Data Mining Final Report

Member: 李承諹、湯豐元、黃彥維、陳祺允、曾毓豪、林純守

Part one-
source of information

"111 Years of Casualty Traffic Accident Data" in the Government Open Data

https://data.gov.tw/dataset/161199

Index	理單位名稱警局層	天候名稱	光線名稱	別-第1當事者	限-第1當事者	型態大類別名	型態子
0	雲林縣警察局	晴	夜間(或隧道、地下道、涵洞)有照明	市區道路	50	單路部分	直路
1	國道公路警察局	晴	晨或暮光	國道	110	單路部分	直路
2	國道公路警察局	晴	晨或暮光	國道	110	單路部分	直路
3	國道公路警察局	晴	晨或暮光	國道	110	單路部分	直路
4	宜蘭縣政府警察局	晴	夜間(或隧道、地下道、涵洞)有照明	省道	40	單路部分	隧道
5	南投縣政府警察局	晴	夜間(或隧道、地下道、涵洞)有照明	省道	30	單路部分	直路
6	南投縣政府警察局	晴	夜間(或隧道、地下道、涵洞)有照明	省道	30	單路部分	直路
7	屏東縣政府警察局	晴	夜間(或隧道、地下道、涵洞)有照明	市區道路	50	交岔路	三岔路
8	屏東縣政府警察局	晴	夜間(或隧道、地下道、涵洞)有照明	市區道路	50	交岔路	三岔路
9	宜蘭縣政府警察局	晴	日間自然光線	其他	50	交岔路	四岔路
10	宜蘭縣政府警察局	晴	日間自然光線	其他	50	交岔路	四岔路
11	嘉義縣警察局	晴	夜間(或隧道、地下道、涵洞)有照明	縣道	50	單路部分	地下道

發生年度	發生月份	發生日期	發生時間	事故類別	處理單位	發生地點	天候名稱	光線名稱	道路類別-	速限-第1當	道路型態	道路型態	事故位置	事故位置	路面狀況-	路面
2022	1	20220101	23000	A1	雲林縣警	雲林縣虎	晴	夜間(或隧	市區道路	50	單路部分	直路	路段	路肩、路	柏油	乾燥
2022	1	20220101	23000	A1	雲林縣警	雲林縣虎	晴	夜間(或隧	市區道路	50	單路部分	直路	路段	路肩、路	柏油	乾燥
2022	1	20220101	60742	A1	國道公路	屏東縣九	晴	晨或暮光	國道	110	單路部分	直路	路段	快車道	柏油	乾燥
2022	1	20220101	60742	A1	國道公路	屏東縣九	晴	晨或暮光	國道	110	單路部分	直路	路段	快車道	柏油	乾燥
2022	1	20220101	60742	A1	國道公路	屏東縣九	晴	晨或暮光	國道	110	單路部分	直路	路段	快車道	柏油	乾燥
2022	1	20220101	60742	A1	國道公路	屏東縣九	晴	晨或暮光	國道	110	單路部分	直路	路段	快車道	柏油	乾燥
2022	1	20220101	165000	A1	宜蘭縣政	宜蘭縣南	晴	夜間(或隧	省道	40	單路部分	隧道	路段	一般車道(	柏油	乾燥
2022	1	20220101	165000	A1	宜蘭縣政	宜蘭縣南	晴	夜間(或隧	省道	40	單路部分	隧道	路段	一般車道(	柏油	乾燥
2022	1	20220101	192000	A1	南投縣政	南投縣仁	晴	夜間(或隧	省道	30	單路部分	直路	其他	其他	柏油	乾燥
2022	1	20220101	192000	A1	南投縣政	南投縣仁	晴	夜間(或隧	省道	30	單路部分	直路	其他	其他	柏油	乾燥
2022	1	20220101	192000	A1	南投縣政	南投縣仁	晴	夜間(或隧	省道	30	單路部分	直路	其他	其他	柏油	乾燥
2022	1	20220101	195500	A1	屏東縣政	屏東縣屏	晴	夜間(或隧	市區道路	50	交岔路	三岔路	交叉路口	交叉口附	柏油	乾燥
2022	1	20220101	195500	A1	屏東縣政	屏東縣屏	晴	夜間(或隧	市區道路	50	交岔路	三岔路	交叉路口	交叉口附	柏油	乾燥
2022	1	20220102	80053	A1	宜蘭縣政	宜蘭縣五	晴	日間自然	其他	50	交岔路	四岔路	交叉路口	交叉路口	柏油	乾燥
2022	1	20220102	80053	A1	宜蘭縣政	宜蘭縣五	晴	日間自然	其他	50	交岔路	四岔路	交叉路口	交叉路口	柏油	乾燥
2022	1	20220102	101200	A1	嘉義縣警	嘉義縣水	晴	夜間(或隧	縣道	50	單路部分	地下道	路段	快車道	柏油	乾燥
2022	1	20220102	101200	A1	嘉義縣警	嘉義縣水	晴	夜間(或隧	縣道	50	單路部分	地下道	路段	快車道	柏油	乾燥
2022	1	20220102	120029	A1	新竹縣政	新竹縣橫	晴	日間自然	省道	60	單路部分	直路	路段	一般車道(	柏油	乾燥
2022	1	20220102	120029	A1	新竹縣政	新竹縣橫	晴	日間自然	省道	60	單路部分	直路	路段	一般車道(	柏油	乾燥
2022	1	20220102	183000	A1	桃園市政	桃園市楊	晴	夜間(或隧	省道	70	單路部分	直路	路段	慢車道	柏油	乾燥
2022	1	20220102	183000	A1	桃園市政	桃園市楊	晴	夜間(或隧	省道	70	單路部分	直路	路段	慢車道	柏油	乾燥
2022	1	20220103	61900	A1	新竹市警	新竹市北	晴	晨或暮光	市區道路	30	單路部分	直路	路段	一般車道(	柏油	乾燥
2022	1	20220103	61900	A1	新竹市警	新竹市北	晴	晨或暮光	市區道路	30	單路部分	直路	路段	一般車道(	柏油	乾燥
2022	1	20220103	143613	A1	臺南市政	臺南市學	晴	日間自然	市區道路	50	交岔路	四岔路	交叉路口	交叉路口	柏油	乾燥
2022	1	20220103	143613	A1	臺南市政	臺南市學	晴	日間自然	市區道路	50	交岔路	四岔路	交叉路口	交叉路口	柏油	乾燥
2022	1	20220103	151100	A1	雲林縣警	雲林縣二	晴	日間自然	縣道	50	交岔路	三岔路	交叉路口	交叉路口	柏油	乾燥

Part two-Data cleaning and preprocessing

1.Remove unused fields

2.Fill in fields with missing or null

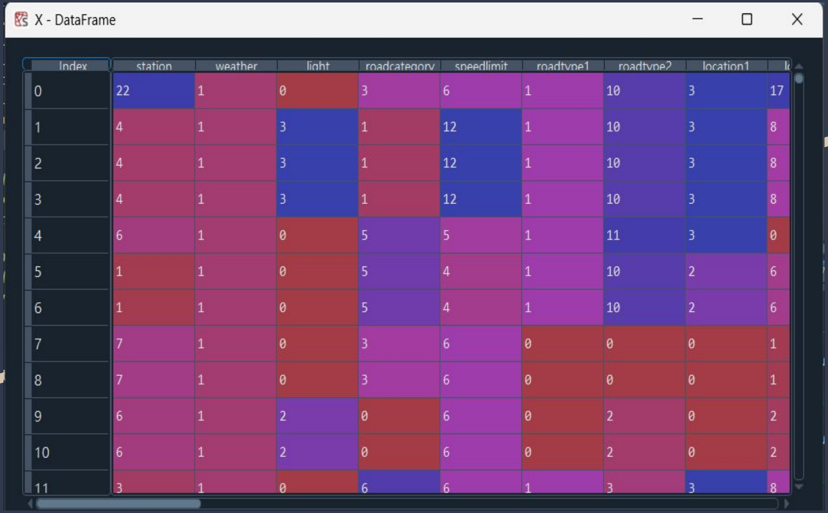
3.Define the target variable as a weight function

y_label - List (3 elements)

