In [1]:

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

In [2]:

```
data = pd.read_csv('files/weather.csv', parse_dates=True, index_col=0)
data.head()
```

Out[2]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humi
Date												
2008-02-01	19.5	22.4	15.6	6.2	0.0	NaN	NaN	S	SSW	17.0	...	
2008-02-02	19.5	25.6	6.0	3.4	2.7	NaN	NaN	W	E	9.0	...	
2008-02-03	21.6	24.5	6.6	2.4	0.1	NaN	NaN	ESE	ESE	17.0	...	
2008-02-04	20.2	22.8	18.8	2.2	0.0	NaN	NaN	NNE	E	22.0	...	
2008-02-05	19.7	25.7	77.4	NaN	0.0	NaN	NaN	NNE	W	11.0	...	

5 rows × 22 columns

In [3]:

```
data.dtypes
```

Out[3]:

```
MinTemp          float64
MaxTemp          float64
Rainfall         float64
Evaporation      float64
Sunshine         float64
WindGustDir       object
WindGustSpeed    float64
WindDir9am        object
WindDir3pm        object
WindSpeed9am     float64
WindSpeed3pm     float64
Humidity9am      float64
Humidity3pm      float64
Pressure9am      float64
Pressure3pm      float64
Cloud9am         float64
Cloud3pm         float64
Temp9am          float64
Temp3pm          float64
RainToday         object
RISK_MM          float64
RainTomorrow      object
dtype: object
```

In [7]:

```
#Choose 3 columns to create datasets
header = ['Humidity3pm', 'Pressure3pm', 'Cloud3pm', 'RainTomorrow']
dataset = data[header]
```

In [5]:

```
dataset.head()
```

Out[5]:

	Humidity3pm	Pressure3pm	Cloud3pm	RainTomorrow
Date				
2008-02-01	84.0	1017.4	8.0	Yes
2008-02-02	73.0	1016.4	7.0	Yes
2008-02-03	86.0	1015.6	8.0	Yes
2008-02-04	90.0	1011.8	8.0	Yes
2008-02-05	74.0	1004.8	8.0	Yes

In [16]:

```
#Deal with remaining missing data
dataset_clean = data.dropna()
```

```
len(dataset), len(dataset_clean)
```

Out[16]:

```
(3337, 1690)
```

In [17]:

```
#Create training and test datasets
X = dataset_clean[header[:3]]
y = dataset_clean[header[3]]
y = np.array([0 if value == 'No' else 1 for value in y])
```

In [18]:

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

In [19]:

```
#Train and test the model
model = KNeighborsClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
metrics.accuracy_score(y_test, y_pred)
```

Out[19]:

```
0.8108747044917257
```

In [ ]:

In [ ]: