# Лабораторна робота №3 Системи DS.

Класифікація днів за метеорологічними озанками. КМ-83, Лозко Олександр Олексійович.
Розуміння бізнесу - розуміння даних. Підготовка даних для вирішення конкретної задачі.

1.1 Предметна область - метеорологія, а саме - синоптика. Нашою задачею є спрогнозувати, чи буде в Австралії наступного дня дощ, за допомогою класифікації ознак, даних в датасеті.

Для вирішення задачі використовуватимемо такі бібліотеки як pandas, numpy, sklearn, seaborn та інші.

В якості датасету беремо файл з Kaggle: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package Назви ознак інтуїтивно зрозумілі, наприклад:

- Date - The date of observation
- MinTemp - The minimum temperature in degrees celsius
- MaxTemp - The maximum temperature in degrees celsius і тд.

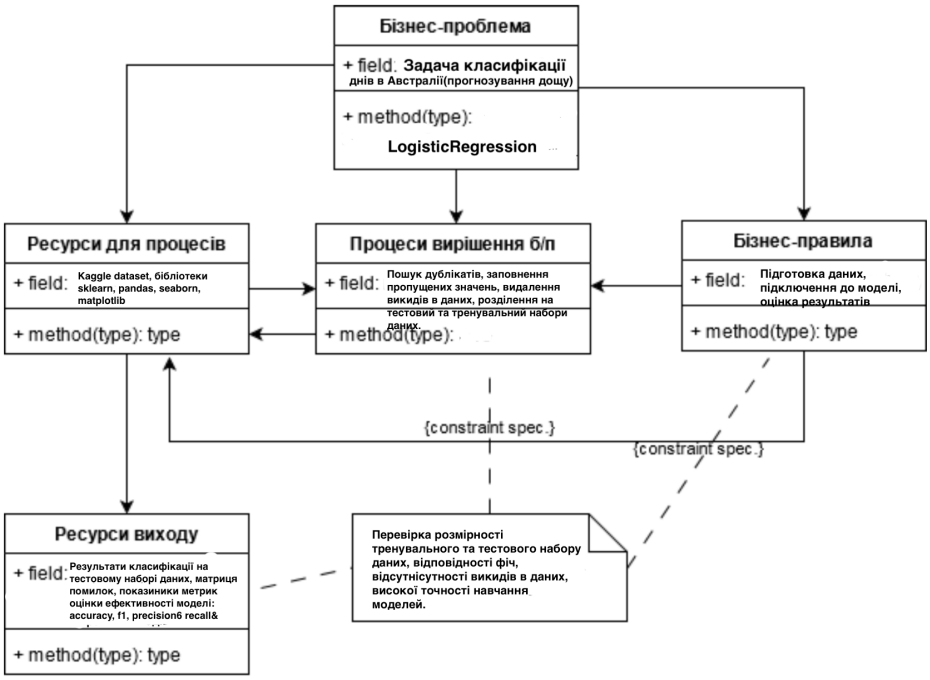
Цей набір даних містить близько 10 років щоденних спостережень за погодою з багатьох місць в Австралії.

RainTomorrow - цільова змінна для прогнозування. Це означає - чи пішов дощ наступного дня, так чи ні? Цей стовпець так, якщо дощ за цей день був 1 мм або більше.

In [2]:

```
Image(filename='diagram.png')
```

Out[2]:



In [1]:

```
import matplotlib.pyplot as plt

import pandas as pd
import pandas_profiling as pp

import seaborn as sns
import numpy as np

from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accur
acy_score

from IPython.display import Image
```

In [15]:

```
df = pd.read_csv('weatherAUS.csv')
df
```

Out[15]:

| | Date | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | Win |
|---|---|---|---|---|---|---|---|---|
| **0** | 2008-12-01 | Albury | 13.4 | 22.9 | 0.6 | NaN | NaN | |
| **1** | 2008-12-02 | Albury | 7.4 | 25.1 | 0.0 | NaN | NaN | |
| **2** | 2008-12-03 | Albury | 12.9 | 25.7 | 0.0 | NaN | NaN | |
| **3** | 2008-12-04 | Albury | 9.2 | 28.0 | 0.0 | NaN | NaN | |
| **4** | 2008-12-05 | Albury | 17.5 | 32.3 | 1.0 | NaN | NaN | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **145455** | 2017-06-21 | Uluru | 2.8 | 23.4 | 0.0 | NaN | NaN | |
| **145456** | 2017-06-22 | Uluru | 3.6 | 25.3 | 0.0 | NaN | NaN | |
| **145457** | 2017-06-23 | Uluru | 5.4 | 26.9 | 0.0 | NaN | NaN | |
| **145458** | 2017-06-24 | Uluru | 7.8 | 27.0 | 0.0 | NaN | NaN | |
| **145459** | 2017-06-25 | Uluru | 14.9 | NaN | 0.0 | NaN | NaN | |

145460 rows × 23 columns

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Date           145460 non-null  object 
 1   Location       145460 non-null  object 
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object 
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object 
 10  WindDir3pm     141232 non-null  object 
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object 
 22  RainTomorrow   142193 non-null  object 
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

### 1.2 Підготовка даних

In [18]:

```
# видаляємо дублікати

df = df.drop_duplicates()
```

In [19]:

```
# застосування описової статистики

df.describe()
```

Out[19]:

| | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | V |
|---|---|---|---|---|---|---|
| **count** | 143975.000000 | 144199.000000 | 142199.000000 | 82670.000000 | 75625.000000 | |
| **mean** | 12.194034 | 23.221348 | 2.360918 | 5.468232 | 7.611178 | |
| **std** | 6.398495 | 7.119049 | 8.478060 | 4.193704 | 3.785483 | |
| **min** | -8.500000 | -4.800000 | 0.000000 | 0.000000 | 0.000000 | |
| **25%** | 7.600000 | 17.900000 | 0.000000 | 2.600000 | 4.800000 | |
| **50%** | 12.000000 | 22.600000 | 0.000000 | 4.800000 | 8.400000 | |

| | | | | | |
|---|---|---|---|---|---|
| **75%** | 16.900000 | 28.200000 | 0.800000 | 7.400000 | 10.600000 |
| **max** | 33.900000 | 48.100000 | 371.000000 | 145.000000 | 14.500000 |

In [20]:

```
# проведення кореляційного та причинно-наслідкового аналізів

df.corr()
```

Out[20]:

