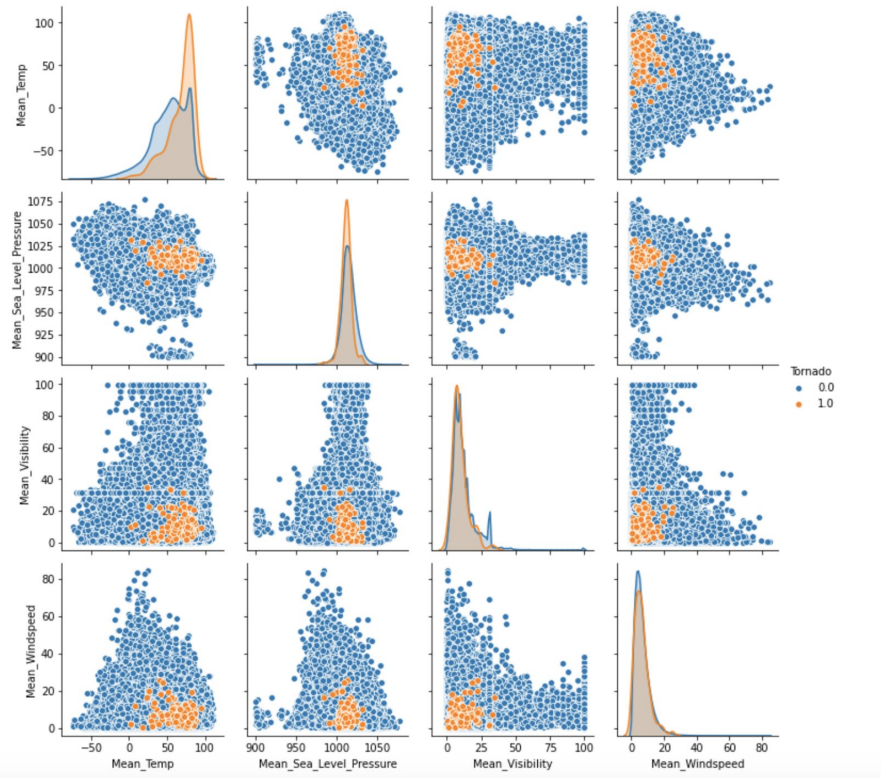

Figure 5: pairplot of Tornado and other features

Our first model is to predict tornado, which is a classification problem. In the data set, we selected 13 predictors.

```
features = ['Mean_Sea_Level_Pressure',
 'Max_Temp_Quality_Flag',
 'Elevation',
 'Min_Temp',
 'Mean_Temp',
 'Fog',
 'Max_Temp',
 'Latitude',
 'Min_Temp_Quality_Flag',
 'Mean_Visibility',
 'Longitude',
 'weights',
 'Mean_Windspeed']
```

Figure 6: Selected Features for Random Forest

For the section of data preprocessing, we firstly choose to fill the null value by their mean or just drop it and then use a standardscaler. we try a different collection of predictors, and one of them we select all 13 features, and another way is to delete windspeed, because we know there may be a high correlation between Tornado and wind speed. The biggest challenge of this classification problem is imbalanced target features, so we added a column called weight, when tornado is equal to 1, the weight in this row is equal to the ratio of tornado happened, and when tornado is equal to 0, the weight in this row is the ratio of tornado not to occur. Finally, through the random forest, the roc with wind speed is 1, and without wind speed is 0.99. However, because of an imbalanced dataset, if we just use ROC, it may lead to an accuracy paradox, resulting in meaningless prediction. Therefore, the f1 score or confusion matrix may be better.

