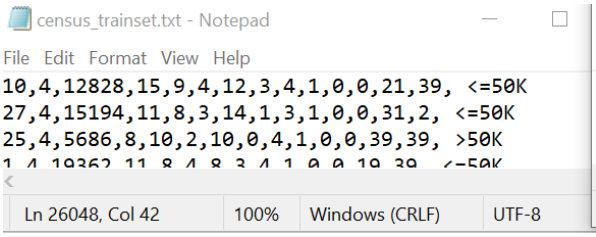
Project 1 Report

Zhenyu Lin

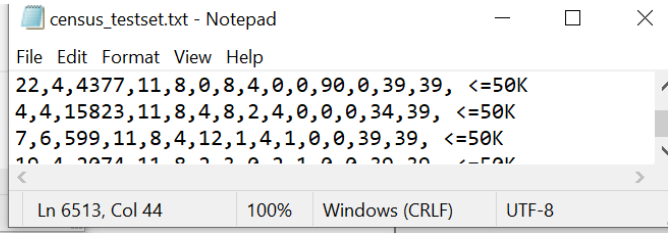
```
# Please install
# pip install scikit-learn
# pip install pandas
```

Task 2: Split data to 80% training, and 20% testing

Census_trainset: Ln: 26048

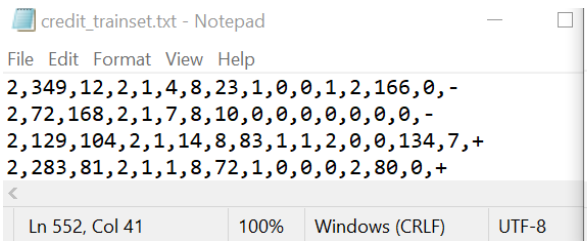


Census_testset: Ln: 6513

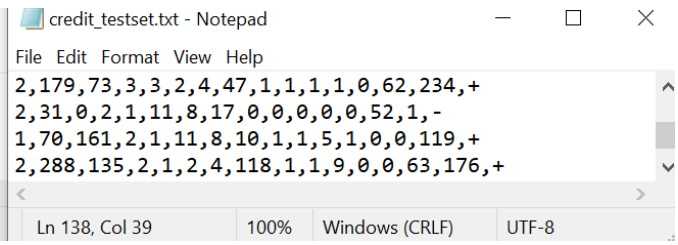


26048+6513 = 32561; 6513/32561 = 0.20002 = 20%

Credit_trainset: Ln: 552



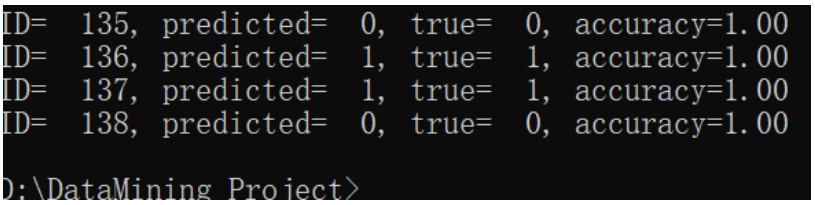
Credit_testset: Ln: 138



552+138 = 690; 138/690 = 0.2 = 20%

Task 3: See code

Task4: See code and a peek for Credit:



Task 5: Using Classifier Report from scikit-learn (sklearn) to auto print the report. Comment out my task_4 function to see it clearer:

Credit:

```
D:\DataMining_Project>python DTvsRFvsNB.py D:\DataMining_Project\credit_trainset.txt D:\DataMining_Project\credit_tes
tset.txt
Number of instances in the training dataset: 552
Number of instances in the test dataset: 138
Decision Tree Results:
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.81      0.79        63
           1       0.83      0.80      0.82        75

    accuracy                           0.80       138
   macro avg       0.80      0.80      0.80       138
weighted avg       0.81      0.80      0.80       138

Runtime: 0.0060002803802490234s

Random Forest Results:
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.84      0.88        63
           1       0.88      0.95      0.91        75

    accuracy                           0.90       138
   macro avg       0.90      0.89      0.90       138
weighted avg       0.90      0.90      0.90       138

Runtime: 0.132002592086792s

Naive Bayes Results:
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88        63
           1       0.89      0.91      0.90        75

    accuracy                           0.89       138
   macro avg       0.89      0.89      0.89       138
weighted avg       0.89      0.89      0.89       138

Runtime: 0.003000020980834961s

D:\DataMining_Project>
```

a. Training is 552, test is 138

Decision Tree Results:

b. Accuracy = 0.8, precision (TC) = 0.83, recall (TC) = 0.95
c. 0.0060002803802490234s

Random Forest Results:

b. Accuracy = 0.9, precision (TC) = 0.88, recall (TC) = 0.95
c. 0.132002592086792s

Naive Bayes Results:

b. Accuracy = 0.89, precision (TC) = 0.89, recall (TC) = 0.91
c. 0.003000020980834961s
d. Comment: Overall, for the Credit dataset with 552 training sets and 138 test sets, the results demonstrate that Random Forest is the most accurate and sensitive algorithm, although it has the longest runtime of the three algorithms, being approximately 20-30 times slower. The fastest algorithm is Naïve Bayes, which is about 30 times faster than Random Forest, and its accuracy is only 1% lower than that of Random Forest.

Census:

```
D:\DataMining_Project>python DTvsRFvsNB.py D:\DataMining_Project\census_trainset.txt D:\DataMining_Project\census_tes
tset.txt
Number of instances in the Credit training dataset: 552
Number of instances in the Credit test dataset: 138
Number of instances in the Credit training dataset: 26048
Number of instances in the Credit test dataset: 6513
Decision Tree Results:
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      5026
           1       0.58      0.63      0.61      1487

    accuracy                           0.81      6513
   macro avg       0.74      0.75      0.74      6513
weighted avg       0.82      0.81      0.82      6513

Runtime: 0.19997453689575195s

Random Forest Results:
Classification Report:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89      5026
           1       0.99      0.14      0.25      1487

    accuracy                           0.80      6513
   macro avg       0.89      0.57      0.57      6513
weighted avg       0.84      0.80      0.74      6513

Runtime: 0.6626431941986084s

Naive Bayes Results:
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.93      0.89      5026
           1       0.66      0.45      0.54      1487

    accuracy                           0.82      6513
   macro avg       0.76      0.69      0.71      6513
weighted avg       0.81      0.82      0.81      6513

Runtime: 0.011000871658325195s
```

a. Training is 26048, test is 6513

Decision Tree Results:

b. Accuracy = 0.81, precision (TC) = 0.58, recall (TC) = 0.63
c. 0.19997453689575195s

Random Forest Results:

b. Accuracy = 0.80, precision (TC) = 0.88, recall (TC) = 0.14
c. 0.6626431941986084s

Naive Bayes Results:

b. Accuracy = 0.82, precision (TC) = 0.89, recall (TC) = 0.45
c. 0.011000871658325195s
d. Comment: For Census when we have 26048 datasets for training and 6513 for test, the results indicate that the most accurate and fastest algorithm is Naïve Bayes. However, compared to Credit dataset, where the number of examples increased but the number of features decreased, both the accuracy of RF and NB decreased, while DT's accuracy increased. At the same time, the running times for DT and NB increased significantly, unlike RF. DT is particularly interesting because, despite having a precision of 0.58 for the True Class, it still achieved high accuracy.

Task 6:

```
********** Task6 **********
Deleted 37 rows with missing values.
Number of instances in the new Credit training dataset after handling missing data: 522
Number of instances in the new Credit test dataset after handling missing data: 131

Decision Tree Results on cleaned data:
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.81      0.78        58
           1       0.84      0.79      0.82        73

    accuracy                           0.80       131
   macro avg       0.80      0.80      0.80       131
weighted avg       0.80      0.80      0.80       131

Runtime: 0.0060007572174072266s

Random Forest Results on cleaned data:
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.78      0.83        58
           1       0.84      0.93      0.88        73

    accuracy                           0.86       131
   macro avg       0.87      0.85      0.86       131
weighted avg       0.87      0.86      0.86       131

Runtime: 0.12599730491638184s

Naive Bayes Results on cleaned data:
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.74      0.79        58
           1       0.81      0.89      0.85        73

    accuracy                           0.82       131
   macro avg       0.83      0.82      0.82       131
weighted avg       0.83      0.82      0.82       131

Runtime: 0.0030040740966796875s
********** Task6 **********
```

	Instances Train	Instances Test
Task 5	552	138
Task 6	522	131
T5-T6	30	7

Table 1

Next Page

	DT	RF	NB
T5 Accuracy	0.8	0.9	0.89
T6 Accuracy	0.8	0.86	0.82
T5 precision (TC)	0.83	0.88	0.89
T6 precision (TC)	0.84	0.84	0.81
T5 recall (TC)	0.95	0.95	0.91
T6 recall (TC)	0.79	0.93	0.89
T5 Run Time	0.0060002803802490234s	0.132002592086792s	0.0030000209808349 61s
T6 Run Time	0.0060007572174072266s	0.12599730491638184s	0.0030040740966796875s

Table 2

a. For task 6 I am using ignore the tuple, to delete/drop the missing data.
b. Table 2
c. From the two tables above and the screenshot, student can observe that after using the ignore the tuple method, new datasets dropped 37 rows. The run time between the original dataset and the T6 dataset was not significant, students believe this is because the original dataset was small, and the amount of data dropped for T6 cleaned dataset was not substantial. Since dropped 37 missing values, there was less data to train compared to the original dataset. This is reflected in the decreased accuracy, precision, and recall for the RF and NB algorithms. As for DT, the accuracy remained unchanged; however, while the precision increased, the recall decreased. Yet, compared to the other two algorithms, this was a significant shift.
   Overall, ignoring tuples has led to a more precise but less sensitive Decision Tree model, a slight trade-off for Random Forest, and a generally decreased performance for Naive Bayes.