A MINI PROJECT REPORT

ON

WEATHER PREDICTION USING TIME SERIES

Submitted By

MISS. KILARI SIRISHA

Regd. Number: 21B91A05F3

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

S. R. K. R ENGINEERING COLLEGE (A)

(Affiliated to JNTU, KAKINADA) BHIMAVARAM-534204

(2024-2025)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

S. R. K.R ENGINEERING COLLEGE BHIMAVARAM

CERTIFICATE

This is to certify that this is a bonafide work on "**Weather Prediction Using Time Series**" and has been submitted by MISS. KILARI SIRISHA(**21B91A05F3**), in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering, during the academic year 2024-2025.

TABLE OF CONTENTS

ABSTRACT

‘Weather prediction using time series’ aims to develop a predictive model for weather forecasting using a time series analysis approach on the Seattle Weather dataset. Weather prediction is a crucial aspect of modern meteorology, affecting various sectors such as agriculture, transportation, and disaster management. Accurate weather forecasting helps in better planning and preparedness, thereby reducing potential risks and optimizing resources.

The dataset includes attributes like date, precipitation, maximum and minimum temperatures, wind, and weather conditions. The project explores various machine learning algorithms including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Gradient Boosting Classifier, and XGBoost Classifier to determine the most accurate model for predicting weather conditions.

The methodology involves comprehensive data pre-processing including handling missing values, removing outliers, and transforming categorical variables. Exploratory data analysis (EDA) is conducted using visualization techniques such as histograms, box plots, and heatmaps to understand the distribution and relationships within the data. The dataset is split into training and testing sets, with 80% of the data used for training and 20% for testing.

Various models are trained and evaluated based on their accuracy. The XGBoost Classifier demonstrates superior performance with an accuracy of over 75%, making it the chosen model for the final predictive analysis. The project concludes with the development and deployment of the predictive model, providing accurate weather forecasts based on the given historical data.

This project highlights the importance of machine learning in enhancing weather prediction accuracy and showcases the potential of advanced algorithms in handling time series data for meteorological applications.

INTRODUCTION ABOUT PROJECT

Introduction

Weather forecasting is an essential aspect of modern life, influencing various sectors such as agriculture, aviation, transportation, and emergency management. Accurate weather predictions can save lives, improve efficiency, and enhance decision-making processes. This project aims to leverage machine learning algorithms to predict weather conditions accurately using the Seattle Weather dataset. By analyzing historical weather data and applying advanced predictive models, the project seeks to enhance the accuracy and reliability of weather forecasts.

Objectives

The primary objectives of this project are:

1. To clean and preprocess the Seattle Weather dataset to ensure high-quality data for analysis.
2. To explore and visualize the dataset to gain insights into weather patterns and relationships among different weather attributes.
3. To implement and evaluate various machine learning algorithms for weather prediction.
4. To identify the most accurate predictive model for forecasting weather conditions in Seattle.
5. To deploy the selected model for practical weather forecasting applications.

Methodology

The methodology for this project involves the following steps:

1. Data Collection: The dataset used in this project is the Seattle Weather dataset, which contains daily weather observations.
2. Data Preprocessing: This step includes cleaning the data, handling missing values, removing outliers, and transforming categorical variables into numerical formats.
3. Exploratory Data Analysis (EDA): Various visualization techniques such as histograms, box plots, and heatmaps are used to explore the data and understand the distribution and relationships among variables.
4. Model Implementation: Several machine learning algorithms are implemented, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Gradient Boosting Classifier, and XGBoost Classifier.
5. Model Evaluation: The performance of each model is evaluated using accuracy metrics, and the best-performing model is selected for final deployment.
6. Model Deployment: The selected model is used to make weather predictions, and its practical applicability is demonstrated.

Expected Outcomes

The expected outcomes of this project are:

1. A cleaned and well-prepared dataset ready for analysis and modeling.
2. Visualizations that provide insights into the weather patterns and relationships within the dataset.
3. A comparative analysis of different machine learning algorithms for weather prediction.
4. Identification of the most accurate predictive model for weather forecasting.
5. Deployment of the selected model for practical weather predictions.

Significance

Accurate weather forecasting is crucial for various sectors, including agriculture, aviation, and emergency management. By improving the accuracy of weather predictions, this project can contribute to better planning, resource management, and risk mitigation. Additionally, the use of machine learning in weather forecasting highlights the potential of data-driven approaches in enhancing traditional meteorological methods.

Background

Weather forecasting has traditionally relied on numerical weather prediction models, which use mathematical equations to simulate atmospheric processes. While these models have proven effective, they can be complemented by machine learning techniques that can analyze vast amounts of historical weather data and identify patterns that may not be evident through traditional methods. Machine learning algorithms, such as KNN, SVM, and XGBoost, have shown promise in various predictive tasks and are increasingly being applied in meteorology to improve the accuracy of weather forecasts.

By combining historical weather data with advanced machine learning algorithms, this project aims to develop a robust predictive model for weather forecasting in Seattle. The insights gained from this project can potentially be extended to other regions, further enhancing the field of meteorology with data-driven approaches.

ALGORITHM USED IN THIS PROJECT

Algorithms Used in This Project

This project employs several machine learning algorithms to predict weather conditions based on historical weather data. Each algorithm offers unique advantages and is evaluated to identify the most accurate predictive model. The algorithms used in this project include:

1. K-Nearest Neighbors (KNN):

- **Description:** KNN is a simple, non-parametric algorithm used for classification and regression. It works by finding the 'k' closest data points (neighbors) to a given input and assigning the most common class among the neighbors.
- **Advantages:** Easy to understand and implement, effective for small datasets, no need for training phase.
- **Disadvantages:** Computationally intensive for large datasets, sensitive to irrelevant features and the choice of 'k'.

2. Support Vector Machine (SVM):

- **Description:** SVM is a powerful classification algorithm that finds the optimal hyperplane separating different classes in the feature space. It uses kernel functions to handle non-linear relationships.
- **Advantages:** Effective in high-dimensional spaces, robust to overfitting, works well with a clear margin of separation.
- **Disadvantages:** Computationally expensive, requires careful tuning of parameters and choice of kernel.

3. Gradient Boosting Classifier:

- **Description:** Gradient Boosting is an ensemble learning technique that builds multiple weak learners (typically decision trees) sequentially. Each new tree corrects the errors made by the previous ones, improving the model's accuracy.
- **Advantages:** High accuracy, handles various types of data, robust to overfitting with proper tuning.
- **Disadvantages:** Computationally intensive, sensitive to noise and overfitting without proper parameter tuning.

4. XGBoost Classifier:

- **Description:** XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting that is designed for speed and performance. It includes regularization to prevent overfitting and can handle missing data internally.
- **Advantages:** High performance, fast execution, handles missing values, regularization to prevent overfitting.
- **Disadvantages:** Requires careful parameter tuning, can be complex to implement and understand.

Why These Algorithms Are Chosen:

1. KNN: Chosen for its simplicity and effectiveness in handling small datasets. It provides a good baseline for comparison with more complex models.
2. SVM: Selected for its ability to handle high-dimensional data and its robustness in classification tasks. It is particularly effective when there is a clear margin of separation between classes.
3. Gradient Boosting Classifier: Used for its high accuracy and ability to improve prediction performance through sequential learning. It is effective in handling various types of data and provides good predictive power.
4. XGBoost Classifier: Chosen for its superior performance and speed. It is an advanced implementation of gradient boosting that includes regularization and can handle large datasets efficiently.

By evaluating these diverse algorithms, the project aims to identify the most accurate and reliable model for weather prediction, leveraging the strengths of each algorithm to achieve optimal results.

REQUIREMENTS

SOFTWARE REQUIREMENTS:

1. **TEXT EDITOR:** Jupyter Notebook, Visual Studio Code
2. **Internet Browser:** Google Chrome, Mozilla Firefox, Internet Browser
3. **Programming Language:** Python 3.7
4. **Libraries: Pandas,** NumPy, Scikit-Learn, XGBoost, Matplotlib, Seaborn

HARDWARE REQUIREMENTS:

1. Processor-Intel Core I5
2. RAM-8GB

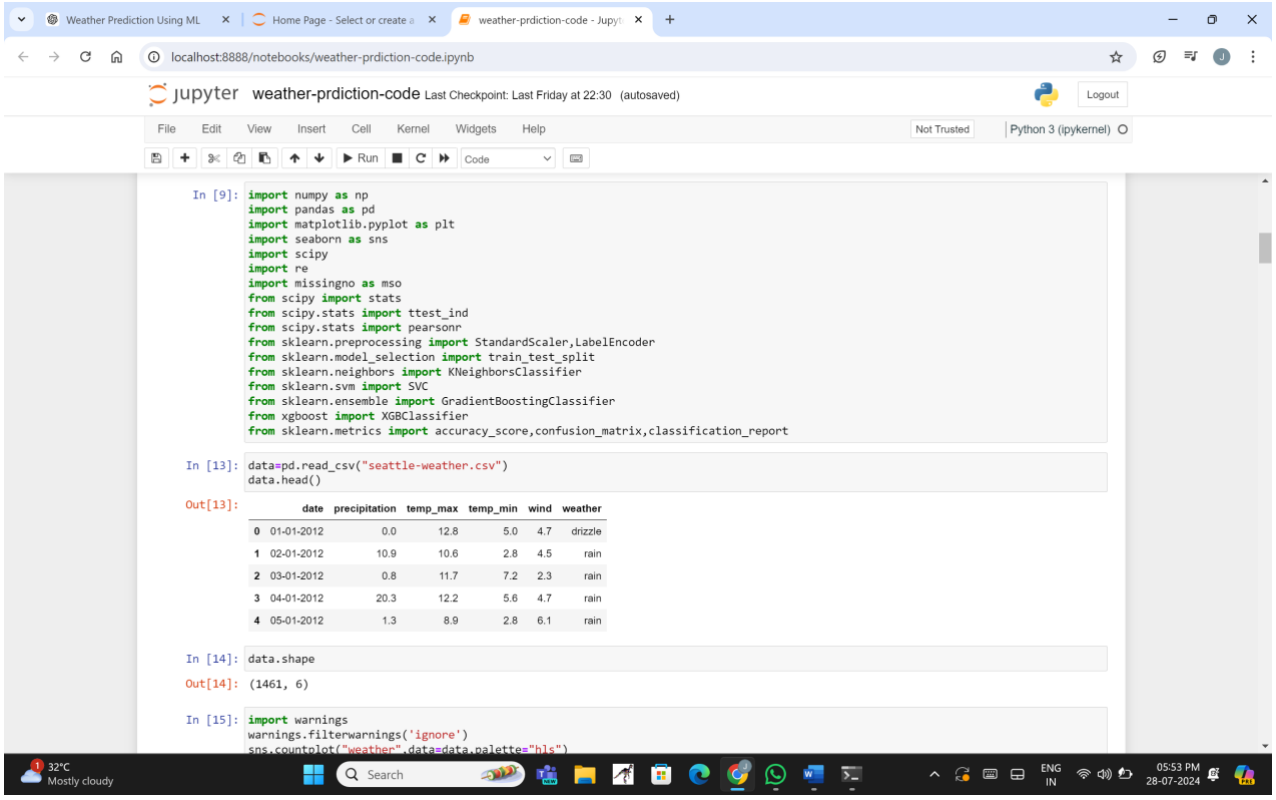
SAMPLE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import re
import missingno as mso
from scipy import stats
from scipy.stats import ttest_ind
from scipy.stats import pearsonr
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

