Machine Learning Project 1 (NCTU)

By: 費蓋德 0516251
Omodou Njie 0210895
方妮可 0416256

Environment and Packages we use

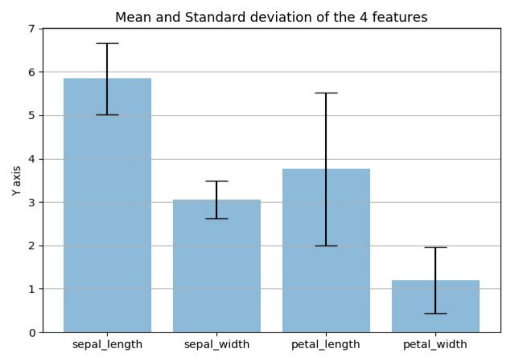
Programming Language: Python3

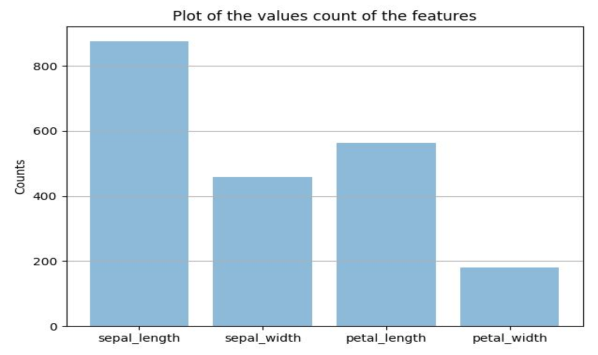
Packages: sklearn and its dependencies, numpy array, matplotlib, pandas, seaborn.

Compiler: SublimeText3 (run via macbook terminal)

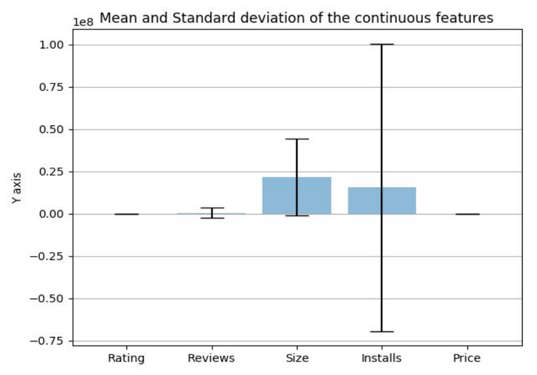
Visualization of the basic statistics

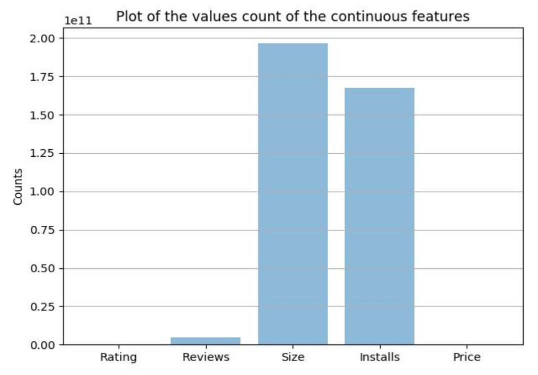
Iris Dataset





Googleplaystore





Data Preprocessing methods

We used various Data Preprocessing methods and for both datasets.

For iris Dataset:

This dataset was already quite clean. We just retrieve the data from the url: http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data, convert it to pandas dataframe for visualization then to numpy array before passing it into the algorithm.

For googleplaystore:

This dataset had some irrelevant informations. The first one we spotted was the row 10472 Thus we removed that one. For the column *"Price"*, some values which should be '0' are 'anyone' so we replace them accordingly. And also "*varies with device*" are all replaced by NaN for more convenience.

We then convert all of the features from discrete to continuous features except for our target class ("Category").

At last, as we did for iris dataset, we created a np.array out of the dataframe before passing it to the algorithm.

How we generated Decision Trees

We split the data into training and testing set

We called: DecisionTreeClassifier() from sklearn

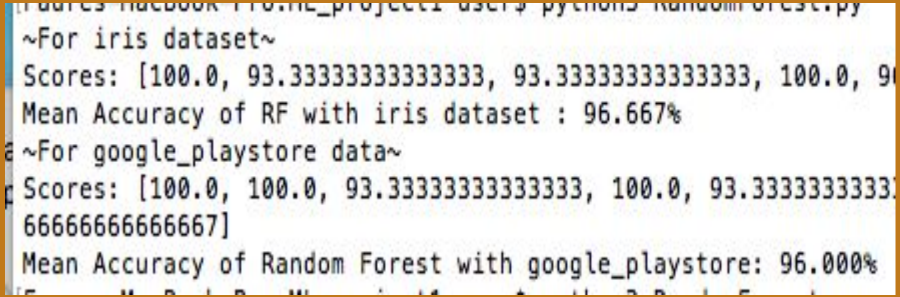
Fit our training data

Predict the performance of the tested data

How we generated Random Forests

Since we were not authorized to use any packages, we implemented the random forest in our own in the following steps:

- ★ Splitted the dataset into K-Folds (5 for Iris, 10 for googlePlaystore data)
- ★ Buit a decision tree (Just a model for testing many decision trees)
- ★ Split a dataset based on an attribute and an attribute value
- ★ Calculated the Gini index for a split dataset
- ★ Created child splits
- ★ Selected the best split point for a dataset based on the score of each tree
- ★ Evaluated the algorithm using a cross validation split
- ★ Loaded and tested our preprocessed data

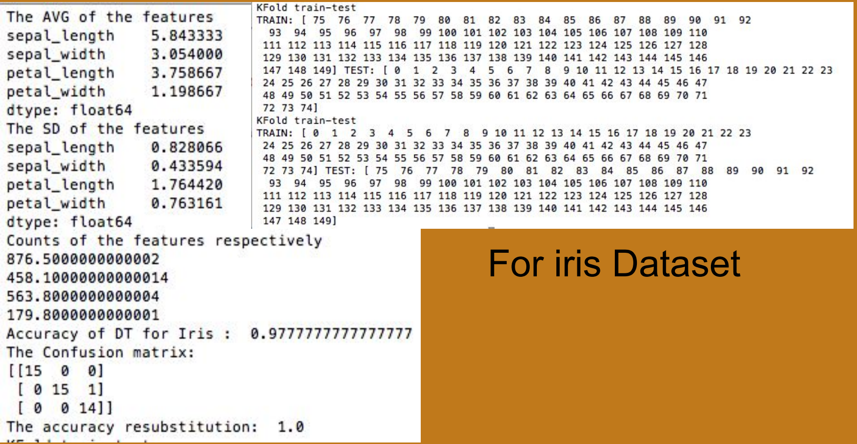


```
[Fadies-MacBook-Pro:ML_project1 user$ python3 RandomForest.py
~For iris dataset~
Scores: [100.0, 93.33333333333333, 93.33333333333333, 100.0, 9
Mean Accuracy of RF with iris dataset : 96.667%
~For google_playstore data~
Scores: [100.0, 100.0, 93.33333333333333, 100.0, 93.3333333333
6666666666667]
Mean Accuracy of Random Forest with google_playstore: 96.000%
```

Performance Evaluation

To evaluate the performance of our models, we used:

- ➔ Confusion Matrix and calculated TP, FP, FN, TN to confirm the accuracy obtained
- ➔ K-Fold cross Validation, Splitting our data into many testing and training sets and swapped training and testing to check if everything is correct.
- ➔ Resubstitution validation. Validate our model by using the same data as training and testing data.