Inde	Type	Size	
0	str	5	1.9~3
1	str	5	3.1~4
2	str	5	4.1~9

```python
import pandas as pd
data = pd.read_csv('111年度A1交通事故資料.csv',encoding='big5')
columns_to_drop = [0, 1, 2, 3, 4, 6, 31, 33, 39, 41, 43, 44, 45, 46, 47, 48, 49, 50]
data = data.drop(data.columns[columns_to_drop], axis=1)
data[['死','傷']]=data['死亡受傷人數'].str.split(';',expand=True)
data['死'].str.slice(start=2)
data['傷'].str.slice(start=2)
data['當事者屬-性-別名稱'].fillna('', inplace=True)
 # 找出包含"無"、"或物"的行索引
rows_to_drop = data[data['當事者屬-性-別名稱'].str.contains('無或物')].index
data.drop(rows_to_drop, inplace=True)
rows_to_drop = data[data['當事者屬-性-別名稱'].str.contains('肇事逃逸尚未查獲')].index
 # 刪除相應的行
data.drop(rows_to_drop, inplace=True)
 # 刪除指定行
data = data.drop([4544, 4545])
data = data.reset_index(drop=True)
data.loc[1225, "車輛撞擊部位大類別名稱-最初"] = "機車"
data.loc[1783, "車輛撞擊部位大類別名稱-最初"] = "汽車"
missing_values = data.isnull().sum()
print(missing_values)
data['死']=data['死'].str.slice(start=2)
data['傷']=data['傷'].str.slice(start=2)
data = data.drop("死亡受傷人數", axis=1)
data.dtypes
data['死'] = data['死'].astype(int)
data['傷'] = data['傷'].astype(int)
data["死傷權重比"] = data['死'] * 2 + data['傷'] * 1
```

Use LabelEncorder to transform the data


X - DataFrame

Index	station	weather	light	roadcategory	speedlimit	roadtype1	roadtype2	location1	l
0	22	1	0	3	6	1	10	3	17
1	4	1	3	1	12	1	10	3	8
2	4	1	3	1	12	1	10	3	8
3	4	1	3	1	12	1	10	3	8
4	6	1	0	5	5	1	11	3	0
5	1	1	0	5	4	1	10	2	6
6	1	1	0	5	4	1	10	2	6
7	7	1	0	3	6	0	0	0	1
8	7	1	0	3	6	0	0	0	1
9	6	1	2	0	6	0	2	0	2
10	6	1	2	0	6	0	2	0	2
11	3	1	0	6	6	1	3	3	8

```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
le_station = le.fit_transform(data['處理單位名稱警局層'])
le_weather = le.fit_transform(data['天候名稱'])
le_light = le.fit_transform(data['光線名稱'])
le_roadcategory = le.fit_transform(data['道路類別-第1當事者-名稱'])
le_speedlimit = le.fit_transform(data['速限-第1當事者'])
le_roadtype1 = le.fit_transform(data['道路型態大類別名稱'])
le_roadtype2 = le.fit_transform(data['道路型態子類別名稱'])
le_location1 = le.fit_transform(data['事故位置大類別名稱'])
le_location2 = le.fit_transform(data['事故位置子類別名稱'])


le_Pavement1 = le.fit_transform(data["路面狀況-路面鋪裝名稱"])
le_Pavement2 = le.fit_transform(data["路面狀況-路面狀態名稱"])
le_Pavement3 = le.fit_transform(data["路面狀況-路面缺陷名稱"])
le_RoadObstacles1 = le.fit_transform(data["道路障礙-障礙物名稱"])
le_RoadObstacles2 = le.fit_transform(data["道路障礙-視距品質名稱"])
le_RoadObstacles3 = le.fit_transform(data["道路障礙-視距名稱"])
le_Traffic_signal = le.fit_transform(data["號誌-號誌種類名稱"])
le_Traffic_signa2 = le.fit_transform(data["號誌-號誌動作名稱"])
le_LaneDivisionFacilities1=le.fit_transform(data["車道劃分設施-分向設施大類別名稱"])
le_LaneDivisionFacilities2=le.fit_transform(data["車道劃分設施-分向設施子類別名稱"])
le_LaneDivisionFacilities3=le.fit_transform(data["車道劃分設施-分道設施-快車道或一般車道間名稱"])
le_LaneDivisionFacilities4=le.fit_transform(data["車道劃分設施-分道設施-快慢車道間名稱"])
le_LaneDivisionFacilities5=le.fit_transform(data["車道劃分設施-分道設施-路面邊線名稱"])
le_AccidentType1=le.fit_transform(data["事故類型及型態大類別名稱"])

le_AccidentType2=le.fit_transform(data["事故類型及型態子類別名稱"])
le_CauseJudgment=le.fit_transform(data["肇因研判大類別名稱-主要"])
le_PartiesInvolved1=le.fit_transform(data["當事者區分-類別-大類別名稱-車種"])
le_PartiesInvolved2=le.fit_transform(data["當事者區分-類別-子類別名稱-車種"])
le_Gender=le.fit_transform(data["當事者屬-性-別名稱"])
le_Protect=le.fit_transform(data["保護裝備名稱"])
```

Take y into three part

G1: 1.9 ~ 3 (mild)
G2: 3.1 ~ 4 (moderate)
G3: 4.1 ~ 9 (severe)

y - Series

Index	死傷權重比
0	1.9~3
1	1.9~3
2	1.9~3
3	1.9~3
4	1.9~3
5	1.9~3
6	1.9~3
7	1.9~3
8	1.9~3
9	1.9~3
10	1.9~3
11	1.9~3

```
le_AccidentType2=le.fit_transform(data["事故類型及型態子類別名稱"])
le_CauseJudgment=le.fit_transform(data["肇因研判大類別名稱-主要"])
le_PartiesInvolved1=le.fit_transform(data["當事者區分-類別-大類別名稱-車種"])
le_PartiesInvolved2=le.fit_transform(data["當事者區分-類別-子類別名稱-車種"])
le_Gender=le.fit_transform(data["當事者屬-性-別名稱"])
le_Protect=le.fit_transform(data["保護裝備名稱"])
le_State=le.fit_transform(data["當事者行動狀態大類別名稱"])
le_VehicleImpact=le.fit_transform(data["車輛撞擊部位大類別名稱-最初"])


age_label=["0~32",'32~65','65~98']
age=pd.cut(data["當事者事故發生時年齡"],bins=3, labels=age_label)
le_age=le.fit_transform(age)

X=pd.DataFrame([le_station,le_weather,le_light,le_roadcategory,le_speedlimit,
                le_roadtype1,le_roadtype2,le_location1,le_location2,
                le_Pavement1,le_Pavement2,le_Pavement3,le_RoadObstacles1,
                le_RoadObstacles2,le_RoadObstacles3,le_Traffic_signal,le_Traffic_signa2,
                le_LaneDivisionFacilities1,le_LaneDivisionFacilities2,le_LaneDivisionFacilities3,
                le_LaneDivisionFacilities4,le_LaneDivisionFacilities5,
                le_AccidentType1,le_AccidentType2,le_CauseJudgment,
                le_PartiesInvolved1,le_PartiesInvolved2,le_Gender,le_Protect,le_State,
                le_VehicleImpact,le_age,data['速限-第1當事者']

    ]).T
X.columns=['station','weather','light','roadcategory','speedlimit'
           ,'roadtype1','roadtype2','location1','location2','Pavement1',
           'Pavement2','Pavement3','RoadObstacles1','RoadObstacles2',
           'RoadObstacles3','Traffic signal','Traffic signa2','LaneDivisionFacilities1'
           ,'LaneDivisionFacilities2','LaneDivisionFacilities3','LaneDivisionFacilities4'
           ,'LaneDivisionFacilities5','AccidentType1','AccidentType2','CauseJudgment',
           'PartiesInvolved1','PartiesInvolved2','Gender','Protect','State','VehicleImpact'
           ,'age','Speed limit']

y_label=["1.9~3",'3.1~4','4.1~9']
y = pd.qcut(data["死傷權重比"], q=7, duplicates='drop',labels=y_label) #實際上只分成3個level

X.to_csv('X.csv', index=False)
y.to_csv('y.csv', index=False)
```

