```
from google.colab import files
upload = files.upload()
```

Choose Files weather.csv

- **weather.csv**(text/csv) - 29462 bytes, last modified: 25/1/2024 - 100% done

```
Saving weather.csv to weather.csv
```

```
import pandas as pd
import seaborn as sns
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt

import io
```

```
data = pd.read_csv(io.BytesIO(upload['weather.csv']))
data.head()
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDi
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	V
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	

5 rows × 22 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MinTemp        366 non-null    float64
 1   MaxTemp        366 non-null    float64
 2   Rainfall       366 non-null    float64
 3   Evaporation    366 non-null    float64
 4   Sunshine       363 non-null    float64
 5   WindGustDir    363 non-null    object 
 6   WindGustSpeed  364 non-null    float64
 7   WindDir9am     335 non-null    object 
 8   WindDir3pm     365 non-null    object 
 9   WindSpeed9am   359 non-null    float64
 10  WindSpeed3pm   366 non-null    int64  
 11  Humidity9am    366 non-null    int64  
 12  Humidity3pm    366 non-null    int64  
 13  Pressure9am    366 non-null    float64
 14  Pressure3pm    366 non-null    float64
 15  Cloud9am       366 non-null    int64  
 16  Cloud3pm       366 non-null    int64  
 17  Temp9am        366 non-null    float64
 18  Temp3pm        366 non-null    float64
 19  RainToday      366 non-null    object 
 20  RISK_MM        366 non-null    float64
 21  RainTomorrow   366 non-null    object 
dtypes: float64(12), int64(5), object(5)
memory usage: 63.0+ KB
```

```
data.isnull().sum()   #checking for null values or missing values
```

```
MinTemp           0
MaxTemp           0
Rainfall          0
Evaporation       0
Sunshine          3
WindGustDir       3
WindGustSpeed     2
WindDir9am       31
WindDir3pm        1
WindSpeed9am      7
WindSpeed3pm      0
Humidity9am       0
```

```
     Humidity3pm       0
     Pressure9am       0
     Pressure3pm       0
     Cloud9am          0
     Cloud3pm          0
     Temp9am           0
     Temp3pm           0
     RainToday         0
     RISK_MM           0
     RainTomorrow      0
     dtype: int64
```

```
data.duplicated().sum() #checking for duplicated entries
```

```
     0
```

```
#Clearing Missing Values for columns with string Datatype

mode_WindGustDir = data['WindGustDir'].mode()[0]
data['WindGustDir'].fillna(mode_WindGustDir, inplace=True)

mode_WindDir9am = data['WindDir9am'].mode()[0]
data['WindDir9am'].fillna(mode_WindDir9am, inplace=True)

mode_WindDir3pm = data['WindDir3pm'].mode()[0]
data['WindDir3pm'].fillna(mode_WindDir3pm, inplace=True)

# Clearing missing values in other columns

to_be_cleaned_columns = ['Sunshine', 'WindGustSpeed', 'WindSpeed9am']
imputer = SimpleImputer(strategy='mean')
data[to_be_cleaned_columns] = imputer.fit_transform(data[to_be_cleaned_columns])
```

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

```
     MinTemp          0
     MaxTemp          0
     Rainfall         0
     Evaporation      0
     Sunshine         0
     WindGustDir      0
     WindGustSpeed    0
     WindDir9am       0
     WindDir3pm       0
     WindSpeed9am     0
     WindSpeed3pm     0
     Humidity9am      0
     Humidity3pm      0
     Pressure9am      0
     Pressure3pm      0
     Cloud9am         0
     Cloud3pm         0
     Temp9am          0
     Temp3pm          0
     RainToday        0
     RISK_MM          0
     RainTomorrow     0
     dtype: int64
```

```
data.to_csv('cleaned_data.csv', index=False)
```