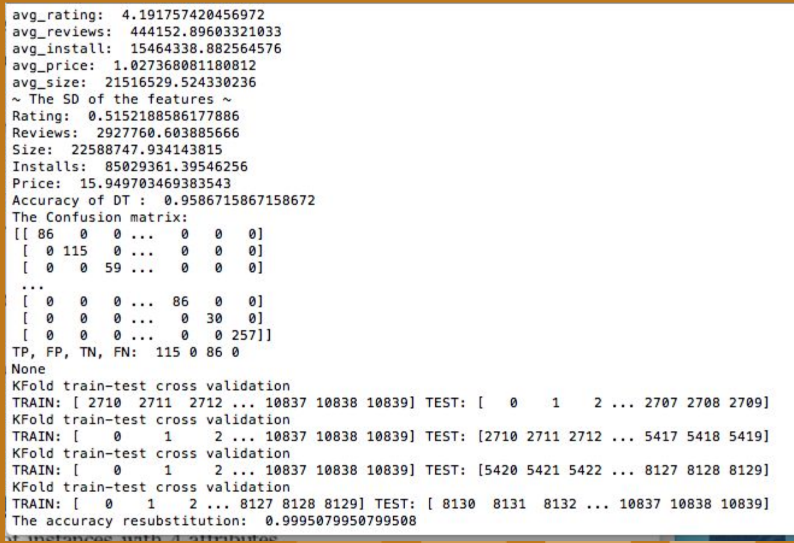


```
The AVG of the features
sepal_length    5.843333
sepal_width     3.054000
petal_length    3.758667
petal_width     1.198667
dtype: float64
The SD of the features
sepal_length    0.828066
sepal_width     0.433594
petal_length    1.764420
petal_width     0.763161
dtype: float64
Counts of the features respectively
876.5000000000002
458.10000000000014
563.8000000000004
179.8000000000001
Accuracy of DT for Iris :  0.9777777777777777
The Confusion matrix:
[[15  0  0]
 [ 0 15  1]
 [ 0  0 14]]
The accuracy resubstitution:  1.0
```

```
KFold train-test
TRAIN: [ 75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92
  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110
 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146
 147 148 149] TEST: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74]
KFold train-test
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74] TEST: [ 75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92
  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110
 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146
 147 148 149]
```

For iris Dataset

For Google PlayStore



```
avg_rating:  4.191757420456972
avg_reviews:  444152.89603321033
avg_install:  15464338.882564576
avg_price:  1.027368081180812
avg_size:  21516529.524330236
~ The SD of the features ~
Rating:  0.5152188586177886
Reviews:  2927760.603885666
Size:  22588747.934143815
Installs:  85029361.39546256
Price:  15.949703469383543
Accuracy of DT :  0.9586715867158672
The Confusion matrix:
[[ 86   0   0 ...   0   0   0]
 [  0 115   0 ...   0   0   0]
 [  0   0  59 ...   0   0   0]
 ...
 [  0   0   0 ...  86   0   0]
 [  0   0   0 ...   0  30   0]
 [  0   0   0 ...   0   0 257]]
TP, FP, TN, FN:  115 0 86 0
None
KFold train-test cross validation
TRAIN: [ 2710  2711  2712 ... 10837 10838 10839] TEST: [    0     1     2 ... 2707 2708 2709]
KFold train-test cross validation
TRAIN: [    0     1     2 ... 10837 10838 10839] TEST: [2710 2711 2712 ... 5417 5418 5419]
KFold train-test cross validation
TRAIN: [    0     1     2 ... 10837 10838 10839] TEST: [5420 5421 5422 ... 8127 8128 8129]
KFold train-test cross validation
TRAIN: [   0    1    2 ... 8127 8128 8129] TEST: [ 8130  8131  8132 ... 10837 10838 10839]
The accuracy resubstitution:  0.9995079950799508
```

Conclusion

This is an interesting project where we had our hands on real data and learned how to deal with them. Iris dataset is very well documented and also preprocessed so it did not give us much trouble. This was not the case for googleplaystore dataset which had a few irrelevant entries. We learned how to remove the unnecessary ones, clean our data, transform it into various different forms according to our needs. We also learned about some must-know libraries for ML practitioners and Data Scientists.

The most challenging part was implementing the Random Forest manually. This took us some time and lots of researches but it was worth it since we came up with a very performant RF.

Thank you !!!

The source code contains: iris_analysis.py (iris visualization, preprocessing and DT, gp_analysis.py(vis, preproc, DT), RandomForest.py (RF for both datasets) and also the visualization files(mean, SD, DT...)