In [250]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score , mean_squared_error, r2_sc
```

In [230]:
```
aus = pd.read_csv("AUS_Cleaned2.csv", index_col=0)
```

In [211]:
```
aus.head()
```

Out[211]:

Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustS
2008-12-01	Albury	13.4	22.9	0.6	5.469824	7.624853	W	
2008-12-02	Albury	7.4	25.1	0.0	5.469824	7.624853	WNW	
2008-12-03	Albury	12.9	25.7	0.0	5.469824	7.624853	WSW	
2008-12-04	Albury	9.2	28.0	0.0	5.469824	7.624853	NE	
2008-12-05	Albury	17.5	32.3	1.0	5.469824	7.624853	W	

5 rows × 22 columns

In [231]:
```
cols_to_remove = ['Location', 'WindGustDir',
                  'WindGustSpeed', 'WindDir9am','WindDir3pm','RainToda
removed = aus.loc[:, cols_to_remove]
aus = aus.loc[:, ~aus.columns.isin(cols_to_remove)]
```

In [232]:
```
scaler = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(aus.iloc[:,:]), columns
```

In [233]:
```
aus = pd.concat([df_scaled, removed], axis=1)
```

In [234]:
```
for column in ['Location', 'WindGustDir', 'WindGustSpeed', 'WindDir9am
    dummy_cols = pd.get_dummies(aus[column], prefix=column, drop_first
    aus = pd.concat([aus, dummy_cols], axis=1)
    aus.drop(column, axis=1, inplace=True)
```

In [235]:
```
my_col = aus.pop('RainToday')
df_new = aus.loc[:, aus.columns != 'RainToday']

# Concatenate the slice and the 'my_col' column
aus = pd.concat([df_new, my_col], axis=1)
```

In [236]:
```
my_col = aus.pop('RainTomorrow')
df_new = aus.loc[:, aus.columns != 'RainTomorrow']

# Concatenate the slice and the 'my_col' column
aus = pd.concat([df_new, my_col], axis=1)
```

In [237]:
```
aus
```

Out[237]:

Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindSpeed9am	WindSpeed3pm
2008-12-01	0.516509	0.523629	0.001617	0.037723	0.525852	0.153846	0.275862
2008-12-02	0.375000	0.565217	0.000000	0.037723	0.525852	0.030769	0.252874
2008-12-03	0.504717	0.576560	0.000000	0.037723	0.525852	0.146154	0.298851
2008-12-04	0.417453	0.620038	0.000000	0.037723	0.525852	0.084615	0.103448
2008-12-05	0.613208	0.701323	0.002695	0.037723	0.525852	0.053846	0.229885
...	...	...	...	...	...	...	...
2017-06-20	0.283019	0.502836	0.000000	0.037723	0.525852	0.115385	0.149425
2017-06-21	0.266509	0.533081	0.000000	0.037723	0.525852	0.100000	0.126437
2017-06-22	0.285377	0.568998	0.000000	0.037723	0.525852	0.100000	0.103448
2017-06-23	0.327830	0.599244	0.000000	0.037723	0.525852	0.069231	0.103448
2017-06-24	0.384434	0.601134	0.000000	0.037723	0.525852	0.100000	0.080460

142193 rows × 177 columns

In [238]:
```
# drops around 1000 rows, not too detrimental

aus = aus.dropna()
```

In [ ]:
```

```

In [239]:
```
X = aus.iloc[:, :-1]
y = aus[['RainTomorrow']]
```

In [240]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [241]:
```
logr = LogisticRegression(max_iter = 1000)
logr_fit = logr.fit(X_train,y_train.values.ravel())
logr_predict = logr.predict(X_test)
```

In [242]:
```
accuracy_score(y_test, logr_predict)
```

Out[242]: 0.8515519568151148

In [243]:
```
mean_squared_error(y_test, logr_predict)
```

Out[243]: 0.1484480431848853

In [252]:
```
#this is really bad

r2_score(y_test, logr_predict)
```

Out[252]: 0.1435298337787545

In [253]:
```
#all of the coefficients

column_labels = X.columns.tolist()
coef = logr.coef_.squeeze().tolist()

# Zip together
labels_coef = list(zip(column_labels, coef))
```

In [249]:
```
coefficients = logr.coef_[0]

abs_coefficients = np.abs(coefficients)

# Get the indices that would sort the absolute values in descending or
sorted_indices = np.argsort(abs_coefficients)[::-1]

# Get the column names and corresponding coefficients
columns = X.columns
sorted_columns = columns[sorted_indices]
sorted_coefficients = coefficients[sorted_indices]

# Print the top 5 coefficients and their corresponding columns
for col, coef in zip(sorted_columns[:5], sorted_coefficients[:5]):
    print(f'{col}: {coef:.4f}')
```

```
Pressure3pm: -10.0235
Pressure9am: 6.1767
Humidity3pm: 5.9053
MaxTemp: -2.7073
Rainfall: 2.6019
```

In [256]:
```
#------- What if I didn't include wind direction?
aus = pd.read_csv("AUS_Cleaned2.csv", index_col=0)
aus = aus.drop(columns = ['WindGustDir',
                  'WindGustSpeed', 'WindDir9am','WindDir3pm'])
```

In [257]:
```
cols_to_remove = ['Location','RainToday','RainTomorrow'  ]
removed = aus.loc[:, cols_to_remove]
aus = aus.loc[:, ~aus.columns.isin(cols_to_remove)]

scaler = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(aus.iloc[:,:]), columns


aus = pd.concat([df_scaled, removed], axis=1)

for column in ['Location' ]:
    dummy_cols = pd.get_dummies(aus[column], prefix=column, drop_first
    aus = pd.concat([aus, dummy_cols], axis=1)
    aus.drop(column, axis=1, inplace=True)

my_col = aus.pop('RainToday')
df_new = aus.loc[:, aus.columns != 'RainToday']

# Concatenate the slice and the 'my_col' column
aus = pd.concat([df_new, my_col], axis=1)

my_col = aus.pop('RainTomorrow')
df_new = aus.loc[:, aus.columns != 'RainTomorrow']

# Concatenate the slice and the 'my_col' column
aus = pd.concat([df_new, my_col], axis=1)

aus
```

Out[257]:

Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindSpeed9am	WindSpeed3pm
2008-12-01	0.516509	0.523629	0.001617	0.037723	0.525852	0.153846	0.275862
2008-12-02	0.375000	0.565217	0.000000	0.037723	0.525852	0.030769	0.252874
2008-12-03	0.504717	0.576560	0.000000	0.037723	0.525852	0.146154	0.298851
2008-12-04	0.417453	0.620038	0.000000	0.037723	0.525852	0.084615	0.103448
2008-12-05	0.613208	0.701323	0.002695	0.037723	0.525852	0.053846	0.229885
...	...	...	...	...	...	...	...
2017-	0.283019	0.502836	0.000000	0.037723	0.525852	0.115385	0.149425

In [258]:
```
aus = aus.dropna()
```

In [259]:
```
X = aus.iloc[:, :-1]
y = aus[['RainTomorrow']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
logr = LogisticRegression(max_iter = 1000)
logr_fit = logr.fit(X_train,y_train.values.ravel())
logr_predict = logr.predict(X_test)
```

In [260]:
```
accuracy_score(y_test, logr_predict)
```

Out[260]:
```
0.8426379714468357
```

In [261]:
```
mean_squared_error(y_test, logr_predict)
```

Out[261]:
```
0.15736202855316428
```

In [262]:
```
r2_score(y_test, logr_predict)
```

Out[262]:
```
0.08342214645559387
```

In [263]:
```
coefficients = logr.coef_[0]

abs_coefficients = np.abs(coefficients)

# Get the indices that would sort the absolute values in descending or
sorted_indices = np.argsort(abs_coefficients)[::-1]

# Get the column names and corresponding coefficients
columns = X.columns
sorted_columns = columns[sorted_indices]
sorted_coefficients = coefficients[sorted_indices]

# Print the top 5 coefficients and their corresponding columns
for col, coef in zip(sorted_columns[:5], sorted_coefficients[:5]):
    print(f'{col}: {coef:.4f}')
```

```
Pressure3pm: -12.5218
Pressure9am: 6.8232
Humidity3pm: 5.5889
Rainfall: 3.4347
WindSpeed9am: 1.9955
```