Advanced Survival and Risk Analysis
- Cleaning the TRANING_DATA -

Firstly I am importing packages that I am going to use and loading the data:

In [31]:

```
# importing packages
import pandas as pd
import numpy as np
import difflib as diff


# loading the data frame
score = pd.read_excel('~/Desktop/unsw/score_prediction/Training_data.csv.xlsx'
)
```

Now lets analyse the head of the data to see what it does look like:

In [32]:

```
#analysing the head of the data
score.head()
```

Out[32]:

	Fruit_Ate	Name	Why_Eat	Where_Eat	How_Fast_Eat	Time_Eatten	Sco
0	lychee	Jacky	Lunch or dI_%nner substI_%tute	Home	NaN	4	44459
1	strawberry	Nicholai	Yu34@mmy so ate it	Family	normally	4	24223
2	nectarine	Rachel	Hungry so ate anything	H"$"nging out	slowly	5	30535
3	lychee	Da!!Niel	Lunch or dinner substitute	Home	normally	7	23532
4	lychee	Daniel	Cheapest to buy	Guy/girl friend's place	slowly	14	25474

From that it is clear that the column names can be changed into lower case, also there are some spelling mistakes. The Time_Eatten column as well as Score column must be stored as integers as I wish to deal with numbers.

Checking the formatation of column names:

In [33]:

```
# checking the formatation of column names:
score.columns
```

Out[33]:

```
Index(['Fruit_Ate', 'Name', 'Why_Eat', 'Where_Eat', 'How_Fast_Eat'
,
       'Time_Eatten', 'Score'],
      dtype='object')
```

Changing these column names into lowercase:

In [34]:

```
# changing column names:
score.rename(columns={'Fruit_Ate': 'fruit_ate', 'Name': 'name', 'Why_Eat': 'wh
y_eat', 'Where_Eat': 'where_eat', 'How_Fast_Eat': 'eating_velocity', 'Time_Eat
ten': 'time_eatten', 'Score': 'score'}, inplace=True)

# printing the head of the dataframe:
score.head()
```

Out[34]:

	fruit_ate	name	why_eat	where_eat	eating_velocity	time_eatten	scor
0	lychee	Jacky	Lunch or dI_%nner substI_%tute	Home	NaN	4	44459
1	strawberry	Nicholai	Yu34@mmy so ate it	Family	normally	4	24223
2	nectarine	Rachel	Hungry so ate anything	H"$"nging out	slowly	5	30535
3	lychee	Da!!Niel	Lunch or dinner substitute	Home	normally	7	23532
4	lychee	Daniel	Cheapest to buy	Guy/girl friend's place	slowly	14	25474

Analysing basic information about data such as type of column and non-null entries:

In [35]:

```
#analysing basic information about data
score.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37500 entries, 0 to 37499
Data columns (total 7 columns):
fruit_ate          36865 non-null object
name               36864 non-null object
why_eat            36889 non-null object
where_eat          36881 non-null object
eating_velocity    36841 non-null object
time_eatten        36884 non-null object
score              37380 non-null object
dtypes: object(7)
memory usage: 2.0+ MB
```

Both time_eatten and score columns are stored as object, therefore we have to change it to integer type. So lets have a closer look to both columns:

In [36]:

```
# counting the number of times that each different value appear:
score.time_eatten.value_counts(dropna=False)
```

```
Out[36]:

16      1596
17      1576
5       1576
13      1564
2       1545
20      1545
12      1542
10      1541
4       1538
18      1535
22      1535
3       1529
24      1521
9       1514
6       1513
19      1510
1       1496
15      1490
11      1485
7       1482
14      1474
23      1466
21      1455
8       1454
NaN      616
NILL     190
???      174
--        38
Name: time_eatten, dtype: int64
```

From that we can see that the observations such as 'Nan', 'NILL', '???' and '--' should be all Nan type which corresponds to missing values. Changing this observations to Nan:

In [37]:

```
# grouping 'NILL', '--' and '???' into the missing values:
score['time_eatten' == 'NILL', 'time_eatten' == '???', 'time_eatten' == '--']
= score.time_eatten.replace({'NILL': np.nan, '???': np.nan, '--': np.nan}, inp
lace=True)

# counting the number of times that each different value appear:
score.time_eatten.value_counts(dropna=False)
```