Rain Prediction

This part of this project selects 14 columns as the predictors and *Rain_or_Drizzle* as the label.

```
root
 |-- Elevation: float (nullable = true)
 |-- Latitude: float (nullable = true)
 |-- Longitude: float (nullable = true)
 |-- Month: float (nullable = true)
 |-- Mean_Temp: float (nullable = true)
 |-- Max_Temp: float (nullable = true)
 |-- Min_Temp: float (nullable = true)
 |-- Mean_Dewpoint: float (nullable = true)
 |-- Mean_Sea_Level_Pressure: float (nullable = true)
 |-- Mean_Station_Pressure: float (nullable = true)
 |-- Mean_Visibility: float (nullable = true)
 |-- Fog: float (nullable = true)
 |-- Mean_Windspeed: float (nullable = true)
 |-- Precipitation: float (nullable = true)
 |-- Rain_or_Drizzle: float (nullable = true)
```

Figure 7. Selected 14 features for rain prediction

Since there are many variables, we only select the ‘mean’ variables but not ‘max’ or ‘min’, like ‘mean_temp’, ‘mean_windspeed’. This task mainly implements the Logistic Regression, Decision Tree Classifier and Random Forest Classifier to do the classification models. The evaluation method of these above classifiers is BinaryClassificationEvaluator.

At the beginning, there is only one model for the rain prediction, which is Logistic Regression. However, we are not satisfied with its performance, and the ROC curve:

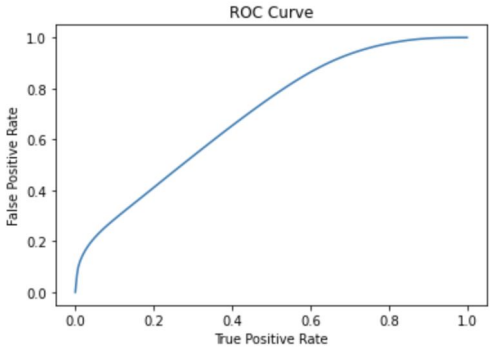

Figure 8. ROC curve of Logistic Regression model in rain prediction

Therefore, we decide to use more models and test them to improve our prediction result. Since there are many binary categorical columns in our data, we choose the tree methods. And here are the AUC results of LogisticRegression, DecisionTreeClassifier and RandomForestClassifier:

Model	AUC
Logistic Regression	**0.6993**
Decision Tree	**0.7993**
Random Forest	**0.8695**

Table 1. AUC of 3 models in rain prediction

As we can see, the best model in rain prediction is Random Forest.

Temperature Prediction

This part of the task selects 10 columns as features and *Mean_Temp* as the label.

This task mainly implements the Lasso Regression to select the features that are significantly correlated to the mean temperature and predict the mean temperature of a day given the other weather measurements. Since it is a penalized linear regression, there are some assumptions of the data:

1. Linearity: The relationship between the other weather measurements and the mean temperature is linear.
2. Homoscedasticity: The variance of residual is the same for any value of the other weather measurements.
3. Independence: Observations are independent of each other.
4. Normality: For any fixed value of the other weather measurements, the mean temperature is normally distributed.

The pyspark pipeline includes VectorAssembler, StandardScaler and LinearRegression (with *elasticNetParam* as 1 and *regParam* as 0.5).

Predictors	Coefficient
Elevation	-0.0051
Latitude	-0.0591
Longitude	0
Month	0
Mean_Dewpoint	0.9464
Mean_Sea_Level_Pressure	-0.0662
Mean_Station_Pressure	0
Mean_Visibility	0.1136
Mean_Windspeed	0
Precipitation	-2.3783

Figure 9. Coefficients of the Lasso Regression

Figure 9 shows the features selected by Lasso, from the coefficient we find that a higher elevation, a higher latitude, a higher mean sea level pressure and a higher precipitation are correlated to a lower mean temperature. While a higher mean dewpoint and a higher mean visibility are correlated to a higher mean temperature. Other predictors like longitude, month, mean station pressure and mean wind speed have little correlation with mean temperature.

The r-squared of the Lasso Regression model on the test set is 87%. It means that 87% of the variance for the mean temperature is predictable from the model's input weather measurements, so the model is good for predicting the mean temperature.

Conclusion

For classification models, since we have some categorical predictors in our data, tree models like random forest perform way more better than logistic regression.
Feature selection does really improve machine learning models and makes models more efficient(save running time), especially when we have a lot of features (or high-dimensional) in our big data tasks

Appendix

Codes:
Github: https://github.com/gu-yuxuanyao/502Project