Introduction to Machine Learning Assignment #1

Team ID: 15
Team member: 0416004 郭羽喬, 0416208 黃士軒, 0416025 呂翊愷, 0416022 楊旻學

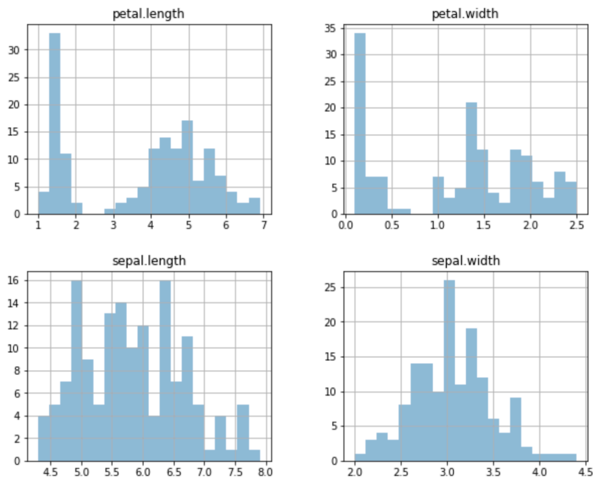
1. What environments the members are using

NCHC-TWGC Machine Learning Cloud service with V100 GPU

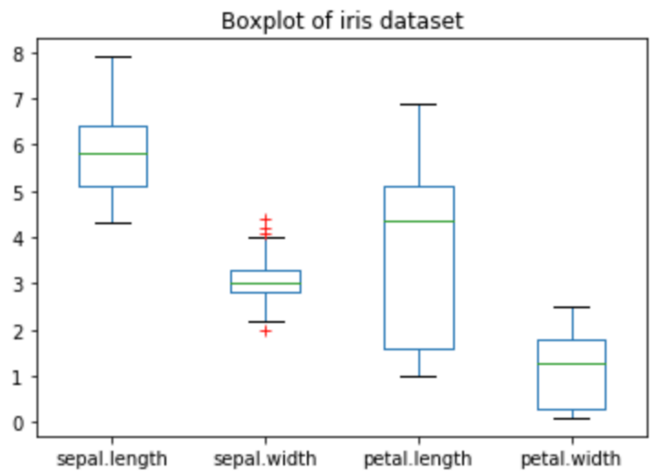
2. Basic statistic visualization of the data

Iris & Google Play:

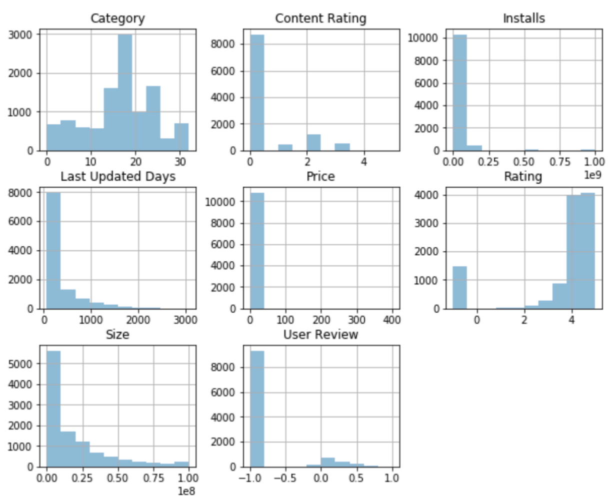
We use the same way to visualize Iris dataset and Google play dataset. After using **pandas** to read .csv files, we use **matplotlib.pyplot** to figure the histogram and plot box of the processed dataset. Through the histogram, we can visualize the distribution of every feature in the dataset, and average the data we want to use as a feature. Through the box plot, we can visualize the average and standard deviation of every feature in the dataset.



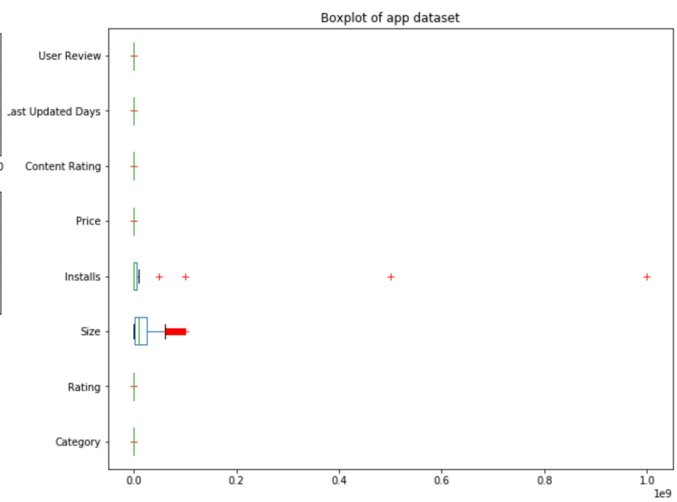
(a) Histogram of Iris dataset



(b) Boxplot of Iris dataset



(c) Histogram of Google Play dataset



(d) Boxplot of Google Play dataset

3. Data preprocessing methods

Iris:

The Iris features are put together in the original Iris data set. To build a training model, it is necessary to split the features into descriptive features and target features. Descriptive features is composed of sepal size and petal size, and target feature is composed of Iris class. We delete the forth column in the dataset and save the rest of the features as descriptive features. The forth column, which is Iris class, is served as target features.

Google Play:

Because the original google play store data contains many of un-analysis data format, we need to transform it. We replace all the string format to the numerical format. We abandon a corrupt data '**Life Made WI-Fi Touchscreen Photo Frame**' due to its unrecoverable format, additionally, we remove the data instances contain nan value in the "Rating" feature, because "Rating" is our predicted target, using nan "Rating" value is non-sense. The feature we will use are the Category, Reviews, Size, Installs, Price, Content Rating, Last Updated Days and User Review. After transforming the data format, we append these feature to google play dataset.

About "User Review" dataset, it is the custom attribute we obtained from the original attribute **Sentiment_Polarity** and **Sentiment_Subjectivity**. This new attribute is represented as user's feedback after using the APPs, and we use this feature with other feature above to conduct the following training.

4. How you generate decision tree and random forest models

The way we generate decision tree and random forest in Iris dataset and Google Play dataset is same. However, we use different data in K-fold Validation and Resubstituion Validation, so we are going to explain the difference between K-fold Validation and Resubstituion Validation.

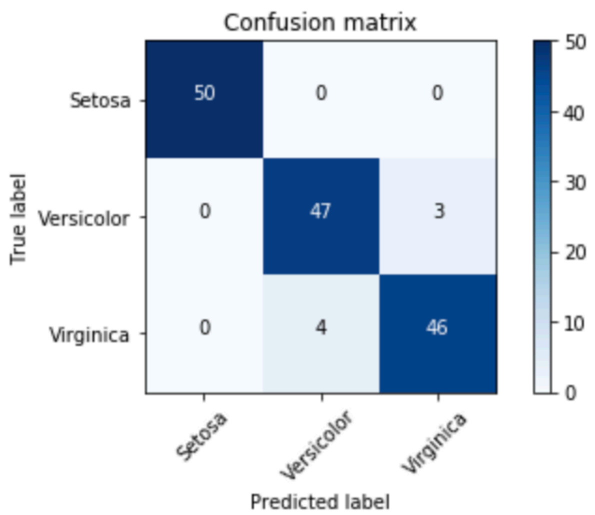
Decision Tree model with K-fold Validation:

First, we load the dataset and split the dataset into descriptive features and target feature. In Iris dataset, descriptive features is composed of sepal size and petal size, and target feature is composed of Iris class. Then, we use **sklearn.model_selection.KFold** to randomly choose 9/10 dataset to be the training set and the other 1/10 dataset to be the validation set. Training set is used to build the training model, and validation set is used to evaluate the performance of the prediction model in final.

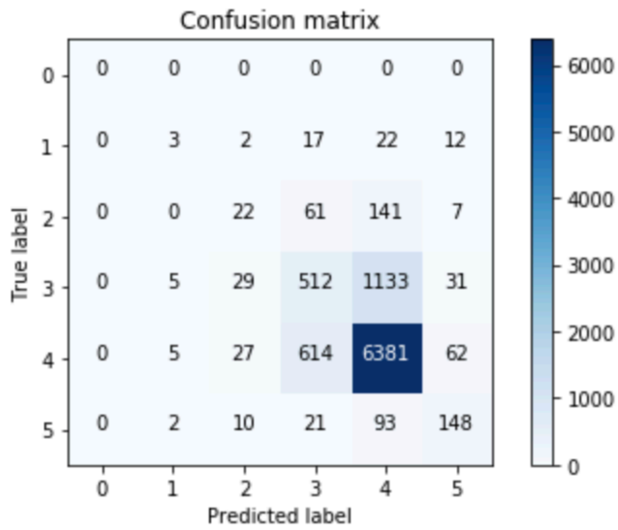
To generate the random forest, we randomly choose 70% of the data to build a decision tree 10 times so that we can get 10 different decision trees, which is also a random forest model. In order to implement attribute bagging, we randomly choose half of the attributes to delete and use the rest of the attributes to generate a decision tree every time. Then, we use **DecisionTreeClassifier()** in **sklearn.tree** to generate the decision tree classifier, and the classifier is be able to train the model by **clf.fit**. In addition, we also have a mode without doing attribute bagging, which simply puts 70% data into the prediction model without deleting the attributes. By doing so, we can observe the performance of attribute bagging.