Part Three & Four–
Variable Selection and Decision Tree Modeling

Splitting the data into proportion (8:2)

X_train - DataFrame

Index	station	weather	light	roadcategory	speedlimit	roadtype1	roadtype2	location1	location2	Pavement1	Pavement2
1899	11	1	2	6	6	1	10	3	0	1	0
665	23	1	2	3	4	1	10	3	0	1	0
3701	22	1	2	4	6	1	10	3	0	1	0
3423	12	1	2	3	6	1	7	3	0	1	0
3505	14	1	2	3	6	1	10	3	0	1	0
145	11	1	2	7	6	1	7	3	0	1	0
3938	14	1	0	3	6	1	10	3	0	1	0
2834	8	1	0	4	6	1	7	3	0	1	0
3188	17	1	2	3	6	1	10	3	0	1	0
2465	9	1	2	3	6	1	10	3	0	1	0
1803	22	1	0	6	7	1	7	3	0	1	0

```python
#-------------------------------------
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,random_state=19911223)

'''
X_train.to_csv('X_train.csv', index=False)
X_test.to_csv('X_test.csv', index=False)
y_train.to_csv('y_train.csv', index=False)
y_test.to_csv('y_test.csv', index=False)
'''

from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
clf.fit(X_train,y_train)
print("分類樹的訓練正確率=",format(clf.score(X_train,y_train)*100,".2f"),"%")
print("分類樹的測試正確率=",format(clf.score(X_test,y_test)*100,".2f"),"%")

#分類樹的訓練正確率= 99.87 %
#分類樹的測試正確率= 84.20 %
```

Put train&test data into decision tree model

Use chi-square test to filter out the top k features

k=5 is the best choice

```
In [415]: print(clf.feature_importances_)
[0.04727185 0.01593263 0.08009521 0.          0.53219206
0.32450824]
```

X_train_new1 - DataFrame

Index	›DivisionFacilit	AccidentType2	'artiesInvolved	'artiesInvolved	Speed limit
0	3	2	6	26	40
1	0	9	7	23	50
2	0	3	10	15	30
3	4	25	6	23	110
4	0	9	6	23	50
5	0	4	10	16	50
6	1	2	7	23	70
7	0	24	10	16	50
8	3	3	6	23	50
9	1	11	6	23	60
10	0	3	9	20	50

Format Resize ☑ Background color ☑ Column min/max Save and Close Close

```
#卡方分配挑選變數過程(K=5)
from sklearn.feature_selection import SelectKBest, chi2
sk=SelectKBest(chi2,k=5)

sk.fit(X,y)
selected_features = X.columns[sk.get_support()]
print(selected_features)

X_train_new1=sk.transform(X_train)
X_train_new1=pd.DataFrame(X_train_new1)
X_train_new1.columns=['LaneDivisionFacilities3', 'AccidentType2', 'PartiesInvolved1',
       'PartiesInvolved2', 'Speed limit']

from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(criterion="gini",min_samples_split=0.2,
                           min_samples_leaf=2,random_state=20230410)

clf.fit(X_train_new1,y_train)
print(clf.feature_importances_)
print("卡方分配挑出top5建模資料集正確率=",format(clf.score(X_train_new1,y_train)*100,".2f"),"%")
#卡方分配挑出top5建模資料集正確率= 83.83 %

X_test_new1=sk.transform(X_test)
X_test_new1=pd.DataFrame(X_test_new1)
X_test_new1.columns=['LaneDivisionFacilities3', 'AccidentType2', 'PartiesInvolved1',
       'PartiesInvolved2', 'Speed limit']


clf.fit(X_test_new1,y_test)
print(clf.feature_importances_)
print("卡方分配挑出top5測試資料集正確率=",format(clf.score(X_test_new1,y_test)*100,".2f"),"%")
#卡方分配挑出top5測試資料集正確率= 80.40 %
```

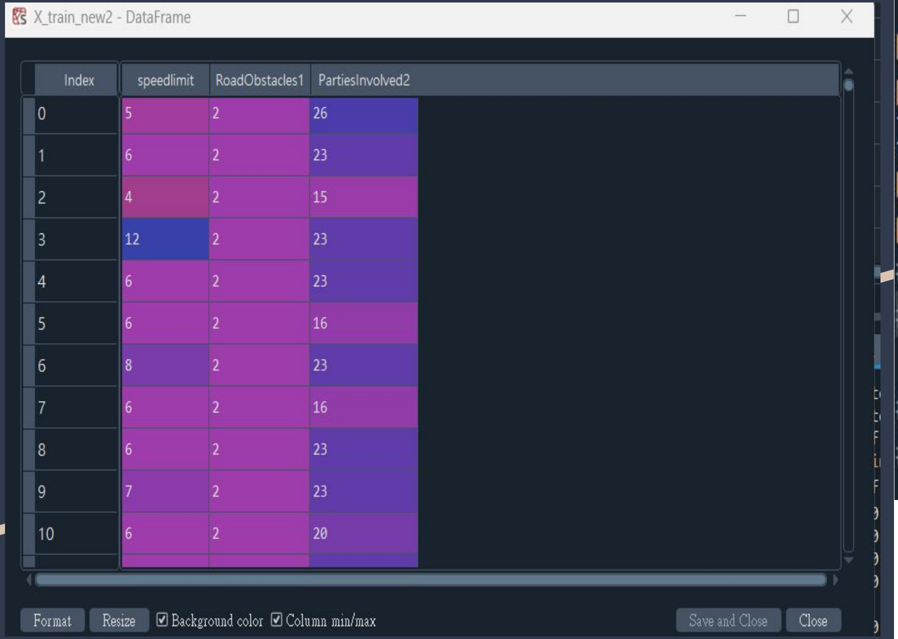
Use Decision Tree Classifier(Gini) to select features

3 variables are being selected

```
In [419]: sm=SelectFromModel(clf,max_features=8)

In [420]: sm.fit(X,y)
Out[420]:
SelectFromModel(estimator=DecisionTreeClassifier(min_samples_le
af=2,
                                                 min_samples_sp
lit=0.2,
                                                 random_state=2
0230410),
                max_features=8)

In [421]: print(sm.get_feature_names_out())#並沒有排序
['speedlimit' 'RoadObstacles1' 'PartiesInvolved2']
```

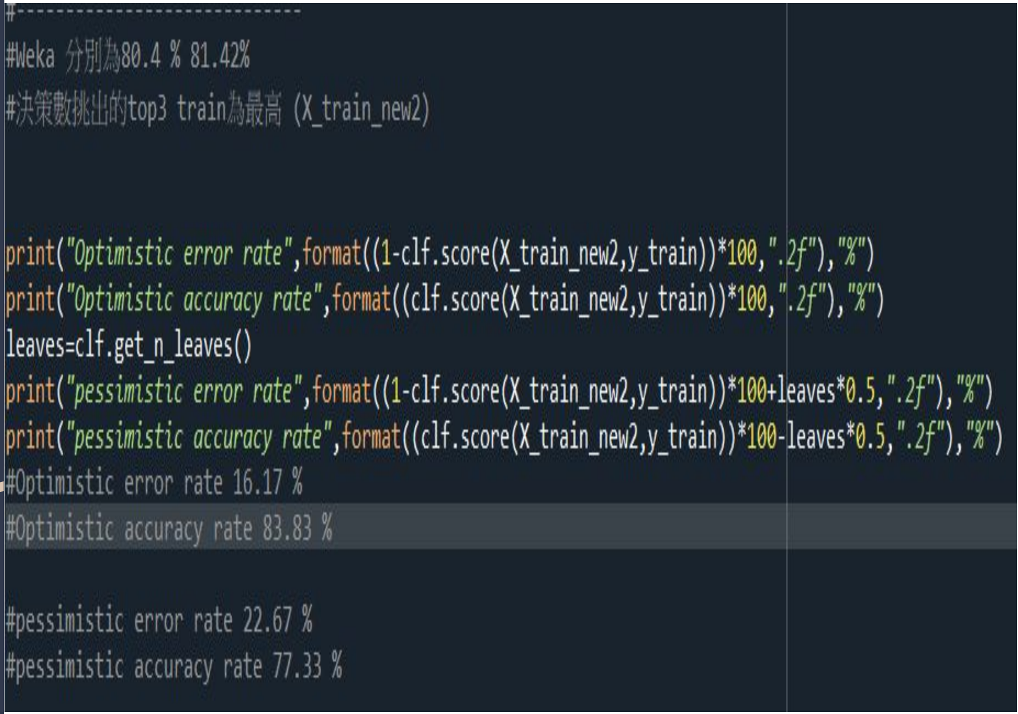
```
clf.fit(X,y)
print(clf.feature_importances_) #0代表沒挑選 他用gini係數  值越大影響越大

from sklearn.feature_selection import SelectFromModel
sm=SelectFromModel(clf,max_features=4)
sm.fit(X,y)
print(sm.get_feature_names_out())#並沒有排序

X_train_new2=sm.transform(X_train)
X_train_new2=pd.DataFrame(X_train_new2)
X_train_new2.columns=['speedlimit', 'RoadObstacles1', 'PartiesInvolved2']
clf.fit(X_train_new2,y_train)
print("決策樹模型自行挑出top3正確率=",format(clf.score(X_train_new2,y_train)*100,".2f"),"%")
#決策樹模型自行挑出top3正確率= 83.90 %

X_test_new2=sm.transform(X_test)
X_test_new2=pd.DataFrame(X_test_new2)
clf.fit(X_test_new2,y_test)
print("決策樹模型自行挑出top3正確率=",format(clf.score(X_test_new2,y_test)*100,".2f"),"%")
#最好的是決策數自己挑的train model
#決策樹模型自行挑出top3正確率= 80.40 %
```

Counting the accuracy rates for the optimistic method and the pessimistic method



X_train_new2 - DataFrame

Index	speedlimit	RoadObstacles1	PartiesInvolved2
0	5	2	26
1	6	2	23
2	4	2	15
3	12	2	23
4	6	2	23
5	6	2	16
6	8	2	23
7	6	2	16
8	6	2	23
9	7	2	23
10	6	2	20



```
#----------------------------
#Weka 分別為80.4 % 81.42%
#決策數挑出的top3 train為最高 (X_train_new2)


print("Optimistic error rate",format((1-clf.score(X_train_new2,y_train))*100,".2f"),"%")
print("Optimistic accuracy rate",format((clf.score(X_train_new2,y_train))*100,".2f"),"%")
leaves=clf.get_n_leaves()
print("pessimistic error rate",format((1-clf.score(X_train_new2,y_train))*100+leaves*0.5,".2f"),"%")
print("pessimistic accuracy rate",format((clf.score(X_train_new2,y_train))*100-leaves*0.5,".2f"),"%")
#Optimistic error rate 16.17 %
#Optimistic accuracy rate 83.83 %

#pessimistic error rate 22.67 %
#pessimistic accuracy rate 77.33 %
```

C4.5 in R weka

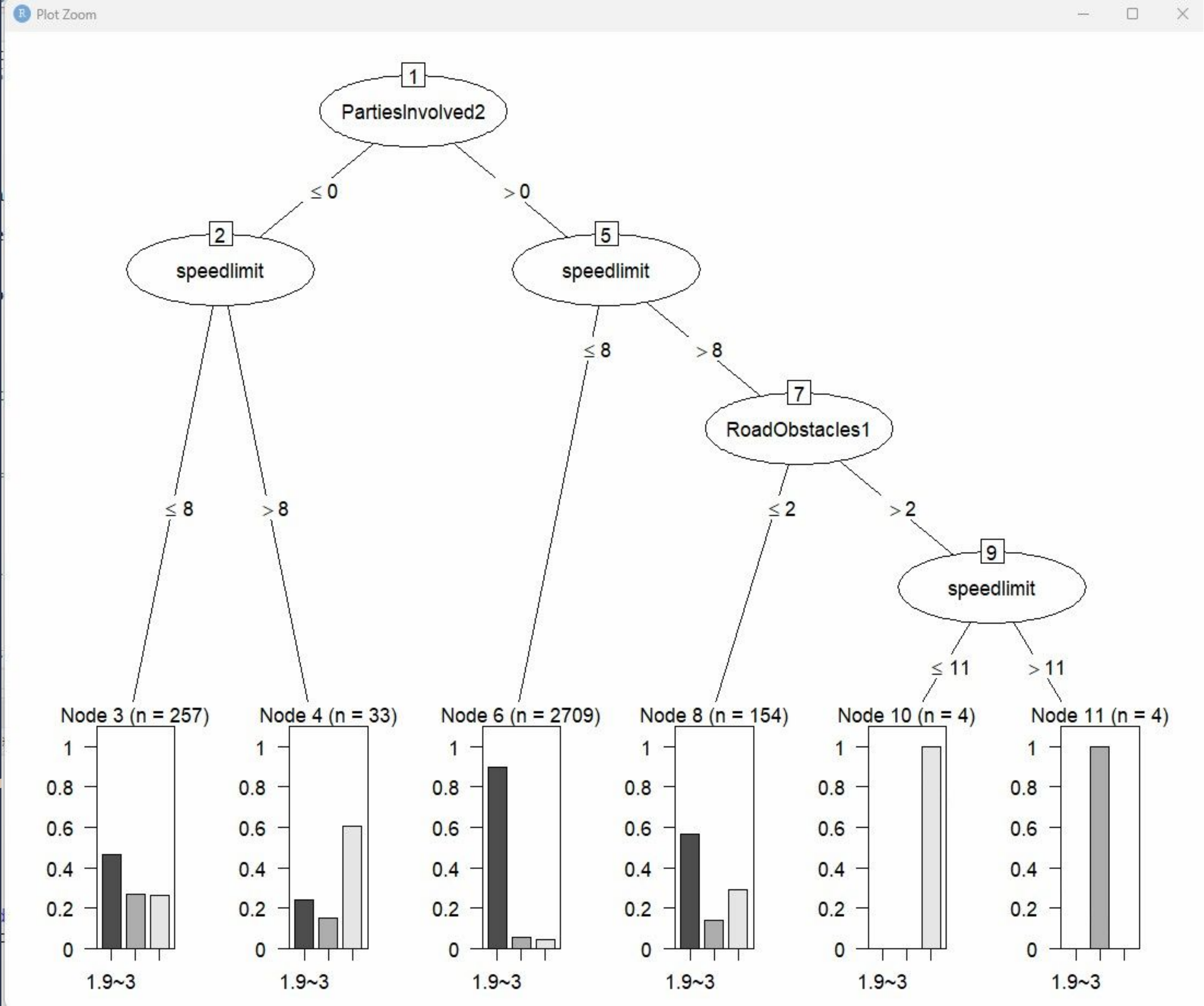
```
$ X0         : num  6 6 4 5 4 4 6 5 6 4 ...
$ X1         : num  2 2 2 2 2 2 2 2 2 2 ...
$ X2         : num  23 16 13 16 23 23 16 23...
$ 死傷權重比: Factor w/ 3 levels "1.9~3","3....
```

