NEXUS PROJECT – 2

NAME : BANDREDDY NITHIN

Project Title : Weather Analysis

Summary Of Weather Data :

The dataframe df1 has 366 entries (rows) and 22 columns. The columns are:

- MinTemp: Minimum temperature (float64)
- MaxTemp: Maximum temperature (float64)
- Rainfall: Amount of rainfall (float64)
- Evaporation: Amount of evaporation (float64)
- Sunshine: Amount of sunshine (float64, 3 missing values)
- WindGustDir: Wind gust direction (object, 3 missing values)
- WindGustSpeed: Wind gust speed (float64, 2 missing values)
- WindDir9am: Wind direction at 9am (object, 31 missing values)
- WindDir3pm: Wind direction at 3pm (object, 1 missing value)
- WindSpeed9am: Wind speed at 9am (float64, 7 missing values)
- WindSpeed3pm: Wind speed at 3pm (int64)
- Humidity9am: Humidity at 9am (int64)
- Humidity3pm: Humidity at 3pm (int64)
- Pressure9am: Atmospheric pressure at 9am (float64)
- Pressure3pm: Atmospheric pressure at 3pm (float64)
- Cloud9am: Cloudiness at 9am (int64)
- Cloud3pm: Cloudiness at 3pm (int64)
- Temp9am: Temperature at 9am (float64)
- Temp3pm: Temperature at 3pm (float64)
- RainToday: Whether it rained today (object)
- RISK_MM: Risk millimeters (float64)
- RainTomorrow: Whether it will rain tomorrow (object)

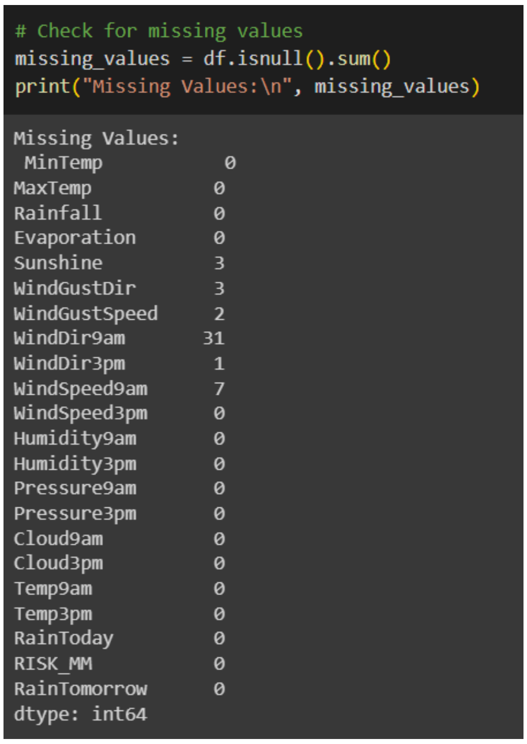
There are some missing values in the columns Sunshine, WindGustDir, WindGustSpeed, WindDir9am, WindDir3pm, and WindSpeed9am. The data types are float64, int64, and object (which usually means string).

- **MinTemp:** The minimum temperature ranges from -5.3 to 20.9 with an average of 7.27. The 25th percentile is 2.3, the median is 7.45, and the 75th percentile is 12.5.
- **MaxTemp:** The maximum temperature ranges from 7.6 to 35.8 with an average of 20.55. The 25th percentile is 15.03, the median is 19.65, and the 75th percentile is 25.5.
- **Rainfall:** The rainfall ranges from 0 to 39.8 with an average of 1.43. The 25th percentile is 0, the median is 0, and the 75th percentile is 0.2.
- Evaporation: The evaporation ranges from 0.2 to 13.8 with an average of 4.52. The 25th percentile is 2.2, the median is 4.2, and the 75th percentile is 6.4.
- **Sunshine:** The sunshine ranges from 0 to 13.6 with an average of 7.91. The 25th percentile is 5.95, the median is 8.6, and the 75th percentile is 10.5.
- **WindGustDir:** The most frequent wind gust direction is NW, occurring 73 times.
- **WindGustSpeed:** The wind gust speed ranges from 13 to 98 with an average of 39.84. The 25th percentile is 31, the median is 39, and the 75th percentile is 46.
- **WindDir9am:** The most frequent wind direction at 9am is SE, occurring 47 times.
- **WindDir3pm:** The most frequent wind direction at 3pm is NW, occurring 61 times.
- **WindSpeed9am:** The wind speed at 9am ranges from 0 to 41 with an average of 9.65. The 25th percentile is 6, the median is 7, and the 75th percentile is 13.
- **WindSpeed3pm:** The wind speed at 3pm ranges from 0 to 52 with an average of 17.99. The 25th percentile is 11, the median is 17, and the 75th percentile is 24.
- **Humidity9am:** The humidity at 9am ranges from 36 to 99 with an average of 72.04. The 25th percentile is 64, the median is 72, and the 75th percentile is 81.
- **Humidity3pm:** The humidity at 3pm ranges from 13 to 96 with an average of 44.52. The 25th percentile is 32.25, the median is 43, and the 75th percentile is 55.

- **Pressure9am:** The atmospheric pressure at 9am ranges from 996.5 to 1035.7 with an average of 1019.71. The 25th percentile is 1015.35, the median is 1020.15, and the 75th percentile is 1024.48.
- **Pressure3pm:** The atmospheric pressure at 3pm ranges from 996.8 to 1033.2 with an average of 1016.81. The 25th percentile is 1012.8, the median is 1017.4, and the 75th percentile is 1021.48.
- **Cloud9am:** The cloudiness at 9am ranges from 0 to 8 with an average of 3.89. The 25th percentile is 1, the median is 3.5, and the 75th percentile is 7.
- **Cloud3pm:** The cloudiness at 3pm ranges from 0 to 8 with an average of 4.02. The 25th percentile is 1, the median is 4, and the 75th percentile is 7.
- **Temp9am:** The temperature at 9am ranges from 0.1 to 24.7 with an average of 12.36. The 25th percentile is 7.63, the median is 12.55, and the 75th percentile is 17.
- **Temp3pm:** The temperature at 3pm ranges from 5.1 to 34.5 with an average of 19.23. The 25th percentile is 14.15, the median is 18.55, and the 75th percentile is 24.
- **RainToday:** It rained today in 300 instances.
- **RISK_MM:** The risk millimeters ranges from 0 to 39.8 with an average of 1.43. The 25th percentile is 0, the median is 0, and the 75th percentile is 0.2.
- **RainTomorrow:** It will rain tomorrow in 300 instances.

