HW1 Decision Tree

0853412 吳宛儒

作業基本要求：

資料集 - https://www.kaggle.com/c/sf-crime/data

特徵 - 使用五個欄位 Dates, DayofWeek, PdDistrict, X, Y

標籤 - 使用 Category 欄位

0) 引入基本套件

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1) 將資料讀取進來

```
data = pd.read_csv("train.csv")
```

2) 資料前處理

2-1 把空值以 0 替代

```
data = data.fillna("0")
```

2-2 將 DayofWeek, PdDistrict 轉成 dummy 特徵

```
# dummies
# PdDistrict
PdDistrict = pd.get_dummies(data['PdDistrict'])
print(list(PdDistrict))

# DayofWeek
DayOfWeek = pd.get_dummies(data['DayOfWeek'])
print(list(DayOfWeek))
```

2-3 從 Dates 抓取小時，並轉成 dummy 特徵

```
# Dates 切開
new = data['Dates'].str.split(" ", n = 1, expand = True)
data['Date'] = new[0]
data['Time'] = new[1]
del data['Dates']

# 建立新欄位 Hour
data['Hour'] = data['Time'].str.split(":", n = 2, expand = True)[0]

# Hour dummies
Hour = pd.get_dummies(data['Hour'])
print(list(Hour))
```

將上面兩點步驟整理好的 dummies 加入原本的 data 中：

```
# concat to original one
data = pd.concat([data, PdDistrict, DayOfWeek, Hour], axis = 1)
print(list(data))
```

至此，基本的 features 有以下幾種：

features = ['X', 'Y', 'BAYVIEW', 'CENTRAL', 'INGLESIDE', 'MISSION', 'NORTHERN', 'PARK', 'RICHMOND', 'SOUTHERN', 'TARAVAL', 'TENDERLOIN', 'Friday', 'Monday', 'Saturday', 'Sunday', 'Thursday', 'Tuesday', 'Wednesday', '00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23']

2-4（加分題，5%）額外處理

除了上述所要求的欄位以外，先觀察資料以找出各項 features 和我們所要找出的答案 target(也就是 Category 欄位)有甚麼關係，並做 features 的篩選與討論。

2-4-1 印出現有欄位

```
# 印出欄位
print(list(data))
```

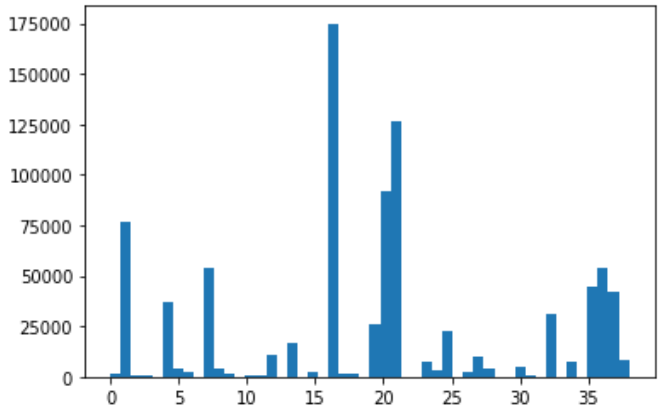
['Category', 'Descript', 'DayOfWeek', 'PdDistrict', 'Resolution', 'Address', 'X', 'Y', 'Date', 'Time', 'Hour', 'BAYVIEW', 'CENTRAL', 'INGLESIDE', 'MISSION', 'NORTHERN', 'PARK', 'RICHMOND', 'SOUTHERN', 'TARAVAL', 'TENDERLOIN', 'Friday', 'Monday', 'Saturday', 'Sunday', 'Thursday', 'Tuesday', 'Wednesday', '00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23']

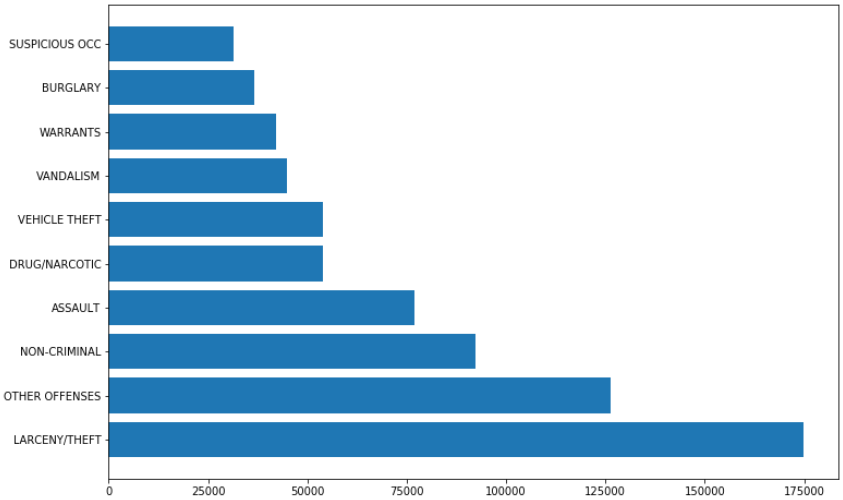
2-4-2 觀察犯罪種類的特色

下圖：觀察各種犯罪的數量，下二圖：觀察最多的犯罪種類

```
# Distribution of crimes
from sklearn.preprocessing import LabelEncoder
le_cat = LabelEncoder()
le_cat.fit(data.Category)
Ytrain = le_cat.transform(data.Category)

plt.hist(Ytrain, bins = 50)
```





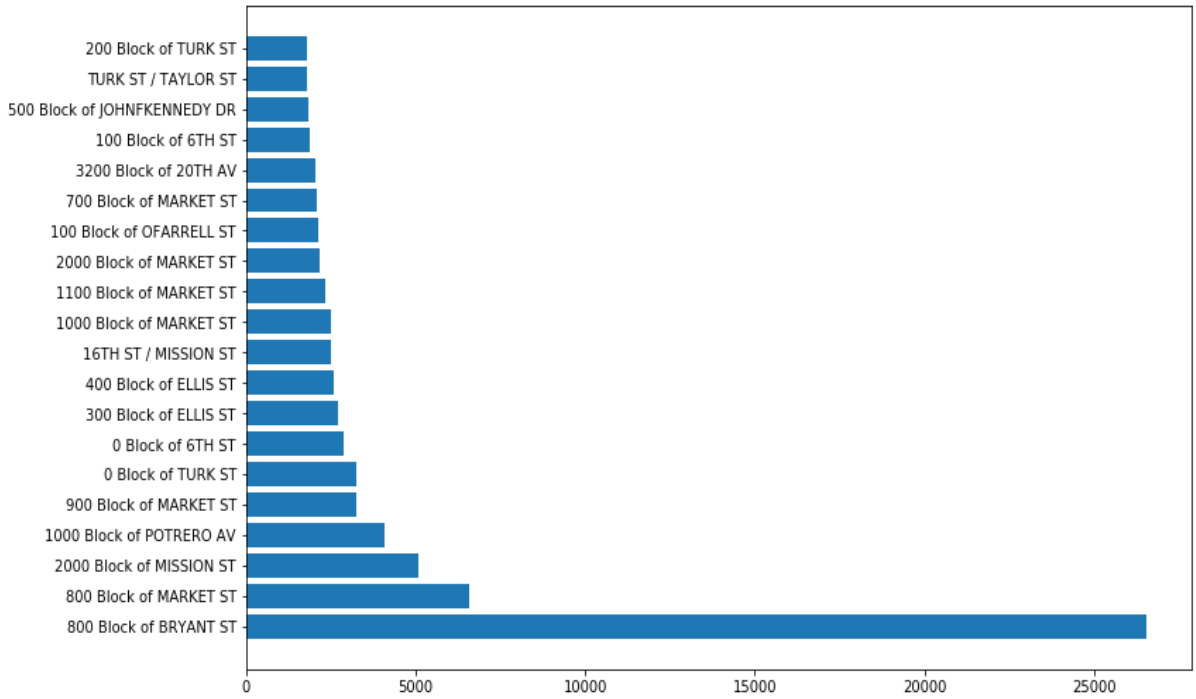

```
# The most popular crimes
top_crimes = data.Category.value_counts()[:10]
plt.figure(figsize=(12, 8))
pos = np.arange(len(top_crimes))
plt.barh(pos, top_crimes.values)
plt.yticks(pos, top_crimes.index)
```

2-4-3 Address 和 Category 的關係

```
# Address 獨特值的數量
print(len(data.Address.unique()))

# The most criminal locations
top_addresses = data.Address.value_counts()[:20]
plt.figure(figsize=(12, 8))

pos = np.arange(len(top_addresses))
plt.barh(pos, top_addresses.values)
plt.yticks(pos, top_addresses.index)
```



可以看出 800 Block of BRYANT ST 特別多，

因此新增一欄位 most 分出：800 Block of BRYANT ST 和其他。

```
data['most'] = 0
data.loc[data['Address'] == top_addresses.index[0], 'most'] = 1
```

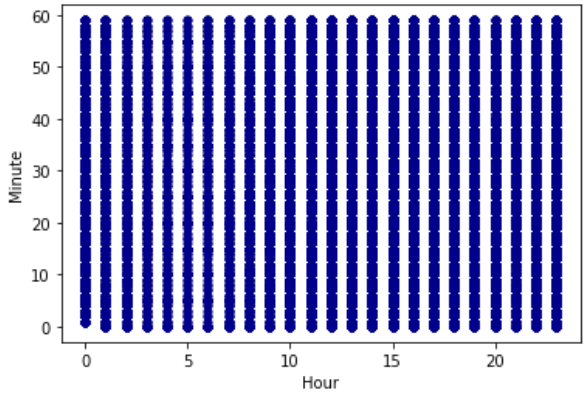
2-4-4 Hour 和 Category 的關係

先整理欄位

```
data['Hour'] = data['Time'].str.split(":", n = 2, expand = True)[0].astype(int)
data['Minute'] = data['Time'].str.split(":", n = 2, expand = True)[1].astype(int)
data['Second'] = data['Time'].str.split(":", n = 2, expand = True)[2].astype(int)
```

看小時與分鐘是否有特殊分布，結果並沒有。

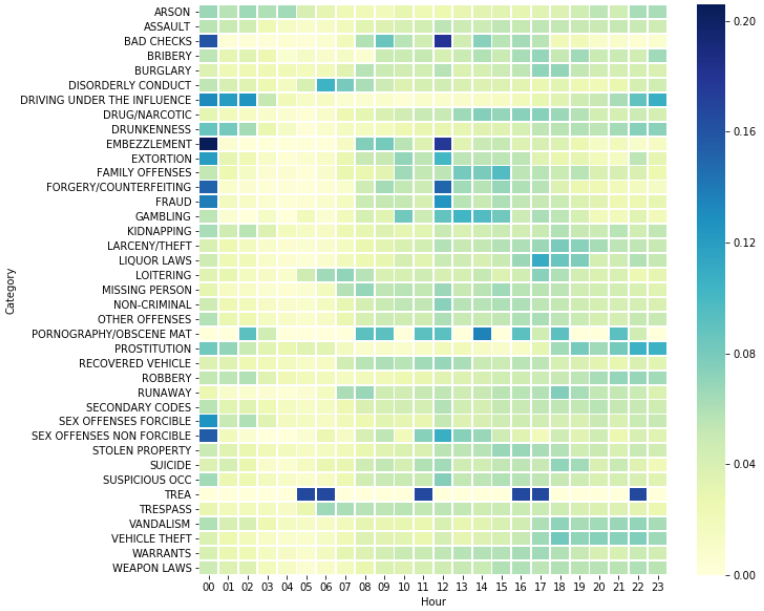
```
data.plot.scatter(x='Hour',y='Minute', color='DarkBlue', alpha=0.1)
```



接著觀察 Hour 和 Category 的關係

```
# Heatmap: Category and Hour
hour_cross_cat = pd.crosstab(data.Category, data.Hour).apply(lambda r: r/r.sum(), axis=1)

plt.figure(figsize=(10, 10))
sns.heatmap(hour_cross_cat, linewidths=.9, cmap="YlGnBu")
```

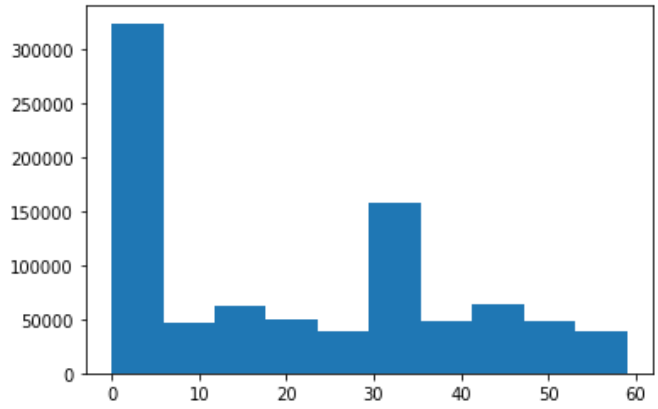


可以看出並沒有特別集中在某些時刻，種類之間也沒有明顯集中的現象。

2-4-5 Minute 和 Category 的關係

```
minute_cross_cat = pd.crosstab(data.Category, data.Minute).apply(lambda r: r/r.sum(), axis=1)

plt.figure(figsize=(10, 10))
sns.heatmap(minute_cross_cat, linewidths=.9, cmap="Greens")
```



可以看出時間有集中在 0 分以及 30 分的現象，可能紀錄的時間是半小時有一統整，因此認為可能可以將資料分為是否在 0 分或 30 分登記與其他。

```
'''
新欄位：min_cat
將Minute拆為兩個：0和30分、非0或30分
'''
data['min_cat'] = 0
mask1 = (data['Minute'] == 0)|(data['Minute'] == 30)
data.loc[mask1, 'min_cat'] = 1
```

2-4-6 Year 和 Category 的關係

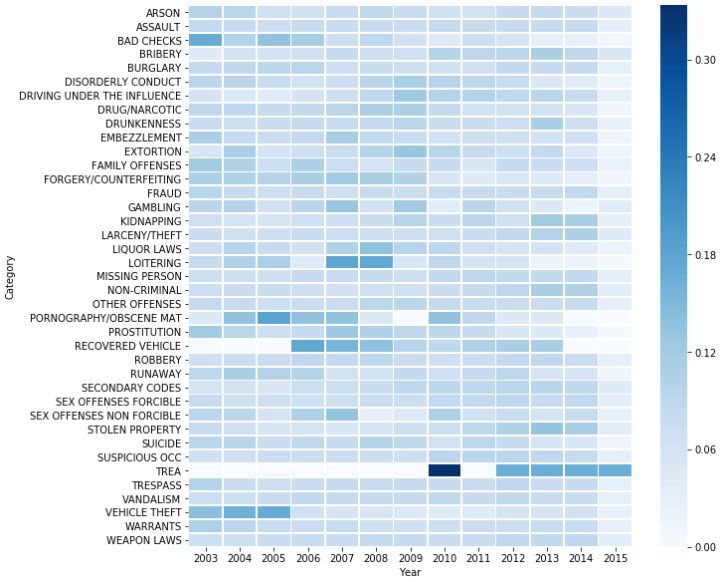
先將日期有關的欄位做分解與整理

```
data['Year'] = data['Date'].str.split("-", n=2, expand = True)[0]
data['Month'] =  data['Date'].str.split("-", n = 2, expand = True)[1]
data['Day'] =  data['Date'].str.split("-", n = 2, expand = True)[2]
```

再來看 Year 和 Category 的關係

```
# Heatmap: Category and Year
year_cross_cat = pd.crosstab(data.Category, data.Year).apply(lambda r: r/r.sum(), axis=1)

plt.figure(figsize=(10, 10))
sns.heatmap(year_cross_cat, linewidths=.9, cmap="Blues")
```



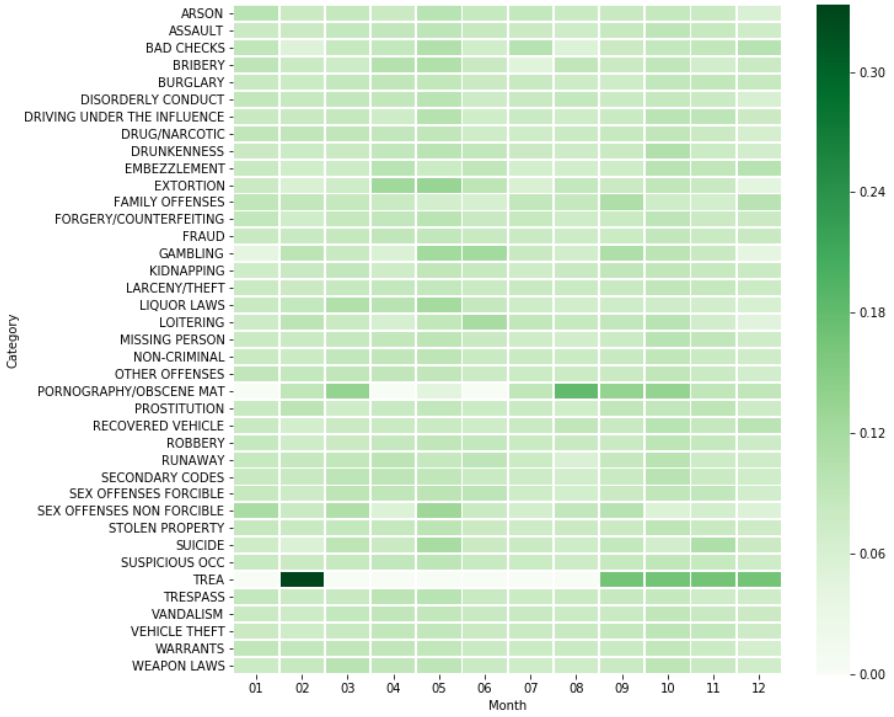
可以看到 2010 年有某種犯罪 TREA 集中的現象，其他犯罪種類也有在某些年份有較為集中的趨勢，因此決定先將 Year 欄位(範圍 2003~2015 年)壓到-6~+6 之間做類似正規化。

```
'''
新建欄位Year_norm：
將Year(範圍2003~ 2015)壓到-6 ~ +6之間
'''
data['Year_norm'] = data['Year'].astype(int)-2009
```

2-4-7 Month 和 Category 的關係

```
# Heatmap: Category and Month
month_cross_cat = pd.crosstab(data.Category, data.Month).apply(lambda r: r/r.sum(), axis=1)

plt.figure(figsize=(10, 10))
sns.heatmap(month_cross_cat, linewidths=.9, cmap="Greens")
```



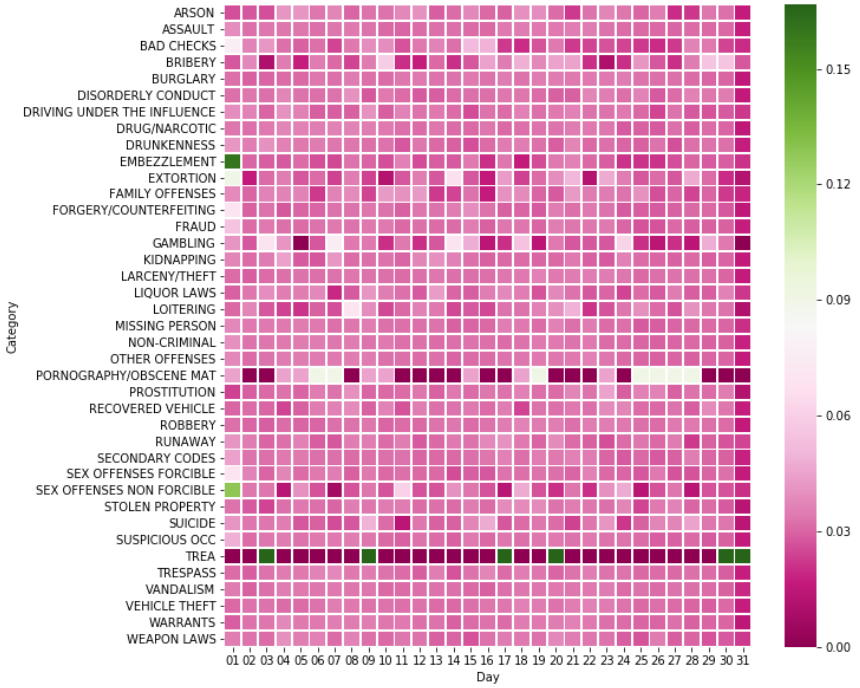
可以明顯看出 2 月份 TREA 種類的犯罪相當多且集中，因此我們新增一個欄位將二月與其他月份的資料分開來。

```
'''
新建欄位FEB：
'''
# 分是否為二月
data['FEB'] = 0
mask1 = (data['Month'] == '02')
data.loc[mask1, 'FEB'] = 1
```

2-4-8 Day 和 Category 的關係

```
# Heatmap: Category and Day
day_cross_cat = pd.crosstab(data.Category, data.Day).apply(lambda r: r/r.sum(), axis=1)

plt.figure(figsize=(10, 10))
sns.heatmap(day_cross_cat, linewidths=.9, cmap="PiYG")
```



可以看出來除了 TREA 犯罪種類非常多以外，依照日期來分，犯罪並沒有特別向某天集中。因此日期欄位我們不再多做處理也不會納入 features。

2-4-9 DayOfWeek 和 Category 的關係

```
# Heatmap: Category and DayOfWeek
import seaborn as sns
day_cross_cat = pd.crosstab(data.Category, data.DayOfWeek).apply(lambda r: r/r.sum(), axis=1)

plt.figure(figsize=(10, 10))
sns.heatmap(day_cross_cat, linewidths=.9, cmap="BuPu")
```

由於一開始便有將 DayOfWeek 做 dummies，因此這邊先不再做其他處理。

2-4-10 Resolution 和 Category 的關係

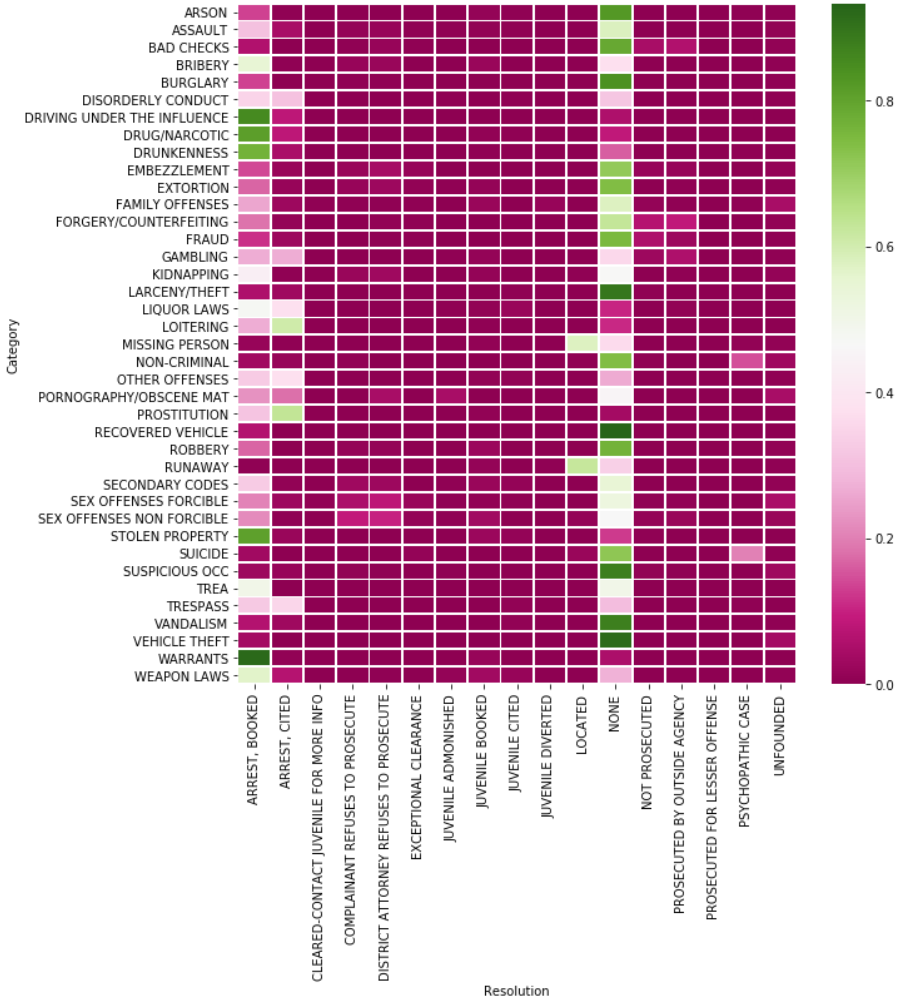
先查看有多少種 Resolution

```
print(data.Resolution.unique())
```

```
array(['ARREST, BOOKED', 'NONE', 'ARREST, CITED', 'PSYCHOPATHIC CASE',
       'JUVENILE BOOKED', 'UNFOUNDED', 'EXCEPTIONAL CLEARANCE', 'LOCATED',
       'CLEARED-CONTACT JUVENILE FOR MORE INFO', 'NOT PROSECUTED',
       'JUVENILE DIVERTED', 'COMPLAINANT REFUSES TO PROSECUTE',
       'JUVENILE ADMONISHED', 'JUVENILE CITED',
       'DISTRICT ATTORNEY REFUSES TO PROSECUTE',
       'PROSECUTED BY OUTSIDE AGENCY', 'PROSECUTED FOR LESSER OFFENSE'],
      dtype=object)
```

```
# Heatmap: Category and Resolution
resolution_cross_cat = pd.crosstab(data.Category, data.Resolution).apply(lambda r: r/r.sum(), axis=1)

plt.figure(figsize=(10, 10))
sns.heatmap(resolution_cross_cat, linewidths=.9, cmap="PiYG")
```



可以看出資料大部分集中在 ARREST，BOOKED、NONE，採取做法：分為 **ARREST, BOOKED、NONE、OTHERS 三類**去做 dummies。

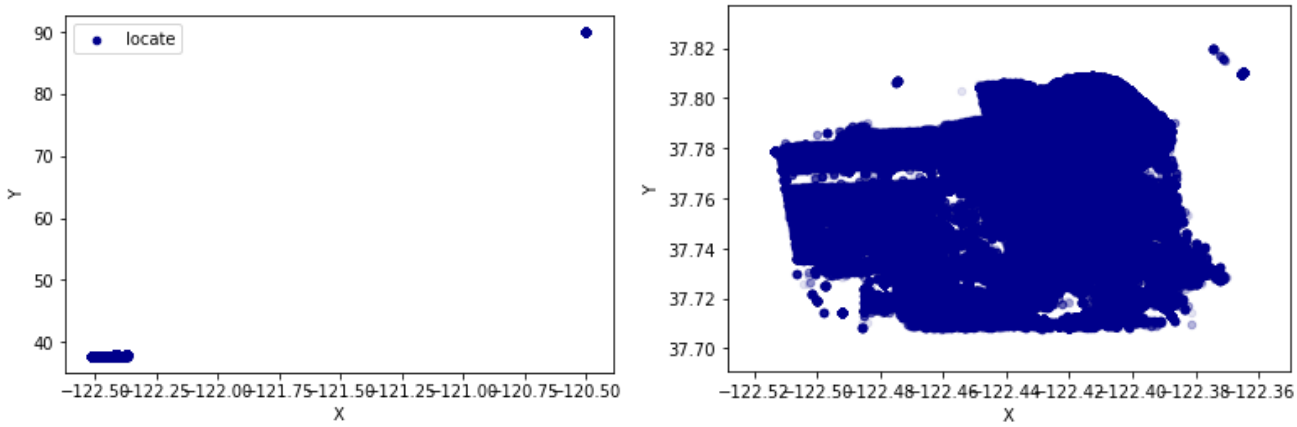
```
# dummies
data['res'] = data['Resolution']
data.loc[(data.Resolution != 'NONE') & (data.Resolution != 'ARREST, BOOKED'), 'res'] = 'other'
print(data.res.unique())
res = pd.get_dummies(data['res'])
print(list(res))

data = pd.concat([data,res],axis=1,ignore_index=False)
```

2-4-11 X&Y 和 Category 的關係

```
data.plot.scatter(x='X',y='Y',color='DarkBlue',label='Locate')
```

可能因為同位於舊金山的關係，所有資料中僅有 67 筆集中在別的經緯度上，其餘皆為在經度-122.25~-122.50、緯度 40 左右，如下左圖。



除去獨特的 67 筆以外，其他的座標(如上右圖)沒有特別明顯的分布，且有地址相關資訊列入 features，因此這邊的經緯度不會特別處理，只會嘗試放入與不放入兩種看準確度是否會提高。

2-5 亂數後拆成訓練集(75%)與測試集(25%)

Features 由下面第 3 點討論結果得出，目前最好的準確度之 features 為：

```
features = ['X', 'Y', 'BAYVIEW', 'CENTRAL', 'INGLESIDE', 'MISSION', 'NORTHERN', 'PARK', 'RICHMOND', 'SOUTHERN',
'TARAVAL', 'TENDERLOIN', 'Friday', 'Monday', 'Saturday', 'Sunday', 'Thursday', 'Tuesday', 'Wednesday', '00', '01',
'02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20',
'21', '22', '23', 'Year','ARREST, BOOKED', 'NONE', 'other','most','min_cat']
```

```
X = data[features]
Y = data['Category']

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=0)
```

3) 使用 DecisionTreeClassifier 進行預測

由於除了基本作業要求的 features 以外，也新建立了很多 features，因此在提高準確度上，嘗試了多次實驗，紀錄如下：(*代表取 dummies)

3-1 區域*、星期*、小時*→ acc: 0.22

3-2 **經緯度**、區域*、星期*、小時*→ acc: 0.2683713

3-3 經緯度、區域*、星期*、小時*、**季節(分成四季)** → acc: 0.264394364

3-4 經緯度、區域*、星期*、小時*、**年分(-6~+6)** → acc: 0.2685763

3-5 **經緯度(取四捨五入)**、區域*、星期*、小時*、年分(-6~+6) → acc: 0.23218

3-6 經緯度、區域*、星期*、小時*、年分(-6~+6)、**Resolution*** → acc: 0.319047

3-7 經緯度、區域*、星期*、小時*、年分(-6~+6)、Resolution*、**上下半年** → acc: 0.3163

3-8 經緯度、區域*、星期*、小時*、年分(-6~+6)、Resolution*、**是否二月** → acc: 0.31829

3-9 經緯度、區域*、星期*、小時*、年分(-6~+6)、Resolution*、是否二月、**是否為最多犯罪的地址** → acc: 0.31839

3-10 經緯度、區域*、星期*、小時*、年分(-6~+6)、Resolution*、是否為最多犯罪的地址 → acc: 0.318901

3-11 經緯度、區域*、星期*、小時*、年分(-6~+6)、Resolution*、是否為最多犯罪的地址、**是否為 0 分或 30 分** → acc: **0.3203136**

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

clf = DecisionTreeClassifier(criterion="entropy", max_depth=5,random_state=0)
clf = clf.fit(X_train,Y_train)
Y_pred = clf.predict(X_test)
```

因為以下第 5 點提到的生成圖的問題，因此在第四點提高準確率上，使用了 max_depth=20，而生成圖的時候使用的深度為 5。

```
clf = DecisionTreeClassifier(criterion="entropy", max_depth=20,random_state=0)
clf = clf.fit(X_train,Y_train)
Y_pred = clf.predict(X_test)
```

4) 計算各犯罪類型的 Precision, Recall, Accuracy

```
print(metrics.accuracy_score(Y_test, Y_pred))
print (metrics.classification_report(Y_test, Y_pred))
confusion_matrix = metrics.confusion_matrix( Y_test, Y_pred, labels=None, sample_weight=None)
```

	precision	recall	f1-score	support
ARSON	0.03	0.02	0.02	375
ASSAULT	0.21	0.22	0.21	19217
BAD CHECKS	0.01	0.01	0.01	103
BRIBERY	0.00	0.00	0.00	63
BURGLARY	0.12	0.08	0.09	9156
DISORDERLY CONDUCT	0.04	0.04	0.04	999
DRIVING UNDER THE INFLUENCE	0.06	0.03	0.04	591
DRUG/NARCOTIC	0.38	0.55	0.45	13610
DRUNKENNESS	0.02	0.01	0.02	1061
EMBEZZLEMENT	0.02	0.01	0.02	290
EXTORTION	0.00	0.00	0.00	68
FAMILY OFFENSES	0.08	0.04	0.06	115
FORGERY/COUNTERFEITING	0.13	0.10	0.12	2645
FRAUD	0.09	0.06	0.07	4089
GAMBLING	0.00	0.00	0.00	41
KIDNAPPING	0.04	0.02	0.03	597
LARCENY/THEFT	0.39	0.66	0.49	43436
LIQUOR LAWS	0.09	0.05	0.07	496
LOITERING	0.33	0.23	0.28	307
MISSING PERSON	0.51	0.50	0.50	6512
NON-CRIMINAL	0.26	0.21	0.24	23096
OTHER OFFENSES	0.39	0.35	0.37	31556
PORNOGRAPHY/OBSCENE MAT	0.00	0.00	0.00	5
PROSTITUTION	0.58	0.66	0.62	1825
RECOVERED VEHICLE	0.09	0.04	0.06	789
ROBBERY	0.09	0.04	0.06	5730
RUNAWAY	0.29	0.19	0.23	497
SECONDARY CODES	0.01	0.00	0.00	2472
SEX OFFENSES FORCIBLE	0.15	0.08	0.10	1072
SEX OFFENSES NON FORCIBLE	0.00	0.00	0.00	38
STOLEN PROPERTY	0.04	0.01	0.02	1138
SUICIDE	0.02	0.01	0.01	118
SUSPICIOUS OCC	0.08	0.04	0.05	8038
TREA	0.00	0.00	0.00	2
TRESPASS	0.06	0.03	0.04	1862
VANDALISM	0.14	0.06	0.08	11265
VEHICLE THEFT	0.34	0.32	0.33	13503
WARRANTS	0.24	0.20	0.22	10648
WEAPON LAWS	0.15	0.05	0.08	2088
accuracy			0.32	219513
macro avg	0.14	0.13	0.13	219513
weighted avg	0.28	0.32	0.29	219513

此為 max_depth=20 的結果

5）產出決策樹的圖

由於電腦本身記憶體限制，嘗試過若樹的深度大於 5 便會不穩定，可能樹太大生成 pdf 時圖被切掉或是根本生不出來而當掉。因此決定最後以準確率最高的 features 組合印出決策樹，但限制深度只在 5：

```
'''
畫出樹
'''
from sklearn import tree
tree.plot_tree(clf.fit(X_train,Y_train))
```

```
#%%
'''
生成graph檔案
'''
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("result", view=True)
```

因為圖檔太大，附圖在 result.pdf 檔案中。