X0=Speed_limit
X1=RoadObstacles1
X2=PartiesInvolved2

```
> cm
        預測
實際    1.9~3 3.1~4 4.1~9
  1.9~3   636     0     0
  3.1~4    81     0     0
  4.1~9    74     0     0
```

636/791=80.4%

```
library(RWeka)
acc <- numeric(10)
for(i in 2:10) {
  ctree<-J48(死傷權重比~.,data=train,control=Weka_control(M=i,C=0.25,R=FALSE))

  test_predicted<-predict(ctree,test,type="class")
  test$predict<-test_predicted

  cm<-table(test$死傷權重比,test$predict,dnn=c("實際","預測"))
  acc[i]<-sum(diag(cm))/sum(cm)
}

max_acc<-max(acc)
cat("The highest accuracy for testing data set =",round(max_acc*100,2),"%")
library(partykit)

highest_acc_model<-which.max(acc)
ctree <- J48(死傷權重比 ~ .,data =train,control=Weka_control(M=highest_acc_mod
rparty.tree <- as.party(ctree)
plot(rparty.tree)
```


Plot Zoom
1
PartiesInvolved2
≤ 0
> 0
2
speedlimit
5
speedlimit
≤ 8
> 8
≤ 8
> 8
7
RoadObstacles1
≤ 2
> 2
9
speedlimit
≤ 11
> 11
Node 3 (n = 257)
Node 4 (n = 33)
Node 6 (n = 2709)
Node 8 (n = 154)
Node 10 (n = 4)
Node 11 (n = 4)
1
0.8
0.6
0.4
0.2
0
1.9~3

Association Rule

```
setwd("C://Users//user//OneDrive//桌面")
y_train <- read.csv('y_train.csv')
X_train <- read.csv('X_train.csv')
train <- data.frame(X_train, y_train)

train$station <- as.factor(train$station)
train$weather <- as.factor(train$weather)
train$light <- as.factor(train$light)
train$roadcategory <- as.factor(train$roadcategory)
train$speedlimit <- as.factor(train$speedlimit)
train$roadtype1 <- as.factor(train$roadtype1)
train$roadtype2 <- as.factor(train$roadtype2)
train$location1 <- as.factor(train$location1)
train$location2 <- as.factor(train$location2)
train$Pavement1 <- as.factor(train$Pavement1)
train$Pavement2 <- as.factor(train$Pavement2)
train$Pavement3 <- as.factor(train$Pavement3)
train$RoadObstacles1<- as.factor(train$RoadObstacles1)
train$RoadObstacles2<- as.factor(train$RoadObstacles2)
train$RoadObstacles3<- as.factor(train$RoadObstacles3)
train$Traffic.signal<- as.factor(train$Traffic.signal)
train$Traffic.signa2<- as.factor(train$Traffic.signa2)
train$LaneDivisionFacilities1<- as.factor(train$LaneDivisionFacilities1)
train$LaneDivisionFacilities2<- as.factor(train$LaneDivisionFacilities2)
train$LaneDivisionFacilities3<- as.factor(train$LaneDivisionFacilities3)
train$LaneDivisionFacilities4<- as.factor(train$LaneDivisionFacilities4)
train$LaneDivisionFacilities5<- as.factor(train$LaneDivisionFacilities5)
train$AccidentType1<- as.factor(train$AccidentType1)
train$AccidentType2<- as.factor(train$AccidentType2)
train$CauseJudgment<- as.factor(train$CauseJudgment)
train$PartiesInvolved1<- as.factor(train$PartiesInvolved1)
```

```
train$LaneDivisionFacilities2<- as.factor(train$LaneDivisionFacilities2)
train$LaneDivisionFacilities3<- as.factor(train$LaneDivisionFacilities3)
train$LaneDivisionFacilities4<- as.factor(train$LaneDivisionFacilities4)
train$LaneDivisionFacilities5<- as.factor(train$LaneDivisionFacilities5)
train$AccidentType1<- as.factor(train$AccidentType1)
train$AccidentType2<- as.factor(train$AccidentType2)
train$CauseJudgment<- as.factor(train$CauseJudgment)
train$PartiesInvolved1<- as.factor(train$PartiesInvolved1)
train$PartiesInvolved2<- as.factor(train$PartiesInvolved2)
train$Gender<- as.factor(train$Gender)
train$Protect<- as.factor(train$Protect)
train$State<- as.factor(train$State)
train$VehicleImpact<- as.factor(train$VehicleImpact)
train$age<- as.factor(train$age)
train$Speed.limit<- as.factor(train$Speed.limit)
train$死傷權重比 <- as.factor(train$死傷權重比)



require(arules)
rule<-apriori(train,parameter=list(supp=0.1,conf=0.95),
              appearance=list(rhs=c("死傷權重比=1.9~3","死傷權重比=3.1~4","死傷權重比=4.1~9'

sort.rule<-sort(rule,by="support")
subset.matrix<-as.matrix(is.subset(x=sort.rule,y=sort.rule))
subset.matrix[lower.tri(subset.matrix,diag=T)]<-NA

redundant<-colSums(subset.matrix,na.rm=T)>=1

sort.rule<-sort.rule[!redundant]
sort.rules<-as(sort.rule,"data.frame")
```

Association result

	rules	support	confidence	coverage	lift	count
44	{LaneDivisionFacilities3=0,Gender=1,Protect=2,State=1,age=1} => {死傷權重比=1.9~3}	0.1262259	0.9522673	0.1325530	1.135893	399
105	{RoadObstacles3=5,LaneDivisionFacilities3=0,LaneDivisionFacilities4=2,Gender=1,Protect=2,age=1} => {死傷權重比=1.9~3}	0.1154698	0.9505208	0.1214805	1.133810	365
106	{RoadObstacles2=1,LaneDivisionFacilities3=0,LaneDivisionFacilities4=2,Gender=1,Protect=2,age=1} => {死傷權重比=1.9~3}	0.1154698	0.9505208	0.1214805	1.133810	365
2	{roadcategory=4,RoadObstacles3=5,State=1} => {死傷權重比=1.9~3}	0.1148371	0.9502618	0.1208478	1.133501	363
3	{roadcategory=4,RoadObstacles2=1,State=1} => {死傷權重比=1.9~3}	0.1148371	0.9502618	0.1208478	1.133501	363
74	{weather=1,light=2,RoadObstacles1=2,LaneDivisionFacilities2=1,Gender=1,State=1} => {死傷權重比=1.9~3}	0.1094590	0.9505495	0.1151534	1.133844	346
79	{weather=1,light=2,RoadObstacles1=2,LaneDivisionFacilities1=2,Gender=1,State=1} => {死傷權重比=1.9~3}	0.1094590	0.9505495	0.1151534	1.133844	346
80	{light=2,roadcategory=3,RoadObstacles1=2,LaneDivisionFacilities5=1,Gender=1,State=1} => {死傷權重比=1.9~3}	0.1066118	0.9573864	0.1113572	1.141999	337
37	{RoadObstacles3=5,Traffic.signa2=3,LaneDivisionFacilities5=1,CauseJudgment=7,State=1} => {死傷權重比=1.9~3}	0.1062955	0.9545455	0.1113572	1.138611	336
38	{RoadObstacles2=1,Traffic.signa2=3,LaneDivisionFacilities5=1,CauseJudgment=7,State=1} => {死傷權重比=1.9~3}	0.1062955	0.9545455	0.1113572	1.138611	336
42	{RoadObstacles3=5,Traffic.signal=0,LaneDivisionFacilities5=1,CauseJudgment=7,State=1} => {死傷權重比=1.9~3}	0.1062955	0.9545455	0.1113572	1.138611	336
43	{RoadObstacles2=1,Traffic.signal=0,LaneDivisionFacilities5=1,CauseJudgment=7,State=1} => {死傷權重比=1.9~3}	0.1062955	0.9545455	0.1113572	1.138611	336
31	{LaneDivisionFacilities2=1,LaneDivisionFacilities3=0,Gender=1,State=1,age=1} => {死傷權重比=1.9~3}	0.1037646	0.9534884	0.1088263	1.137350	328
32	{LaneDivisionFacilities1=2,LaneDivisionFacilities3=0,Gender=1,State=1,age=1} => {死傷權重比=1.9~3}	0.1037646	0.9534884	0.1088263	1.137350	328
12	{Traffic.signa2=3,LaneDivisionFacilities5=1,Gender=1,State=1} => {死傷權重比=1.9~3}	0.1028156	0.9530792	0.1078773	1.136862	325
13	{Traffic.signal=0,LaneDivisionFacilities5=1,Gender=1,State=1} => {死傷權重比=1.9~3}	0.1028156	0.9530792	0.1078773	1.136862	325
11	{light=2,roadtype1=0,AccidentType2=2,State=1} => {死傷權重比=1.9~3}	0.1024992	0.9529412	0.1075609	1.136697	324
10	{light=2,location1=0,AccidentType2=2,State=1} => {死傷權重比=1.9~3}	0.1018665	0.9526627	0.1069282	1.136365	322
1	{roadcategory=4,Gender=1,State=1} => {死傷權重比=1.9~3}	0.1002847	0.9519520	0.1053464	1.135517	317

Showing 1 to 19 of 19 entries, 6 total columns

Part6 SVM(Support Vector Machine)

```python
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse = False)
encoder.fit(data[['車道劃分設施-分道設施-快車道或一般車道間名稱']])
LaneDivisionFacilities3 = encoder.transform(data[['車道劃分設施-分道設施-快車道或一般車道間名稱']
LaneDivisionFacilities3 = pd.DataFrame(LaneDivisionFacilities3)
LaneDivisionFacilities3.columns = encoder.categories_

encoder.fit(data[['事故類型及型態子類別名稱']])
AccidentType2 = encoder.transform(data[['事故類型及型態子類別名稱']])
AccidentType2 = pd.DataFrame(AccidentType2)
AccidentType2.columns = encoder.categories_


encoder.fit(data[['當事者區分-類別-大類別名稱-車種']])
PartiesInvolved1 = encoder.transform(data[['當事者區分-類別-大類別名稱-車種']])
PartiesInvolved1 = pd.DataFrame(PartiesInvolved1)
PartiesInvolved1.columns = encoder.categories_

encoder.fit(data[['當事者區分-類別-子類別名稱-車種']])
PartiesInvolved2 = encoder.transform(data[['當事者區分-類別-子類別名稱-車種']])
PartiesInvolved2 = pd.DataFrame(PartiesInvolved2)
PartiesInvolved2.columns = encoder.categories_


x1=pd.concat([LaneDivisionFacilities3,AccidentType2,PartiesInvolved1,PartiesInvolved2,data['速
y_label=["1.9~3",'3.1~4','4.1~9']
y = pd.qcut(data["死傷權重比"], q=7, duplicates='drop',labels=y_label) #實際上只分成3個level
```

```python
from sklearn.preprocessing import StandardScaler
scalar=StandardScaler()
scalar.fit(x1)
X_train_std=scalar.transform(x1)

from sklearn.svm import LinearSVC

m=LinearSVC(C=0.1, dual=False,class_weight="balanced")
m.fit(X_train_std,y)

y_pred=m.predict(X_train_std)

print("分類錯誤的資料筆數有=",(y!=y_pred).sum())
#分類錯誤的資料筆數有= 698

from sklearn.metrics import accuracy_score
print("正確率=",accuracy_score(y,y_pred))
#正確率= 0.8233805668016194

from sklearn.metrics import f1_score
print("F1-score=",f1_score(y,y_pred,average="weighted"))
#F1-score= 0.8101866336926598
```

```
分類錯誤的資料筆數有= 698
正確率= 0.8233805668016194
F1-score= 0.8101866336926598
```

Part Seven- RandomForest

```python
#7.Random forest
from sklearn.ensemble import RandomForestClassifier
#隨機森林裡深度統一=10

clf1=RandomForestClassifier(n_estimators=10,max_depth=10,random_state=19911223)
clf1.fit(X_train_new1,y_train)
print("樹木數量=10的建模正確率=",format(clf1.score(X_train_new1,y_train)*100,".2f"),"%")
print("樹木數量=10的測試正確率=",format(clf1.score(X_test_new1,y_test)*100,".2f"),"%")
#樹木數量=10的建模正確率= 88.45 %
#樹木數量=10的測試正確率= 81.67 %

clf2=RandomForestClassifier(n_estimators=50,max_depth=10,random_state=19911223)
clf2.fit(X_train_new1,y_train)
print("樹木數量=50的建模正確率=",format(clf2.score(X_train_new1,y_train)*100,".2f"),"%")
print("樹木數量=50的測試正確率=",format(clf2.score(X_test_new1,y_test)*100,".2f"),"%")
#樹木數量=50的建模正確率= 88.67 %
#樹木數量=50的測試正確率= 81.67 %
```

Part Seven- RandomForest

```
clf3=RandomForestClassifier(n_estimators=80,max_depth=10,random_state=19911223)
clf3.fit(X_train_new1,y_train)
print("樹木數量=80的建模正確率=",format(clf3.score(X_train_new1,y_train)*100,".2f"),"%")
print("樹木數量=80的測試正確率=",format(clf3.score(X_test_new1,y_test)*100,".2f"),"%")
#樹木數量=80的建模正確率= 88.71 %
#樹木數量=80的測試正確率= 81.67 %

clf4=RandomForestClassifier(n_estimators=100,max_depth=10,random_state=19911223)
clf4.fit(X_train_new1,y_train)
print("樹木數量=100的建模正確率=",format(clf4.score(X_train_new1,y_train)*100,".2f"),"%")
print("樹木數量=100的測試正確率=",format(clf4.score(X_test_new1,y_test)*100,".2f"),"%")
#樹木數量=100的建模正確率= 88.71 %
#樹木數量=100的測試正確率= 81.54 %

clf5=RandomForestClassifier(n_estimators=120,max_depth=10,random_state=19911223)
clf5.fit(X_train_new1,y_train)
print("樹木數量=120的建模正確率=",format(clf5.score(X_train_new1,y_train)*100,".2f"),"%")
print("樹木數量=120的測試正確率=",format(clf5.score(X_test_new1,y_test)*100,".2f"),"%")
#樹木數量=120的建模正確率= 88.64 %
#樹木數量=120的測試正確率= 81.54 %
```

