```
print("HI")
```

```
HI
```

```
Start coding or generate with AI.
```

```
import numpy as np
import pandas as pd


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

pd.set_option('display.max_columns', None)
```

```
# ************* Misc. *************
import random

from prettytable import PrettyTable

# ************* Plotting *************
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt

# ************* Data Manipulation *************
from sklearn.preprocessing import LabelEncoder
```

Double-click (or enter) to edit

```
df = pd.read_csv("/content/weather_classification_data.csv")

df.describe()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	V
count	13200.000000	13200.000000	13200.000000	13200.000000	13200.000000	13200.000000	132
mean	19.127576	68.710833	9.832197	53.644394	1005.827896	4.005758	
std	17.386327	20.194248	6.908704	31.946541	37.199589	3.856600	
min	-25.000000	20.000000	0.000000	0.000000	800.120000	0.000000	
25%	4.000000	57.000000	5.000000	19.000000	994.800000	1.000000	
50%	21.000000	70.000000	9.000000	58.000000	1007.650000	3.000000	
75%	31.000000	84.000000	13.500000	82.000000	1016.772500	7.000000	
max	109.000000	109.000000	48.500000	109.000000	1199.210000	14.000000	

```
df.head()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibilit (km
0	14	73	9.5	82	partly cloudy	1010.82	2	Winter	3
1	39	96	8.5	71	partly cloudy	1011.43	7	Spring	10
2	30	64	7.0	16	clear	1018.72	5	Spring	5
3	38	83	1.5	82	clear	1026.25	7	Spring	1
4	27	74	17.0	66	overcast	990.67	1	Winter	2

Next steps: Generate code with df New interactive sheet

```
df
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visi
0	14	73	9.5	82	partly cloudy	1010.82	2	Winter	
1	39	96	8.5	71	partly cloudy	1011.43	7	Spring	
2	30	64	7.0	16	clear	1018.72	5	Spring	
3	38	83	1.5	82	clear	1026.25	7	Spring	
4	27	74	17.0	66	overcast	990.67	1	Winter	
...	...	...	...	...	...	...	...	...	
13195	10	74	14.5	71	overcast	1003.15	1	Summer	
13196	-1	76	3.5	23	cloudy	1067.23	1	Winter	
13197	30	77	5.5	28	overcast	1012.69	3	Autumn	
13198	3	76	10.0	94	overcast	984.27	0	Winter	
13199	-5	38	0.0	92	overcast	1015.37	5	Autumn	

13200 rows × 11 columns

Next steps: Generate code with df New interactive sheet

```
df.head(200)
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibi
0	14	73	9.5	82	partly cloudy	1010.82	2	Winter	
1	39	96	8.5	71	partly cloudy	1011.43	7	Spring	
2	30	64	7.0	16	clear	1018.72	5	Spring	
3	38	83	1.5	82	clear	1026.25	7	Spring	
4	27	74	17.0	66	overcast	990.67	1	Winter	
...	...	...	...	...	...	...	...	...	
195	26	68	4.0	39	partly cloudy	1016.39	4	Summer	
196	10	99	16.0	58	overcast	995.85	2	Autumn	
197	20	42	4.0	13	partly cloudy	1028.30	5	Autumn	
198	-2	32	1.5	17	overcast	930.32	1	Winter	
199	-1	94	13.0	74	overcast	981.13	0	Winter	

200 rows × 11 columns

Next steps: Generate code with df New interactive sheet

```
df.tail()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visi
13195	10	74	14.5	71	overcast	1003.15	1	Summer	
13196	-1	76	3.5	23	cloudy	1067.23	1	Winter	
13197	30	77	5.5	28	overcast	1012.69	3	Autumn	
13198	3	76	10.0	94	overcast	984.27	0	Winter	
13199	-5	38	0.0	92	overcast	1015.37	5	Autumn	

```
df.tail(200)
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visi
13000	-6	65	6.0	85	overcast	980.58	0	Winter	
13001	-10	86	9.5	98	overcast	985.98	1	Winter	
13002	-5	61	14.0	63	overcast	999.86	0	Winter	
13003	10	51	4.5	31	partly cloudy	1008.56	4	Autumn	
...	...	...	...	...	...	...	...	...	
			14.5	71	overcast	1003.15	1	Summer	
13196	-1	76	3.5	23	cloudy	1067.23	1	Winter	
			5.5	28	overcast	1012.69	3	Autumn	
13198	3	76	10.0	94	overcast	984.27	0	Winter	
			0.0	92	overcast	1015.37	5	Autumn	

200 rows × 11 columns

```
df.count()
```

	0
Temperature	13200
Humidity	13200
Wind Speed	13200
Precipitation (%)	13200
Cloud Cover	13200
Atmospheric Pressure	13200
UV Index	13200
Season	13200
Visibility (km)	13200
Location	13200
Weather Type	13200

dtype: int64

```
print(df.shape)
```

```
(13200, 11)
```

```
df.describe()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	V
count	13200.000000	13200.000000	13200.000000	13200.000000	13200.000000	13200.000000	132
mean	19.127576	68.710833	9.832197	53.644394	1005.827896	4.005758	
std	17.386327	20.194248	6.908704	31.946541	37.199589	3.856600	
min	-25.000000	20.000000	0.000000	0.000000	800.120000	0.000000	
25%	4.000000	57.000000	5.000000	19.000000	994.800000	1.000000	
50%	21.000000	70.000000	9.000000	58.000000	1007.650000	3.000000	
75%	31.000000	84.000000	13.500000	82.000000	1016.772500	7.000000	
max	109.000000	109.000000	48.500000	109.000000	1199.210000	14.000000	

```
df.mean()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-3698961737.py in <cell line: 0>()
----> 1 df.mean()

                                    10 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/nanops.py in _ensure_numeric(x)
   1684             if inferred in ["string", "mixed"]:
   1685                 # GH#44008, GH#36703 avoid casting e.g. strings to numeric
-> 1686                 raise TypeError(f"Could not convert {x} to numeric")
   1687             try:
   1688                 x = x.astype(np.complex128)

TypeError: Could not convert ['partly cloudypartly
cloudyclearclearovercastovercastovercastpartly cloudyovercastclearpartly cloudyclearpartly
cloudyovercastclearpartly cloudypartly
cloudyovercastclearclearovercastovercastclearovercastpartly cloudypartly
cloudyovercastpartly cloudypartly cloudypartly
cloudyovercastovercastovercastovercastclearclearpartly cloudyovercastpartly cloudypartly
cloudypartly cloudyclearpartly cloudypartly cloudyclearclearpartly cloudypartly
cloudyovercastovercastovercastpartly cloudypartly
cloudyovercastovercastovercastovercastclearovercastovercastovercastovercastpartly
cloudyovercastpartly cloudyovercastpartly cloudyclearpartly cloudyclearovercastpartly
cloudyovercastovercastclearclearpartly cloudyovercastpartly cloudyovercastpartly
cloudypartly cloudypartly cloudyovercastovercastpartly cloudyovercastclearpartly
cloudyovercastpartly cloudypartly cloudyovercastpartly cloudypartly cloudypartly
cloudyovercastpartly cloudyovercastpartly cloudyclearclearovercastovercastpartly
cloudyovercastclearovercastclearovercastclearpartly
cloudyovercastovercastovercastovercastpartly
cloudyovercastovercastclearovercastovercastpartly cloudypartly
cloudyovercastovercastovercastovercastovercastovercastpartly
cloudyovercastovercastovercastcloudypartly cloudypartly cloudyclearovercastpartly
cloudyovercastclearovercastovercastclearovercastpartly cloudypartly cloudycloudypartly
cloudyovercastovercastpartly cloudyovercastovercastovercastovercastpartly cloudypart...