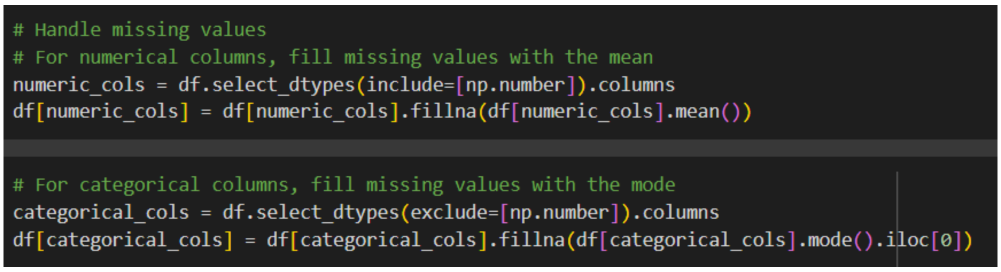
Python Codes :

Data Cleaning



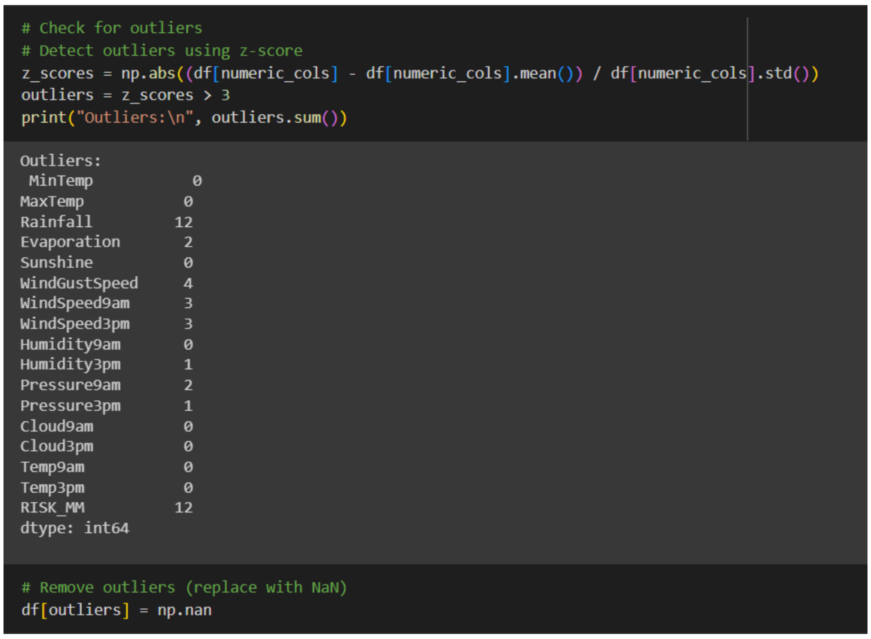
```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
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
```



```
# Handle missing values
# For numerical columns, fill missing values with the mean
numeric_cols = df.select_dtypes(include=[np.number]).columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
```

```
# For categorical columns, fill missing values with the mode
categorical_cols = df.select_dtypes(exclude=[np.number]).columns
df[categorical_cols] = df[categorical_cols].fillna(df[categorical_cols].mode().iloc[0])
```



```
# Check for outliers
# Detect outliers using z-score
z_scores = np.abs((df[numeric_cols] - df[numeric_cols].mean()) / df[numeric_cols].std())
outliers = z_scores > 3
print("Outliers:\n", outliers.sum())
```

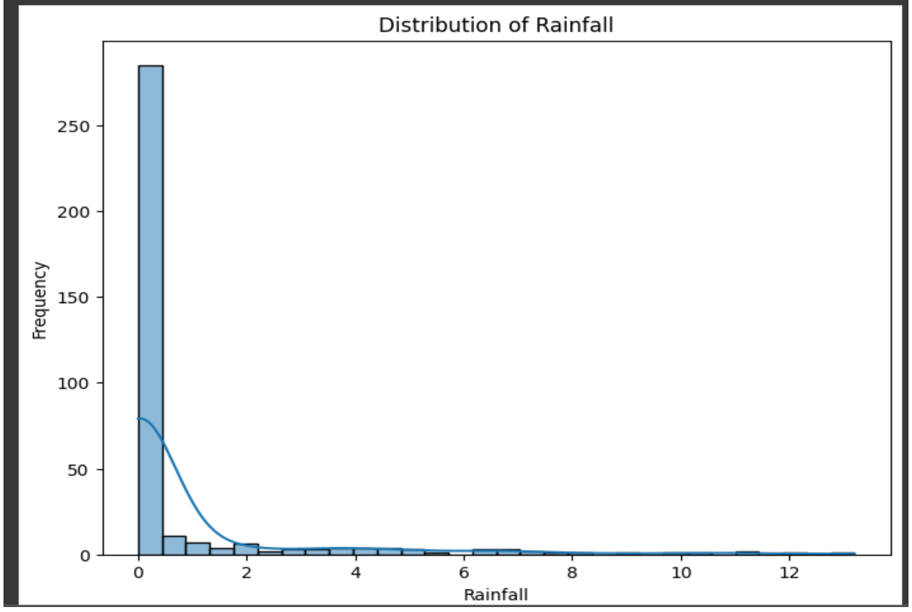
```
Outliers:
 MinTemp          0
MaxTemp          0
Rainfall        12
Evaporation      2
Sunshine         0
WindGustSpeed    4
WindSpeed9am     3
WindSpeed3pm     3
Humidity9am      0
Humidity3pm      1
Pressure9am      2
Pressure3pm      1
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RISK_MM         12
dtype: int64
```

```
# Remove outliers (replace with NaN)
df[outliers] = np.nan
```