RandomForest

```
樹木數量=10的建模正確率= 88.45 %
樹木數量=10的測試正確率= 81.67 %
樹木數量=50的建模正確率= 88.67 %
樹木數量=50的測試正確率= 81.67 %
樹木數量=80的建模正確率= 88.71 %
樹木數量=80的測試正確率= 81.67 %
樹木數量=100的建模正確率= 88.71 %
樹木數量=100的測試正確率= 81.54 %
樹木數量=120的建模正確率= 88.64 %
樹木數量=120的測試正確率= 81.54 %
```

```
#當樹木數量為80實有最佳的測試正確率
#當樹木數量為100時，測試正確率已開始收斂
#當樹木數量為120時，建模正確率已開始降低
#因此決定設置80棵樹，深度10作為模型的參數
```

K-Nearest Neighbor

```python
X1_train, X1_test, y_train, y_test = train_test_split(x1,y,test_size=0.2
                                                      ,random_state=19911223)
#訓練資料集
acc=[]
for i in range (2,792):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X1_train,y_train)
    y_pred = knn.predict(X1_train)
    acc.append(accuracy_score(y_train, y_pred))

#print("k=",i,"訓練資料集的正確率 = ",knn.score(y_train_new1,y_pred))
#print(acc)
print("最佳的K=",acc.index(max(acc))+2,"訓練資料集最佳正確率=",max(acc))
#最佳的K= 4 訓練資料集最佳正確率= 0.8712432774438469
```

Index	Type	Size	Value
2	float64	1	0.8712432774438469
0	float64	1	0.8702942106928188
3	float64	1	0.8699778551091427
4	float64	1	0.8655488769376779
5	float64	1	0.8639670990192977
1	float64	1	0.8636507434356216
6	float64	1	0.8611198987662132
7	float64	1	0.8608035431825372
8	float64	1	0.859538120847833
9	float64	1	0.859538120847833
10	float64	1	0.8563745650110725
11	float64	1	0.8563745650110725
13	float64	1	0.853527364757988

```python
#測試資料集
acc = []
for i in range (1,792):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X1_train,y_train)
    y_pred = knn.predict(X1_test)
    acc.append(accuracy_score(y_test, y_pred))

#print("k=",i,"測試資料集的正確率 = ",accuracy_score(y_test,y_pred))


print("最佳的K=",acc.index(max(acc))+1,"測試資料集最佳正確率=",max(acc))

#最佳的K= 8 測試資料集最佳正確率= 0.8217446270543616
```

acc - List (791 elements)

Index	Type	Size	Value
7	float64	1	0.8217446270543616
8	float64	1	0.8217446270543616
9	float64	1	0.8166877370417194
5	float64	1	0.8141592920353983
6	float64	1	0.8141592920353983
4	float64	1	0.8128950695322377
11	float64	1	0.8128950695322377
10	float64	1	0.8116308470290771
12	float64	1	0.8116308470290771

Part Nine- Comparison of Accuracy for Various Testing Datasets in Different Methods

- SVM
- Random Forest
- KNN
- Hard Voting
- Soft Voting

Voting Model-Hard

```python
#9綜合比較voting

from sklearn.ensemble import RandomForestClassifier
clf1=RandomForestClassifier(n_estimators=10,random_state=19911223)
clf1.fit(X_train,y_train)


from sklearn.neighbors import KNeighborsClassifier
clf2=KNeighborsClassifier(n_neighbors=4)

from sklearn.svm import SVC
clf3=SVC(gamma=0.1,kernel="rbf",probability=True)

from sklearn.ensemble import VotingClassifier

clf4=VotingClassifier(estimators=[("RF",clf1),("KNN",clf2),("SVC",clf3)],voting="hard",n_jobs=-1)

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)
X_train_std=scaler.transform(X_train)
clf4.fit(X_train_std,y_train)

print("訓練資料集的正確率=",clf4.score(X_train_std,y_train))

std_x_test=scaler.transform(X_test)
y_pred=clf4.predict(std_x_test)

from sklearn.metrics import accuracy_score
print("測試資料集的正確率=",accuracy_score(y_test,y_pred))
#Hard_Voting測試資料集的正確率= 0.8647281921618205
```

Voting Model-Soft

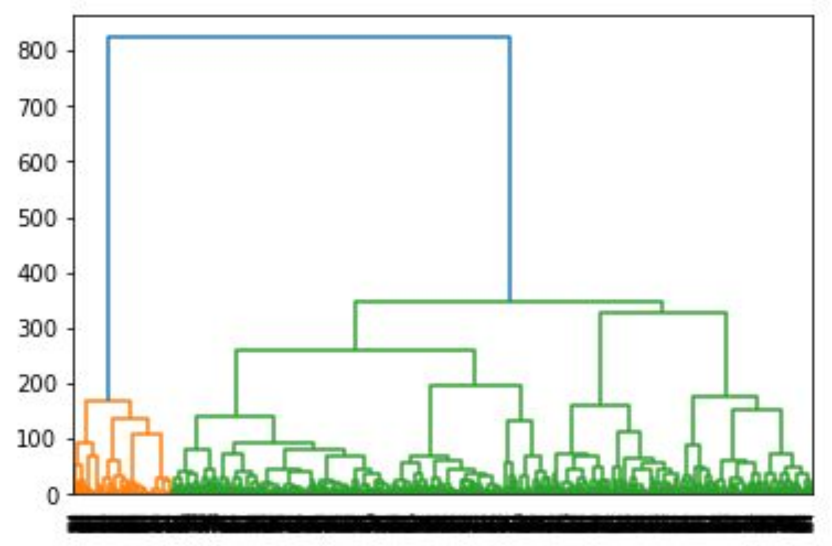
```
clf5=VotingClassifier(estimators=[("RF",clf1),("KNN",clf2),("SVC",clf3)],voting="soft",n_jobs=-1)

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)
X_train_std=scaler.transform(X_train)
clf5.fit(X_train_std,y_train)

print("訓練資料集的正確率=",clf5.score(X_train_std,y_train))
#調整上三模型參數看結果

std_x_test=scaler.transform(X_test)
y_pred=clf5.predict(std_x_test)

from sklearn.metrics import accuracy_score
print("測試資料集的正確率=",accuracy_score(y_test,y_pred))
#Soft_Voting測試資料集的正確率= 0.8596713021491783

```

Comparing Five Methods

Predictive Models	Testing Accuracy
SVM	0.8234
Random Forest	0.8167
KNN	0.8217
Hard Voting	0.8647
Soft Voting	0.8597

Part Ten-
Hierarchical Clustering

we suggest that if the threshold=29.5, number of clusters will be 203.




```
#10.階層式分群分析
import numpy as np
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering

HC=AgglomerativeClustering(n_clusters=203,affinity='euclidean',linkage='ward',compute_distances=True)
HC.fit(X)
#print(HC,children_)
print(HC.distances_)

#畫圖看分幾群
dis=sch.linkage(X,metric="euclidean",method='ward')
sch.dendrogram(dis)#距離長代表是好的切割點

#算出要分幾群
HC=AgglomerativeClustering(n_clusters=None,affinity='euclidean',linkage='ward',distance_threshold=29.5)
HC.fit(X)
print(HC.n_clusters_)  #叫他算分幾群比較好
#應該分203群


y_pred=HC.fit_predict(X)

for i in range(203):
    print("第",i+1,"群有",np.sum(y_pred==i))
```

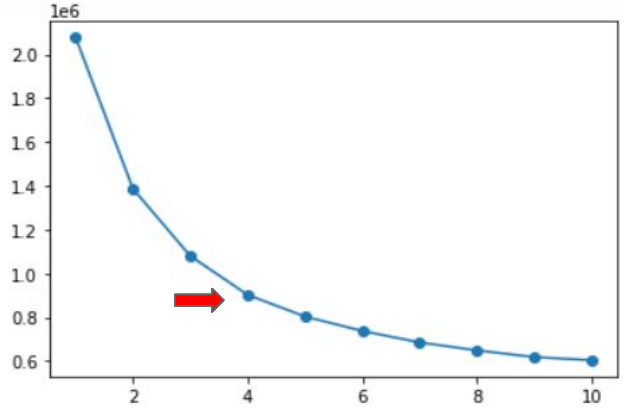
Part Eleven-Kmeans

Group number determination of K:

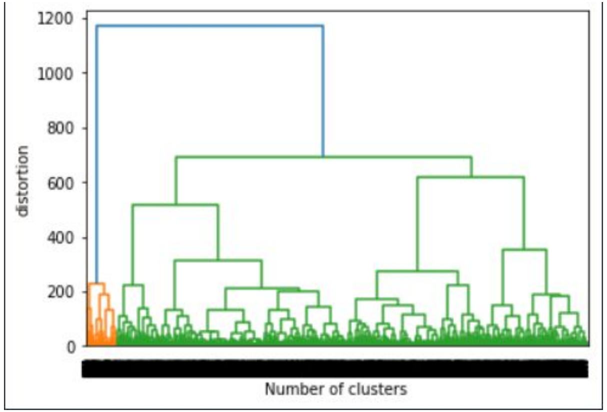
1. Steep slope map (K=4)

2. Hierarchical (K=203)

Steep slope map



Hierarchical



SSE calculation

SSE of Steep Slope map
= 902425.1423689526

SSE of Hierarchical
=139716.95423965578

After comparison, smaller SSE is better result
Hierarchical clustering

```
col_0  0    1    2    3    4    5    6   ...  196  197  198  199  200  201  \
202
死傷權重比                                      ...
1.9~3   80   22   30    9   23    6   19  ...   20   10   10    3    3    7
13
3.1~4    8    8    0    1    1    7    5  ...    2    0    0    1    0    2
0
4.1~9   11   10    1    0    1    8   16  ...    3    0    0    2    1    4
0

[3 rows x 203 columns]
階層式的SSE= 140228.24169195286
```

```
col_0     0     1    2    3
死傷權重比
1.9~3   981  1530  119  656
3.1~4   120   136   40   40
4.1~9    93    95   97   45
K=4時 accuracy= 0.492153443766347
陡坡圖的SSE= 902425.1423689527
```

```python
#陡坡圖SSE計算與正確率
from sklearn.cluster import KMeans

distortion=[]

for i in range(10):
    kmeans=KMeans(n_clusters=i+1, init="k-means++",random_state=19911223,
                  n_init=15, max_iter=200)
    kmeans.fit(X)
    distortion.append(kmeans.inertia_)
print(distortion)

import matplotlib.pyplot as plt
plt.plot(range(1,11),distortion,marker="o")
plt.xlabel("Number of clusters")
plt.ylabel("distortion")

kmeans2=KMeans(n_clusters=4, init="k-means++",random_state=19911223,
              n_init=15, max_iter=200)
kmeans2.fit(X)
centroid=pd.DataFrame(kmeans2.cluster_centers_, columns=X.columns)

X_pred=kmeans2.predict(X)
print(pd.crosstab(y,X_pred))
print("K=4時 accuracy=",(670+1082+61+445)/4588) #取比較多的
print("陡坡圖的SSE=",kmeans2.inertia_)

#accuracy= 0.492153443766347
#陡坡圖的SSE= 902425.1423689526
```

```python
#階層式分群結果(203群)的SSE計算
kmeans3=KMeans(n_clusters=203, init="k-means++",random_state=19911223,
              n_init=15, max_iter=200)
kmeans3.fit(X)
centroid=pd.DataFrame(kmeans3.cluster_centers_, columns=X.columns)

X_pred=kmeans3.predict(X)
print(pd.crosstab(y,X_pred))
print("階層式的SSE=",kmeans3.inertia_)
#階層式的SSE= 140228.24169195286
```

The accuracy of hierarchical

The final accuracy of hierarchical
= 0.728204010462075

```
col_0
0       80
1       22
2       30
3        9
4       23
        ..
198     10
199      3
200      3
201      7
202     13
Length: 203, dtype: int64
3341
K=203時 accuracy= 0.728204010462075
```

```
#階層式分群的正確率計算
a = pd.crosstab(y, X_pred)
column_max = a.max()
print(column_max)
total_sum = column_max.sum()
print(total_sum)
print("K=203時 accuracy=",(3341)/4588) #取比較多的
#accuracy= 0.728204010462075
```

a - DataFrame

死傷權重比	0	1	2	3	4
1.9~3	80	22	30	9	23
3.1~4	8	8	0	1	1
4.1~9	11	10	1	0	1

Description of each category of grouping

centroid - DataFrame

Index	station	weather	light	oadcategory	speedlimit	roadtype1	roadtype2	location1	lo
0	13.1414	1.50505	1.16162	3.13131	6	0	1.50505	0.0606061	1.67
1	4	1.875	0.95	1	12	1	10	3	8
2	21.7742	1.09677	1	3.6129	6	0	1.87097	2.22045e-16	1.96
3	10	1.8	1.7	3.3	4	0	0.4	2.22045e-16	1.7
4	7.52	1.88	0.84	3.8	6	1	9.24	2.96	0.24
5	19.4762	1.66667	1.57143	3.90476	5.80952	1	10.0476	3	3.90
6	4	1.65	0.575	1	12	1	10.05	2.9	7.95
7	7.875	3.25	0	4	7.875	1	10	3	17
8	15.2308	1.69231	1.53846	3.92308	8	0.0769231	1.30769	0.153846	2.15
9	22.4	1.4	1.5	3.4	6	1	9.5	3	13.9
10	13.3333	1.82222	1.02222	3.13333	6	-1.11022e-16	1.04444	0.0666667	1.95
11	6.56522	1.17391	1.21739	3.91304	4.13043	0.608696	5.95652	1.78261	0.95

Characteristics of the 死傷權重比 is 1.9~3 (group 35):

The station value is too small, the Speedlimit is too large, and the PartiesInvolved2 is too large

Characteristics of the 死傷權重比 is 3.1~4 (group 73):

The light is too small, the LaneDivisionFacilities3 is too large, and the Speedlimit is too large

Characteristics of the 死傷權重比 is 4.1~9(group 21):

Light is too small, Speedlimit is too big, Pavement2 is too big

#---

The group with 死傷權重比 1.9~3 tends to have a larger Speedlimit value

Groups with 死傷權重比 3.1~4 tend to have an abnormally small Trafficsignal value

Groups with 死傷權重比 4.1 to 9 tend to have a smaller light value

Part Twelve-best classification prediction model

C4.5(RWEKA)	0.8040
CART	0.8040
SVM	0.8233
RANDOM FOREST	0.8167
KNN	0.8217
HARD VOTING	0.8597
SOFT VOTING	0.8647
K-means	0.7282

Part Thirteen-conclusion

Importantvariables

speedlimit

RoadObstacles1

PartiesInvolved2