'WinterSpringSpringSpringWinterSummerWinterWinterWinterWinterSpringAutumnAutumnWinterWinterSu

'inlandinlandmountaincoastalmountaininlandinlandinlandmountaincoastalmountaininlandmountainin
```

Next steps: Explain error

```
df.min()
```

```
                              0
Temperature                 -25
Humidity                     20
Wind Speed                  0.0
Precipitation (%)             0
Cloud Cover               clear
Atmospheric Pressure     800.12
UV Index                      0
Season                   Autumn
Visibility (km)             0.0
Location                coastal
Weather Type             Cloudy

dtype: object
```

```
df.max()
```

```
                                   0
Temperature                      109
Humidity                         109
Wind Speed                      48.5
Precipitation (%)                109
Cloud Cover            partly cloudy
Atmospheric Pressure         1199.21
UV Index                          14
Season                        Winter
Visibility (km)                 20.0
Location                    mountain
Weather Type                   Sunny

dtype: object
```

```
df.isnull()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibi (
0	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	
13195	False	False	False	False	False	False	False	False	F
13196	False	False	False	False	False	False	False	False	F
13197	False	False	False	False	False	False	False	False	F
13198	False	False	False	False	False	False	False	False	F
13199	False	False	False	False	False	False	False	False	F

13200 rows × 11 columns

```
df.notnull()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibi (
0	True	True	True	True	True	True	True	True	
1	True	True	True	True	True	True	True	True	
2	True	True	True	True	True	True	True	True	
3	True	True	True	True	True	True	True	True	
4	True	True	True	True	True	True	True	True	
...	...	...	...	...	...	...	...	...	
13195	True	True	True	True	True	True	True	True	
13196	True	True	True	True	True	True	True	True	
13197	True	True	True	True	True	True	True	True	
13198	True	True	True	True	True	True	True	True	
13199	True	True	True	True	True	True	True	True	

13200 rows × 11 columns

```
df.dropna()
```

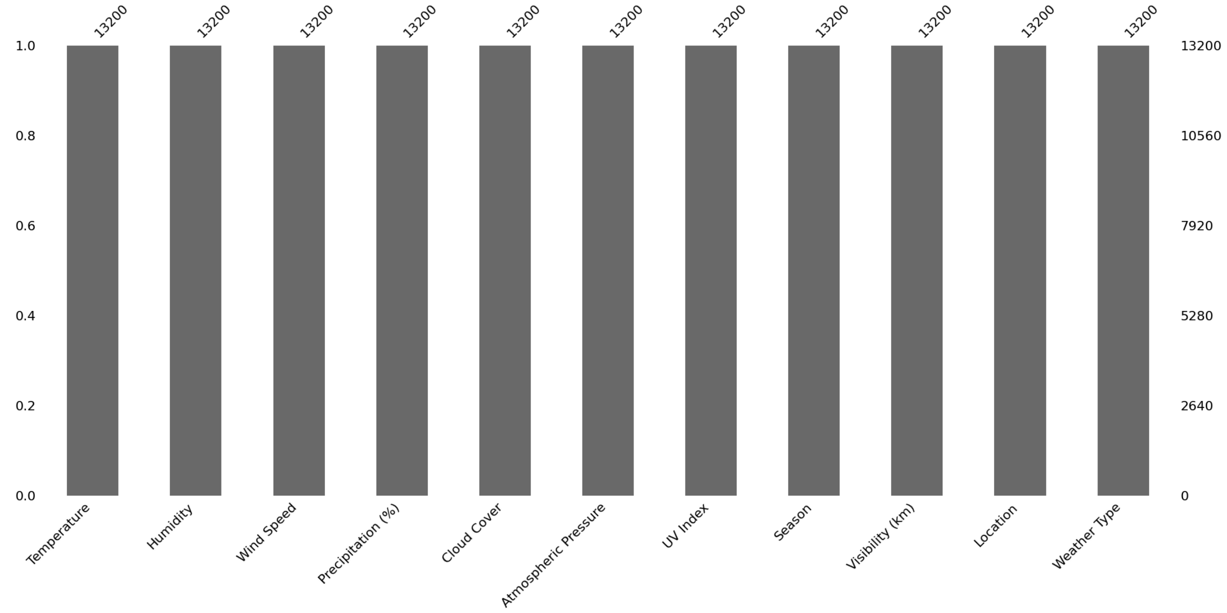
	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visi
0	14	73	9.5	82	partly cloudy	1010.82	2	Winter	
1	39	96	8.5	71	partly cloudy	1011.43	7	Spring	
2	30	64	7.0	16	clear	1018.72	5	Spring	
3	38	83	1.5	82	clear	1026.25	7	Spring	
4	27	74	17.0	66	overcast	990.67	1	Winter	
...	...	...	...	...	...	...	...	...	
13195	10	74	14.5	71	overcast	1003.15	1	Summer	
13196	-1	76	3.5	23	cloudy	1067.23	1	Winter	
13197	30	77	5.5	28	overcast	1012.69	3	Autumn	
13198	3	76	10.0	94	overcast	984.27	0	Winter	
13199	-5	38	0.0	92	overcast	1015.37	5	Autumn	

13200 rows × 11 columns

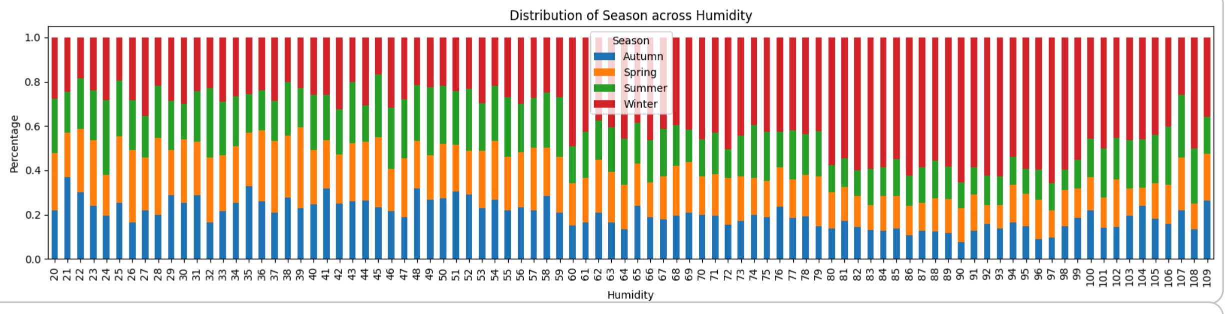
```
table = PrettyTable()
table.field_names = ['Feature', 'Data Type']

for column in df.columns:
    column_dtype = str(df[column].dtype)
    table.add_row([column, column_dtype])

print(table)
```

```
+----------------------+-----------+
|       Feature        | Data Type |
+----------------------+-----------+
|     Temperature      |   int64   |
|       Humidity       |   int64   |
|      Wind Speed      |  float64  |
|  Precipitation (%)   |   int64   |
|     Cloud Cover      |   object  |
| Atmospheric Pressure |  float64  |
|       UV Index       |   int64   |
|        Season        |   object  |
|    Visibility (km)   |  float64  |
|       Location       |   object  |
|     Weather Type     |   object  |
+----------------------+-----------+
```

```
msno.bar(df)

plt.show()
```

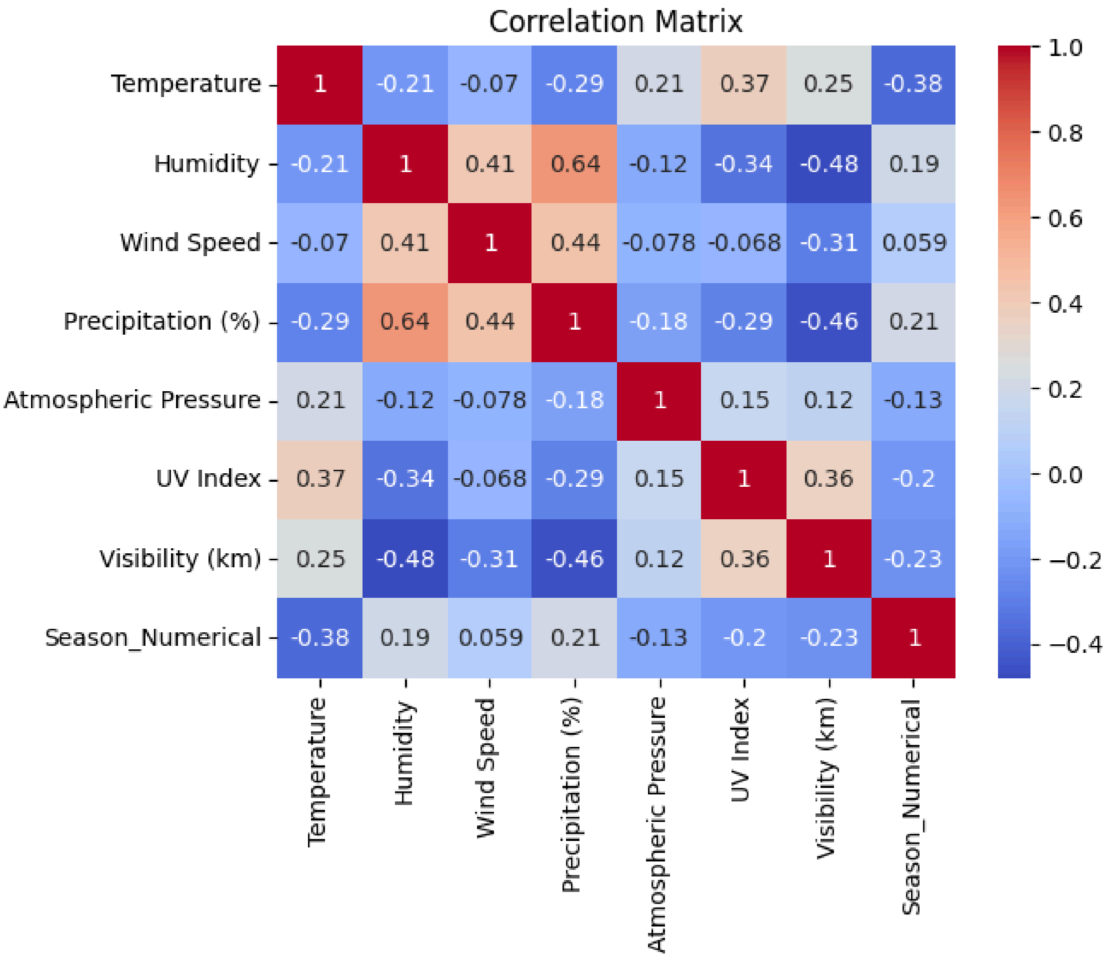


```
def distribution_of_target(target, dataframe):
    cat_cols = [feature
                for feature in dataframe.columns
                if (dataframe[feature].dtype != 'O' and dataframe[feature].nunique() <100)
                or (dataframe[feature].dtype == 'O' and feature not in [target])
                ]

    for column in cat_cols:
        contingency_table = pd.crosstab(dataframe[column], dataframe[target], normalize='inc
        contingency_table.plot(kind="bar", stacked=True, figsize=(20, 4))

        plt.title(f"Distribution of {target} across {column}")
        plt.xlabel(column)
        plt.ylabel("Percentage")
        plt.legend(title=target)

        plt.show()

distribution_of_target("Season", df)
```



```
#correlation matrix
```

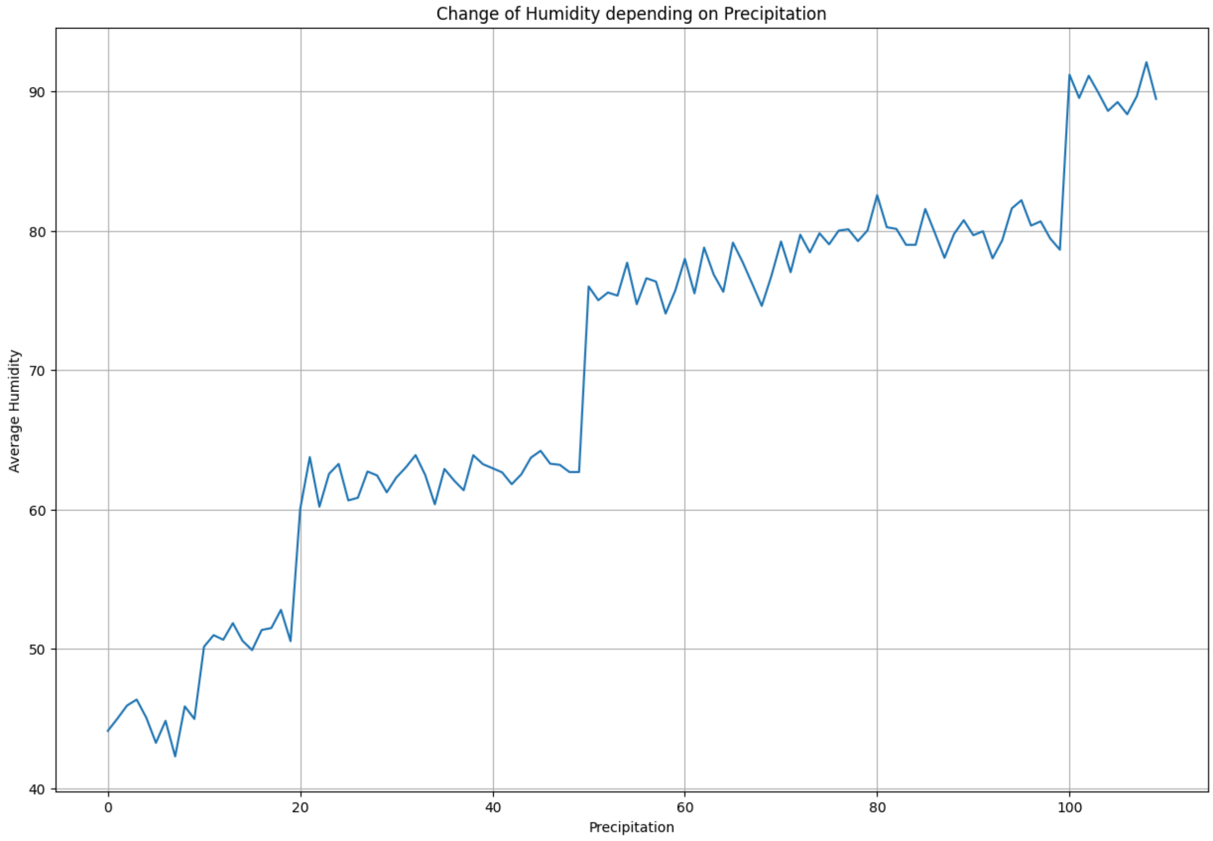
```
label_encoder = LabelEncoder()
df["Season_Numerical"] = label_encoder.fit_transform(df["Season"])
numerical_df = df.select_dtypes(include=["int", "float"])
corr_matrix = numerical_df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix

	Temperature	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	Visibility (km)	Season_Numerical
Temperature	1	-0.21	-0.07	-0.29	0.21	0.37	0.25	-0.38
Humidity	-0.21	1	0.41	0.64	-0.12	-0.34	-0.48	0.19
Wind Speed	-0.07	0.41	1	0.44	-0.078	-0.068	-0.31	0.059
Precipitation (%)	-0.29	0.64	0.44	1	-0.18	-0.29	-0.46	0.21
Atmospheric Pressure	0.21	-0.12	-0.078	-0.18	1	0.15	0.12	-0.13
UV Index	0.37	-0.34	-0.068	-0.29	0.15	1	0.36	-0.2
Visibility (km)	0.25	-0.48	-0.31	-0.46	0.12	0.36	1	-0.23
Season_Numerical	-0.38	0.19	0.059	0.21	-0.13	-0.2	-0.23	1

```
#relationships like change of humiditydepending on various factors like prcipitation
```

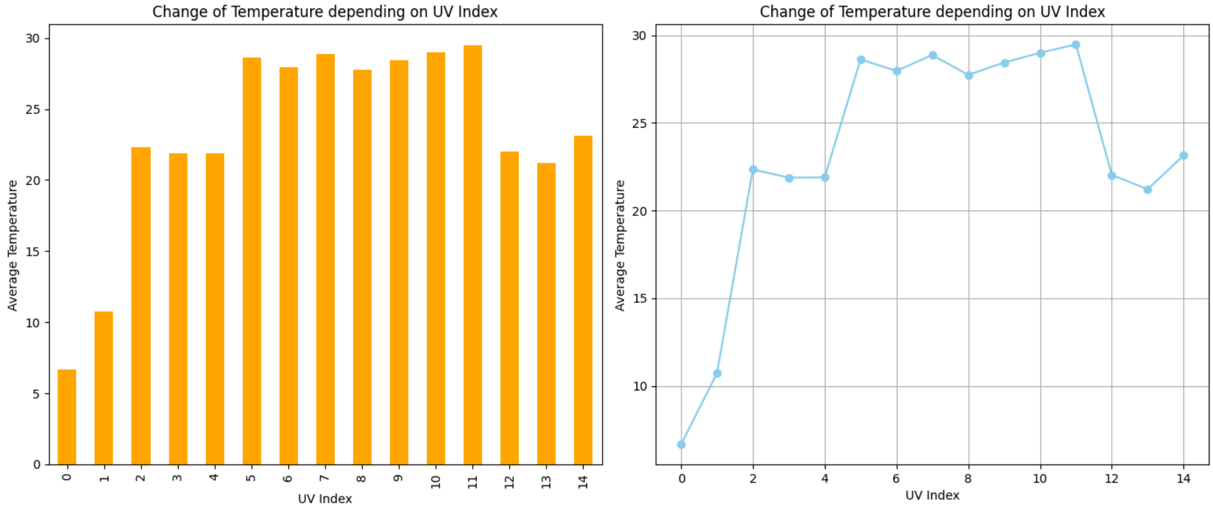
```
plt.figure(figsize=(15, 10))

precipitation_on_humidity = df.groupby("Precipitation (%)")["Humidity"].mean()

precipitation_on_humidity.plot(kind="line")

plt.title('Change of Humidity depending on Precipitation')
plt.xlabel('Precipitation')
plt.ylabel('Average Humidity')
plt.grid(True)

plt.show()
```



```
#change of temperarute based on uv index
```

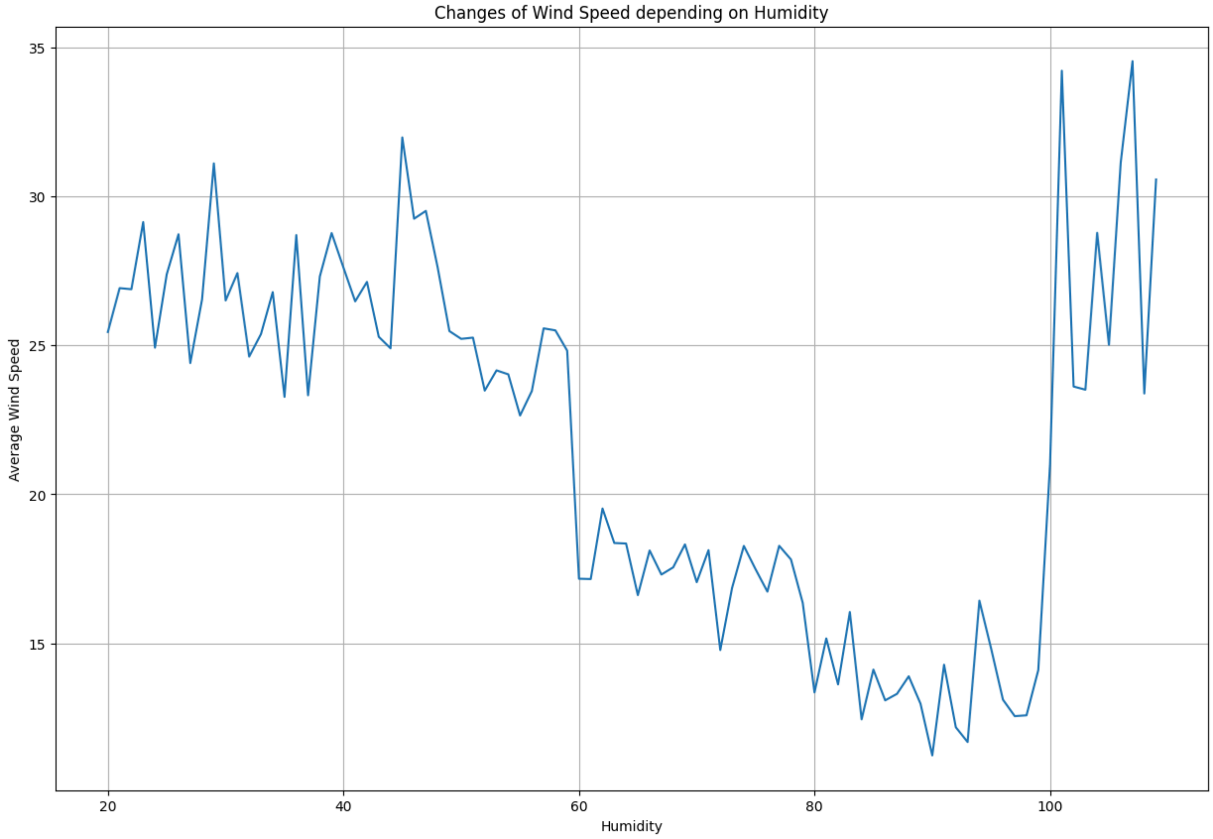
```
plt.figure(figsize=(15, 10))

temperature_on_uv = df.groupby("UV Index")["Temperature"].mean()

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Bar Chart
temperature_on_uv.plot(kind='bar', ax=axes[0], color='orange')
axes[0].set_title('Change of Temperature depending on UV Index')
axes[0].set_xlabel('UV Index')
axes[0].set_ylabel('Average Temperature')

# Line Chart
temperature_on_uv.plot(kind='line', ax=axes[1], color='skyblue', marker='o')
axes[1].set_title('Change of Temperature depending on UV Index')
axes[1].set_xlabel('UV Index')
axes[1].set_ylabel('Average Temperature')
axes[1].grid(True)

plt.tight_layout()
plt.show()
```

```
<Figure size 1500x1000 with 0 Axes>
```



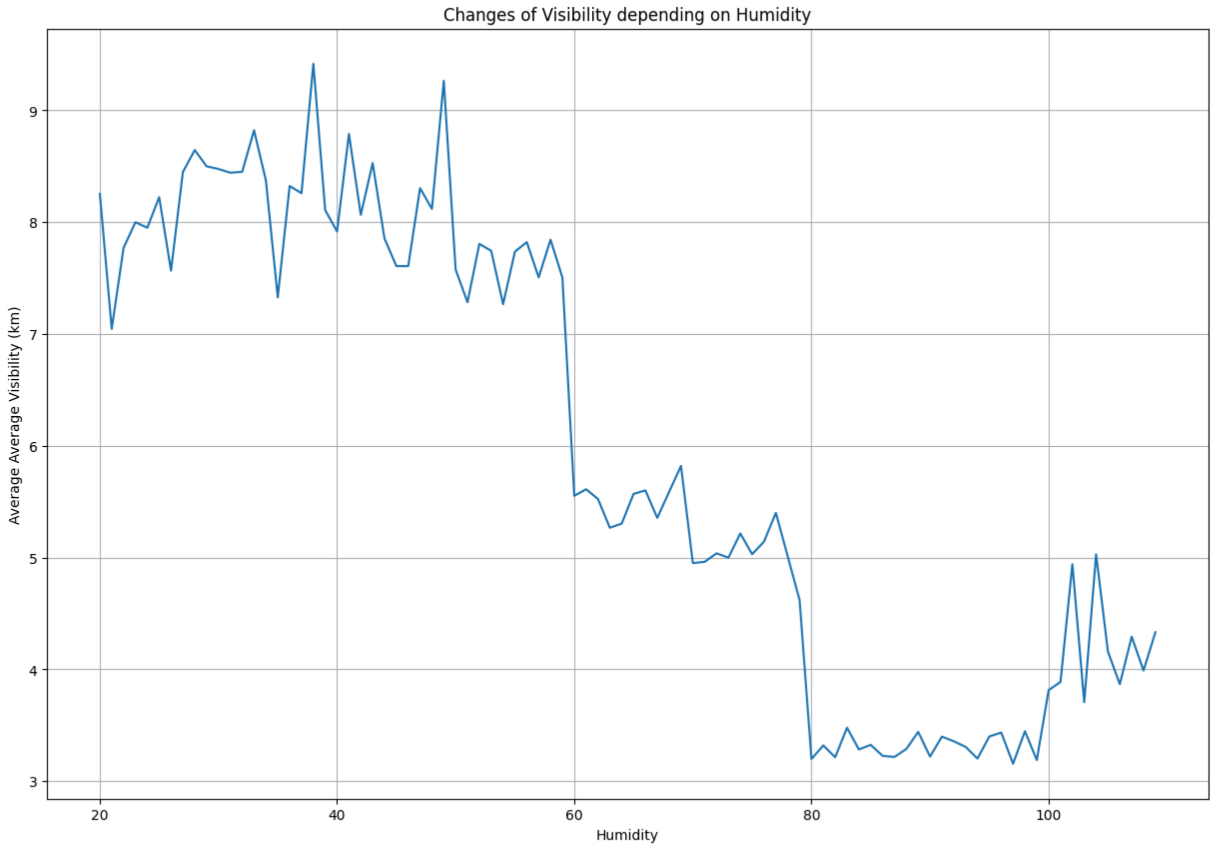
```
#change od wind speed depending on humidity
```

```
plt.figure(figsize=(15, 10))

wind_on_humidity = df.groupby("Humidity")["Temperature"].mean()

wind_on_humidity.plot(kind="line")

plt.title('Changes of Wind Speed depending on Humidity')
plt.xlabel('Humidity')
plt.ylabel('Average Wind Speed')
plt.grid(True)

plt.show()
```



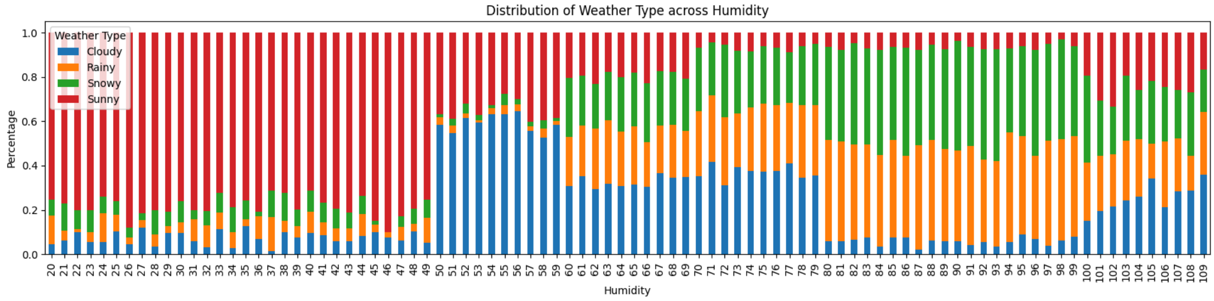
```
#Change of visibility based on humidity
```

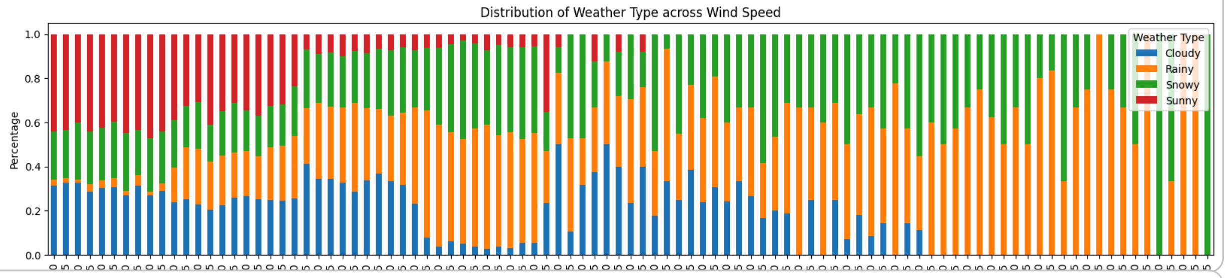
```
plt.figure(figsize=(15, 10))

visibility_on_humidity = df.groupby("Humidity")["Visibility (km)"].mean()

visibility_on_humidity.plot(kind="line")

plt.title('Changes of Visibility depending on Humidity')
plt.xlabel('Humidity')
plt.ylabel('Average Average Visibility (km)')
plt.grid(True)

plt.show()
```



```
#distribution of weather type over all columns
```

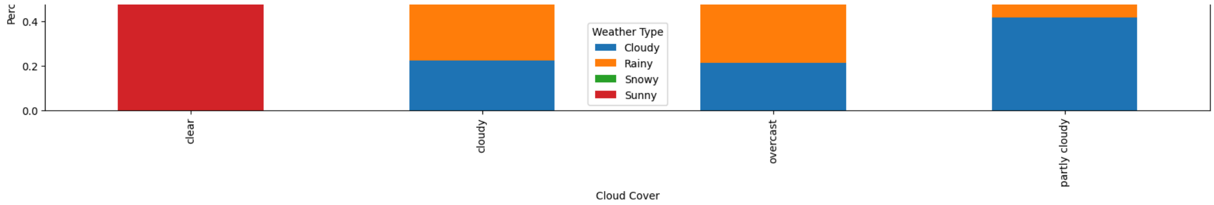
```
distribution_of_target("Weather Type", df)
```

