An Application of Discrete Probability

1. Introduction

1.1. What is Probability?

In most people's cognition, the probability is only a subject of studying and analyzing random phenomena. Most of them think that probability theory belongs to a more professional but out of touch with the reality of the theoretical discipline. Not only that, most people have a wrong understanding of probability theory. But after further study, we not only realize these common pseudo-probability views but also realize that probability is also a discipline closely related to real life. Probability theory is widely used in people's daily life and business operations, such as insurance claims, forensic analysis, and so on. It can even be applied to more common and mundane aspects of everyday life, such as predicting our daily weather, which is exactly what we want to discuss in this proposal.

1.2. The Question of "Will it rains in Seattle?"

In our daily lives, we find an interesting fact. Many students haven't lived in a climate-like area like Seattle before, so after the rainy season in Seattle begins. The talk about the weather is becoming more and more frequent among students, there are many complaints and ridicule. Many students said that the first time they recognized that rain can be almost every day. Also, they are more eager to go out on a sunny day to enjoy the sunshine, so as to relieve the pressure of study.

In order to help people better plan their daily travel, avoid getting sick in the rain and cold, and live up to the precious sunshine, and at the same time, to deepen the understanding and application of probability science, we want to study the application of probability in rainfall prediction in this final after discussion. In this report, we plan to explain how to use Bayesian equations and linear regression models for rainfall probability calculations and show the relevant code and models we designed, as well as the final data conclusions.

1.3. Project Progress

After we set up our topic of predicting if it's going to rain in Seattle, we came to an agreement that we'll need a good amount of data in order to avoid any contingencies and build the perfect model. We were able to find a dataset collected by the Seattle Tacoma International Airport that contains complete records of daily rainfall patterns from 1948 to 2017. Besides whether or not it rained, it also tells the amount of precipitation, and the maximum and minimum temperature on that day which is also crucial information to our analysis.

Next, we applied what we learned about probability and functions from the class but with an emphasis on methods of the Naive Bayesian Model and Logistic Regression to the collected dataset. After dividing the dataset into train set and test set, we trained the prediction models and tested their accuracy. In this process, software of Python was used to implement our application.

2. Analysis

2.1. The Bayes' Rule in Discrete Probability

Usually, when we predict the probability of a certain event with some evidence, we use Bayes' Rule, which is an important probability theory proposed by Thomas Bayes in the 18th century. The common characteristic of its application in computer science is, according to the existing input, finding the items with the highest matching degree (that is, the probability of occurrence) in the huge existing database.

Let A, B two events, according to conditional probability, the relative probability of A and B occurring in the condition of the other occurs:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (1)$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \quad (2)$$

According to (1) and (2):

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A) \quad (3)$$

If P(A) is not equal to zero, then let (3) divide P(A) :

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \quad (4)$$

Therefore, when building the model of the target Y and its related feature vector X_1 ~ X_n:

$$P(y|x, \cdots x_n) = \frac{P(y)\Pi_{i=1}^{n} P(x_i|y)}{P(x_1, \cdots x_n)} \quad (5)$$

2.2. Naive Bayesian Model

The naïve Bayesian Model (NBM) is based on Bayes' Rule with the hypothesis of variables are independent. For a given training dataset, firstly we train the combined probability distribution of input and output and build up the model. Secondly, based on this model, for a given input x, we use NBM to find out the output y with the highest probability of post-test.

$$P(x|y \quad) = P(x_1|y)xP(x_2|y)x \cdots P(x_n|y) \quad (6)$$

The NBM algorithm is simple and convenient and useful for huge databases, where we use this method to predict the probability of the target variable. In python, we use the Gaussian method from the Sklearn Naïve Bayes package, which assumes that the feature follows a normal distribution.

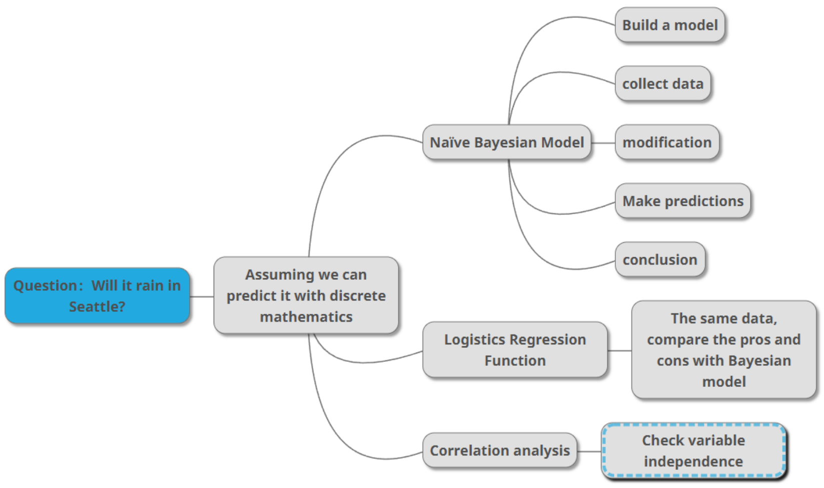
2.3. Logistics Regression Function

Building functions are important in discrete math, used in solving real-world problems when we need to explore the relationships between variables. Logistic regression is a useful function to establish a regression formula for the classification boundary line based on the existing data, to classify the classification. In this project, we use this logistic regression method because its algorithm code is simple, easy to implement, with a low cost of calculation and storage resources but a fast speed. In python, we can use Sklearn. Logistics regression package.

2.4. Scientific Question

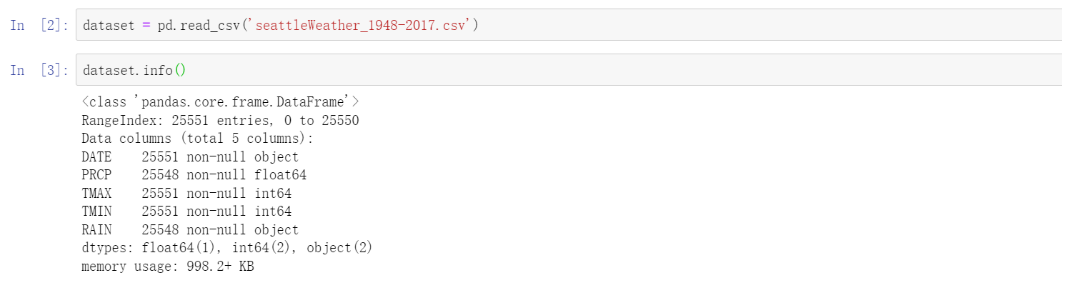
In this project, we plan to build a model that gives information on the previous days we can predict whether it will rain on a specific day. We collected data from a publicly available dataset on the NOAA website…..

2.5. Strategy


Question: Will it rain in Seattle?
Assuming we can predict it with discrete mathematics
Naïve Bayesian Model
Build a model
collect data
modification
Make predictions
conclusion
Logistics Regression Function
The same data, compare the pros and cons with Bayesian model
Correlation analysis
Check variable independence

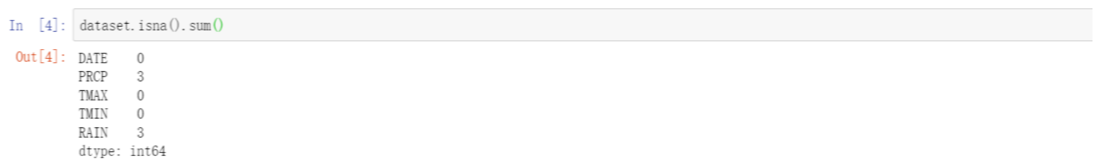
2.6. Data Cleaning

Using Jupyter to prompt all codes. We will use naive Bayes and logic regression models to make our predictions. First, read the CSV file and look at the dataset information, which contains 25551 rows×5 columns, with different types. Because each column is different in length, there is missing data in the column. Then the next step is we'll do the data cleaning.