Visualization

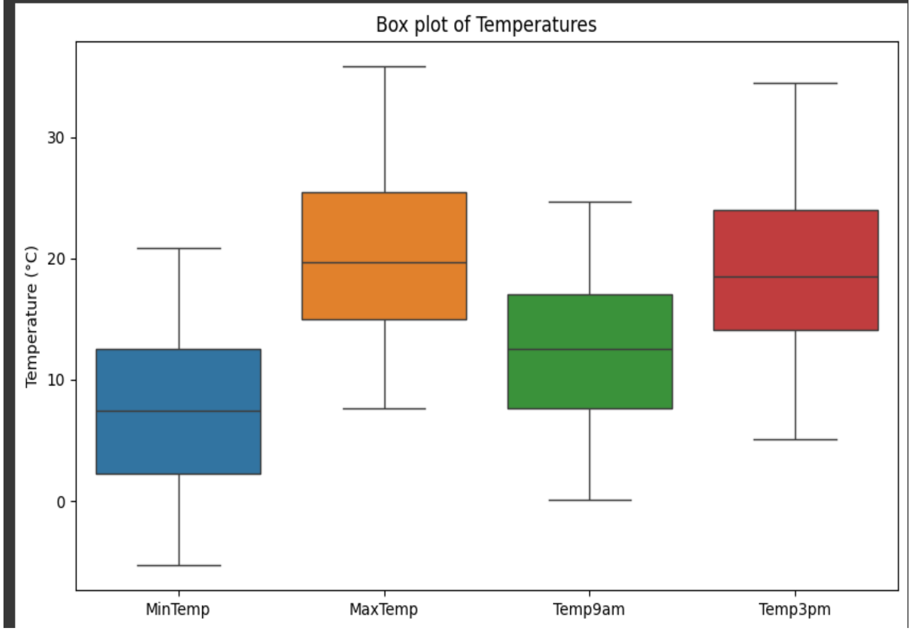
Histogram of Rainfall :

```
plt.figure(figsize=(8, 6))
sns.histplot(df['Rainfall'], bins=30, kde=True)
plt.title("Distribution of Rainfall")
plt.xlabel("Rainfall")
plt.ylabel("Frequency")
plt.show()
```



Box Plot of Temperature

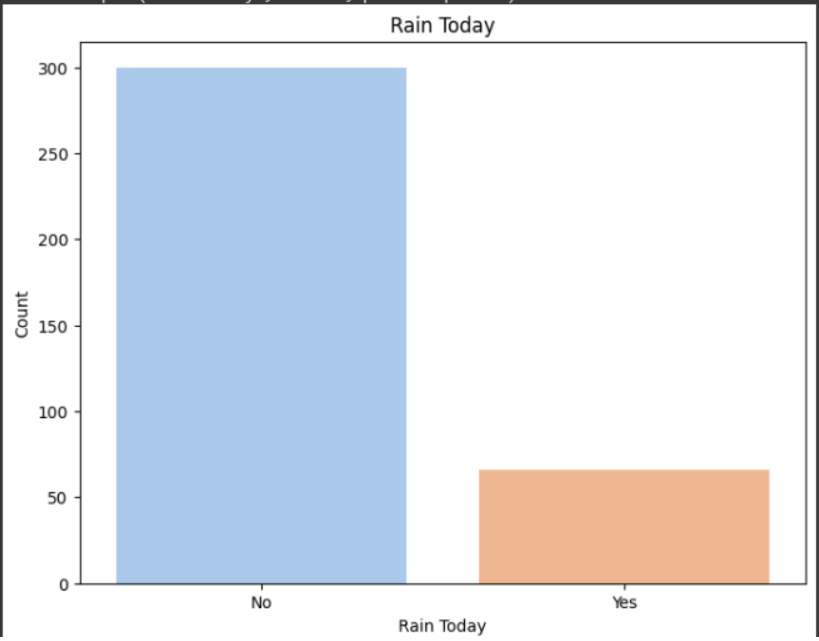
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['MinTemp', 'MaxTemp', 'Temp9am', 'Temp3pm']])
plt.title("Box plot of Temperatures")
plt.ylabel("Temperature (°C)")
plt.show()
```



Bar plot of RainToday

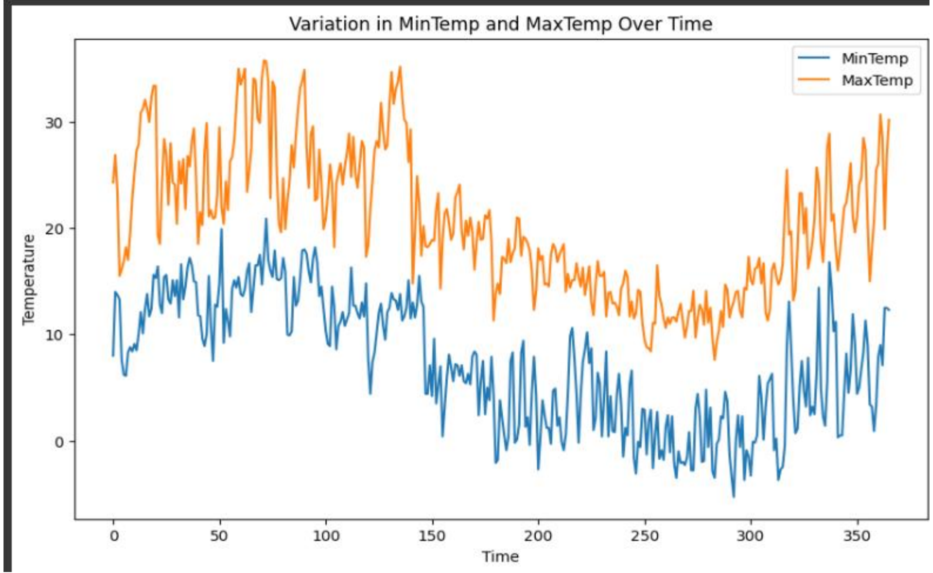
```
plt.figure(figsize=(8, 6))
sns.countplot(x='RainToday', data=df, palette='pastel')
plt.title("Rain Today")
plt.xlabel("Rain Today")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-11-95260d9c5457>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` var