After getting the random forest and the prediction model, we can use the model to get the prediction of validation set by **clf.predict** and then take the outcome of prediction as a vote. The prediction with the most votes wins.

Finally, we get the whole prediction generated by the random forest. To calculate the confusion matrix, we use **sklearn.metrics.confusion_matrix**. Pass the actual class and the predicted class into **sklearn.metrics.confusion_matrix**, and we can get a confusion matrix. Each row of the matrix represents the instances in a actual class while each column represents the instances in an predicted class. The accuracy can be calculated by dividing the number of data into the sum of the diagonal of the confusion matrix. To calculate the precision, we sum each entries in the same column of the confusion matrix up, and divide the summation into the true positive entry in the column. To calculate the recall, we sum each entries in the same row of the confusion matrix up, and divide the summation into the true positive entry in the row.



(e) Confusion matrix of Iris dataset with K-fold Validation



(f) Confusion matrix of Google Play dataset with K-fold Validation

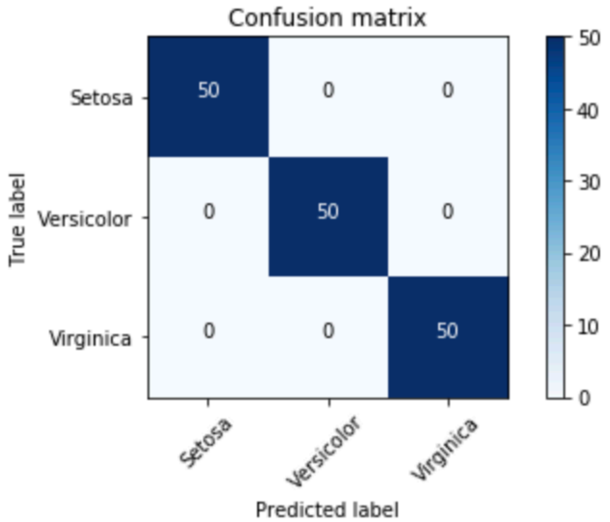
Decision Tree model with Resubstitution Validation:

First, we load the dataset and split the dataset into descriptive features and target feature. In Iris dataset, descriptive features is composed of sepal size and petal size, and target feature is composed of Iris class. As opposed to K-fold Validation, Resubstitution Validation doesn't split the data into training set and validation set. Instead, Resubstitution Validation use the same data to train and validate. Thus, we use **sklearn.model_selection.train_test_split** but the argument of test size is 0, which lead to the data not split.

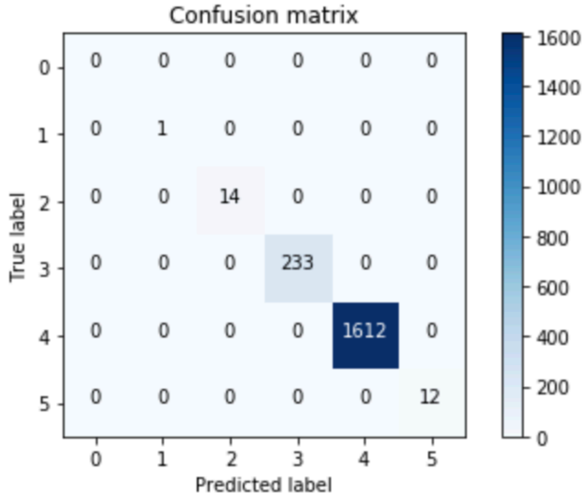
To generate the random forest, we use all of the data to build a decision tree 10 times so that we can get 10 different decision trees, which is also a random forest model. The approach of generating decision tree and attribute bagging is same as K-fold. In addition, we also have a mode without doing attribute bagging, which simply puts all of the data into the prediction model without deleting the attributes. By doing so, we can observe the performance of attribute bagging.

After getting the random forest and the prediction model, we can use the model to get the prediction of validation set by **clf.predict** and then take the outcome of prediction as a vote. The prediction with the most votes wins.

Finally, the approach of calculating confusion matrix, accuracy, precision, and recall is same as K-fold Validation.



(g) Confusion matrix of Iris dataset with Resubstitution Validation



(h) Confusion matrix of Google Play dataset with Resubstitution Validation

5. The performance

Iris:

Decision Tree model with K-fold Validation

accuracy : 0.953333
Setosa precision : 1.000000
Setosa recall : 1.000000
Versicolor precision : 0.921569
Versicolor recall : 0.940000
Virginica precision : 0.938776
Virginica recall : 0.920000

Decision Tree model with Resubstitution Validation

accuracy : 1.000000
Setosa precision : 1.000000
Setosa recall : 1.000000
Versicolor precision : 1.000000
Versicolor recall : 1.000000
Virginica precision : 1.000000
Virginica recall : 1.000000

Google Play:

Decision Tree model with K-fold Validation

accuracy : 0.754915
rating 0 precision : nan
rating 0 recall : nan
rating 1 precision : 0.200000
rating 1 recall : 0.053571
rating 2 precision : 0.244444

rating 2 recall : 0.095238
rating 3 precision : 0.417959
rating 3 recall : 0.299415
rating 4 precision : 0.821236
rating 4 recall : 0.900127
rating 5 precision : 0.569231
rating 5 recall : 0.540146

Decision Tree model with Resubstitution Validation

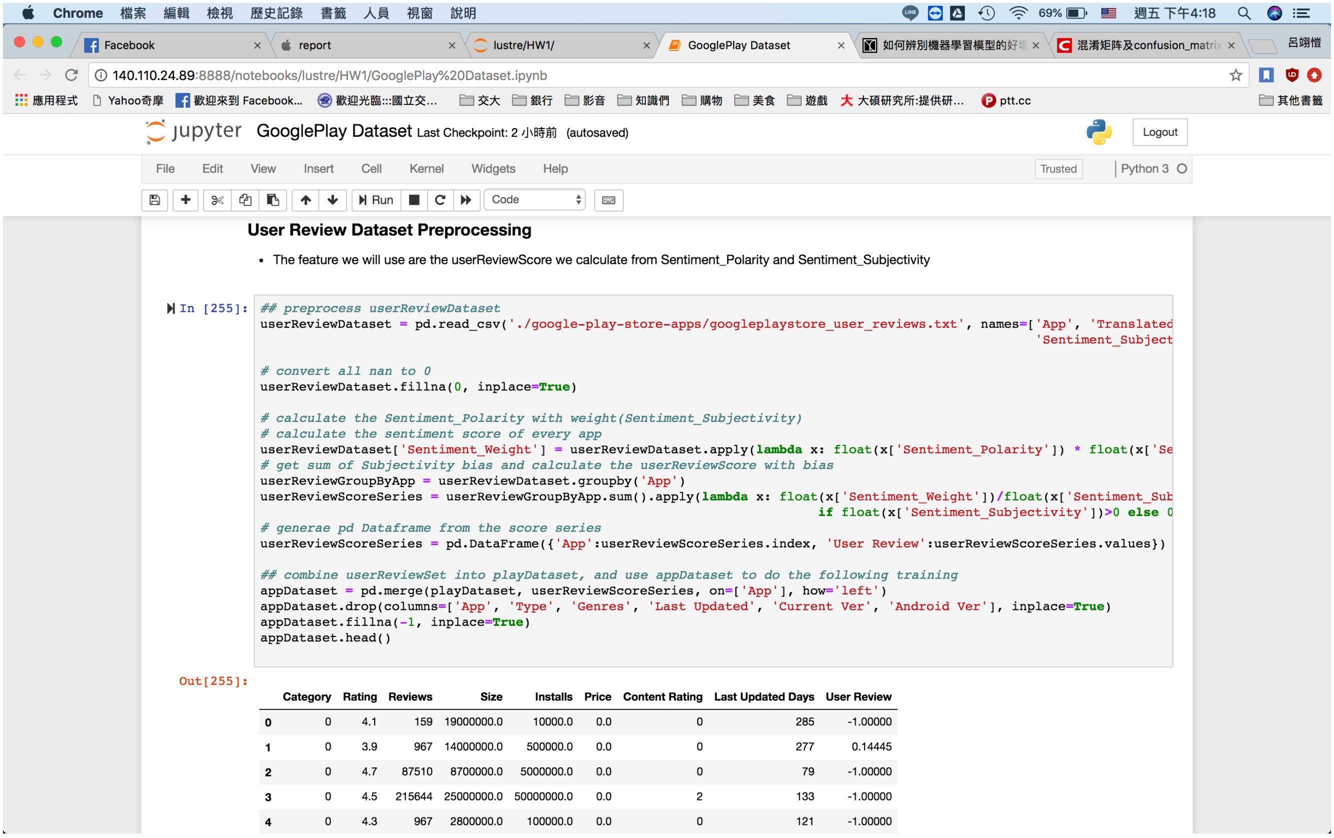
accuracy : 1.000000
rating 0 precision : nan
rating 0 recall : nan
rating 1 precision : 1.000000
rating 1 recall : 1.000000
rating 2 precision : 1.000000
rating 2 recall : 1.000000
rating 3 precision : 1.000000
rating 3 recall : 1.000000
rating 4 precision : 1.000000
rating 4 recall : 1.000000
rating 5 precision : 1.000000
rating 5 recall : 1.000000