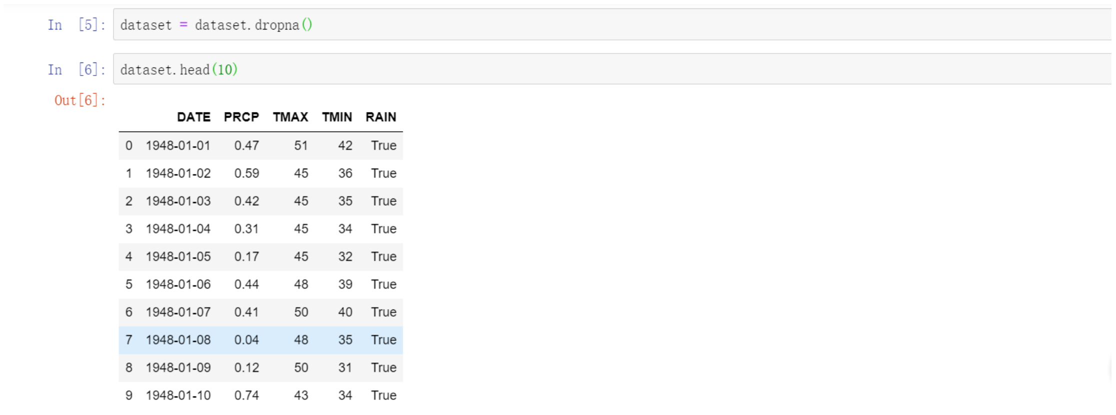
```
In  [2]: dataset = pd.read_csv('seattleWeather_1948-2017.csv')

In  [3]: dataset.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 25551 entries, 0 to 25550
         Data columns (total 5 columns):
         DATE    25551 non-null object
         PRCP    25548 non-null float64
         TMAX    25551 non-null int64
         TMIN    25551 non-null int64
         RAIN    25548 non-null object
         dtypes: float64(1), int64(2), object(2)
         memory usage: 998.2+ KB
```

Secondly, calculate the total number of nulls in each column.



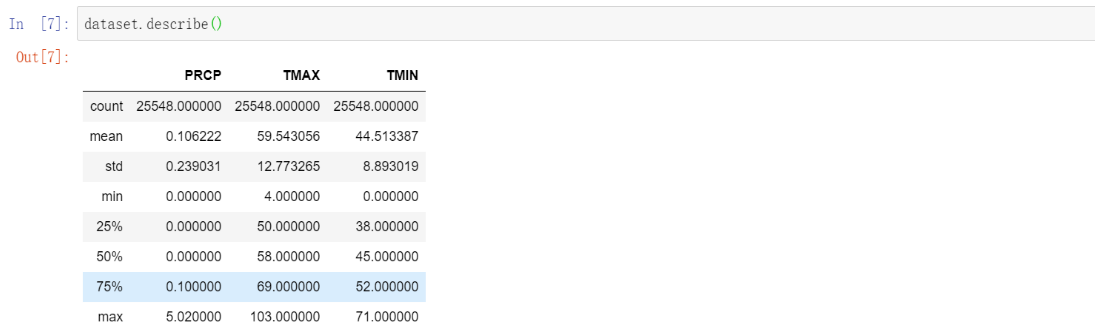
```
In  [4]: dataset.isna().sum()

Out[4]: DATE    0
        PRCP    3
        TMAX    0
        TMIN    0
        RAIN    3
        dtype: int64
```

Then, since the number of null data is small, which is 3, just drop them. The first 10 rows are displayed to check for the new dataset. There are 25548 rows × 5 columns in the dataset.



```
In  [5]: dataset = dataset.dropna()

In  [6]: dataset.head(10)

Out[6]:
```

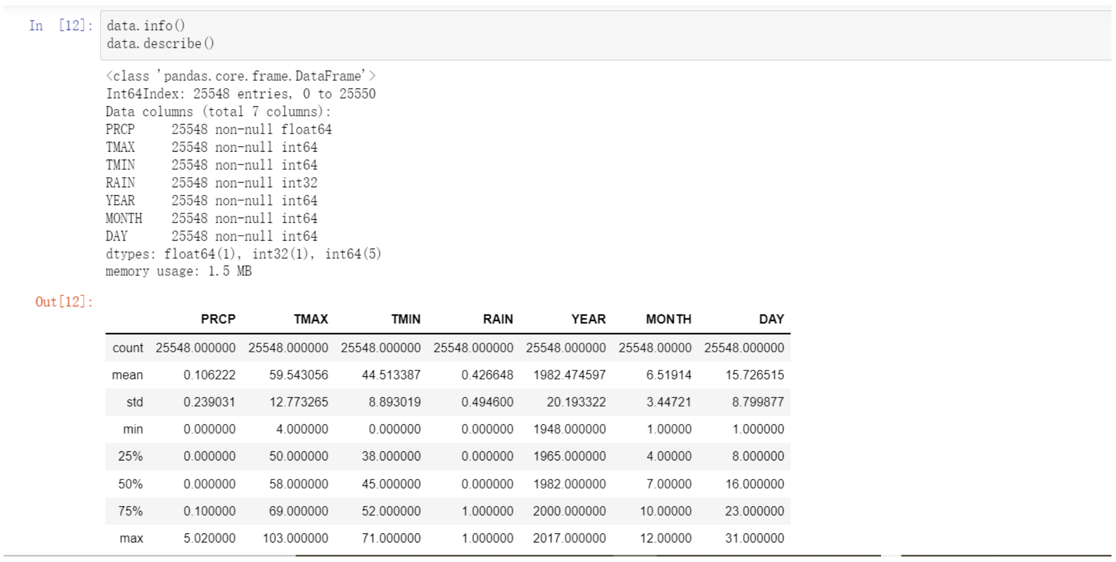
	DATE	PRCP	TMAX	TMIN	RAIN
0	1948-01-01	0.47	51	42	True
1	1948-01-02	0.59	45	36	True
2	1948-01-03	0.42	45	35	True
3	1948-01-04	0.31	45	34	True
4	1948-01-05	0.17	45	32	True
5	1948-01-06	0.44	48	39	True
6	1948-01-07	0.41	50	40	True
7	1948-01-08	0.04	48	35	True
8	1948-01-09	0.12	50	31	True
9	1948-01-10	0.74	43	34	True



```
In [7]: dataset.describe()
Out[7]:
```

	PRCP	TMAX	TMIN
count	25548.000000	25548.000000	25548.000000
mean	0.106222	59.543056	44.513387
std	0.239031	12.773265	8.893019
min	0.000000	4.000000	0.000000
25%	0.000000	50.000000	38.000000
50%	0.000000	58.000000	45.000000
75%	0.100000	69.000000	52.000000
max	5.020000	103.000000	71.000000

Also, should convert “RAIN” and “DATE” into int type. Now we have completed the data cleaning.



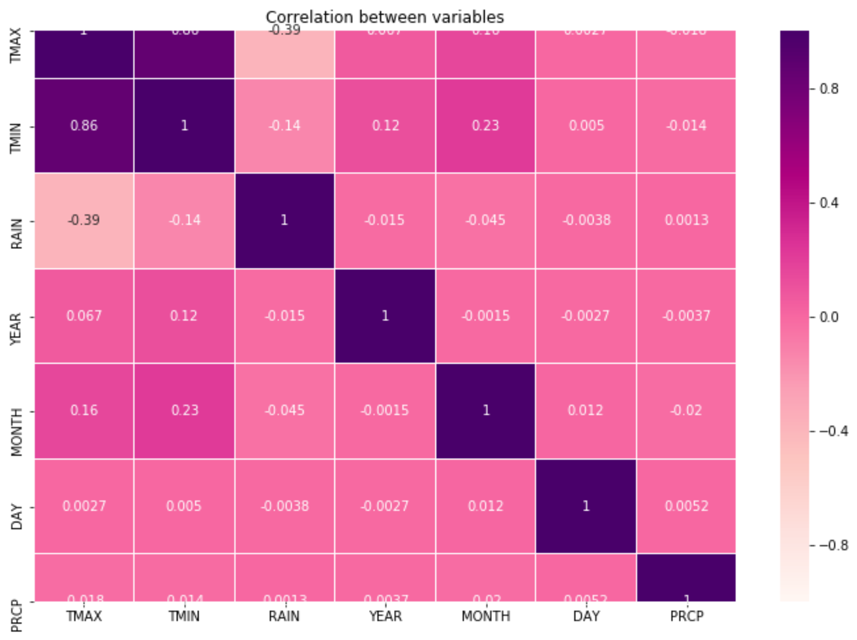
```
In [12]: data.info()
         data.describe()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 25548 entries, 0 to 25550
Data columns (total 7 columns):
PRCP     25548 non-null float64
TMAX     25548 non-null int64
TMIN     25548 non-null int64
RAIN     25548 non-null int32
YEAR     25548 non-null int64
MONTH    25548 non-null int64
DAY      25548 non-null int64
dtypes: float64(1), int32(1), int64(5)
memory usage: 1.5 MB
Out[12]:
```

	PRCP	TMAX	TMIN	RAIN	YEAR	MONTH	DAY
count	25548.000000	25548.000000	25548.000000	25548.000000	25548.000000	25548.00000	25548.000000
mean	0.106222	59.543056	44.513387	0.426648	1982.474597	6.51914	15.726515
std	0.239031	12.773265	8.893019	0.494600	20.193322	3.44721	8.799877
min	0.000000	4.000000	0.000000	0.000000	1948.000000	1.00000	1.000000
25%	0.000000	50.000000	38.000000	0.000000	1965.000000	4.00000	8.000000
50%	0.000000	58.000000	45.000000	0.000000	1982.000000	7.00000	16.000000
75%	0.100000	69.000000	52.000000	1.000000	2000.000000	10.00000	23.000000
max	5.020000	103.000000	71.000000	1.000000	2017.000000	12.00000	31.000000

Next, extract the X values, in this case, "TMAX", "TMIN", "RAIN", and Y values, or "PRCP". Since each variable has different units, for example, "PRCP" is "inches" and "TMAX" is "F", we should transfer all variables to the same scale. Then we get a dataset of X and Y.

2.7. Correlation

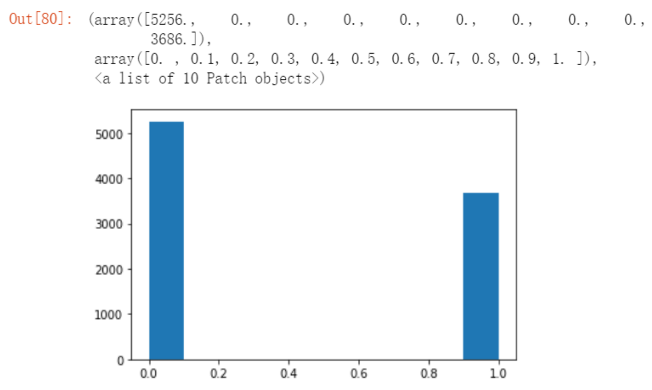
Check the correlation between variables. We defined a strong correlation with a coefficient of 0.8. So, from the results we know other than "TMAX" and "TMIN", there is no strong correlation between each variable, which means that they are independent of each other.



Correlation between variables

2.8. Model Results

In this case, when predicting the "PRCP" column, following the Naive Bayes principle, the false positive is a type II error, the hypothesis is wrong, but we did not reject it. False-negative is a type I error, assuming it is true, but we reject it. A common specification is that Type I is worse than Type II. When we split the dataset into 65% to train and 35% to test, the accuracy of the Navies Bayes is 60%.

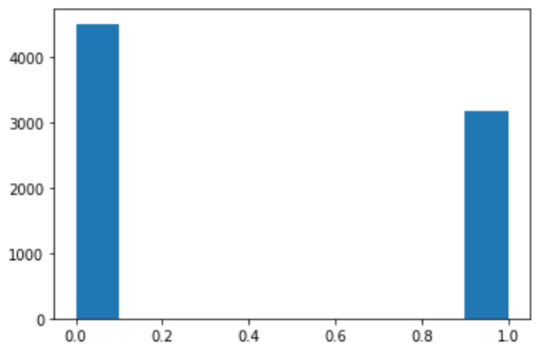


```
Out[80]: (array([5256.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
          3686.]),
         array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
         <a list of 10 Patch objects>)
```

results of Naive bayes: test-size = 0.65

When we split the dataset into 70% to train and 30% to test, the accuracy of the Navies Bayes is 60%.

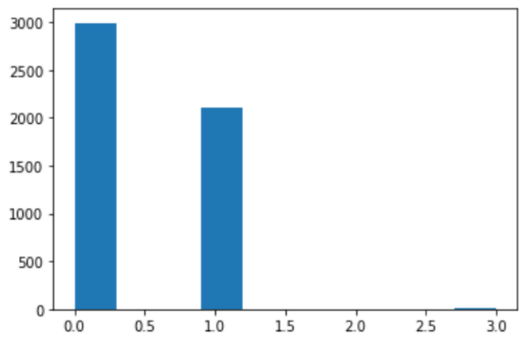
```
Out[77]: (array([4504.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
          3161.]),
          array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
          <a list of 10 Patch objects>)
```



results of Naive bayes: test-size = 0.7

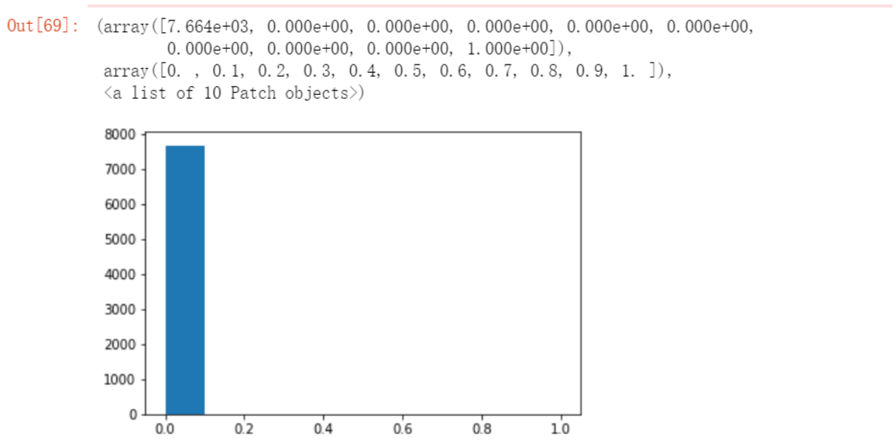
When we split the dataset into 80% to train and 20% to test, the accuracy of the Navies Bayes is 59.6%.

```
Out[73]: (array([2992.,    0.,    0., 2106.,    0.,    0.,    0.,    0.,    0.,
            12.]),
          array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ]),
          <a list of 10 Patch objects>)
```