data = pd.read_csv("seattle-weather.csv")
data.head()

data.shape

import warnings
warnings.filterwarnings('ignore')
sns.countplot("weather", data=data, palette="hls")

countrain = len(data[data.weather=="rain"])
countsun = len(data[data.weather=="sun"])
countdrizzle = len(data[data.weather=="drizzle"])
countsnow = len(data[data.weather=="snow"])
countfog = len(data[data.weather=="fog"])
print("Percent of Rain:{:2f}%".format((countrain / (len(data.weather)) * 100)))
print("Percent of Sun:{:2f}%".format((countsun / (len(data.weather)) * 100)))
print("Percent of Drizzle:{:2f}%".format((countdrizzle / (len(data.weather)) * 100)))
print("Percent of Snow:{:2f}%".format((countsnow / (len(data.weather)) * 100)))
print("Percent of Fog:{:2f}%".format((countfog / (len(data.weather)) * 100)))

data[["precipitation", "temp_max", "temp_min", "wind"]].describe()

sns.set(style="darkgrid")
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
sns.histplot(data=data, x="precipitation", kde=True, ax=axs[0, 0], color='green')
sns.histplot(data=data, x="temp_max", kde=True, ax=axs[0, 1], color='red')
sns.histplot(data=data, x="temp_min", kde=True, ax=axs[1, 0], color='skyblue')
sns.histplot(data=data, x="wind", kde=True, ax=axs[1, 1], color='orange')
```

```
sns.set(style="darkgrid")
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
sns.violinplot(data=data, x="precipitation", kde=True, ax=axs[0, 0], color='green')
sns.violinplot(data=data, x="temp_max", kde=True, ax=axs[0, 1], color='red')
sns.violinplot(data=data, x="temp_min", kde=True, ax=axs[1, 0], color='skyblue')
sns.violinplot(data=data, x="wind", kde=True, ax=axs[1, 1], color='yellow')

plt.figure(figsize=(12, 6))
sns.boxplot("precipitation", "weather", data=data, palette="YlOrBr")

plt.figure(figsize=(12, 6))
sns.boxplot("temp_max", "weather", data=data, palette="inferno")

plt.figure(figsize=(12, 6))
sns.boxplot("wind", "weather", data=data, palette="inferno")

plt.figure(figsize=(12, 6))
sns.boxplot("temp_min", "weather", data=data, palette="YlOrBr")

plt.figure(figsize=(12, 7))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')

data.plot("precipitation", "temp_max", style='o')
print("Pearson correlation:", data["precipitation"].corr(data["temp_max"]))
print("T Test and P value:", stats.ttest_ind(data["precipitation"], data["temp_max"]))

data.plot("wind", "temp_max", style='o')
print("Pearson correlation:", data["wind"].corr(data["temp_max"]))
print("T Test and P value:", stats.ttest_ind(data["wind"], data["temp_max"]))

data.plot("temp_max", "temp_min", style='o')
data.isna().sum()

plt.figure(figsize=(12, 6))
axz = plt.subplot(1, 2, 2)
mso.bar(data.drop(["date"], axis=1), ax=axz, fontsize=12)

df = data.drop(["date"], axis=1)

Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
df = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]

sns.set(style="darkgrid")
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
sns.histplot(data=df, x="precipitation", kde=True, ax=axs[0, 0], color='green')
sns.histplot(data=df, x="temp_max", kde=True, ax=axs[0, 1], color='red')
```

```
sns.histplot(data=df, x="temp_min", kde=True, ax=axs[1, 0], color='skyblue')
sns.histplot(data=df, x="wind", kde=True, ax=axs[1, 1], color='orange')

df.head()

lc = LabelEncoder()
df["weather"] = lc.fit_transform(df["weather"])

df.head()

x = ((df.loc[:, df.columns != "weather"]).astype(int)).values[:, 0:]
y = df["weather"].values

df.weather.unique()