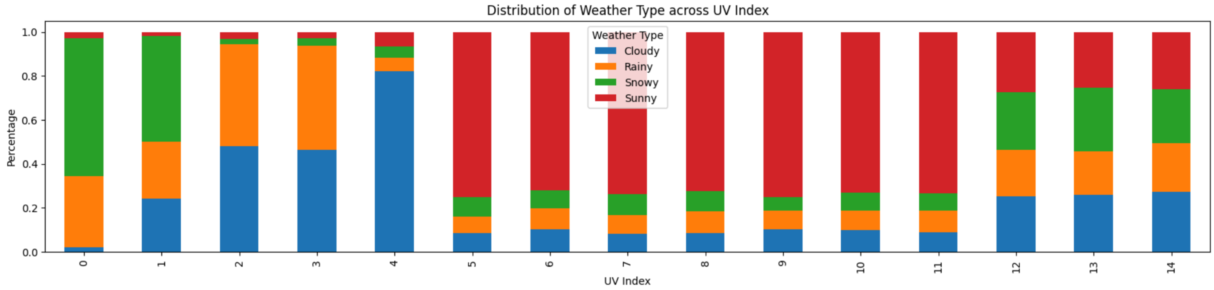


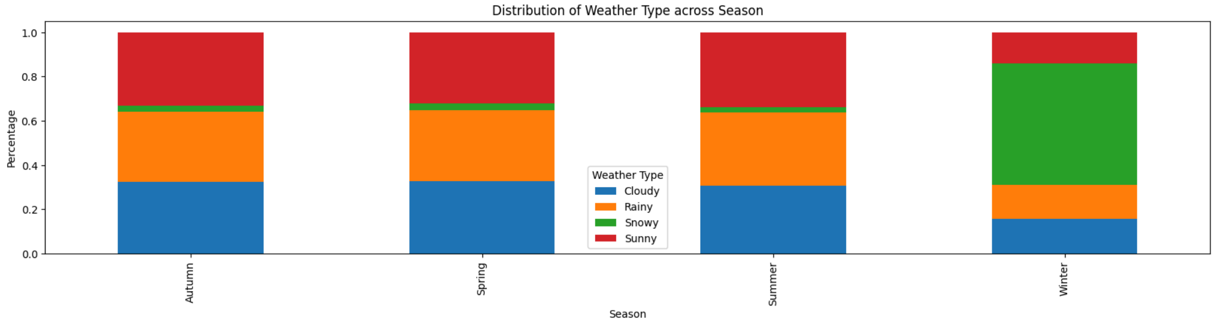


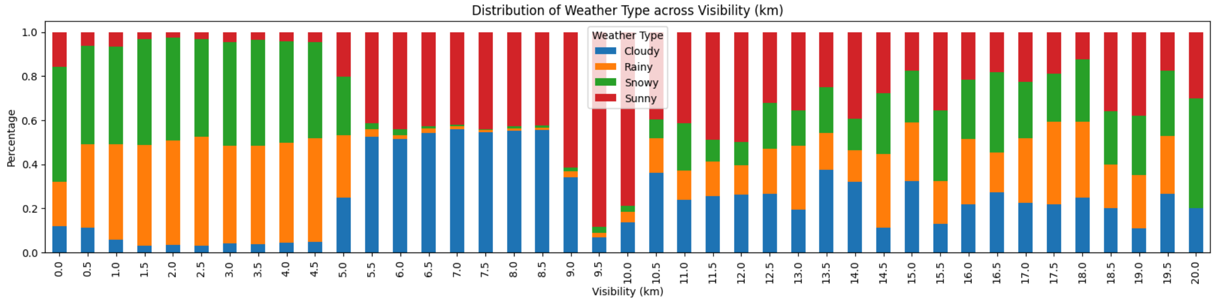
#Distribution of cloud cover over all coulmns

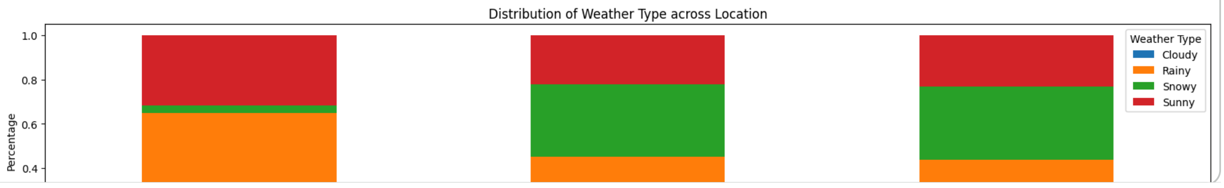
distribution_of_target("Cloud Cover", df)