  sns.countplot(x='RainToday', data=df, palette='pastel')
```



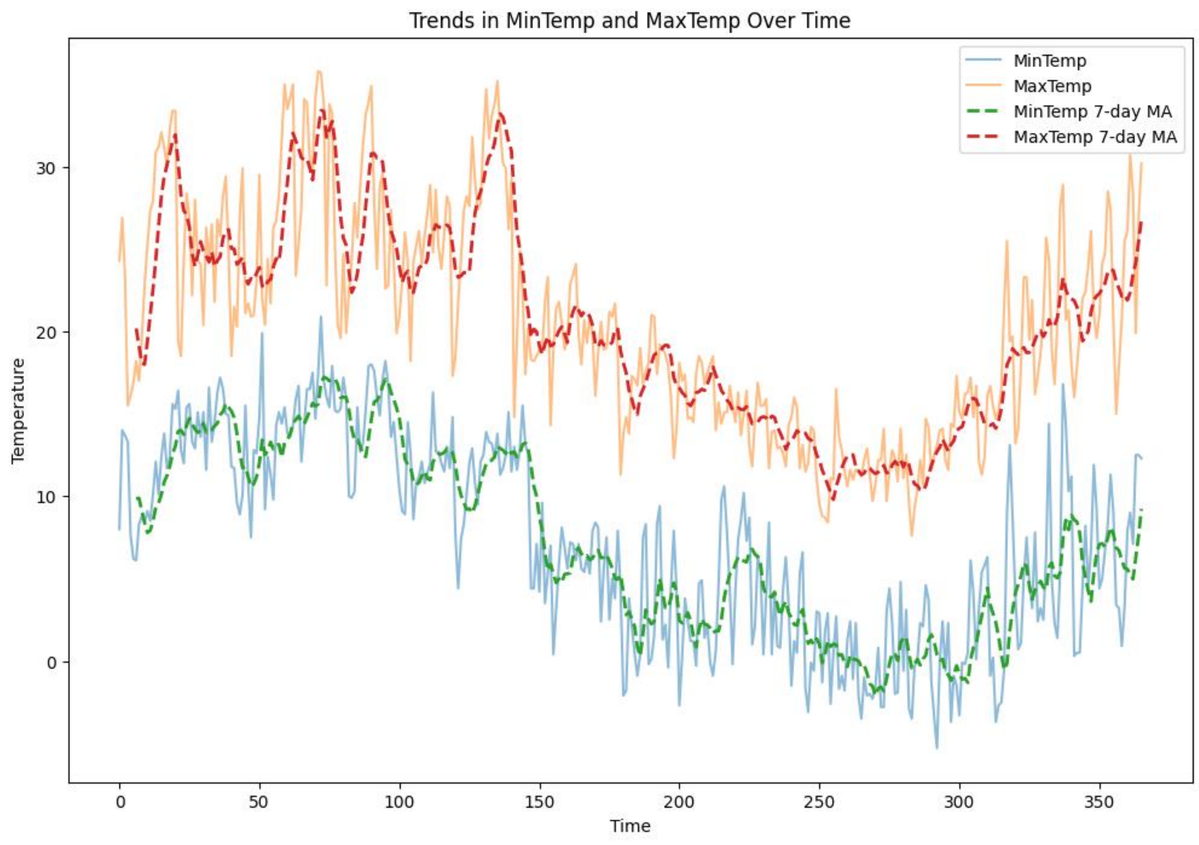
Variation in MinTemp and MaxTemp Over Time

```
time_index = range(len(df))

plt.figure(figsize=(10, 6))
plt.plot(time_index, df['MinTemp'], label='MinTemp')
plt.plot(time_index, df['MaxTemp'], label='MaxTemp')
plt.title('Variation in MinTemp and MaxTemp Over Time')
plt.xlabel('Time')
plt.ylabel('Temperature')
plt.legend()
plt.show()
```



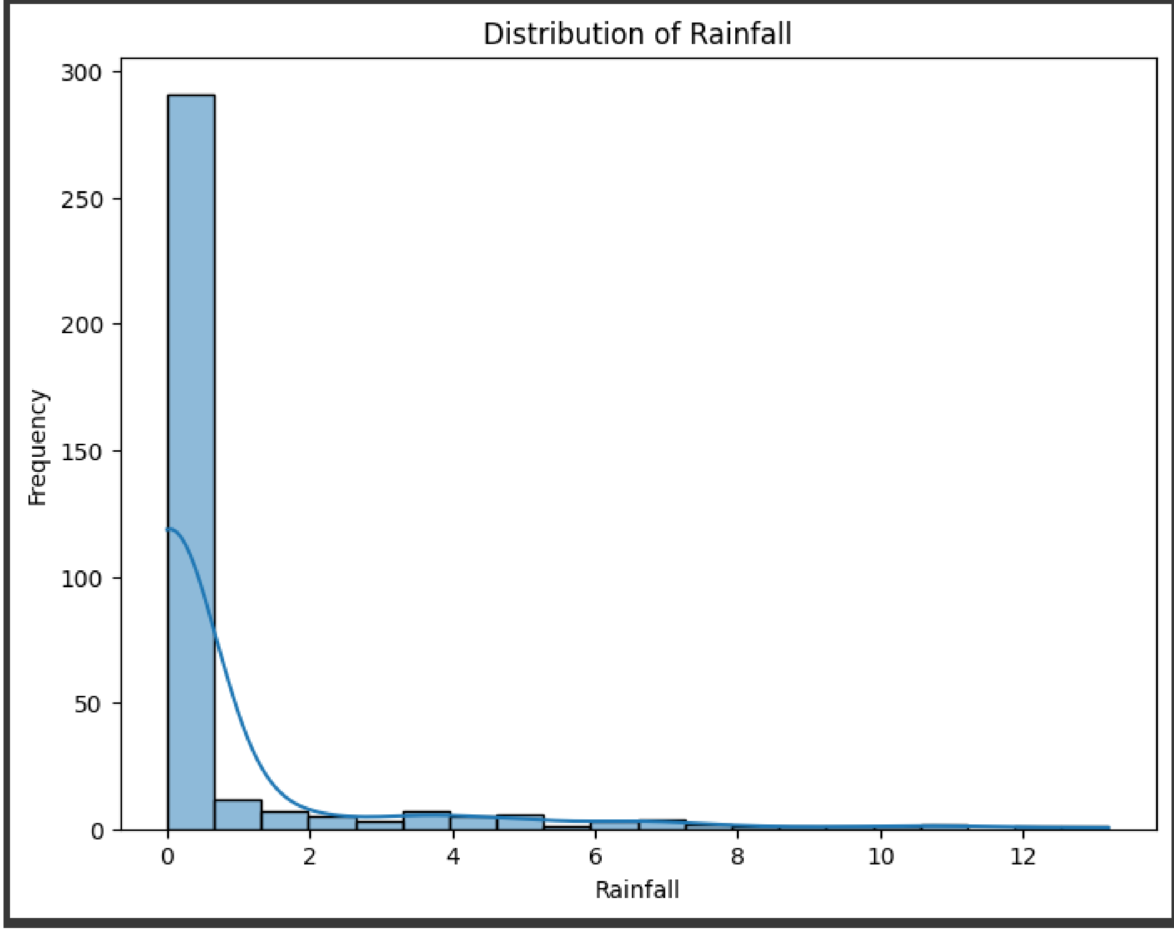
Trends in MinTemp and MaxTemp Over Time

