Weather Prediction<\h1>

In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
# Importing the dataset
data = pd.read_csv('../input/weather-dataset/weatherHistory.csv')
```

In [4]:

```
data.head()
```

Out[4]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00	Mostly	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy through

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
	.000 +0200	Cloudy										out the day.
3	2006-04-01 03:00:00 .000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00 .000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

In [5]:

```
data.shape
```

Out[5]:

Preparing Data for training the model

In [6]:

```
data['Loud Cover'].value_counts()
```

Out[6]:

In [7]:

```
# Removing variable which wont have much impact on the prediction. value of loud co
ver is same for all rows, so it wont affect the prediction.
data.drop(['Formatted Date','Daily Summary','Loud Cover','Wind Bearing (degrees)'],
axis=1,inplace=True)
```

In [8]:

```
data
```

Out[8]:

	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Visibility (km)	Pressure (millibars)
0	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	15.8263	1015.13
1	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	15.8263	1015.63
2	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	14.9569	1015.94
3	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	15.8263	1016.41
4	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	15.8263	1016.51
...	...	...	...	...	...	...	...	...
96448	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	16.1000	1014.36

	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Visibility (km)	Pressure (millibars)
96449	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	15.5526	1015.16
96450	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	16.1000	1015.66
96451	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	16.1000	1015.95
96452	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	15.5204	1016.16

96453 rows × 8 columns

In [9]:

```
data.shape
```

Out[9]:

In [10]:

```
data.isnull().sum()
```

Out[10]:

In [11]:

```
data['Precip Type'].value_counts()
```

Out[11]:

In [12]:

```
data['Precip Type'].fillna(method='ffill',inplace=True,axis=0)
```

In [13]:

```
data['Precip Type'].value_counts()
```

Out[13]:

In [14]:

```
# Converting categorical data into numerical data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Precip Type']=le.fit_transform(data['Precip Type'])
data['Summary']=le.fit_transform(data['Summary'])
```

In [15]:

```
data
```

Out[15]:

	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Visibility (km)	Pressure (millibars)
0	19	0	9.472222	7.388889	0.89	14.1197	15.8263	1015.13

	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Visibility (km)	Pressure (millibars)
1	19	0	9.355556	7.227778	0.86	14.2646	15.8263	1015.63
2	17	0	9.377778	9.377778	0.89	3.9284	14.9569	1015.94
3	19	0	8.288889	5.944444	0.83	14.1036	15.8263	1016.41
4	17	0	8.755556	6.977778	0.83	11.0446	15.8263	1016.51
...	...	...	...	...	...	...	...	...
96448	19	0	26.016667	26.016667	0.43	10.9963	16.1000	1014.36
96449	19	0	24.583333	24.583333	0.48	10.0947	15.5526	1015.16
96450	19	0	22.038889	22.038889	0.56	8.9838	16.1000	1015.66
96451	19	0	21.522222	21.522222	0.60	10.5294	16.1000	1015.95
96452	19	0	20.438889	20.438889	0.61	5.8765	15.5204	1016.16

96453 rows × 8 columns

In [16]:

```
y=data.iloc[:,0] # Dependent Variable
```

In [17]:

```
x = data.iloc[:,1:]    #Independent variable
```

In [18]:

```
x.corr()    # checking correlation to drop unnecessary variable
```

Out[18]:

	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Visibility (km)	Pressure (millibars)
Precip Type	1.000000	-0.562620	-0.565207	0.232113	-0.067928	-0.312875	0.009008
Temperature (C)	-0.562620	1.000000	0.992629	-0.632255	0.008957	0.392847	-0.005447
Apparent Temperature (C)	-0.565207	0.992629	1.000000	-0.602571	-0.056650	0.381718	-0.000219
Humidity	0.232113	-0.632255	-0.602571	1.000000	-0.224951	-0.369173	0.005454
Wind Speed (km/h)	-0.067928	0.008957	-0.056650	-0.224951	1.000000	0.100749	-0.049263
Visibility (km)	-0.312875	0.392847	0.381718	-0.369173	0.100749	1.000000	0.059818
Pressure (millibars)	0.009008	-0.005447	-0.000219	0.005454	-0.049263	0.059818	1.000000

In [19]:

```
# Apparent Temperature and Temperax.drop('Apparent Temperature (C)',axis=1,inplace=
True)ture are highly correlated (correlation almost equal to 1). So, we can drop on
e of them.
x.drop('Apparent Temperature (C)',axis=1,inplace=True)
```

In [20]:

```
x
```

Out[20]:

	Precip Type	Temperature (C)	Humidity	Wind Speed (km/h)	Visibility (km)	Pressure (millibars)
0	0	9.472222	0.89	14.1197	15.8263	1015.13
1	0	9.355556	0.86	14.2646	15.8263	1015.63
2	0	9.377778	0.89	3.9284	14.9569	1015.94
3	0	8.288889	0.83	14.1036	15.8263	1016.41
4	0	8.755556	0.83	11.0446	15.8263	1016.51
...	...	...	...	...	...	...
96448	0	26.016667	0.43	10.9963	16.1000	1014.36
96449	0	24.583333	0.48	10.0947	15.5526	1015.16
96450	0	22.038889	0.56	8.9838	16.1000	1015.66

	Precip Type	Temperature (C)	Humidity	Wind Speed (km/h)	Visibility (km)	Pressure (millibars)
96451	0	21.522222	0.60	10.5294	16.1000	1015.95
96452	0	20.438889	0.61	5.8765	15.5204	1016.16

96453 rows × 6 columns

In [21]:

```
x.shape
```

Out[21]:

In [22]:

```
# Splitting the dataset into train data and test data
# Train dataset is 70% of and Test dataset is 30% of original dataset

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=1)
```

In [23]:

```
x_train.shape
```

Out[23]:

In [24]:

```
x_test.shape
```

Out[24]:

Training the Model

In [25]:

```
# Using Random Forest Classifier algorithm to predict the weather
# Training model on test data

from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(max_depth=32,n_estimators=120,random_state=1)
RF.fit(x_train,y_train)
y_pred = RF.predict(x_test)
```

Measuring Accuracy

In [26]:

```
# Finding accuracy of model using test data
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[26]:

In [27]:

```
# To see importance of each variable in prediction
RF.feature_importances_
```

Out[27]:

In [ ]:

```

```