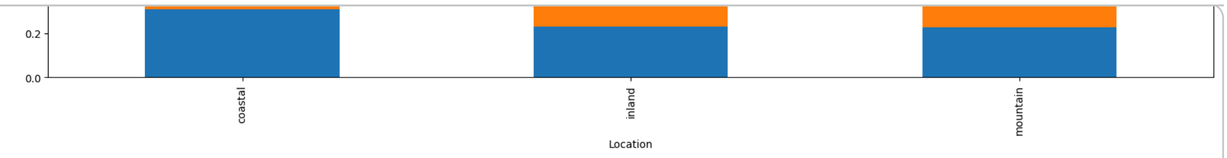


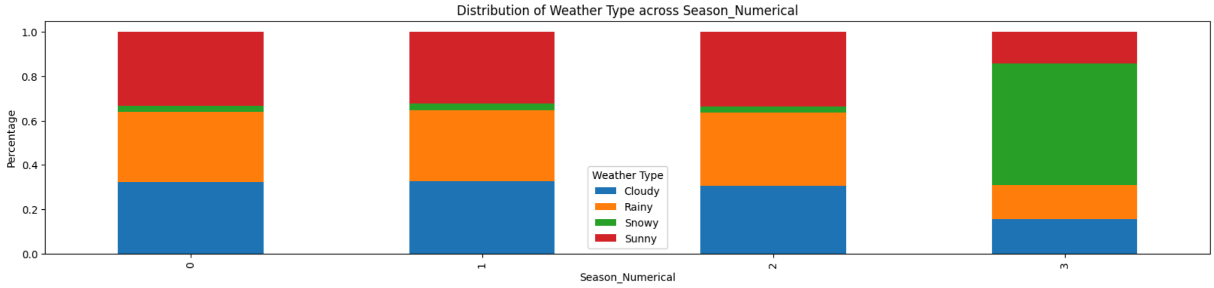


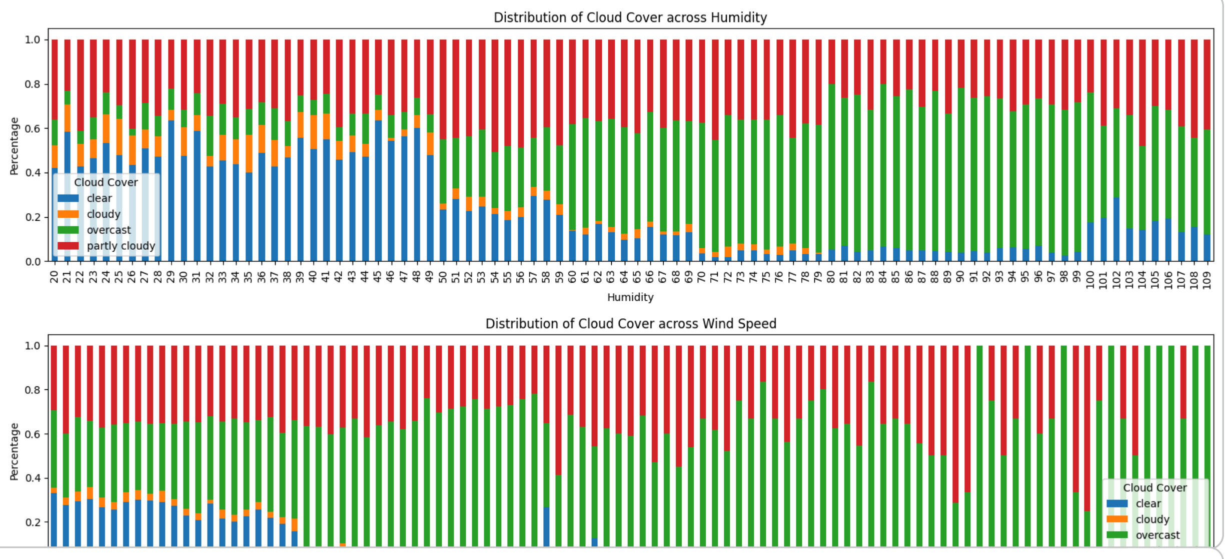






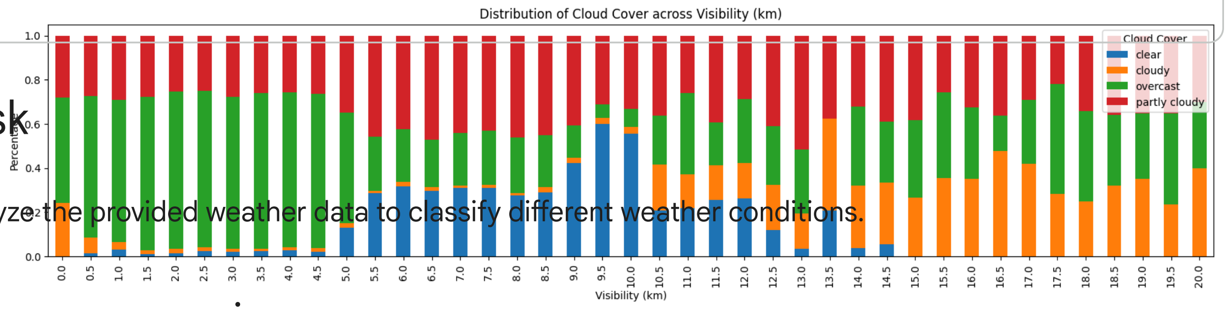



Distribution of Weather Type across Season_Numerical



```
df['Temp_Humidity_Interaction'] = df['Temperature'] * df['Humidity']
df['Wind_Speed_Squared'] = df['Wind Speed']**2
# Using the Steadman's apparent temperature formula for "feels like" temperature
# Formula: AT = 0.885 * Temperature - 22.4 + (1.20 * Humidity + 0.13 * Temperature) * 0.094
# The formula is simplified and may not be perfectly accurate for all conditions.
df['Feels_Like_Temp'] = 0.885 * df['Temperature'] - 22.4 + (1.20 * df['Humidity'] + 0.13 * c
display(df.head())
```

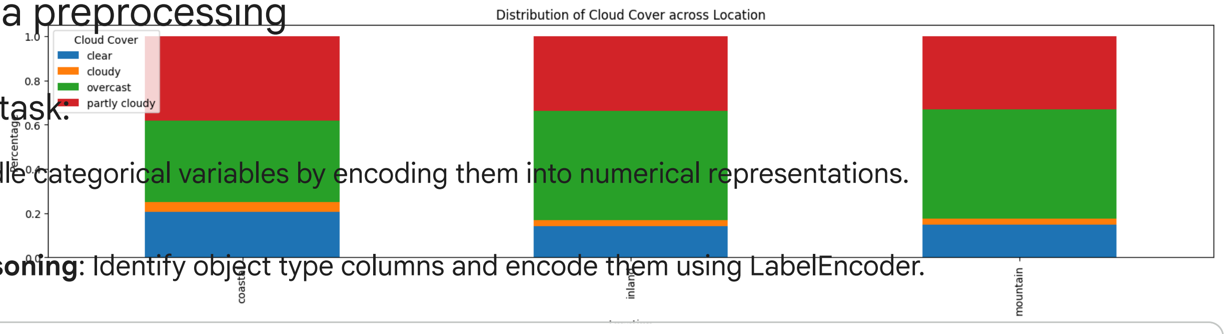
	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibilit (km
0	14	73	9.5	82	partly cloudy	1010.82	2	Winter	3
1	39	96	8.5	71	partly cloudy	1011.43	7	Spring	10
2	30	64	7.0	16	clear	1018.72	5	Spring	5
3	38	83	1.5	82	clear	1026.25	7	Spring	1
4	27	74	17.0	66	overcast	990.67	1	Winter	2



Task

Analyze the provided weather data to classify different weather conditions.

Data preprocessing

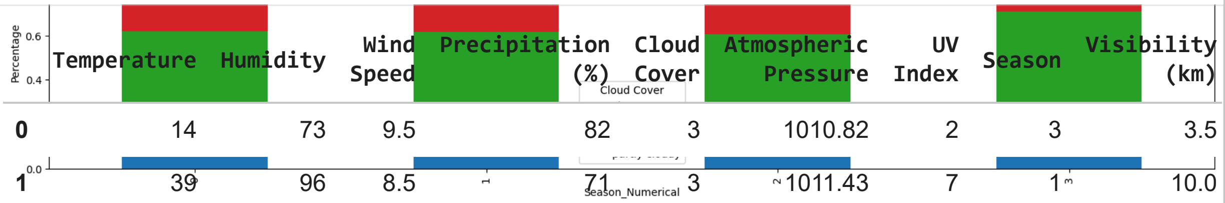


Subtask:

Handle categorical variables by encoding them into numerical representations.

Reasoning: Identify object type columns and encode them using LabelEncoder.

```
for column in df.columns:
    if df[column].dtype == 'object':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])

df.head()
```



	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)
0	14	73	9.5	82	3	1010.82	2	3	3.5
1	39	96	8.5	71	3	1011.43	7	1	10.0
2	30	64	7.0	16	0	1018.72	5	1	5.5
3	38	83	1.5	82	0	1026.25	7	1	1.0
4	27	74	17.0	66	2	990.67	1	3	2.5

Next steps: Generate code with df New interactive sheet

```
for column in df.columns:
    if df[column].dtype == 'object':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])

df.head()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)
0	14	73	9.5	82	3	1010.82	2	3	3.5
1	39	96	8.5	71	3	1011.43	7	1	10.0
2	30	64	7.0	16	0	1018.72	5	1	5.5
3	38	83	1.5	82	0	1026.25	7	1	1.0
4	27	74	17.0	66	2	990.67	1	3	2.5

Next steps: Generate code with df New interactive sheet

Feature engineering

Subtask:

Create new features that might be useful for analysis or modeling.

Reasoning: Create new features based on the existing columns as instructed.

```
df['Temp_Humidity_Interaction'] = df['Temperature'] * df['Humidity']
df['Wind_Speed_Squared'] = df['Wind Speed']**2
# Using the Steadman's apparent temperature formula for "feels like" temperature
# Formula: AT = 0.885 * Temperature - 22.4 + (1.20 * Humidity + 0.13 * Temperature) * 0.094
# The formula is simplified and may not be perfectly accurate for all conditions.
df['Feels_Like_Temp'] = 0.885 * df['Temperature'] - 22.4 + (1.20 * df['Humidity'] + 0.13 * c
display(df.head())
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)
0	14	73	9.5	82	3	1010.82	2	3	3.5
1	39	96	8.5	71	3	1011.43	7	1	10.0
2	30	64	7.0	16	0	1018.72	5	1	5.5
3	38	83	1.5	82	0	1026.25	7	1	1.0
4	27	74	17.0	66	2	990.67	1	3	2.5