```
# Plot the original data and rolling averages
plt.figure(figsize=(12, 8))
plt.plot(df['MinTemp'], label='MinTemp', alpha=0.5)
plt.plot(df['MaxTemp'], label='MaxTemp', alpha=0.5)
plt.plot(df['MinTemp_MA'], label=f'MinTemp {window_size}-day MA', linestyle='--', linewidth=2)
plt.plot(df['MaxTemp_MA'], label=f'MaxTemp {window_size}-day MA', linestyle='--', linewidth=2)

plt.title('Trends in MinTemp and MaxTemp Over Time')
plt.xlabel('Time')
plt.ylabel('Temperature')
plt.legend()
plt.show()
```

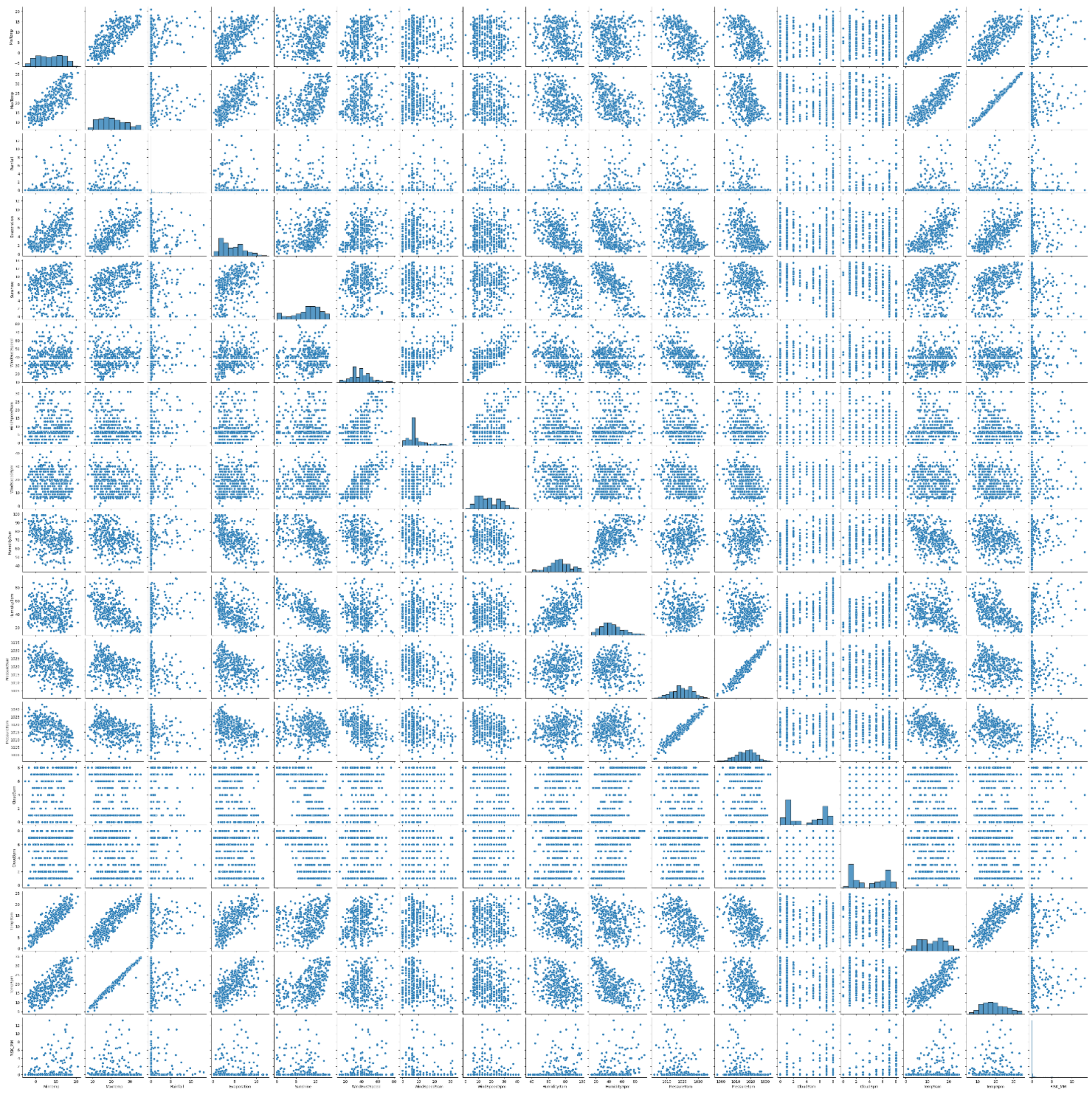


Descriptive Statistics

```
plt.figure(figsize=(8, 6))
sns.histplot(df['Rainfall'], bins=20, kde=True)
plt.title("Distribution of Rainfall")
plt.xlabel("Rainfall")
plt.ylabel("Frequency")
plt.show()
```

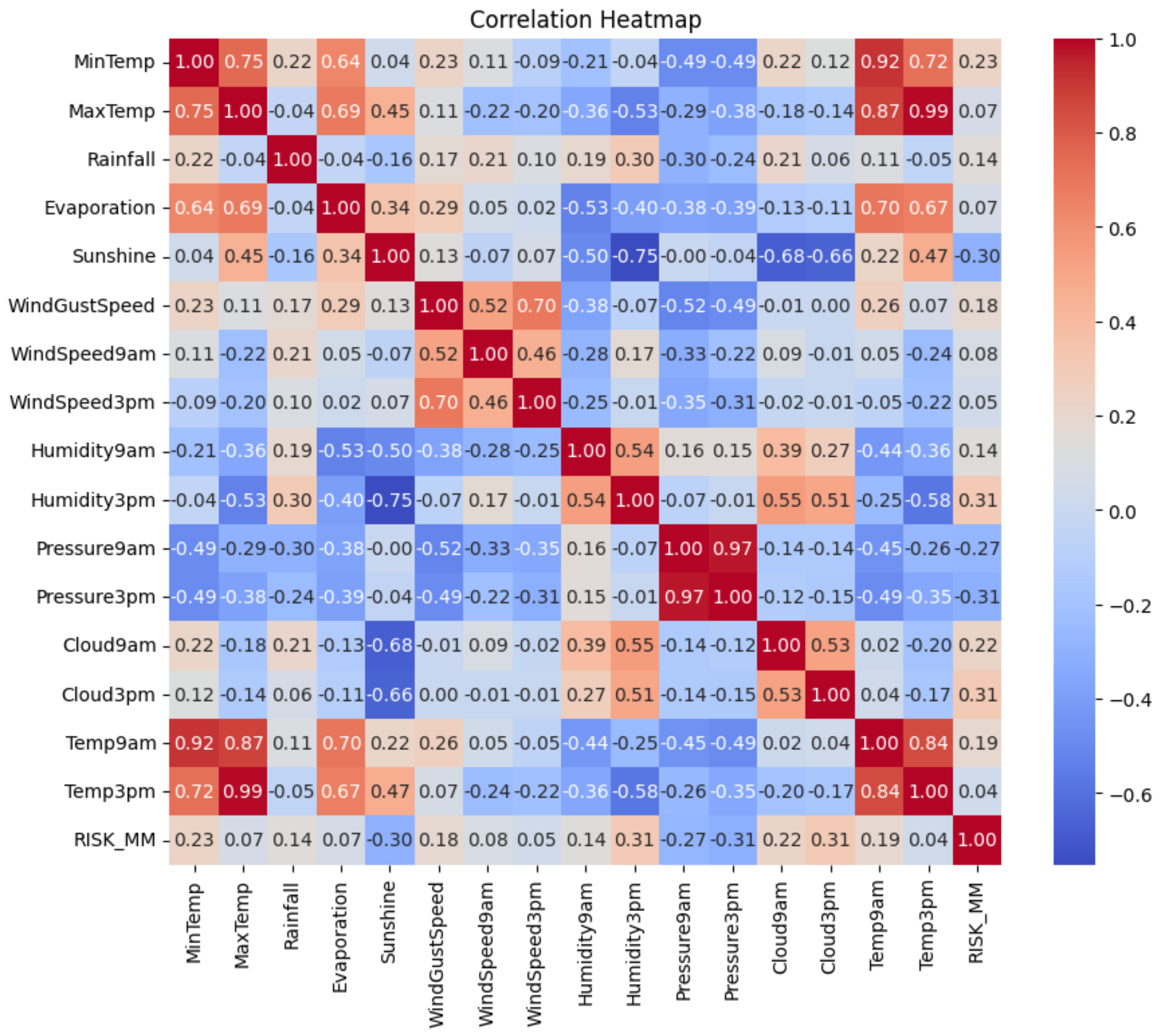


Pairplot of the Columns

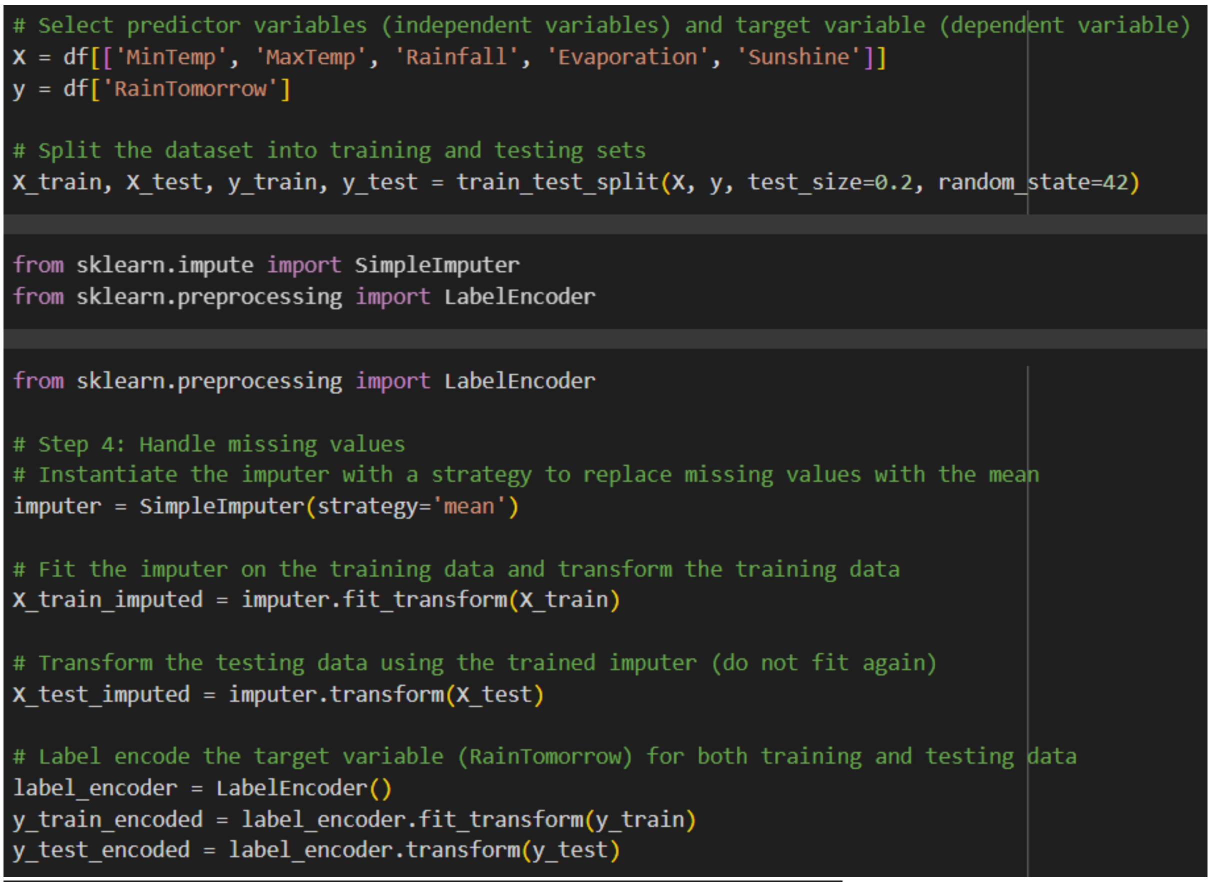


Correlation and Regression Analysis