Out[37]:

```
 16.0    1596
 17.0    1576
 5.0     1576
 13.0    1564
 2.0     1545
 20.0    1545
 12.0    1542
 10.0    1541
 4.0     1538
 22.0    1535
 18.0    1535
 3.0     1529
 24.0    1521
 9.0     1514
 6.0     1513
 19.0    1510
 1.0     1496
 15.0    1490
 11.0    1485
 7.0     1482
 14.0    1474
 23.0    1466
 21.0    1455
 8.0     1454
NaN      1018
Name: time_eatten, dtype: int64
```

Checking the type of data stored in time_eatten column:

In [38]:

```
# checking the type of data stored in time_eatten column:
score.time_eatten.dtype
```

Out[38]:

```
dtype('float64')
```

Now we can see that the type of time_eatten column has changed to type float automatically. This occured because previously the entries corresponded to '???', 'NILL' and '--' was being stored as object as they are strings, so once I changed this entries python recognized the column as type float.

Checking the score column entries:

In [39]:

```
# checking the score column entries:
score.score.value_counts(dropna=False)
```

Out[39]:

```
 22580.696539    145
 23528.916538    143
 22486.881530    137
 1650.863902     137
NaN              120
 608.828894      115
 23273.900904     98
 1905.879537      96
 8563.499662      91
 22231.865895     84
 4483.408730      84
 1557.048893      82
 9605.534671      81
 30535.367307     81
 25413.241367     79
 863.844528       78
 22487.282830     78
 22835.712173     76
 31228.571672     76
 3441.373722      74
 22487.104174     73
 44458.749174     72
 23273.200247     71
 23529.317838     71
 22581.097839     71
 23180.085895     69
 22580.919183     69
 44203.733540     69
 25064.410723     66
 23527.537697     66
                 ...
 30164.337582      1
 3116.765723       1
 3078.955144       1
 3250.368100       1
 2265.654558       1
 1529.835623       1
 44138.628562      1
 22185.350230      1
 3155.765723       1
 165.291787        1
 542.800615        1
 1272.228262       1
 3081.343997       1
 23192.872625      1
 1185.410456       1
 44071.110737      1
 167.680641        1
 7095.937924       1
 23134.497960      1
```

```
 5239.835740       1
 7094.559083       1
 47039.679668      1
 9304.003371       1
 3094.343997       1
 23154.583294      1
 22126.243092      1
 3106.842422       1
 134.984160        1
 23141.278101      1
 10333.961680      1
Name: score, Length: 4264, dtype: int64
```

Because there are many different values we are unable to see each one specifically. I am assuming that all columns in the dataframe will also have entries such '--', 'NILL' and '???' as we had this same pattern on other columns on the dataframe and I also want to check if some entry is stored as string 'Nan' instead of numpy.Nan.

In [42]:

```
#score.loc[score['score'] == '--']
#score.loc[score['score'] == '???']
#score.loc[score['score'] == 'NILL']
#score.loc[score['score'] == 'Nan']

# I located this entries to make sure that they exist in the
# data and after that I commented it out because
# it was locating and printing all entries.
```

So we know that there are values on the score column which are stored as string because they are non numeric types. Changing this observations to numpy.Nan:

In [13]:

```
# changing entry names into missing value np.Nan:
score['score' == 'NILL', 'score' == '???', 'score' == '--'] = score.score.repl
ace({'NILL': np.nan, '???': np.nan, '--': np.nan}, inplace=True)

# checking the score column entries:
score.score.value_counts(dropna=False)
```

Out[13]:

```
NaN               242
 22580.696539     145
 23528.916538     143
 1650.863902      137
 22486.881530     137
 608.828894       115
 23273.900904      98
 1905.879537       96
 8563.499662       91
 22231.865895      84
 4483.408730       84
 1557.048893       82
 30535.367307      81
```

```
9605.534671      81
25413.241367     79
22487.282830     78
863.844528       78
22835.712173     76
31228.571672     76
3441.373722      74
22487.104174     73
44458.749174     72
23273.200247     71
23529.317838     71
22581.097839     71
23180.085895     69
44203.733540     69
22580.919183     69
25064.410723     66
23529.926551     66
                ...
6021.202259       1
1254.839409       1
66097.008820      1
4290.014255       1
66145.008820      1
30153.645210      1
3139.164237       1
44083.807218      1
3081.343997       1
44090.807218      1
1170.019169       1
44089.027077      1
147.680641        1
44078.405918      1
7067.559083       1
10333.038379      1
22516.378929      1
3106.431765       1
44109.027077      1
44098.027077      1
24247.130916      1
4168.199246       1
23141.099445      1
30315.574317      1
5185.623108       1
47043.577711      1
4182.578087       1
2263.874417       1
11189.430156      1
44126.719450      1
Name: score, Length: 4261, dtype: int64
```

Now I want to see if the type of the data on column score has changed to type float:

In [14]:

```
# checking data type of score column:
score.score.dtype
```

Out[14]:

```
dtype('float64')
```

From that I could see that '???', '--' and 'NILL' were the only string present on the score column and by removing those the type of the entries on the score column was automatically changed to float.

Converting the remaining columns into low string:

In [15]:

```
# converting why_eat column into lowercase
score.why_eat = score.why_eat.str.lower()

# converting fruit_ate column into lowercase
score.fruit_ate = score.fruit_ate.str.lower()

# converting where_eat column into lowercase
score.where_eat = score.where_eat.str.lower()

# converting eating_velocity column into lowercase
score.eating_velocity = score.eating_velocity.str.lower()

# printing the head of the dataframe to check the changes:
score.head()
```

Out[15]:

	fruit_ate	name	why_eat	where_eat	eating_velocity	time_eatten	
0	lychee	Jacky	lunch or di_%nner substi_%tute	home	NaN	4.0	44458.
1	strawberry	Nicholai	yu34@mmy so ate it	family	normally	4.0	24223.
2	nectarine	Rachel	hungry so ate anything	h"$"nging out	slowly	5.0	30535.
3	lychee	Da!!Niel	lunch or dinner substitute	home	normally	7.0	23532.
4	lychee	Daniel	cheapest to buy	guy/girl friend's place	slowly	14.0	25474.

Checking the different types of fruit stored in fruit_ate column:

In [16]:

```
# counting occurence of each different fruit:
score.fruit_ate.value_counts()
```

Out[16]:

```
lychee          2275
rockmelon       2256
kiwi            2217
raspberry       2168
peach           2148
grape           2110
logan           2105
apple           2105
orange          2103
pear            2103
strawberry      2085
watermelon      2045
banana          2033
nectarine       1924
durian          1894
jackfruit       1832
???              189
nill             185
pea!!r           114
jackfrui_%t      114
log"$"n          113
a!!pple          110
wa!!termelon     106
gr"$"pe          106
pe"$"ch          105
j"$"ckfruit      105
loga!!n          101
gra!!pe          101
ora!!nge          99
duria!!n          99
duri_%an          99
pe"$"r            98
str"$"wberry      95
necta!!rine       94
ja!!ckfruit       91
nect"$"rine       91
du34@rian         91
w"$"termelon      90
ra!!spberry       89
jackfru34@it      89
r"$"spberry       88
stra!!wberry      87
or"$"nge          87
ki_%wi_%          86
ba!!na!!na!!      86
b"$"n"$"n"$"      86
"$"pple           85
pea!!ch           84
nectari_%ne       81
duri"$"n          78
--                40
Name: fruit_ate, dtype: int64
```

From that I could see that there are some values stored as '--', '???' and 'nill' which I want to change it all to numpy.Nan type (missing value):

In [43]:

```
# changing name of some entries in fruit_ate column:
score['fruit_ate' == 'nill', 'fruit_ate' == '???', 'fruit_ate' == '--'] = scor
e.fruit_ate.replace({'nill': np.nan, '???': np.nan, '--': np.nan}, inplace=Tru
e)
```

Now that I have all missing values stored as numpy.Nan, I am going to remove non-alphabetic characters as we can see that just by removing them some entries will have the correct spelling name of the fruit.

In [18]:

```
# removing characters that are different from alphanumeric characters
# in fruit_ate column:
score.fruit_ate = score.fruit_ate.str.replace('[^a-zA-Z]', '')

# checking if something is changed:
score.fruit_ate.value_counts()
```

Out[18]:

```
kiwi          2303
lychee        2275
raspberry     2257
rockmelon     2256
peach         2232
pear          2217
apple         2215
grape         2211
logan         2206
orange        2202
durian        2183
strawberry    2172
watermelon    2151
jackfruit     2126
banana        2119
nectarine     2099
logn           113
grpe           106
pech           105
jckfruit       105
per             98
strwberry       95
nectrine        91
wtermelon       90
rspberry        88
ornge           87
bnn             86
pple            85
durin           78
Name: fruit_ate, dtype: int64
```

Now it is clear that we have 16 different types of fruits and some of them still have wrong spelling that needs to be fixed (they are missing the letter 'a').

- Note that I didnt finish to clean this column because the way that I was trying to do was creating a list with the correct name of the fruits called "fruits" and an empty list called "newFruit_ate". After that I would loop each entry in fruit_ate column and compare it with "fruits" list, then it would return the best match and apend it into the "newFruit_ate" list. Then I would replace the fruit_ate column with the values in "newFruit_ate" list. The code was: for fruit in fruit_ate: newFruit_ate.append(diff.get_close_matches(fruit, fruits, n=1)) This chunck of code works in a small sample, but when I apply it to the actual column it does not work.

Analysing the name column to see what it does look like:

In [20]:

```
# count the occurence of the same names
score.name.value_counts()
```

Out[20]:

```
saksham     3980
shuning     3976
blake       3959
nicholai    3955
daniel      3850
dean        3849
jacky       3846
rachel      3841
richard     3835
             224
rchel        193
nicholi      182
den          172
skshm        172
richrd       169
blke         159
jcky         159
dniel        154
Name: name, dtype: int64
```

Changing name column to lower case and dropping non aphanumeric characters and changing 'nill' to be Nan:

In [19]:

```
# changing name column to lowcase
score.name = score.name.str.lower()

# removing non alpanumeric characters
score.name = score.name.str.replace('[^a-zA-Z]', '')

# assigning entries stored as 'nill' to be missing value (numpy.Nan)
score['name' == 'nill'] = score.name.replace('nill', np.nan , inplace=True)

# count the occurence of the same names
score.name.value_counts()
```

Out[19]:

```
saksham     3980
shuning     3976
blake       3959
nicholai    3955
daniel      3850
dean        3849
jacky       3846
rachel      3841
richard     3835
             224
rchel        193
nicholi      182
den          172
skshm        172
richrd       169
blke         159
jcky         159
dniel        154
Name: name, dtype: int64
```

Analysing the eating_velocity column:

In [44]:

```
# checking frequency of values in eating_velocity column:
score.eating_velocity.value_counts()
```

Out[44]:

```
slowly         11961
normally       11264
quickly        11220
qui_%ckly        522
norm"$"lly       501
norma!!lly       494
qu34@ickly       483
???              185
nill             177
--                34
Name: eating_velocity, dtype: int64
```

Dropping non aphanumeric characters in eating_velocity column and changing 'nill' to be Nan:

In [45]:

```
# changing eating_velocity column to lowcase
score.eating_velocity = score.eating_velocity.str.lower()

# dropping non alphanumeric characters:
score.eating_velocity = score.eating_velocity.str.replace('[^a-zA-Z]', '')

# grouping 'nill', '???' and '--' entries as missing values (numpy.Nan)
score['eating_velocity' == 'nill'] = score.eating_velocity.replace('nill', np.
nan , inplace=True)

# checking frequency of values in eating_velocity column
# to see what is changed::
score.eating_velocity.value_counts()
```

Out[45]:

```
quickly     12225
slowly      11961
normally    11758
normlly       501
              219
Name: eating_velocity, dtype: int64
```

Now I want to clean why_eat column. Cheking the frequency counts of each different entry:

In [22]:

```
# checking the frequency counts of each entry:
score.why_eat.value_counts()
```

```
Out[22]:

stressed from tests                     5258
nothing else to eat                     4636
lunch or dinner substitute              4604
forced to eat it                        4520
cheapest to buy                         4510
hungry so ate anything                  4423
yummy so ate it                         4303
forced to ea!!t it                       242
nothing else to e"$"t                    241
hungry so "$"te "$"nything               233
lunch or di_%nner substi_%tute           232
yummy so a!!te it                        232
lu34@nch or dinner su34@bstitu34@te      232
nothing else to ea!!t                    231
chea!!pest to buy                        231
yu34@mmy so ate it                       228
forced to eat i_%t                       227
yummy so ate i_%t                        226
forced to e"$"t it                       214
hungry so ate anythi_%ng                 214
yummy so "$"te it                        211
hu34@ngry so ate anything                211
cheapest to bu34@y                       210
hungry so a!!te a!!nything               200
nill                                     197
che"$"pest to buy                        196
???                                      195
nothi_%ng else to eat                    191
--                                        41
Name: why_eat, dtype: int64
```

From that I can see that some entries are stored as '--', '???', and 'nill' and I am changing these entries to numpy.Nan (missing values) as well as dropping non alphanumeric characters.

In [23]:

```
# dropping non alphanumeric characters:
score.why_eat = score.why_eat.str.replace('[^a-zA-Z ]', '')

# grouping 'nill', '???' and '--' entries as missing values (numpy.Nan)
score['why_eat' == 'nill', 'why_eat' == '???', 'why_eat' == '--'] = score.why_
eat.replace({'nill': np.nan, '???': np.nan, '--': np.nan}, inplace=True)


# checking frequency of values in why_eat column
# to see what is changed::
score.why_eat.value_counts()
```

Out[23]:

```
stressed from tests           5258
lunch or dinner substitute    5068
nothing else to eat           5058
hungry so ate anything        5048
forced to eat it              4989
yummy so ate it               4989
cheapest to buy               4951
nothing else to et             241
                               236
hungry so te nything           233
forced to et it                214
yummy so te it                 211
chepest to buy                 196
Name: why_eat, dtype: int64
```

Now I want to clean where_eat column. Cheking the frequency counts of each different entry:

In [24]:

```
# checking the frequency counts of each entry:
score.where_eat.value_counts()
```

Out[24]:

```
work                          4624
home                          4602
friend's                      4313
univeristy                    4303
library                       3941
family                        3928
hanging out                   3865
guy/girl friend's place       3799
f"$"mily                       222
guy/girl friend's pl"$"ce      217
fa!!mily                       215
libra!!ry                      209
gu34@y/girl friend's place     199
h"$"nging out                  196
ha!!nging out                  195
li_%brary                      195
nill                           193
uni_%veri_%sty                 192
guy/girl friend's pla!!ce      190
hanging ou34@t                 189
hangi_%ng out                  182
fri_%end's                     182
???                            178
fami_%ly                       177
libr"$"ry                      177
guy/gi_%rl fri_%end's place    164
--                              34
Name: where_eat, dtype: int64
```

Now we need to do the same cleaning pattern as the previous column that we cleaned, changing entries stored as '--', '???', and 'nill' to numpy.Nan (missing values) as well as dropping non alphanumeric characters.

In [25]:

```
# dropping non alphanumeric characters:
score.where_eat = score.where_eat.str.replace('[^a-zA-Z ]', '')

# grouping 'nill', '???' and '--' entries as missing values (numpy.Nan)
score['where_eat' == 'nill', 'where_eat' == '???', 'where_eat' == '--'] = scor
e.where_eat.replace({'nill': np.nan, '???': np.nan, '--': np.nan}, inplace=Tru
e)


# checking frequency of values in where_eat column
# to see what is changed::
score.where_eat.value_counts()
```

Out[25]:

```
work                    4624
home                    4602
univeristy              4495
friends                 4495
hanging out             4431
guygirl friends place   4352
library                 4345
family                  4320
fmily                    222
guygirl friends plce     217
                         212
hnging out               196
librry                   177
Name: where_eat, dtype: int64
```

This is the end of my data cleaning process.

I did not have the chance to finish the whole project. However it was a great opportunity for me to learn python, which is a language that I have not used before, and also I learned about data cleaning process.