results of Naive bayes: test-size = 0.8

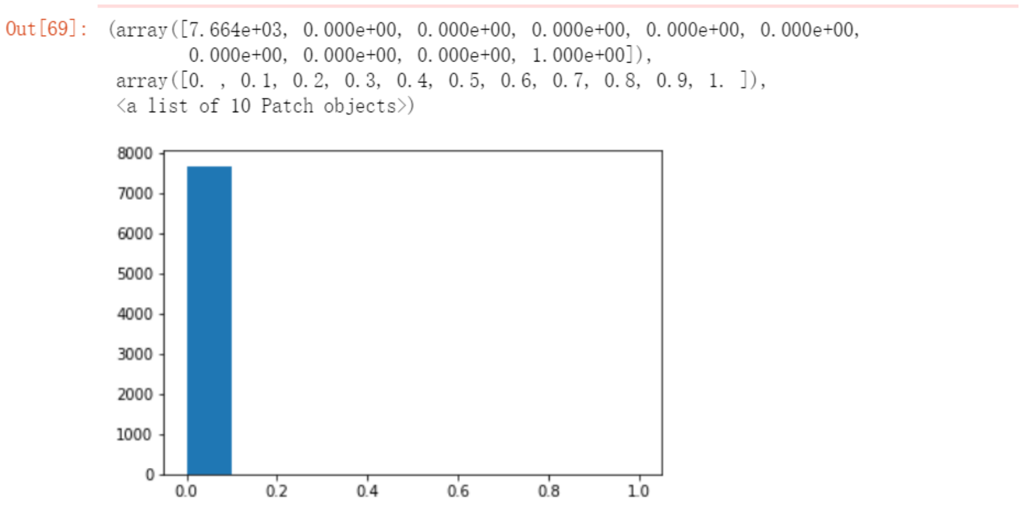
When we split the dataset into 70% to train and 30% to test, the accuracy of the Logistic Regression is 98.59 %.



```
Out[69]: (array([7.664e+03, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
                 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
          array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
          <a list of 10 Patch objects>)
```

results of Logistic Regression: test-size = 0.7

When we split the dataset into 80% to train and 20% to test, the accuracy of the Logistic Regression is 98.69 %.



```
Out[69]: (array([7.664e+03, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
                 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
          array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
          <a list of 10 Patch objects>)
```

results of Logistic Regression: test-size = 0.8

2.9. Compare Analysis

The simple understanding of the R^2 is the distance between the data in the test set and the regression model. The bigger the R^2, the better the model fits and the higher the credibility of the model. Hence we could use R^2 to indicate the accuracy of the model.

For the Bayes model, there is no difference among the 3 tests with different sizes of train datasets. All results are around 60%, the best performance occurred when the test size equals 0.65.

For the Logistic Regression model, there is no difference among the 2 tests with different sizes of train datasets. All results are around 60%, the best performance occurred when the test size equals 0.8.

Compared two models, the Logistic Regression model performs better.

3. Conclusion

3.1. Answer the Question

From the analysis and results above, we conclude the following points:

1. In this project, how to split the data set has little effect on the accuracy of the Naive Bayes model and logistic regression model. Therefore, both two models are acceptable from the perspective of describing data laws.
2. Given the dataset we obtain, the accuracy of the Naïve Bayes Model is around 60%;
3. Given the dataset we obtain, the accuracy of the logistics regression Model is around 98.6%;
4. From the current analysis, the logistic regression model performs better.

3.2. Weaknesses and Limitations

A weakness of our project comes from the Naïve Bayes Model. Though theoretically, compared to other classification techniques the Naïve Bayes Model has the minimum error rate. Under some circumstances, it does not. This is because the Naïve Bayes Model presumes that event attributes are independent, however, this is often not the case in practical applications…

3.3. What did we learn from this project?

….
Each member need to share something here…

4. Appendix

4.1. Reference:

1. Dataset:https://www.ncdc.noaa.gov/cdo-web/datasets/GHCND/stations/GHCND:USW00024233/detail
……

4.2. Source:

Please attach your code screenshot here…