```
# Step 1: Load the dataset
df = pd.read_csv("weather.csv")

# Step 2: Correlation Analysis
correlation_matrix = df.corr()

# Step 3: Visualize Correlations
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



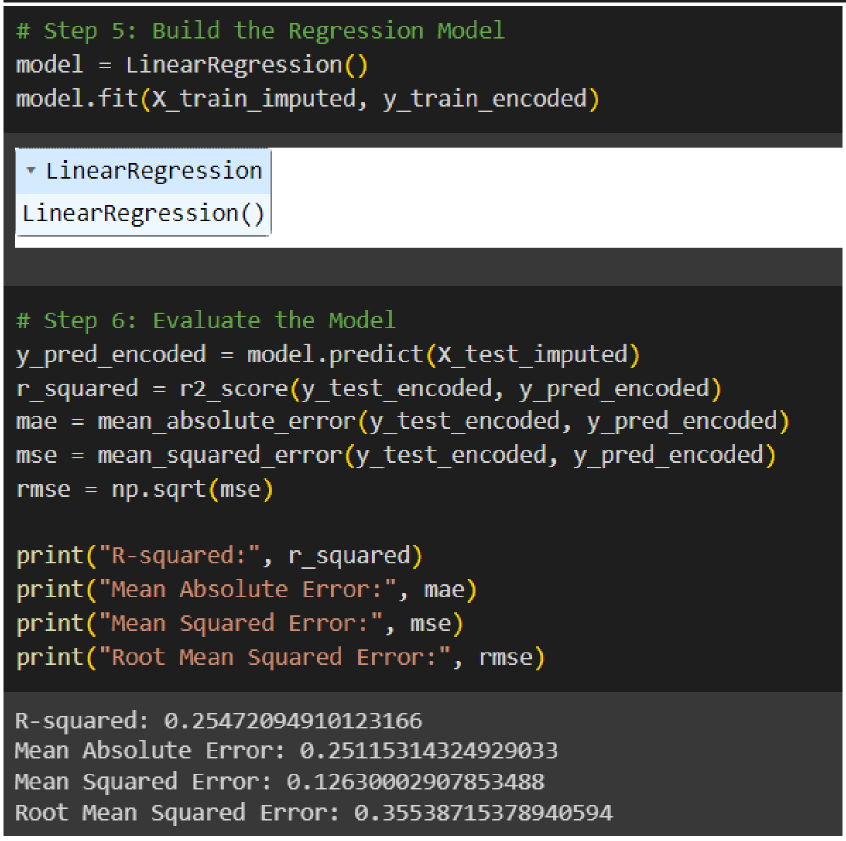
	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RISK_MM
MinTemp	1.00	0.75	0.22	0.64	0.04	0.23	0.11	-0.09	-0.21	-0.04	-0.49	-0.49	0.22	0.12	0.92	0.72	0.23
MaxTemp	0.75	1.00	-0.04	0.69	0.45	0.11	-0.22	-0.20	-0.36	-0.53	-0.29	-0.38	-0.18	-0.14	0.87	0.99	0.07
Rainfall	0.22	-0.04	1.00	-0.04	-0.16	0.17	0.21	0.10	0.19	0.30	-0.30	-0.24	0.21	0.06	0.11	-0.05	0.14
Evaporation	0.64	0.69	-0.04	1.00	0.34	0.29	0.05	0.02	-0.53	-0.40	-0.38	-0.39	-0.13	-0.11	0.70	0.67	0.07
Sunshine	0.04	0.45	-0.16	0.34	1.00	0.13	-0.07	0.07	-0.50	-0.75	-0.00	-0.04	-0.68	-0.66	0.22	0.47	-0.30
WindGustSpeed	0.23	0.11	0.17	0.29	0.13	1.00	0.52	0.70	-0.38	-0.07	-0.52	-0.49	-0.01	0.00	0.26	0.07	0.18
WindSpeed9am	0.11	-0.22	0.21	0.05	-0.07	0.52	1.00	0.46	-0.28	0.17	-0.33	-0.22	0.09	-0.01	0.05	-0.24	0.08
WindSpeed3pm	-0.09	-0.20	0.10	0.02	0.07	0.70	0.46	1.00	-0.25	-0.01	-0.35	-0.31	-0.02	-0.01	-0.05	-0.22	0.05
Humidity9am	-0.21	-0.36	0.19	-0.53	-0.50	-0.38	-0.28	-0.25	1.00	0.54	0.16	0.15	0.39	0.27	-0.44	-0.36	0.14
Humidity3pm	-0.04	-0.53	0.30	-0.40	-0.75	-0.07	0.17	-0.01	0.54	1.00	-0.07	-0.01	0.55	0.51	-0.25	-0.58	0.31
Pressure9am	-0.49	-0.29	-0.30	-0.38	-0.00	-0.52	-0.33	-0.35	0.16	-0.07	1.00	0.97	-0.14	-0.14	-0.45	-0.26	-0.27
Pressure3pm	-0.49	-0.38	-0.24	-0.39	-0.04	-0.49	-0.22	-0.31	0.15	-0.01	0.97	1.00	-0.12	-0.15	-0.49	-0.35	-0.31
Cloud9am	0.22	-0.18	0.21	-0.13	-0.68	-0.01	0.09	-0.02	0.39	0.55	-0.14	-0.12	1.00	0.53	0.02	-0.20	0.22
Cloud3pm	0.12	-0.14	0.06	-0.11	-0.66	0.00	-0.01	-0.01	0.27	0.51	-0.14	-0.15	0.53	1.00	0.04	-0.17	0.31
Temp9am	0.92	0.87	0.11	0.70	0.22	0.26	0.05	-0.05	-0.44	-0.25	-0.45	-0.49	0.02	0.04	1.00	0.84	0.19
Temp3pm	0.72	0.99	-0.05	0.67	0.47	0.07	-0.24	-0.22	-0.36	-0.58	-0.26	-0.35	-0.20	-0.17	0.84	1.00	0.04
RISK_MM	0.23	0.07	0.14	0.07	-0.30	0.18	0.08	0.05	0.14	0.31	-0.27	-0.31	0.22	0.31	0.19	0.04	1.00



```python
# Select predictor variables (independent variables) and target variable (dependent variable)
X = df[['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine']]
y = df['RainTomorrow']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
```

```python
from sklearn.preprocessing import LabelEncoder

# Step 4: Handle missing values
# Instantiate the imputer with a strategy to replace missing values with the mean
imputer = SimpleImputer(strategy='mean')

# Fit the imputer on the training data and transform the training data
X_train_imputed = imputer.fit_transform(X_train)

# Transform the testing data using the trained imputer (do not fit again)
X_test_imputed = imputer.transform(X_test)

# Label encode the target variable (RainTomorrow) for both training and testing data
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

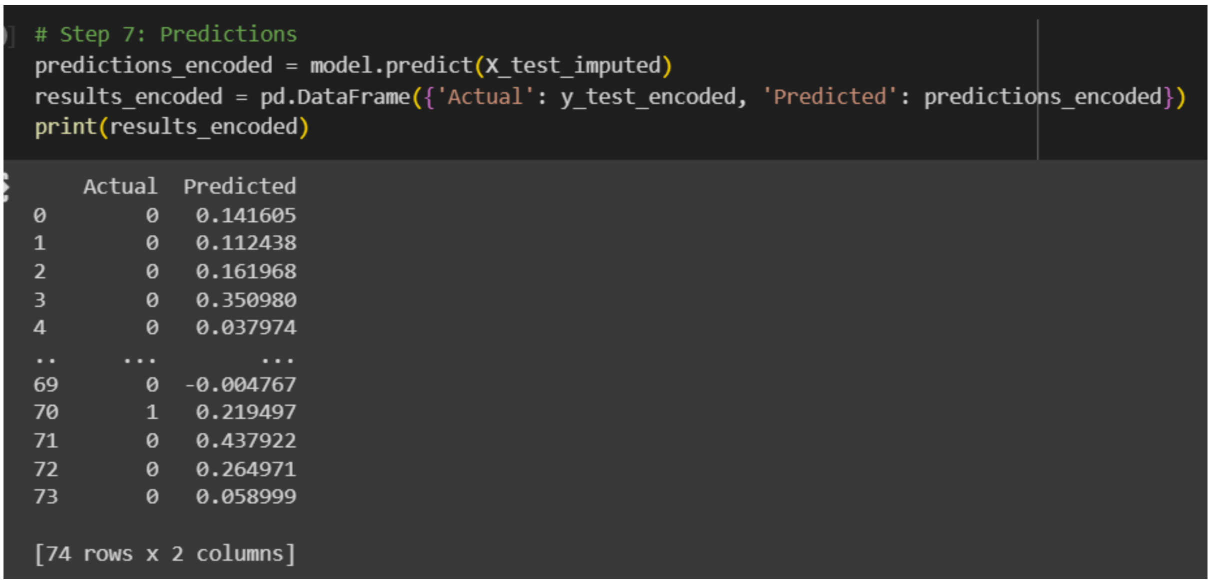


```python
# Step 5: Build the Regression Model
model = LinearRegression()
model.fit(X_train_imputed, y_train_encoded)
```

```
▾ LinearRegression
LinearRegression()
```

```python
# Step 6: Evaluate the Model
y_pred_encoded = model.predict(X_test_imputed)
r_squared = r2_score(y_test_encoded, y_pred_encoded)
mae = mean_absolute_error(y_test_encoded, y_pred_encoded)
mse = mean_squared_error(y_test_encoded, y_pred_encoded)
rmse = np.sqrt(mse)

print("R-squared:", r_squared)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

```
R-squared: 0.25472094910123166
Mean Absolute Error: 0.25115314324929033
Mean Squared Error: 0.12630002907853488
Root Mean Squared Error: 0.35538715378940594
```



```
# Step 7: Predictions
predictions_encoded = model.predict(X_test_imputed)
results_encoded = pd.DataFrame({'Actual': y_test_encoded, 'Predicted': predictions_encoded})
print(results_encoded)
```

```
    Actual  Predicted
0        0   0.141605
1        0   0.112438
2        0   0.161968
3        0   0.350980
4        0   0.037974
..     ...        ...
69       0  -0.004767
70       1   0.219497
71       0   0.437922
72       0   0.264971
73       0   0.058999

[74 rows x 2 columns]
```

Summary of the Python Code:

Data Preparation with Python:

- Loaded the weather dataset using Pandas.
- Handled missing values by filling numerical columns with the mean and categorical columns with the mode.
- Detected outliers using z-score and replaced them with NaN.
- Performed data visualization using Matplotlib and Seaborn to understand the distribution of variables better.
- Saved the cleaned dataset to a new CSV file.

Correlation and Linear Regression:

- Conducted correlation analysis to identify relationships between different weather parameters.
- Used linear regression to predict 'MaxTemp' based on 'MinTemp', splitting the data into training and testing sets, making predictions, and evaluating the model's performance using mean squared error, root mean squared error, and R-squared