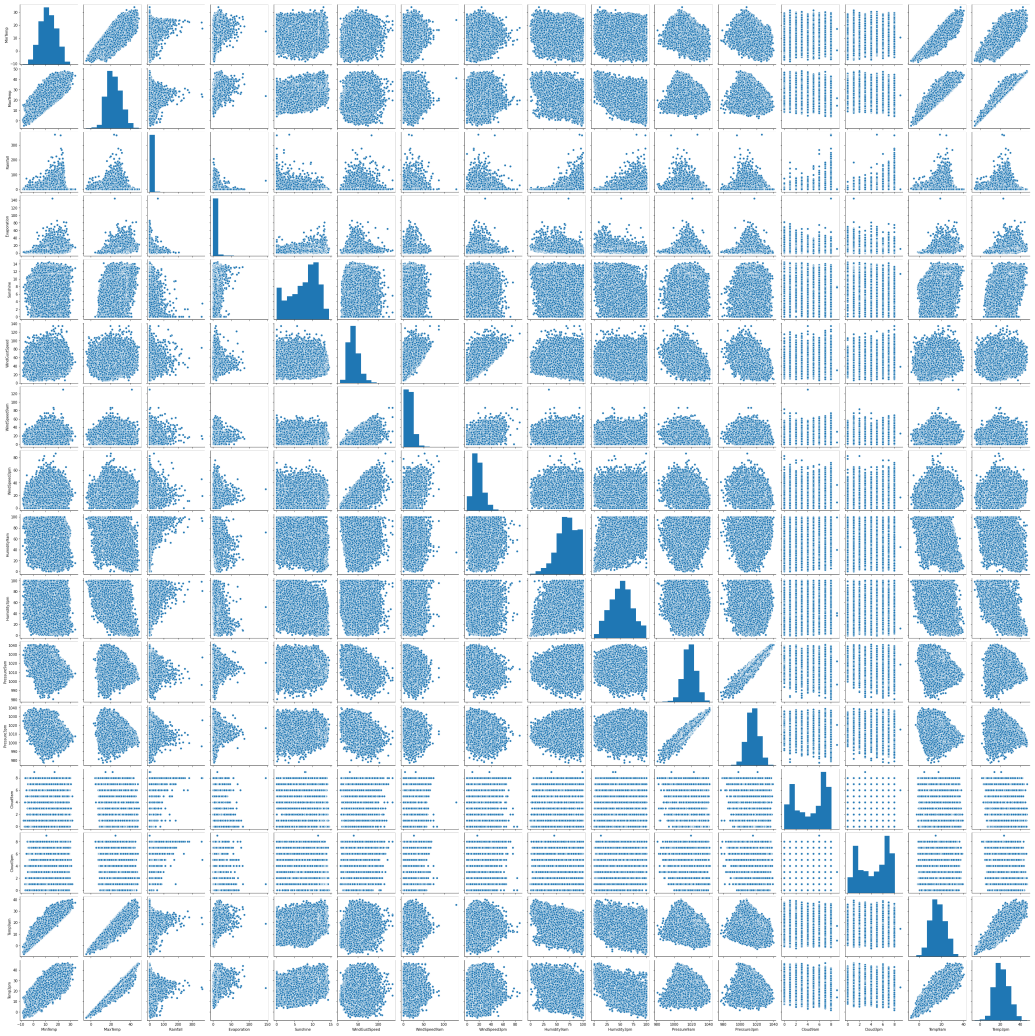
| | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGust |
|---|---|---|---|---|---|---|
| **MinTemp** | 1.000000 | 0.736555 | 0.103938 | 0.466993 | 0.072586 | 0.1 |
| **MaxTemp** | 0.736555 | 1.000000 | -0.074992 | 0.587932 | 0.470156 | 0.0 |
| **Rainfall** | 0.103938 | -0.074992 | 1.000000 | -0.064351 | -0.227549 | 0.1 |
| **Evaporation** | 0.466993 | 0.587932 | -0.064351 | 1.000000 | 0.365602 | 0.2 |
| **Sunshine** | 0.072586 | 0.470156 | -0.227549 | 0.365602 | 1.000000 | -0.0 |
| **WindGustSpeed** | 0.177415 | 0.067615 | 0.133659 | 0.203021 | -0.034750 | 1.0 |
| **WindSpeed9am** | 0.175064 | 0.014450 | 0.087338 | 0.193084 | 0.005499 | 0.6 |
| **WindSpeed3pm** | 0.175173 | 0.050300 | 0.057887 | 0.129400 | 0.053834 | 0.6 |
| **Humidity9am** | -0.232899 | -0.504110 | 0.224405 | -0.504092 | -0.490819 | -0.2 |
| **Humidity3pm** | 0.006089 | -0.508855 | 0.255755 | -0.390243 | -0.629130 | -0.0 |
| **Pressure9am** | -0.450970 | -0.332061 | -0.168154 | -0.270362 | 0.041970 | -0.4 |
| **Pressure3pm** | -0.461292 | -0.427167 | -0.126534 | -0.293581 | -0.019719 | -0.4 |
| **Cloud9am** | 0.078754 | -0.289370 | 0.198528 | -0.183793 | -0.675323 | 0.0 |
| **Cloud3pm** | 0.021605 | -0.277921 | 0.172403 | -0.182618 | -0.703930 | 0.1 |
| **Temp9am** | 0.901821 | 0.887210 | 0.011192 | 0.545115 | 0.291188 | 0.1 |
| **Temp3pm** | 0.708906 | 0.984503 | -0.079657 | 0.572893 | 0.490501 | 0.0 |

In [21]:

```
sns.pairplot(df)
```

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x7f8f62f04e80>
```



In [22]:

```
# конструювання  ознак

df['Date']= pd.to_datetime(df['Date'])
df['year'] = df.Date.dt.year

def encode(df, col, max_val):
    df[col + '_sin'] = np.sin(2 * np.pi * df[col]/max_val)
    df[col + '_cos'] = np.cos(2 * np.pi * df[col]/max_val)
    return df

df['month'] = df.Date.dt.month
df = encode(df, 'month', 12)

df['day'] = df.Date.dt.day
df = encode(df, 'day', 31)

# кодування категоріальних (categorical) характеристик

s = (df.dtypes == "object")
```

```
object_cols = list(s[s].index)

for i in object_cols:
    df[i].fillna(df[i].mode()[0], inplace=True)

label_encoder = LabelEncoder()
for i in object_cols:
    df[i] = label_encoder.fit_transform(df[i])

# заповнюємо пропущені значення середнім колонки

t = (df.dtypes == 'float64')
num_cols = list(t[t].index)

for i in num_cols:
    df[i].fillna(df[i].median(), inplace=True)
```

In [23]:

```
features = df.drop(['RainTomorrow', 'Date','day', 'month'], axis=1)

target = df['RainTomorrow']

col_names = list(features.columns)
s_scaler = preprocessing.StandardScaler()
features = s_scaler.fit_transform(features)
features = pd.DataFrame(features, columns=col_names)

features.describe().T
```

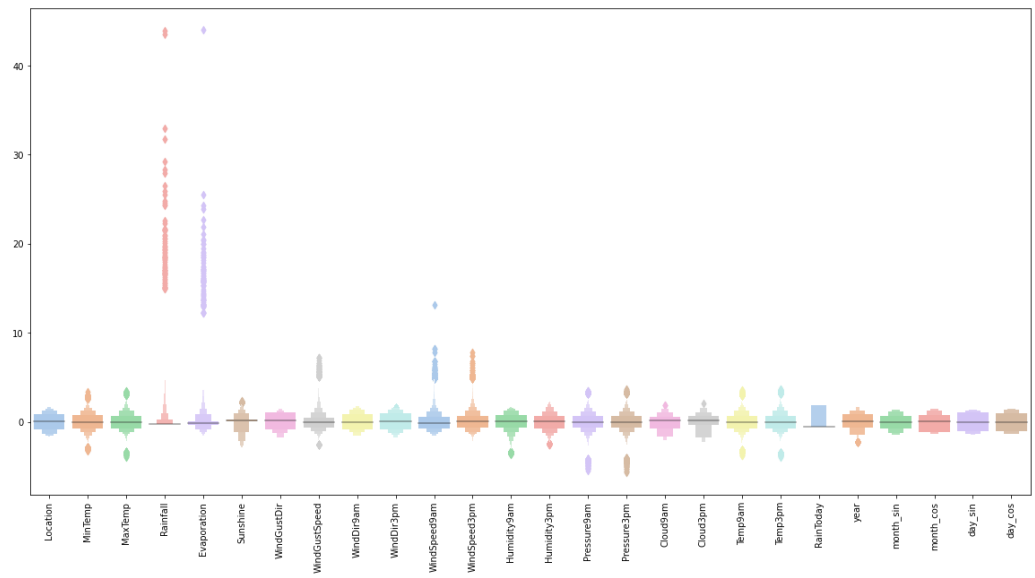
Out[23]:

| | count | mean | std | min | 25% | 50% | |
|---|---|---|---|---|---|---|---|
| **Location** | 145460.0 | 7.815677e-18 | 1.000003 | -1.672228 | -0.899139 | 0.014511 | |
| **MinTemp** | 145460.0 | -4.501830e-16 | 1.000003 | -3.250525 | -0.705659 | -0.030170 | |
| **MaxTemp** | 145460.0 | 3.001220e-16 | 1.000003 | -3.952405 | -0.735852 | -0.086898 | |
| **Rainfall** | 145460.0 | 7.815677e-18 | 1.000003 | -0.275097 | -0.275097 | -0.275097 | -( |
| **Evaporation** | 145460.0 | -3.282584e-17 | 1.000003 | -1.629472 | -0.371139 | -0.119472 | |
| **Sunshine** | 145460.0 | -5.424080e-16 | 1.000003 | -2.897217 | 0.076188 | 0.148710 | |
| **WindGustDir** | 145460.0 | 6.252542e-18 | 1.000003 | -1.724209 | -0.872075 | 0.193094 | |
| **WindGustSpeed** | 145460.0 | 1.824961e-16 | 1.000003 | -2.588407 | -0.683048 | -0.073333 | |
| **WindDir9am** | 145460.0 | 7.190423e-17 | 1.000003 | -1.550000 | -0.885669 | 0.000105 | |
| **WindDir3pm** | 145460.0 | 8.284618e-17 | 1.000003 | -1.718521 | -0.837098 | 0.044324 | |
| **WindSpeed9am** | 145460.0 | 5.627287e-17 | 1.000003 | -1.583291 | -0.793380 | -0.116314 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **WindSpeed3pm** | 145460.0 | 6.565169e-17 | 1.000003 | -2.141841 | -0.650449 | 0.037886 | |
| **Humidity9am** | 145460.0 | 2.250915e-16 | 1.000003 | -3.654212 | -0.631189 | 0.058273 | |
| **Humidity3pm** | 145460.0 | -8.440931e-17 | 1.000003 | -2.518329 | -0.710918 | 0.021816 | |
| **Pressure9am** | 145460.0 | -4.314254e-16 | 1.000003 | -5.520544 | -0.616005 | -0.006653 | |
| **Pressure3pm** | 145460.0 | 5.027043e-15 | 1.000003 | -5.724832 | -0.622769 | -0.007520 | |
| **Cloud9am** | 145460.0 | -1.016038e-16 | 1.000003 | -2.042425 | -0.727490 | 0.149133 | |
| **Cloud3pm** | 145460.0 | 7.346736e-17 | 1.000003 | -2.235619 | -0.336969 | 0.137693 | |
| **Temp9am** | 145460.0 | 7.503050e-17 | 1.000003 | -3.750358 | -0.726764 | -0.044517 | |
| **Temp3pm** | 145460.0 | -6.877796e-17 | 1.000003 | -3.951301 | -0.725322 | -0.083046 | |
| **RainToday** | 145460.0 | -8.988029e-18 | 1.000003 | -0.529795 | -0.529795 | -0.529795 | |
| **year** | 145460.0 | 2.080221e-14 | 1.000003 | -2.273637 | -0.697391 | 0.090732 | |
| **month_sin** | 145460.0 | 1.270048e-17 | 1.000003 | -1.434333 | -0.725379 | -0.016425 | |
| **month_cos** | 145460.0 | -1.602214e-17 | 1.000003 | -1.388032 | -1.198979 | 0.023080 | |
| **day_sin** | 145460.0 | -2.521777e-18 | 1.000003 | -1.403140 | -1.019170 | -0.003198 | |
| **day_cos** | 145460.0 | -1.176015e-17 | 1.000003 | -1.392587 | -1.055520 | -0.044639 | |

In [24]:

```
plt.figure(figsize=(20,10))
sns.boxenplot(data = features,palette = 'pastel')
plt.xticks(rotation=90)
plt.show()
```



In [25]:

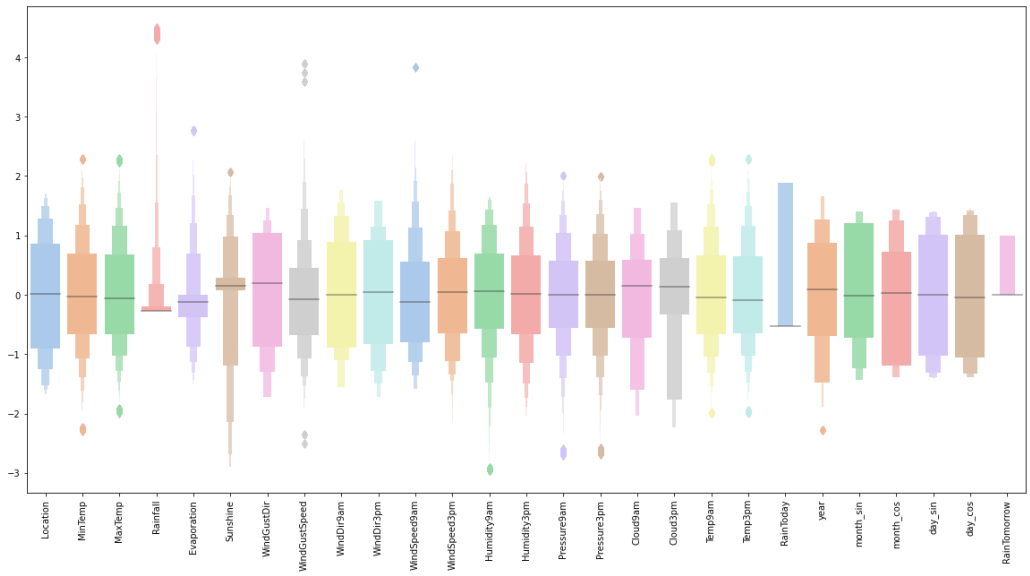
```
features['RainTomorrow'] = target