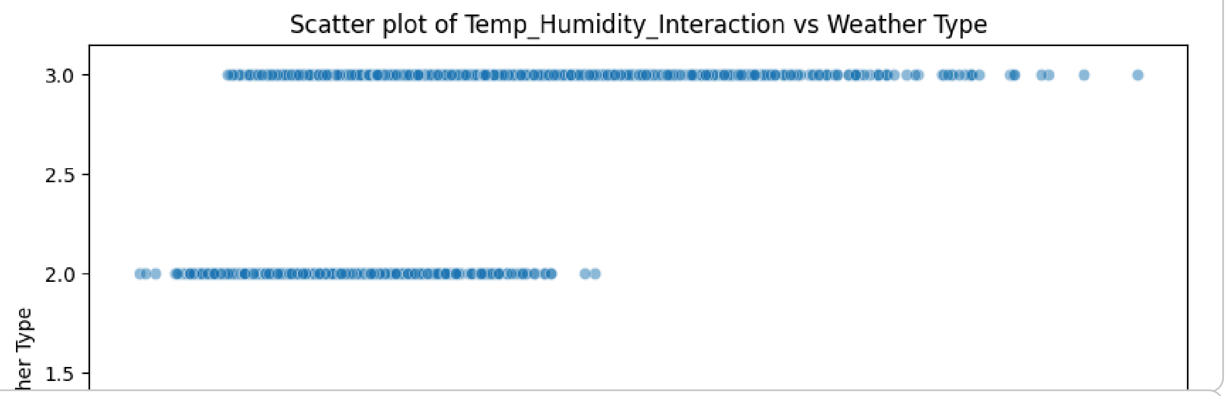
Exploratory data analysis (eda)

Subtask:

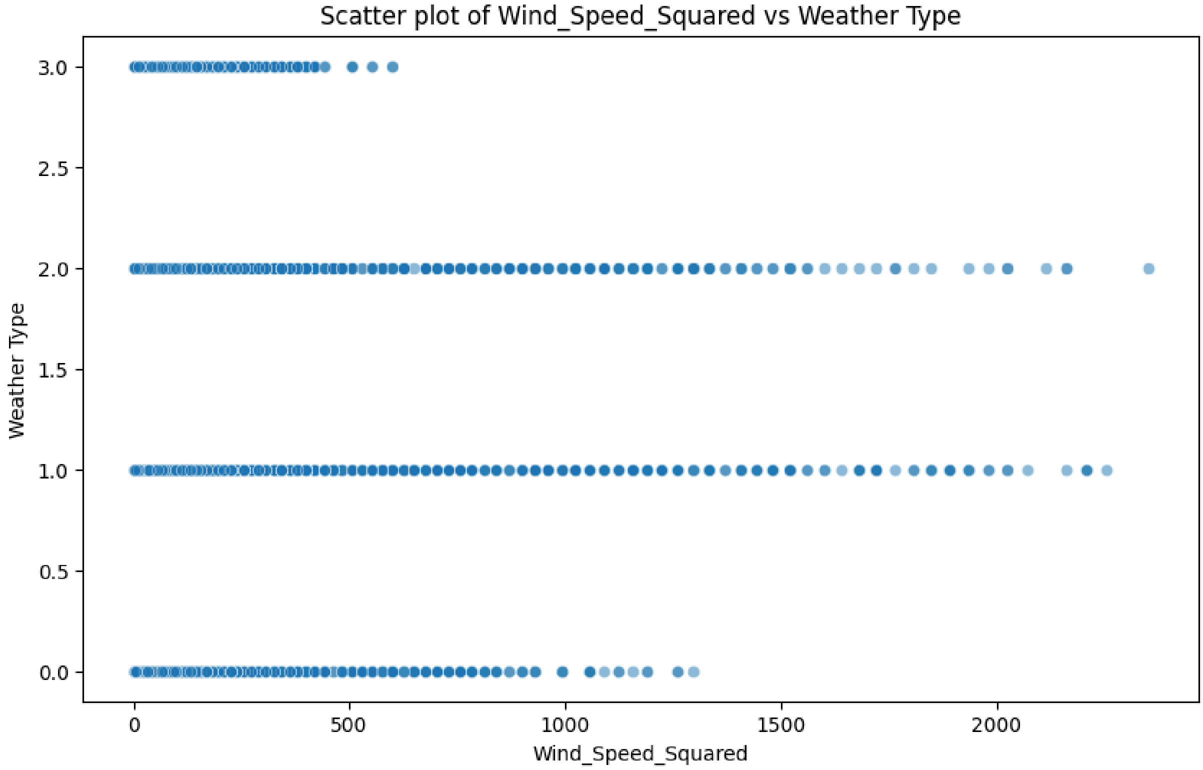
Continue exploring relationships between features and the target variable with visualizations and statistical tests.

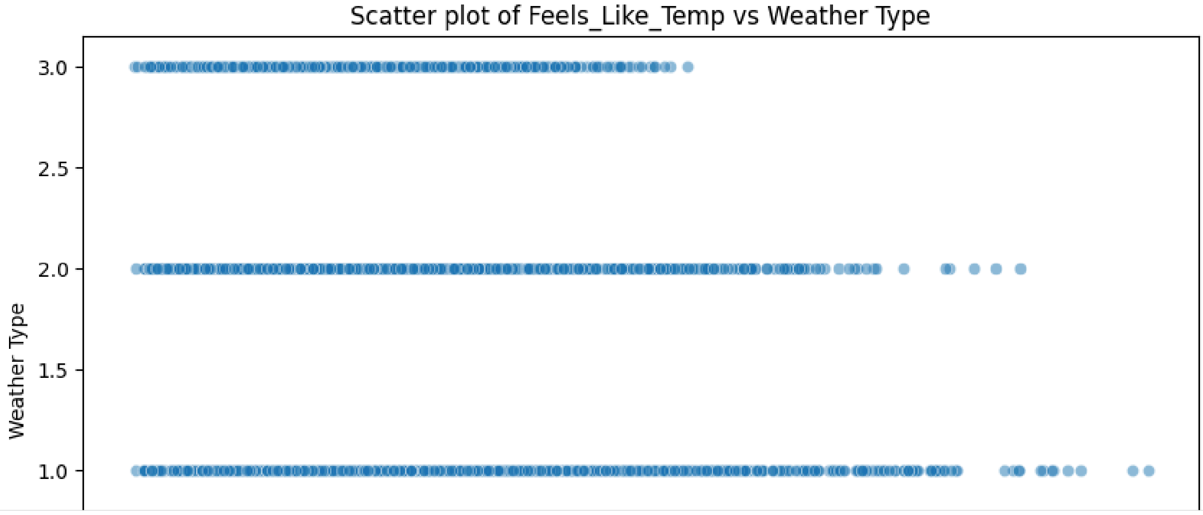
Reasoning: Create scatter plots to visualize the relationship between the newly engineered features and the 'Weather Type' column to identify potential relationships.

```
engineered_features = ['Temp_Humidity_Interaction', 'Wind_Speed_Squared', 'Feels_Like_Temp']
target_variable = 'Weather Type'

for feature in engineered_features:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x=feature, y=target_variable, alpha=0.5)
    plt.title(f'Scatter plot of {feature} vs {target_variable}')
    plt.xlabel(feature)
    plt.ylabel(target_variable)
    plt.show()
```



```
engineered_features = ['Temp_Humidity_Interaction', 'Wind_Speed_Squared', 'Feels_Like_Temp']
target_variable = 'Weather Type'

for feature in engineered_features:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x=feature, y=target_variable, alpha=0.5)
    plt.title(f'Scatter plot of {feature} vs {target_variable}')
    plt.xlabel(feature)
    plt.ylabel(target_variable)
    plt.show()
```


Scatter plot of Wind_Speed_Squared vs Weather Type
Weather Type
Wind_Speed_Squared


Scatter plot of Feels_Like_Temp vs Weather Type
Weather Type