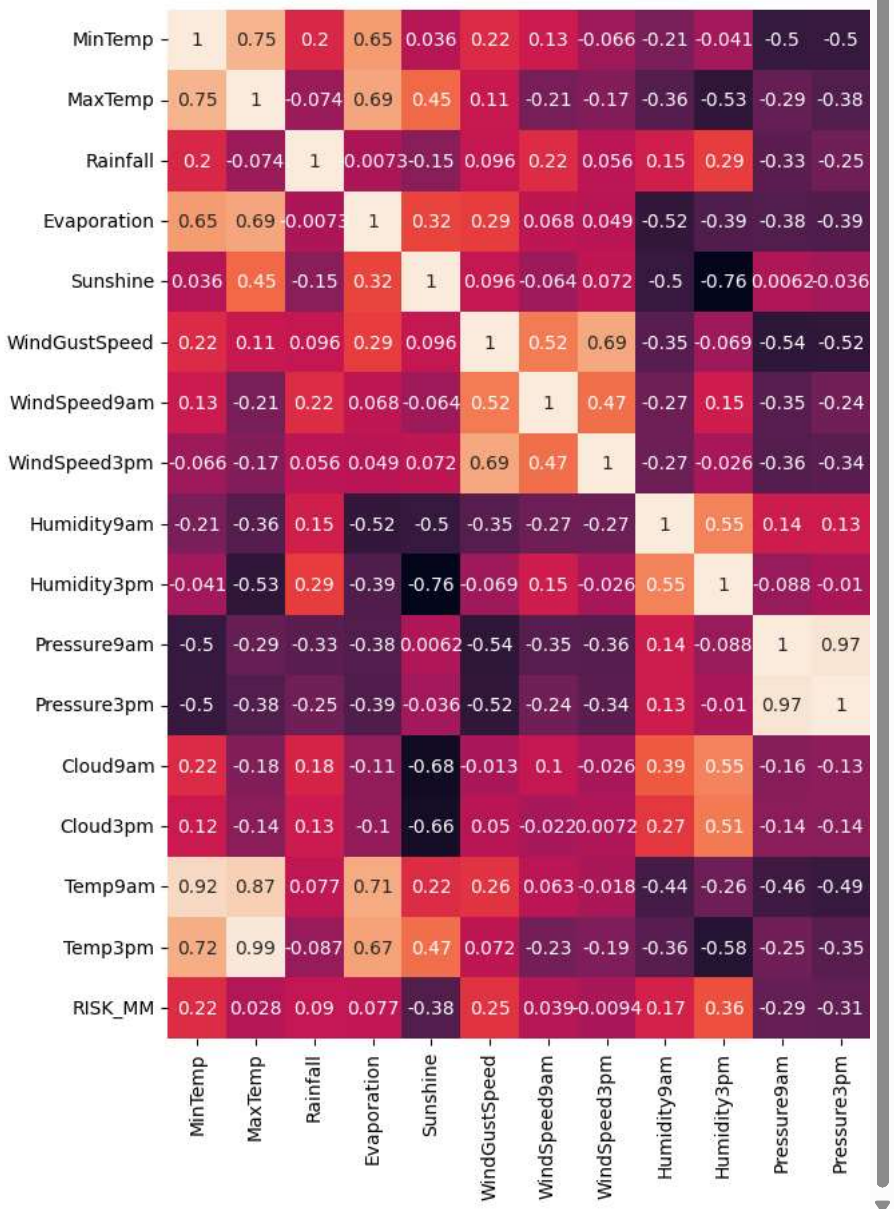
```
data.info()
```

```
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 366 entries, 0 to 365
     Data columns (total 22 columns):
      #   Column         Non-Null Count  Dtype
     ---  ------         --------------  -----
      0   MinTemp        366 non-null    float64
      1   MaxTemp        366 non-null    float64
      2   Rainfall       366 non-null    float64
      3   Evaporation    366 non-null    float64
      4   Sunshine       366 non-null    float64
      5   WindGustDir    366 non-null    object
      6   WindGustSpeed  366 non-null    float64
      7   WindDir9am     366 non-null    object
```

```
 8   WindDir3pm     366 non-null    object 
 9   WindSpeed9am   366 non-null    float64
 10  WindSpeed3pm   366 non-null    int64  
 11  Humidity9am    366 non-null    int64  
 12  Humidity3pm    366 non-null    int64  
 13  Pressure9am    366 non-null    float64
 14  Pressure3pm    366 non-null    float64
 15  Cloud9am       366 non-null    int64  
 16  Cloud3pm       366 non-null    int64  
 17  Temp9am        366 non-null    float64
 18  Temp3pm        366 non-null    float64
 19  RainToday      366 non-null    object 
 20  RISK_MM        366 non-null    float64
 21  RainTomorrow   366 non-null    object 
dtypes: float64(12), int64(5), object(5)
memory usage: 63.0+ KB
```

```
plt.figure(figsize=(12,10))
sns.heatmap(data.corr(), annot=True)
```

```
<ipython-input-12-011d2011025c>:2: FutureWarning: The default value of numeric_only i
  sns.heatmap(data.corr(), annot=True)
<Axes: >
```



Correlation and Regression Analysis Part

```
# @title Correlation and Regression Analysis Part
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv('cleaned_data.csv')

df = pd.get_dummies(df, columns=['WindGustDir', 'WindDir9am', 'WindDir3pm'])

X = df.drop(['RainToday', 'RISK_MM', 'RainTomorrow'], axis=1)
y = df['RainToday']

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

model = LogisticRegression()

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}\n')

print('Classification Report:')
print(classification_report(y_test, y_pred))

df['RainTomorrow'] = df.apply(lambda row: 'Yes' if row['RISK_MM'] > 1 else row['RainToday'], axis=1)

print(df[['RainToday', 'RISK_MM', 'RainTomorrow']])
```

```
Accuracy: 1.00

Classification Report:
              precision    recall  f1-score   support

          No       1.00      1.00      1.00        58
         Yes       1.00      1.00      1.00        16

    accuracy                           1.00        74
   macro avg       1.00      1.00      1.00        74
weighted avg       1.00      1.00      1.00        74

    RainToday  RISK_MM RainTomorrow
0          No      3.6          Yes
1         Yes      3.6          Yes
2         Yes     39.8          Yes
3         Yes      2.8          Yes
4         Yes      0.0          Yes
..        ...      ...          ...
361        No      0.0           No
362        No      0.0           No
363        No      0.0           No
364        No      0.0           No
365        No      0.0           No

[366 rows x 3 columns]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```