1、 資料描述：Kaggle 上的資料分為 training data (訓練資料)及 test data (測試資料)，首先進行資料預處理，處理各欄位資料型態、填補遺失資料，以及觀察資料間相關係數等數據，接著建立 svm 模型對測試資料進行預測，預測隔天是否會降雨（0/1），準確率為 78.881%。

2、SVM 模型介紹：由於資料是監督式特型，僅有 0/1 兩種預測結果，故使用 SVM 模型處理分類問題。該模型是將資料表示為空間中的點，盡可能進行不同類別間的分類。除了執行線性分類外，SVM 還可以有效地執行非線性分類。SVM 模型中 kernel 有 linear, poly, rbf 等分類器，本次作業中 linear 的表現較佳，故使用該模型。

2.1 資料預處理

Step 1 刪除欄位一的日期、轉換資料型態、填補缺失值：

In [5]:
```
#刪除日期
train_data.drop('Attribute1', axis = 1, inplace = True)
test_data.drop('Attribute1', axis = 1, inplace = True)
```

In [9]:
```
#非數字類別資料轉換
categorical = [var for var in df.columns if df[var].dtype == '0']
print('There are {} categorical variables\n'.format(len(categorical)))
print('The categorical variables are :', categorical)

from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

#將NaN以眾數mode取代
df['Attribute8'].fillna(df['Attribute8'].mode()[0], inplace = True)
df['Attribute10'].fillna(df['Attribute10'].mode()[0], inplace = True)
df['Attribute16'].fillna(df['Attribute16'].mode()[0], inplace = True)
df['Attribute17'].fillna(df['Attribute17'].mode()[0], inplace = True)
tf['Attribute8'].fillna(tf['Attribute8'].mode()[0], inplace = True)
tf['Attribute10'].fillna(tf['Attribute10'].mode()[0], inplace = True)
tf['Attribute16'].fillna(tf['Attribute16'].mode()[0], inplace = True)

#非數字類別以數字替代
df['Attribute8']= label_encoder.fit_transform(df['Attribute8'])
df['Attribute10']= label_encoder.fit_transform(df['Attribute10'])
df['Attribute16']= label_encoder.fit_transform(df['Attribute16'])
df['Attribute17']= label_encoder.fit_transform(df['Attribute17'])
tf['Attribute8']= label_encoder.fit_transform(tf['Attribute8'])
tf['Attribute10']= label_encoder.fit_transform(tf['Attribute10'])
tf['Attribute16']= label_encoder.fit_transform(tf['Attribute16'])
df
```

In [11]:
```
#數字類別
numerical = [var for var in df.columns if df[var].dtype != '0']
print('There are {} numerical variables\n'.format(len(numerical)))
print('The numerical variables are :', numerical)

#將NaN以中位數median取代
df = df.fillna(df.median())
tf = tf.fillna(tf.median())
tf
```

確認資料沒有缺失值：

In [12]:
```
df.isnull().sum()
#tf.isnull().sum()
#df.info()
```

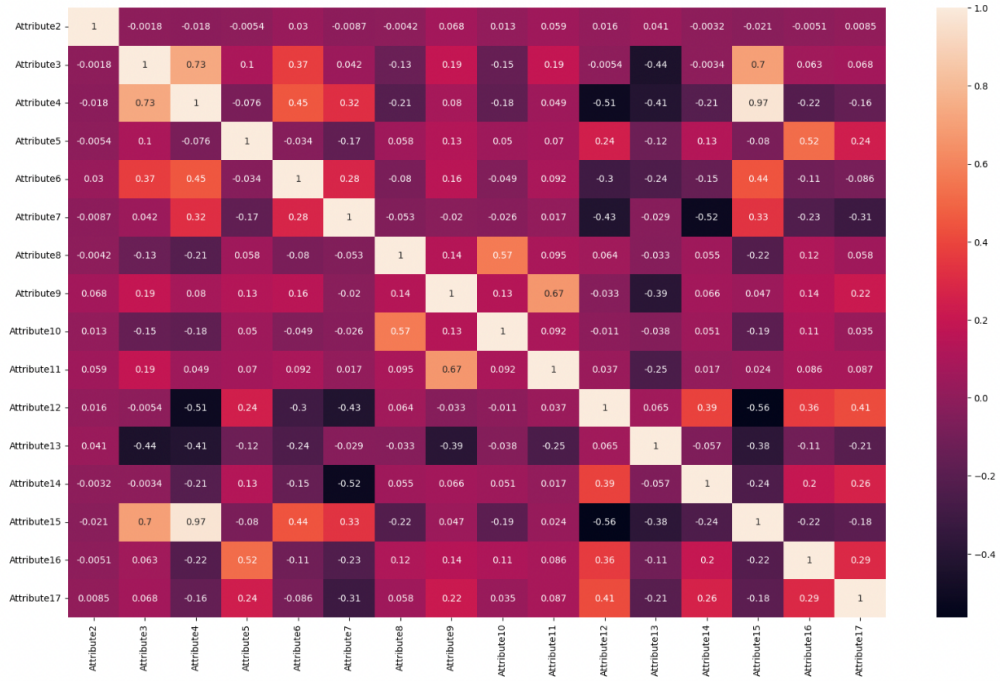
Out[12]:
```
Attribute2     0
Attribute3     0
Attribute4     0
Attribute5     0
Attribute6     0
Attribute7     0
Attribute8     0
Attribute9     0
Attribute10    0
Attribute11    0
Attribute12    0
Attribute13    0
Attribute14    0
Attribute15    0
Attribute16    0
Attribute17    0
dtype: int64
```

Step 2 觀察各欄位資料與明天是否下雨欄位之間相關係數：

In [13]:
```
#增加溫差欄位
df['temp'] = df['Attribute3'] - df['Attribute4']
tf['temp'] = df['Attribute3'] - df['Attribute4']
```

In [13]:
```
cor = df.corr()
plt.figure(figsize = (20,12))
sns.heatmap(cor, annot = True)
```

Out[13]: <AxesSubplot:>



In [60]:
```
#找出相關係數為正的欄位
x = []
for i in range(len(related)):
    if related[i] > 0:
        x.append(related.index[i])
print(x)

#挑選相關係數>0欄位 + 相關係數<-0.10的欄位
x = df[['Attribute12', 'temp', 'Attribute16', 'Attribute14', 'Attribute5',
        'Attribute11', 'Attribute3', 'Attribute8', 'Attribute10', 'Attribut
        'Attribute4', 'Attribute15', 'Attribute13', 'Attribute7']]
xt = tf[['Attribute12', 'temp', 'Attribute16', 'Attribute14', 'Attribute5',
         'Attribute11', 'Attribute3', 'Attribute8', 'Attribute10', 'Attribu
         'Attribute4', 'Attribute15', 'Attribute13', 'Attribute7']]
#x.drop('Attribute17', inplace = True, axis = 1)
xt
```

```
['Attribute17', 'Attribute12', 'temp', 'Attribute16', 'Attribute14', 'Att
ribute5', 'Attribute9', 'Attribute11', 'Attribute3', 'Attribute8', 'Attri
bute10', 'Attribute2']
```

Step 3 平衡資料，0/1 預測結果數量差距太大：

In [16]:
```
#計算數量
y = train_label
y.value_counts()
```

Out[16]:
```
0    13965
1     3138
Name: Attribute17, dtype: int64
```

In [17]:
```
#計算出現機率
y.value_counts()/np.float(len(df))
```

Out[17]:
```
0    0.816523
1    0.183477
Name: Attribute17, dtype: float64
```

In [18]:
```
#平衡資料
from imblearn.over_sampling import SMOTE

bal = SMOTE()
x, y = bal.fit_resample(x, y)
```

In [19]:
```
#已平衡
y.value_counts()
```

Out[19]:
```
0    13965
1    13965
Name: Attribute17, dtype: int64
```

Step 4 Feature Scaling 標準化資料：

In [21]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_sta = scaler.fit_transform(x)
x_sta = pd.DataFrame(x_sta, columns=[x.columns])

xt_sta = scaler.fit_transform(xt)
xt_sta = pd.DataFrame(xt_sta, columns=[xt.columns])
```

In [22]:
```
x_sta.describe()
```

Out[22]:

	Attribute12	Attribute16	Attribute14	Attribute5	Attribute9	Attribute11	Att
count	2.793000e+04	2.793000e+04	2.793000e+04	2.793000e+04	2.793000e+04	2.793000e+04	2.
mean	8.960172e-16	-1.648849e-14	2.569748e-14	-1.111312e-14	1.374188e-15	-2.225614e-15	5.
std	1.000018e+00	1.000018e+00	1.000018e+00	1.000018e+00	1.000018e+00	1.000018e+00	1.
min	-2.651001e+00	-5.753383e-01	-2.597704e+00	-3.427039e-01	-2.470829e+00	-2.169825e+00	-3.
25%	-6.683410e-01	-5.753383e-01	-5.447490e-02	-3.427039e-01	-6.645831e-01	-7.010665e-01	-7.
50%	3.525432e-03	-5.753383e-01	-5.447490e-02	-3.369081e-01	-2.040609e-01	-2.317792e-02	-5.
75%	7.185732e-01	-5.753383e-01	7.957808e-01	-1.393784e-01	5.751405e-01	5.417292e-01	7.
max	2.022412e+00	1.738108e+00	1.471462e+00	1.493491e+01	5.958714e+00	5.964837e+00	3.

2.2 建模

結論：準確度 kernel = linear > rbf > polynomial ;

C = 1, 10, 100 差距不大

Run SVM with rbf kernel and C=1, 10, 100 :

In [24]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# C=1, kernel=rbf, gamma=auto
svc = SVC()
svc.fit(x_sta, y)

test_label = svc.predict(xt_sta)
#test_label
```

In [25]:
```
# C=100
svc = SVC(C = 100.0)
svc.fit(x_sta, y)

test_label_c100 = svc.predict(xt_sta)
#test_label_c100
```

In [26]:
```
# C=10，準確度下降
svc = SVC(C = 10.0)
svc.fit(x_sta, y)

test_label_c10 = svc.predict(xt_sta)
#test_label_c10
```

Run SVM with linear kernel and C=1, 10 :

In [27]:
```
# kernel=linear, C=1
linear_svc = SVC(kernel = 'linear', C = 1.0)

linear_svc.fit(x_sta, y)

test_label_Lc1 = linear_svc.predict(xt_sta)
#test_label_Lc1
```

In [28]:
```
# C=10，輸出資料
linear_svc = SVC(kernel = 'linear', C = 10.0)

linear_svc.fit(x_sta, y)

test_label_Lc10 = linear_svc.predict(xt_sta)
test_label_Lc10
```

Run SVM with polynomial kernel and C=1：

In [29]:
```
#準確度下降
poly_svc = SVC(kernel = 'poly', C = 100.0)

poly_svc.fit(x_sta, y)

test_label_Pc100 = poly_svc.predict(xt_sta)
#test_label_Pc100
```

2.3 輸出資料

In [30]:
```
#整理輸出資料
output = pd.DataFrame(data = test_label_Lc10)
output = output.reset_index()
output['id'] = output['index']
output['ans'] = output[0]
output['id'] = output['id'].astype(float)
del output['index']
del output[0]
output
```

Out[30]:

	id	ans
0	0.0	0
1	1.0	0
2	2.0	0
3	3.0	0
4	4.0	1
...	...	...
801	801.0	0
802	802.0	0
803	803.0	0
804	804.0	0
805	805.0	0

806 rows × 2 columns

In [31]:
```
#輸出資料
import csv
with open('ex_submit.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile) #建立csv檔寫入器
    writer.writerow(['id', 'ans']) #建立標題
    #轉換格式
    for i in range(len(output)):
        writer.writerow([float(output['id'][i]), int(output['ans'][i])])
```