features = features[(features["MinTemp"]<2.3)&(features["MinTemp"]>-2.3)]
features = features[(features["MaxTemp"]<2.3)&(features["MaxTemp"]>-2)]
features = features[(features["Rainfall"]<4.5)]
features = features[(features["Evaporation"]<2.8)]
features = features[(features["Sunshine"]<2.1)]
features = features[(features["WindGustSpeed"]<4)&(features["WindGustSpeed"]>-4)]
features = features[(features["WindSpeed9am"]<4)]
features = features[(features["WindSpeed3pm"]<2.5)]
features = features[(features["Humidity9am"]>-3)]
features = features[(features["Humidity3pm"]>-2.2)]
features = features[(features["Pressure9am"]< 2)&(features["Pressure9am"]>-2.7)]
features = features[(features["Pressure3pm"]< 2)&(features["Pressure3pm"]>-2.7)]
features = features[(features["Cloud9am"]<1.8)]
features = features[(features["Cloud3pm"]<2)]
features = features[(features["Temp9am"]<2.3)&(features["Temp9am"]>-2)]
features = features[(features["Temp3pm"]<2.3)&(features["Temp3pm"]>-2)]

features.shape
```

Out[25]:

```
(127536, 27)
```

In [26]:

```
plt.figure(figsize = (20,10))
sns.boxenplot(data = features, palette = 'pastel')
plt.xticks(rotation = 90)
plt.show()
```




Бачимо, що після підготовки ознаки мають набагато менше викидів.

In [27]:

```
# розбиття набору даних на тренувальний та тестовий набори дани
х

X = features.drop(['RainTomorrow'], axis=1)
y = features["RainTomorrow"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 42)
```

In [28]:

```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 102028 entries, 86273 to 139021
Data columns (total 26 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Location       102028 non-null  float64
 1   MinTemp        102028 non-null  float64
 2   MaxTemp        102028 non-null  float64
 3   Rainfall       102028 non-null  float64
 4   Evaporation    102028 non-null  float64
 5   Sunshine       102028 non-null  float64
 6   WindGustDir    102028 non-null  float64
 7   WindGustSpeed  102028 non-null  float64
 8   WindDir9am     102028 non-null  float64
 9   WindDir3pm     102028 non-null  float64
 10  WindSpeed9am   102028 non-null  float64
 11  WindSpeed3pm   102028 non-null  float64
 12  Humidity9am    102028 non-null  float64
 13  Humidity3pm    102028 non-null  float64
 14  Pressure9am    102028 non-null  float64
 15  Pressure3pm    102028 non-null  float64
 16  Cloud9am       102028 non-null  float64
 17  Cloud3pm       102028 non-null  float64
 18  Temp9am        102028 non-null  float64
 19  Temp3pm        102028 non-null  float64
 20  RainToday      102028 non-null  float64
 21  year           102028 non-null  float64
 22  month_sin      102028 non-null  float64
 23  month_cos      102028 non-null  float64
 24  day_sin        102028 non-null  float64
 25  day_cos        102028 non-null  float64
dtypes: float64(26)
memory usage: 21.0 MB
```

In [29]:

```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25508 entries, 29216 to 11570
Data columns (total 26 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Location       25508 non-null  float64
 1   MinTemp        25508 non-null  float64
 2   MaxTemp        25508 non-null  float64
 3   Rainfall       25508 non-null  float64
```

```
 4   Evaporation    25508 non-null  float64
 5   Sunshine       25508 non-null  float64
 6   WindGustDir    25508 non-null  float64
 7   WindGustSpeed  25508 non-null  float64
 8   WindDir9am     25508 non-null  float64
 9   WindDir3pm     25508 non-null  float64
 10  WindSpeed9am   25508 non-null  float64
 11  WindSpeed3pm   25508 non-null  float64
 12  Humidity9am    25508 non-null  float64
 13  Humidity3pm    25508 non-null  float64
 14  Pressure9am    25508 non-null  float64
 15  Pressure3pm    25508 non-null  float64
 16  Cloud9am       25508 non-null  float64
 17  Cloud3pm       25508 non-null  float64
 18  Temp9am        25508 non-null  float64
 19  Temp3pm        25508 non-null  float64
 20  RainToday      25508 non-null  float64
 21  year           25508 non-null  float64
 22  month_sin      25508 non-null  float64
 23  month_cos      25508 non-null  float64
 24  day_sin        25508 non-null  float64
 25  day_cos        25508 non-null  float64
dtypes: float64(26)
memory usage: 5.3 MB
```

1.3 Верифікація і валідація

In [30]:

```
np.shape(X_train)[0] == np.shape(y_train)[0]
```

Out[30]:

```
True
```

In [31]:

```
np.shape(X_test)[0] == np.shape(y_test)[0]
```

Out[31]:

```
True
```

In [32]:

```
np.shape(X_train)[1] == np.shape(X_test)[1]
```

Out[32]:

```
True
```

1.Перелік існуючих потенційних методів розв’язання поставленої задачі, короткий опис обраного методу моделювання (статистика, ML), його математичного забезпечення.

Для вирішення задачі будемо використовувати Логістичну регресію. Це статистична модель, використовувана для прогнозування ймовірності виникнення деякої події шляхом його порівняння з логістичної кривої. Ця регрессія видає відповідь у вигляді ймовірності бінарного події (1 або 0)

In [34]:

```
Image(filename='LR.jpg')
```

Out[34]:


2.Формалізація цільової функції оптимізації(функція втрат —loss function) в залежності від типу задачі моделювання. Визначення метрик оцінки ефективності.

Для з'ясування ефективності роботи моделі будемо враховувати:

- матрицю помилок
- accuracy(У багаторабельній класифікації ця функція обчислює точність підмножини: набір лейблів, передбачених для зразка, повинен точно відповідати відповідному набору лейблів у y_true.)
- precision(також називається позитивна прогностична цінність - це частка відповідних інстанцій серед витків екземплярів)
- recall(також називається чутливість - це частка відповідних випадків, які були отримані)
- f1(у статистичному аналізі бінарної класифікації F-оцінка або F-міра є кількістю точних тестів)

In [35]:

```
Image(filename='form.jpg')
```

Out[35]:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

1. Імплементація обраного методу моделювання

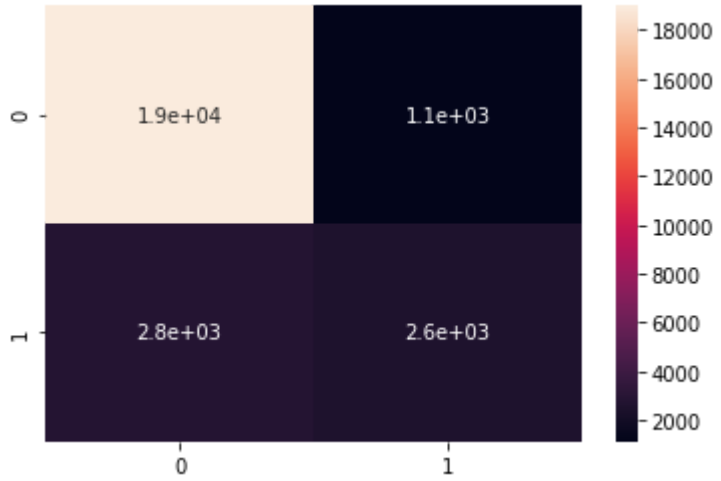
In [38]:

```
model = LogisticRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred), annot=True)
```

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.91     20110
           1       0.70      0.47      0.56      5398

    accuracy                           0.84     25508
   macro avg       0.78      0.71      0.73     25508
weighted avg       0.83      0.84      0.83     25508
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f94d7d9d0>
```




Модель може передбачити дощ з 84% точності, з таблиці бачимо також високі результати за іншими показниками. Згідно до передбачень моделі дощ можна очікувати у 5398 випадках.

1. Верифікація і валідація. При переданні даних до класифікатора, ми не отримали помилок чи попереджень, що свідчить про проходження даними валідації. Оскільки класифікатор дав високу точність за всіми показниками: accuracy, recall, precision, f1, можемо зробити висновок, що дані проходять і верифікацію також. Зазначимо, що це свідчить про коректну підготовку вхідного датасету в першій лабораторній

1. Висновки.

В ході виконання роботи ми провели очищення та передобробку даних для навчання на них обраних типових моделей. Слід зауважити, що дані були досить "хороші", тобто в них було відносно мало викидів та не було пропущених значень. Щоб порівняти ефективність роботи моделей, ми використовували 5 характерістик: матрицю помилок, accuracy, precision, recall, f1. Логістична регресія показала досить високу точність(84%) як і інші показники, що свідчить про високу якість вхідного датасету та правильне налаштування моделі.

In [ ]: