TRAIN TEST SPLITTING THE LENDING CLUB LOAN DATASET

USING PYTHON

DR. ALVIN ANG

CONTENTS

I. Method 1: Normal Way of Train Test Split (Using Scikit Learn) 3

A. Step 1: Reading in the Data 3

1. Import All Libraries 3
2. Browsing the Columns 4

B. Step 2: Selecting Columns 5

C. Step 3: Using Scikit Learn to do Train Test Split 7

1. Previewing X_train 7
2. Previewing X_test 9

II. Method 2: Special Way of Train Test Split (Based on Dates) 10

A. Step 1: Reading in the Data 10

1. Import All Libraries 10
2. Checking out the "Month" Column 11

B. Step 2: Convert the "Month" Column to Date Time Format 12

C. Step 3: Plot the Number of Repayments per Year 13

D. Step 4: Train Test Split Based on Dates 14

E. Step 5: Double Checking the No. of Rows for Train Test Split 14

F. Step 6: Assigning the Train Test Split to X and Y 15

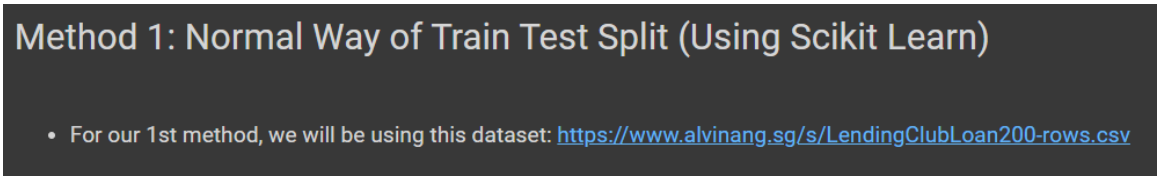
1. Assigning y_train and y_test 15
2. Assigning X_train and X_test 16

About Dr. Alvin Ang 18

I. METHOD 1: NORMAL WAY OF TRAIN TEST SPLIT (USING SCIKIT LEARN)

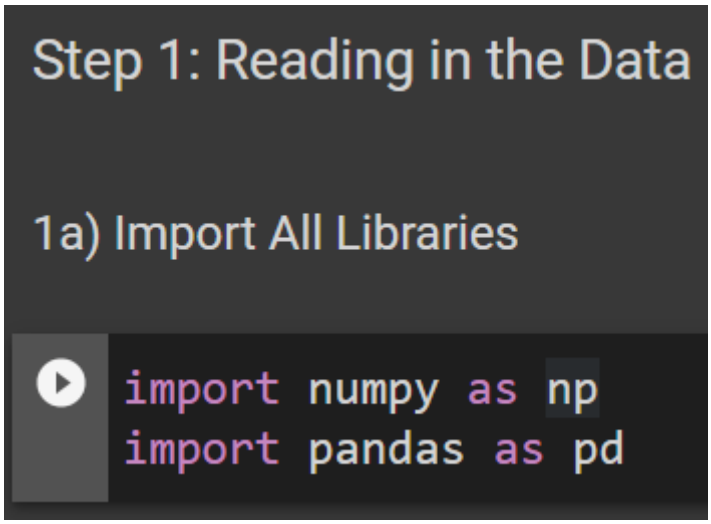
https://www.alvinang.sg/s/LendingClubLoan200-rows.csv

https://www.alvinang.sg/s/Train_Test_Split_the_Lending_Club_Loan_Dataset_by_Dr_Alvin_Ang.ipynb



A. STEP 1: READING IN THE DATA

1. IMPORT ALL LIBRARIES



2. BROWSING THE COLUMNS

```
1b) Browsing the Columns

[2] loans = pd.read_csv('https://www.alvinang.sg/s/LendingClubLoan200-rows.csv')

    loans.info()
```

```
[3] <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 199 entries, 0 to 198
    Data columns (total 74 columns):
     #   Column                      Non-Null Count  Dtype
    ---  ------                      --------------  -----
     0   id                          199 non-null    int64
     1   member_id                   199 non-null    int64
     2   loan_amnt                   199 non-null    int64
     3   funded_amnt                 199 non-null    int64
     4   funded_amnt_inv             199 non-null    float64
     5   term                        199 non-null    object
     6   int_rate                    199 non-null    float64
     7   installment                 199 non-null    float64
     8   grade                       199 non-null    object
     9   sub_grade                   199 non-null    object
     10  emp_title                   190 non-null    object
     11  emp_length                  198 non-null    object
     12  home_ownership              199 non-null    object
     13  annual_inc                  199 non-null    float64
     14  verification_status         199 non-null    object
     15  issue_d                     199 non-null    object
     16  loan_status                 199 non-null    object
     17  pymnt_plan                  199 non-null    object
     18  url                         199 non-null    object
     19  desc                        129 non-null    object
     20  purpose                     199 non-null    object
     21  title                       199 non-null    object
     22  zip_code                    199 non-null    object
     23  addr_state                  199 non-null    object
     24  dti                         199 non-null    object
     25  delinq_2yrs                 199 non-null    float64
     26  earliest_cr_line            199 non-null    object
     27  inq_last_6mths              199 non-null    object
     28  mths_since_last_delinq      47 non-null     float64
     29  mths_since_last_record      5 non-null      float64
     30  open_acc                    198 non-null    float64
     31  pub_rec                     199 non-null    int64
     32  revol_bal                   199 non-null    int64
     33  revol_util                  199 non-null    float64
     34  total_acc                   199 non-null    float64
     35  initial_list_status         199 non-null    object
     36  out_prncp                   199 non-null    object
     37  out_prncp_inv               199 non-null    float64
     38  total_pymnt                 199 non-null    float64
     39  total_pymnt_inv             199 non-null    float64
     40  total_rec_prncp             199 non-null    float64
     41  total_rec_int               199 non-null    float64
     42  total_rec_late_fee          199 non-null    float64
     43  recoveries                  199 non-null    float64
     44  collection_recovery_fee     199 non-null    float64
     45  last_pymnt_d                198 non-null    object
     46  last_pymnt_amnt             199 non-null    object
     47  next_pymnt_d                14 non-null     object
```

B. STEP 2: SELECTING COLUMNS

Step 2: Selecting Columns

```
[4] X = loans[['addr_state',
               'term',
               'int_rate',
               'installment',
               'sub_grade',
               'home_ownership']]

    y = loans['loan_amnt']

    #Presume we have decided to select those columns above as the important ones
    #X are the Predictors while y is the Target (or Prediction)
    #in other words, 'X' are the variables that will help us predict 'y'.
```

```
[16] X.info()

     #X has 6 columns and 199 rows
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   addr_state      199 non-null    object
 1   term            199 non-null    object
 2   int_rate        199 non-null    float64
 3   installment     199 non-null    float64
 4   sub_grade       199 non-null    object
 5   home_ownership  199 non-null    object
dtypes: float64(2), object(4)
memory usage: 9.5+ KB
```

```
y

#y has 1 column and 199 rows
```

```
0       5000
1       2500
2       2400
3      10000
4       3000
       ...
194    14000
195    12000
196    25000
197     9000
198    13250
Name: loan_amnt, Length: 199, dtype: int64
```

C. STEP 3: USING SCIKIT LEARN TO DO TRAIN TEST SPLIT

Step 3: Using Scikit Learn to do Train Test Split

```
[7] from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

    #we decide upon a 20% (test) 80% (train) split
```

1. PREVIEWING X_TRAIN

3a) Previewing X_train

```
X_train.info()

#X_train has 159 rows
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 159 entries, 133 to 59
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   addr_state      159 non-null    object
 1   term            159 non-null    object
 2   int_rate        159 non-null    float64
 3   installment     159 non-null    float64
 4   sub_grade       159 non-null    object
 5   home_ownership  159 non-null    object
dtypes: float64(2), object(4)
memory usage: 8.7+ KB
```

```
X_train

#you can see that the data has been randomized after splitting
#because the row numbers are not in order!!!
```

	addr_state	term	int_rate	installment	sub_grade	home_ownership
133	PA	36 months	14.65	172.48	C3	RENT
92	NY	60 months	19.42	214.62	E3	RENT
159	OR	36 months	12.42	267.33	B4	RENT
101	KS	60 months	17.58	402.65	D4	RENT
178	NY	36 months	12.42	320.79	B4	RENT
...	...	...	...	...	...	...
57	OR	36 months	12.42	334.16	B4	OWN
134	MD	36 months	6.62	271.73	A2	RENT
194	VA	36 months	6.62	429.86	A2	RENT
74	NY	36 months	8.90	457.25	A5	OWN
59	TX	36 months	16.77	252.33	D2	MORTGAGE

159 rows × 6 columns

2. PREVIEWING X_TEST

3b) Previewing X_test

```
[14] X_test.info()

     #X_test has 40 rows (after splitting... 20% of 200 rows....)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 40 entries, 162 to 28
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   addr_state      40 non-null     object
 1   term            40 non-null     object
 2   int_rate        40 non-null     float64
 3   installment     40 non-null     float64
 4   sub_grade       40 non-null     object
 5   home_ownership  40 non-null     object
dtypes: float64(2), object(4)
memory usage: 2.2+ KB
```

```
[21] X_test

     #once again, the rows are not in order.. they have been randomized...
```

35	KY	36 months	10.65	403.91	B2	RENT
142	CA	36 months	14.27	480.33	C2	RENT
189	VA	36 months	12.69	452.86	B5	RENT
129	NJ	36 months	7.51	311.11	A3	RENT
12	VA	36 months	13.49	305.38	C1	RENT
15	MO	36 months	16.29	35.31	D1	RENT
0	AZ	36 months	10.65	162.87	B2	RENT
149	IL	60 months	16.77	208.97	D2	OWN
173	FL	36 months	12.69	245.72	B5	RENT
185	TX	36 months	10.65	1140.07	B2	MORTGAGE
120	AK	60 months	13.49	575.12	C1	MORTGAGE
168	CA	36 months	7.51	77.78	A3	OWN
145	MN	36 months	9.91	386.70	B1	RENT

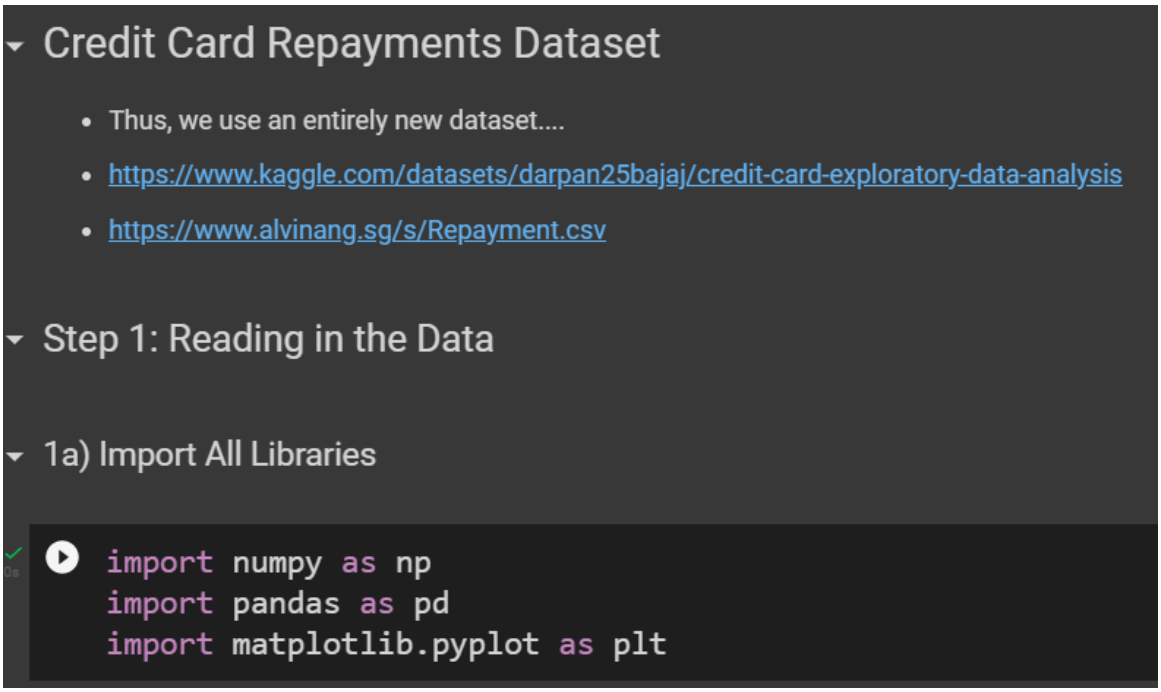
II. METHOD 2: SPECIAL WAY OF TRAIN TEST SPLIT (BASED ON DATES)

https://www.kaggle.com/datasets/darpan25bajaj/credit-card-exploratory-data-analysis

https://www.alvinang.sg/s/Repayment.csv

A. STEP 1: READING IN THE DATA

1. IMPORT ALL LIBRARIES



```
[92] repayment = pd.read_csv('https://www.alvinang.sg/s/Repayment.csv')
```

```
[93] repayment
```

	SL No:	Customer	Month	Amount	Unnamed: 4
0	NaN	A1	12-Jan-04	495414.75	NaN
1	2.0	A1	3-Jan-04	245899.02	NaN
2	3.0	A1	15-Jan-04	259490.06	NaN
3	4.0	A1	25-Jan-04	437555.12	NaN
4	5.0	A1	17-Jan-05	165972.88	NaN
...	...	...	...	...	...
1518	NaN	NaN	NaN	NaN	NaN
1519	NaN	NaN	NaN	NaN	NaN
1520	NaN	NaN	NaN	NaN	NaN
1521	NaN	NaN	NaN	NaN	NaN
1522	NaN	NaN	NaN	NaN	NaN

1523 rows × 5 columns

2. CHECKING OUT THE "MONTH" COLUMN

```
1b) Checking out the "Month" column
```

```
[94] repayment['Month'].isnull().values.sum()

#this dataset has 1523 rows of which 23 rows are NA in the 'Month' column
```

```
23
```

```
repayment['Month'].sample(5)

#note that the 'Month' column is a String format!
#But which it should be a Date Time Format!
```

```
985     29-May-05
1174    23-Feb-05
1346     6-May-06
1382     5-May-04
1407     3-Aug-05
Name: Month, dtype: object
```

B. STEP 2: CONVERT THE "MONTH" COLUMN TO DATE TIME FORMAT

Step 2: Convert the 'Month' column to Date Time format

```
[96] repayment['Month'] = pd.to_datetime(repayment['Month'])
```

```
[97] repayment['Month'].sample(5)

     #it has now been converted to datetime format!
```

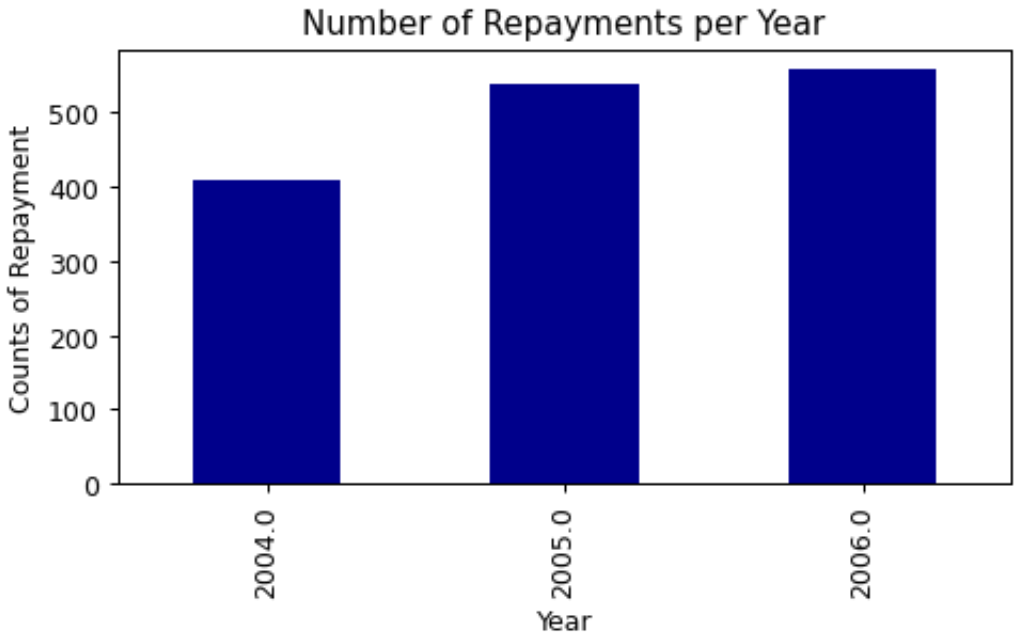
```
309    2004-01-15
1415   2006-04-03
784    2005-02-16
177    2005-12-03
258    2006-05-06
Name: Month, dtype: datetime64[ns]
```

C. STEP 3: PLOT THE NUMBER OF REPAYMENTS PER YEAR

Step 3: Plot the Number of Repayments per Year

```
plt.figure(figsize=(6,3), dpi=90)

repayment['Month'].dt.year.value_counts().sort_index().plot.bar(color = 'darkblue')

plt.xlabel('Year')
plt.ylabel('Counts of Repayment')
plt.title('Number of Repayments per Year')

#the distribution shows us that the Repayments have been increasing
#for the past three years
```

```
Text(0.5, 1.0, 'Number of Repayments per Year')
```



D. STEP 4: TRAIN TEST SPLIT BASED ON DATES

Step 4: Train Test Split based on Dates

```
[99] repayment_train = repayment.loc[repayment['Month'] < repayment['Month'].quantile(0.8)]
     repayment_test = repayment.loc[repayment['Month'] >= repayment['Month'].quantile(0.8)]

     #Train set = 80% of the data
     #in other words, the TRAIN dataset will consist of OLDER DATES

     #Test set = 20% of the data
     #in other words, the TEST dataset will consist of MOST RECENT DATES
     #the 20% of MOST RECENT DATES
```

E. STEP 5: DOUBLE CHECKING THE NO. OF ROWS FOR TRAIN TEST SPLIT

Step 5: Double Checking the No. of Rows for Train Test Split

```
[101] print('Number of Rows in Train Test Split: ', repayment_train.shape[0] + repayment_test.shape[0])
      print('Number of Rows in the full dataset:  ', repayment.shape[0])

      #we see that the Original Dataset has 1523 rows
      #while the Train Test Split has 1500 rows
      #because 23 rows of NaNs have been omitted

      Number of Rows in Train Test Split:  1500
      Number of Rows in the full dataset:   1523
```

F. STEP 6: ASSIGNING THE TRAIN TEST SPLIT TO X AND Y

1. ASSIGNING Y_TRAIN AND Y_TEST

```
Step 6: Assigning the Train Test Split to X and y

6a) Assigning y_train and y_test

y_train = repayment_train['Amount']
y_test =  repayment_test['Amount']

#let's say we are trying to predict the "Amount" of repayment....
#thus "Amount" columns is y
```

```
[106] y_train
      #80% x 1500 rows = 1200 rows

0       495414.75
1       245899.02
2       259490.06
3       437555.12
4       165972.88
          ...
1491    416676.34
1493    230667.34
1495     55638.77
1498    454016.51
1499     56286.33
Name: Amount, Length: 1199, dtype: float64

      y_test
      #20% x 1500 = 300 rows

19      471099.22
21      219264.02
25      470982.16
27      326050.15
29      481343.32
          ...
1488    187041.66
1492    110614.61
1494    113094.58
1496    319836.49
1497    247628.45
Name: Amount, Length: 301, dtype: float64
```

2. ASSIGNING X_TRAIN AND X_TEST

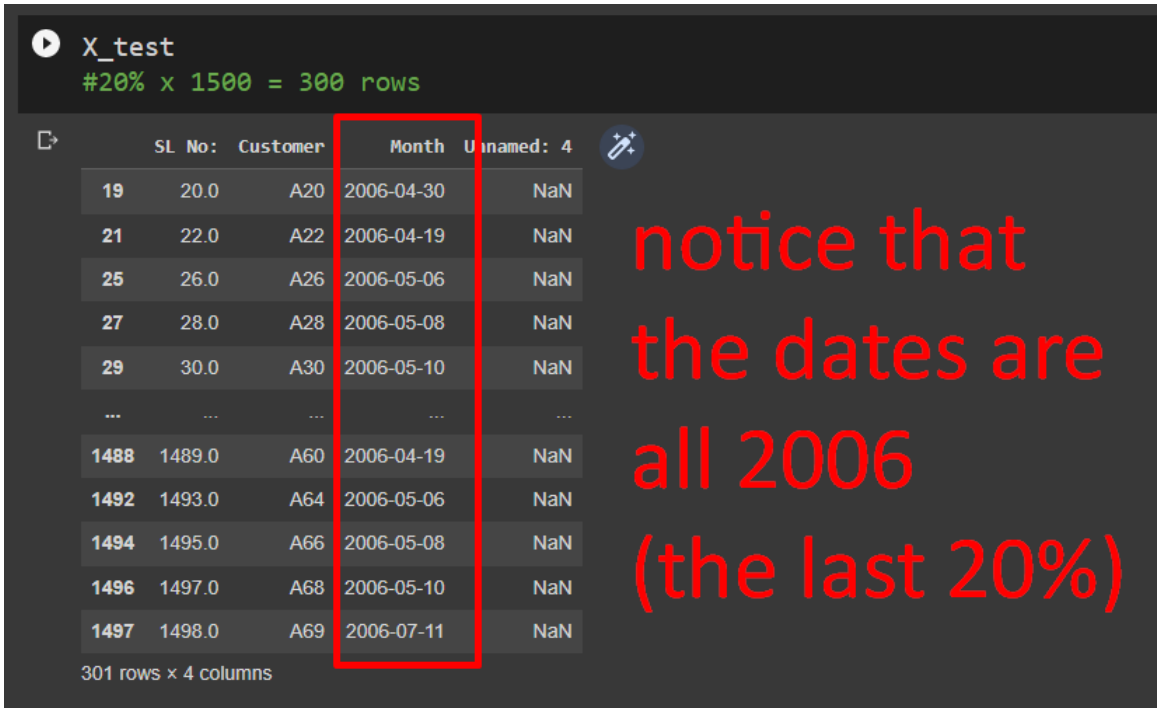
```
6b) Assigning X_train and X_test

[108] X_train = repayment_train.drop('Amount', axis = 1)

[109] X_test =  repayment_test.drop('Amount', axis = 1)
```

```
X_train
#80% x 1500 rows = 1200 rows
```

	SL No:	Customer	Month	Unnamed: 4
0	NaN	A1	2004-01-12	NaN
1	2.0	A1	2004-01-03	NaN
2	3.0	A1	2004-01-15	NaN
3	4.0	A1	2004-01-25	NaN
4	5.0	A1	2005-01-17	NaN
...	...	...	...	...
1491	1492.0	A63	2004-05-05	NaN
1493	1494.0	A65	2005-05-07	NaN
1495	1496.0	A67	2005-05-09	NaN
1498	1499.0	A70	2005-08-12	NaN
1499	1500.0	A71	2004-09-13	NaN

1199 rows × 4 columns

notice that the dates are only 2004 ~ 2005 (the first 80%)



ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.