6. Conclusion

By observing the final result, we discover that the performance of resubstitution validation is much better than K-fold validation, and we think the possible reason is that training data and test data are the same in resubstituion validation. Since it use the same data to train the prediction model, when the model predict the target of the test data, it should not be wrong. Thus, the accuracy of the decision tree model with resubstitution validation is approximately 100%. However, it’s not a case in the real world, reverse data for later validation is more sensible and can really reflect the performance of the model we generate, such as K-fold validation.

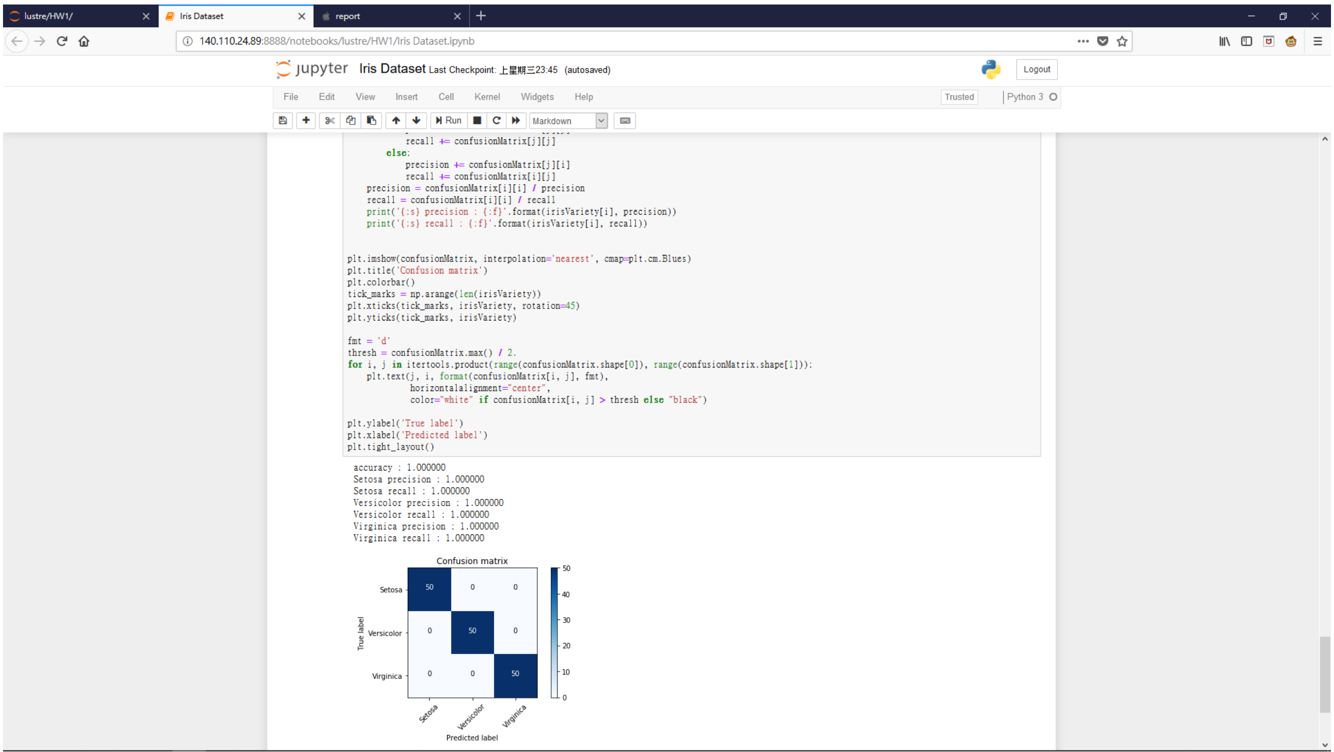
7. Every member's screenshot

0416025 呂翊愷

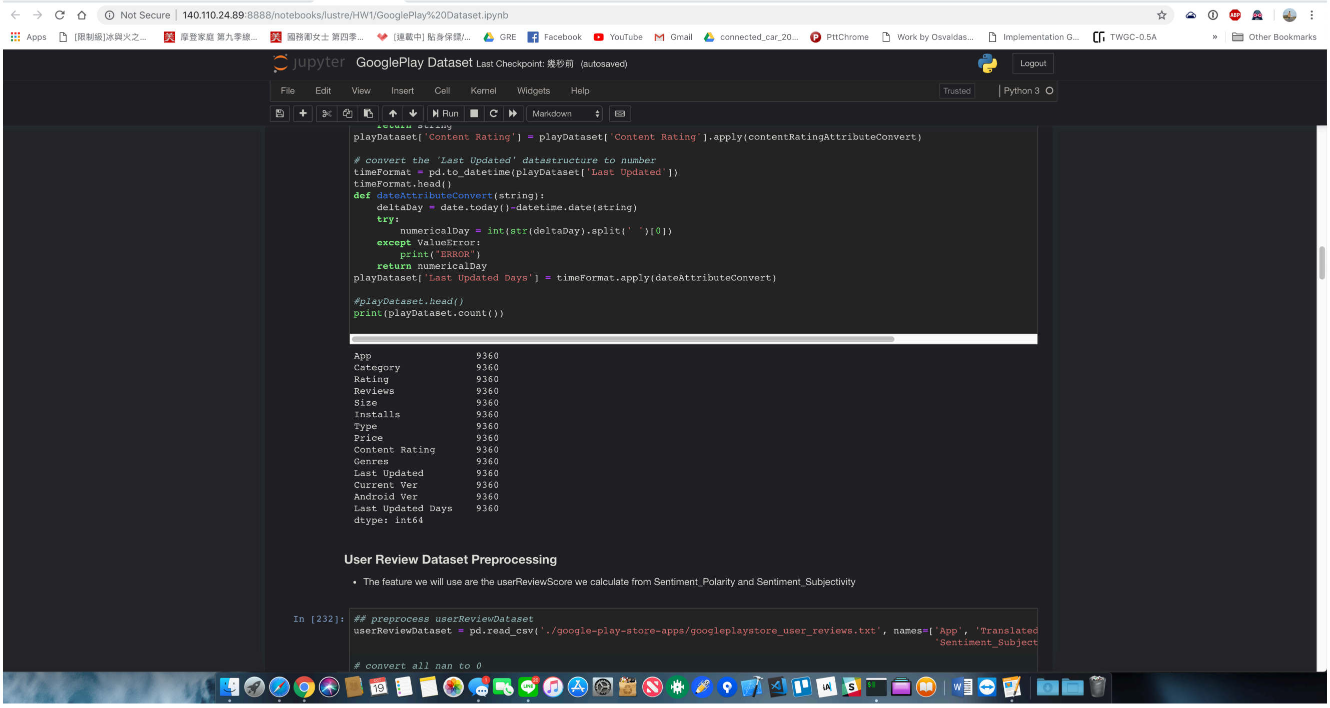


	Category	Rating	Reviews	Size	Installs	Price	Content Rating	Last Updated Days	User Review
0	0	4.1	159	19000000.0	10000.0	0.0	0	285	-1.00000
1	0	3.9	967	14000000.0	500000.0	0.0	0	277	0.14445
2	0	4.7	87510	8700000.0	5000000.0	0.0	0	79	-1.00000
3	0	4.5	215644	25000000.0	50000000.0	0.0	2	133	-1.00000
4	0	4.3	967	2800000.0	100000.0	0.0	0	121	-1.00000

0416022 楊旻學



0416208 黃士軒



0416004 郭羽喬