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=2)

knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
print("KNN Accuracy:{:.2f}%".format(knn.score(x_test, y_test) * 100))

svm = SVC()
svm.fit(x_train, y_train)
print("SVM Accuracy:{:.2f}%".format(svm.score(x_test, y_test) * 100))

gbc = GradientBoostingClassifier(subsample=0.5, n_estimators=450, max_depth=5,
max_leaf_nodes=25)
gbc.fit(x_train, y_train)
print("Gradient Boosting Accuracy:{:.2f}%".format(gbc.score(x_test, y_test) * 100))

import warnings
warnings.filterwarnings('ignore')
xgb = XGBClassifier()
xgb.fit(x_train, y_train)
print("XGB Accuracy:{:.2f}%".format(xgb.score(x_test, y_test) * 100))

input = [[1.140175, 8.9, 2.8, 2.469818]]
ot = xgb.predict(input)
print("The weather is:")
if(ot == 0):
    print("Drizzle")
elif(ot == 1):
    print("Fog")
elif(ot == 2):
    print("Rain")
elif(ot == 3):
    print("Snow")
else:
    print("Sun")
```

SCREENSHOTS

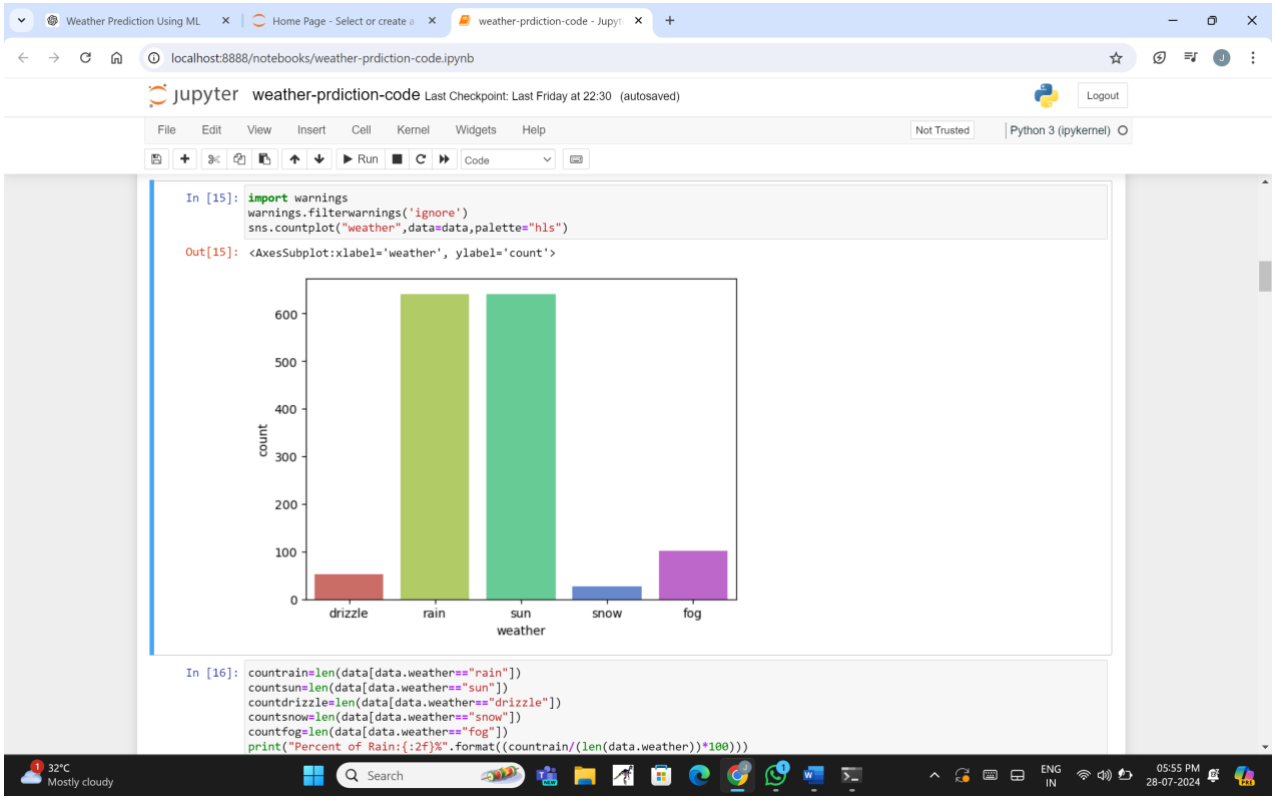


```
In [9]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import scipy
        import re
        import missingno as mso
        from scipy import stats
        from scipy.stats import ttest_ind
        from scipy.stats import pearsonr
        from sklearn.preprocessing import StandardScaler,LabelEncoder
        from sklearn.model_selection import train_test_split
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.svm import SVC
        from sklearn.ensemble import GradientBoostingClassifier
        from xgboost import XGBClassifier
        from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

In [13]: data=pd.read_csv("seattle-weather.csv")
         data.head()
```

Out[13]:

	date	precipitation	temp_max	temp_min	wind	weather
0	01-01-2012	0.0	12.8	5.0	4.7	drizzle
1	02-01-2012	10.9	10.6	2.8	4.5	rain
2	03-01-2012	0.8	11.7	7.2	2.3	rain
3	04-01-2012	20.3	12.2	5.6	4.7	rain
4	05-01-2012	1.3	8.9	2.8	6.1	rain

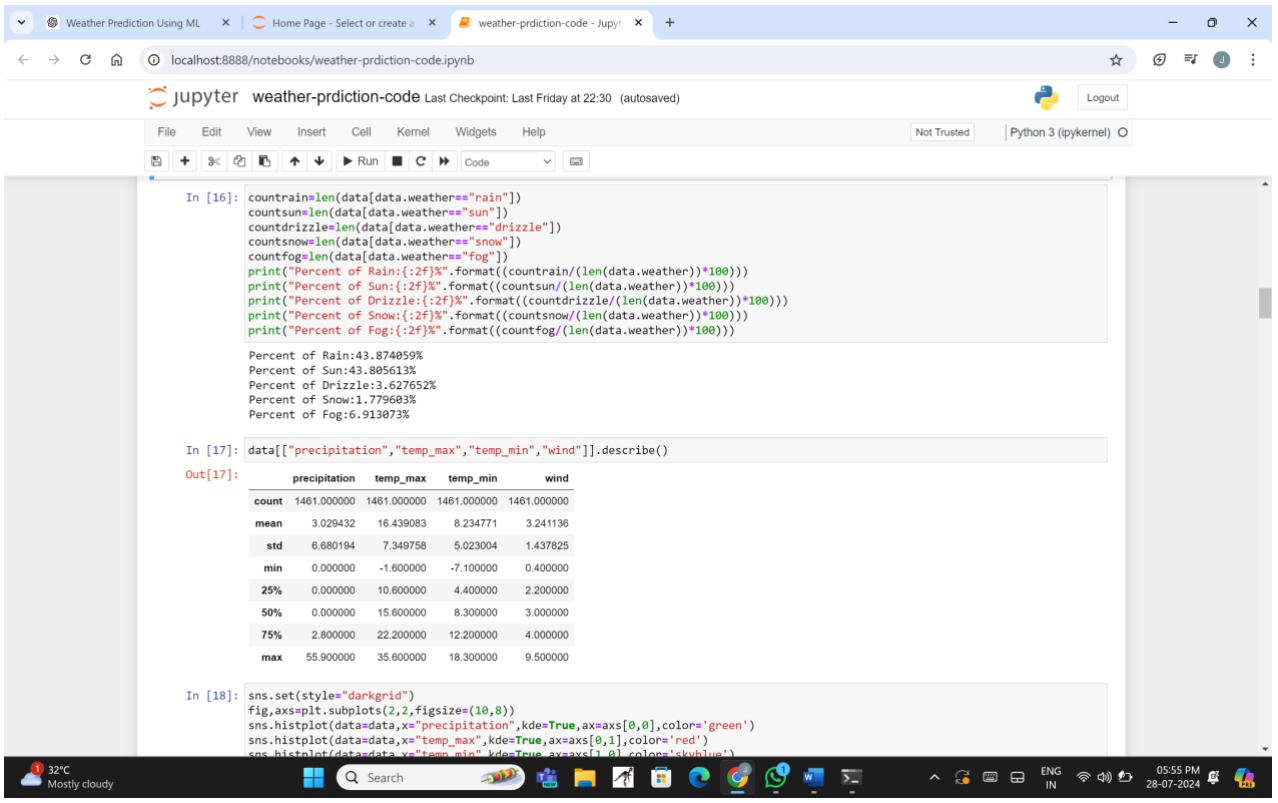
```
In [14]: data.shape
Out[14]: (1461, 6)

In [15]: import warnings
         warnings.filterwarnings('ignore')
         sns.countplot("weather",data=data,palette="hls")
```

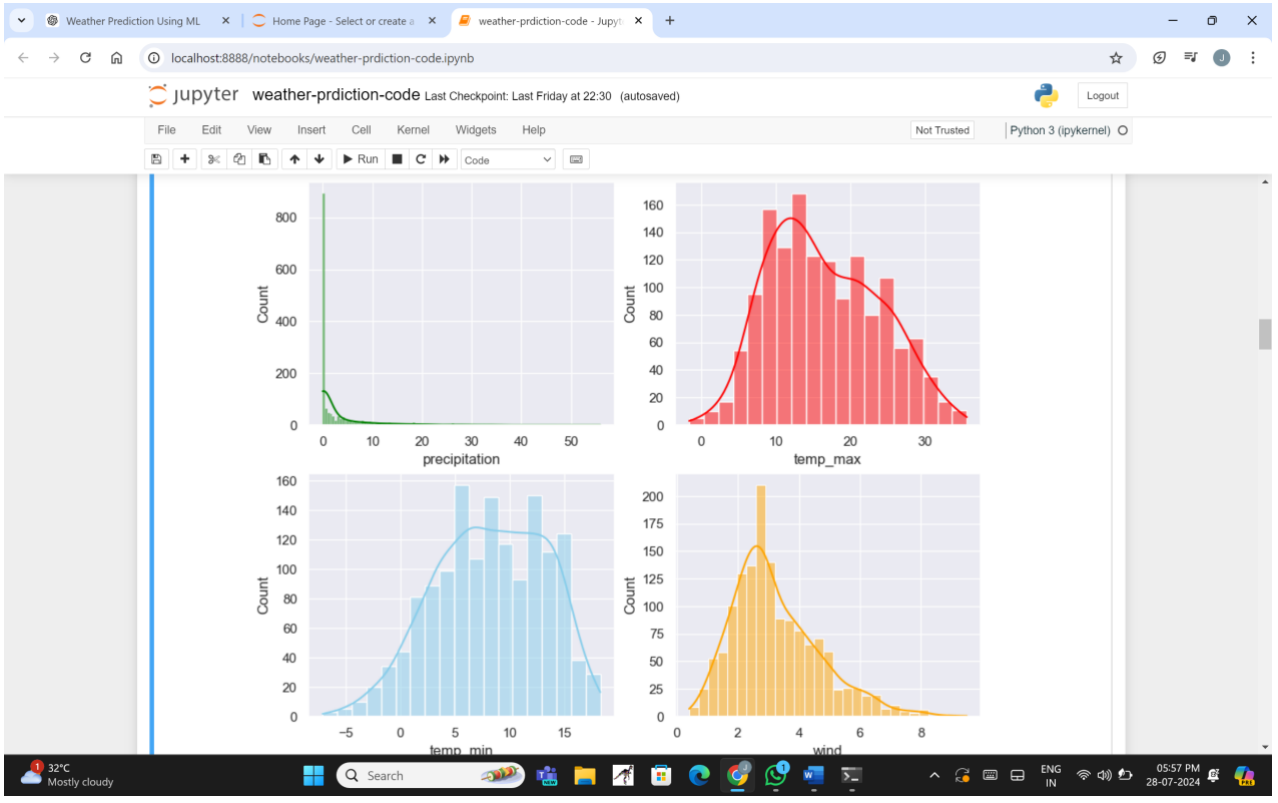


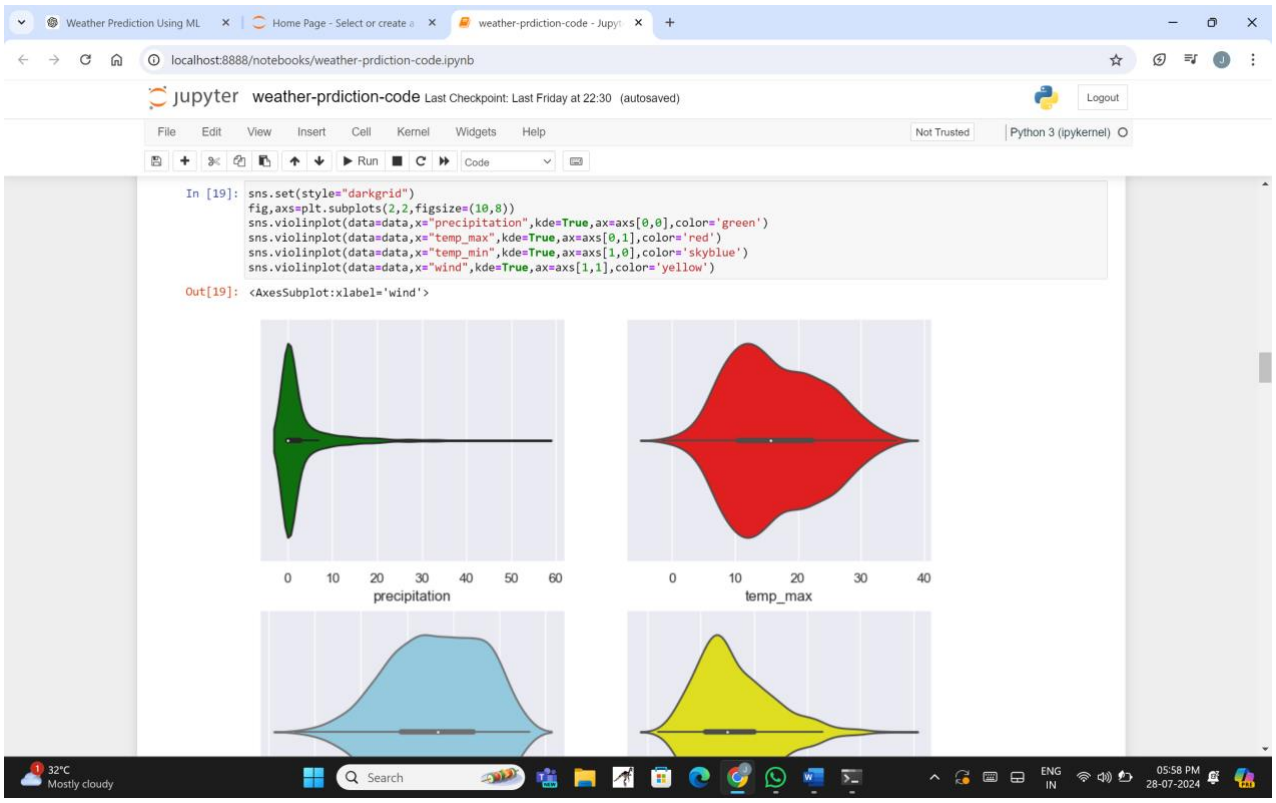
```
In [15]: import warnings
         warnings.filterwarnings('ignore')
         sns.countplot("weather",data=data,palette="hls")
Out[15]: <AxesSubplot:xlabel='weather', ylabel='count'>
```

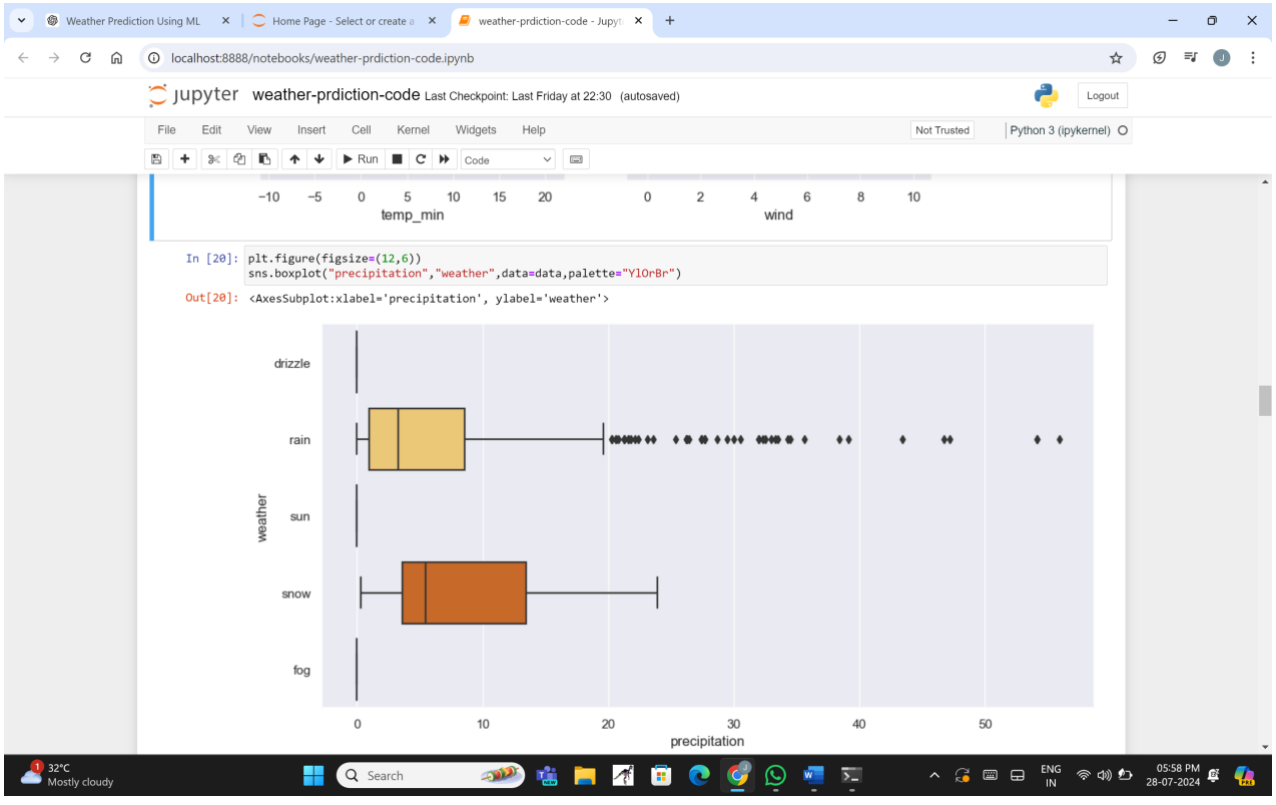


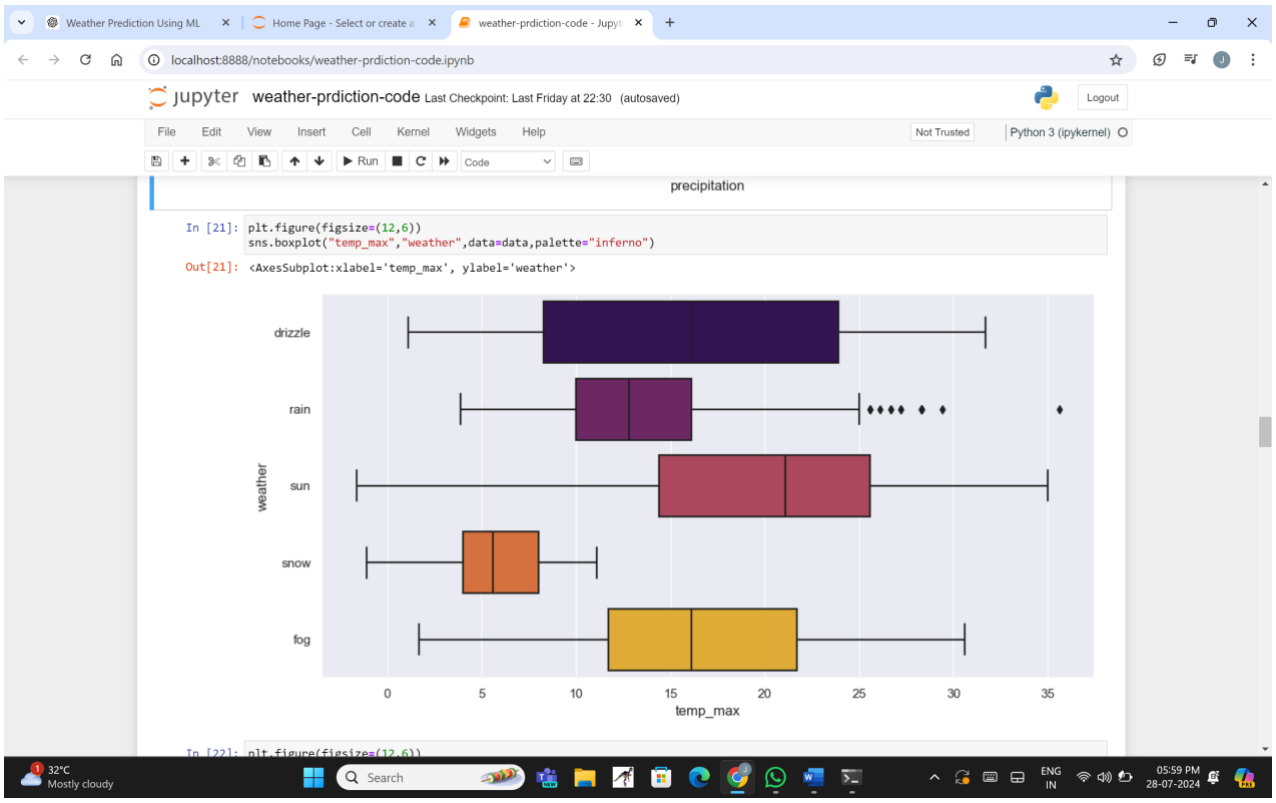
```
In [16]: countrain=len(data[data.weather=="rain"])
         countsun=len(data[data.weather=="sun"])
         countdrizzle=len(data[data.weather=="drizzle"])
         countsnow=len(data[data.weather=="snow"])
         countfog=len(data[data.weather=="fog"])
         print("Percent of Rain:{:2f}%".format((countrain/(len(data.weather))*100)))
```

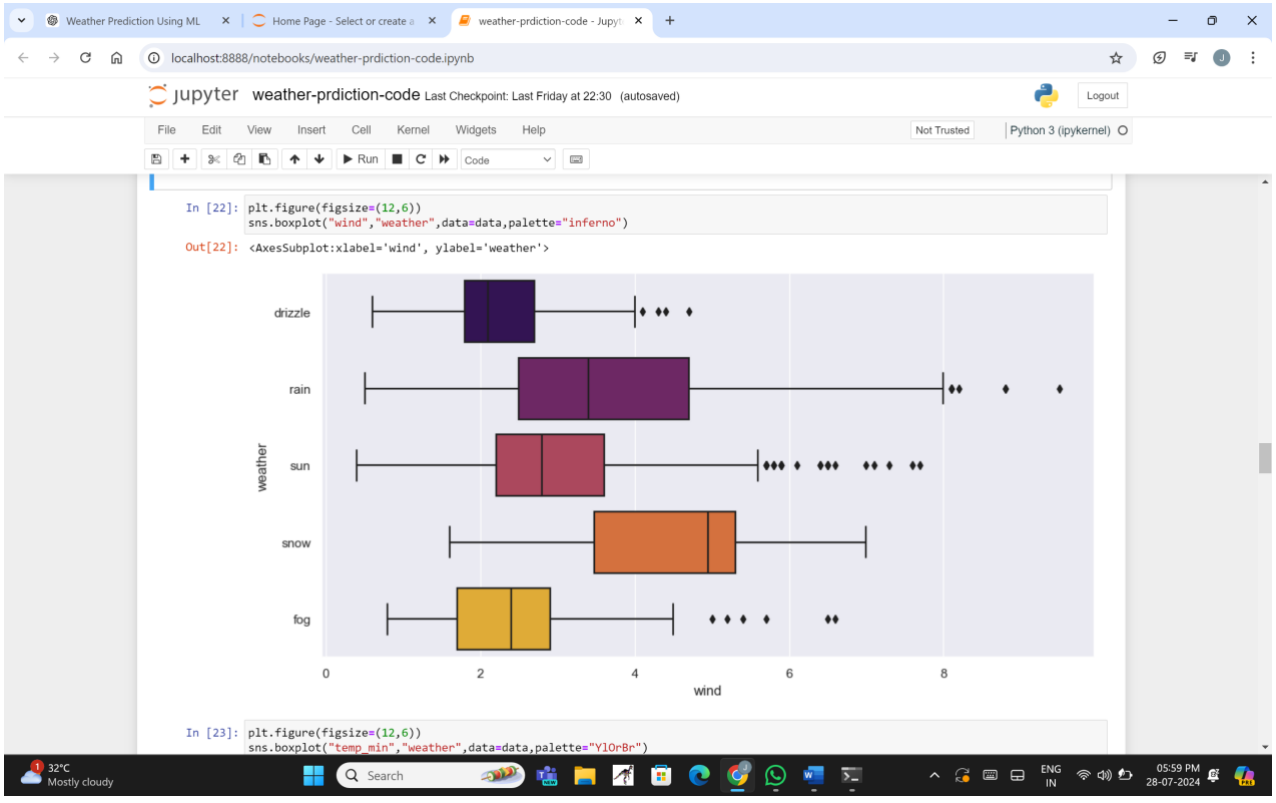

Weather Prediction Using ML
Home Page - Select or create
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [16]:
countrain=len(data[data.weather=="rain"])
countsun=len(data[data.weather=="sun"])
countdrizzle=len(data[data.weather=="drizzle"])
countsnow=len(data[data.weather=="snow"])
countfog=len(data[data.weather=="fog"])
print("Percent of Rain:{:2f}%".format((countrain/(len(data.weather))*100)))
print("Percent of Sun:{:2f}%".format((countsun/(len(data.weather))*100)))
print("Percent of Drizzle:{:2f}%".format((countdrizzle/(len(data.weather))*100)))
print("Percent of Snow:{:2f}%".format((countsnow/(len(data.weather))*100)))
print("Percent of Fog:{:2f}%".format((countfog/(len(data.weather))*100)))
Percent of Rain:43.874059%
Percent of Sun:43.805613%
Percent of Drizzle:3.627652%
Percent of Snow:1.779603%
Percent of Fog:6.913073%
In [17]:
data[["precipitation","temp_max","temp_min","wind"]].describe()
Out[17]:
precipitation temp_max temp_min wind
count 1461.000000 1461.000000 1461.000000 1461.000000
mean 3.029432 16.439083 8.234771 3.241136
std 6.680194 7.349758 5.023004 1.437825
min 0.000000 -1.600000 -7.100000 0.400000
25% 0.000000 10.600000 4.400000 2.200000
50% 0.000000 15.600000 8.300000 3.000000
75% 2.800000 22.200000 12.200000 4.000000
max 55.900000 35.600000 18.300000 9.500000
In [18]:
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=data,x="precipitation",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=data,x="temp_max",kde=True,ax=axs[0,1],color='red')
32°C
Mostly cloudy
Search
ENG
IN
05:55 PM
28-07-2024

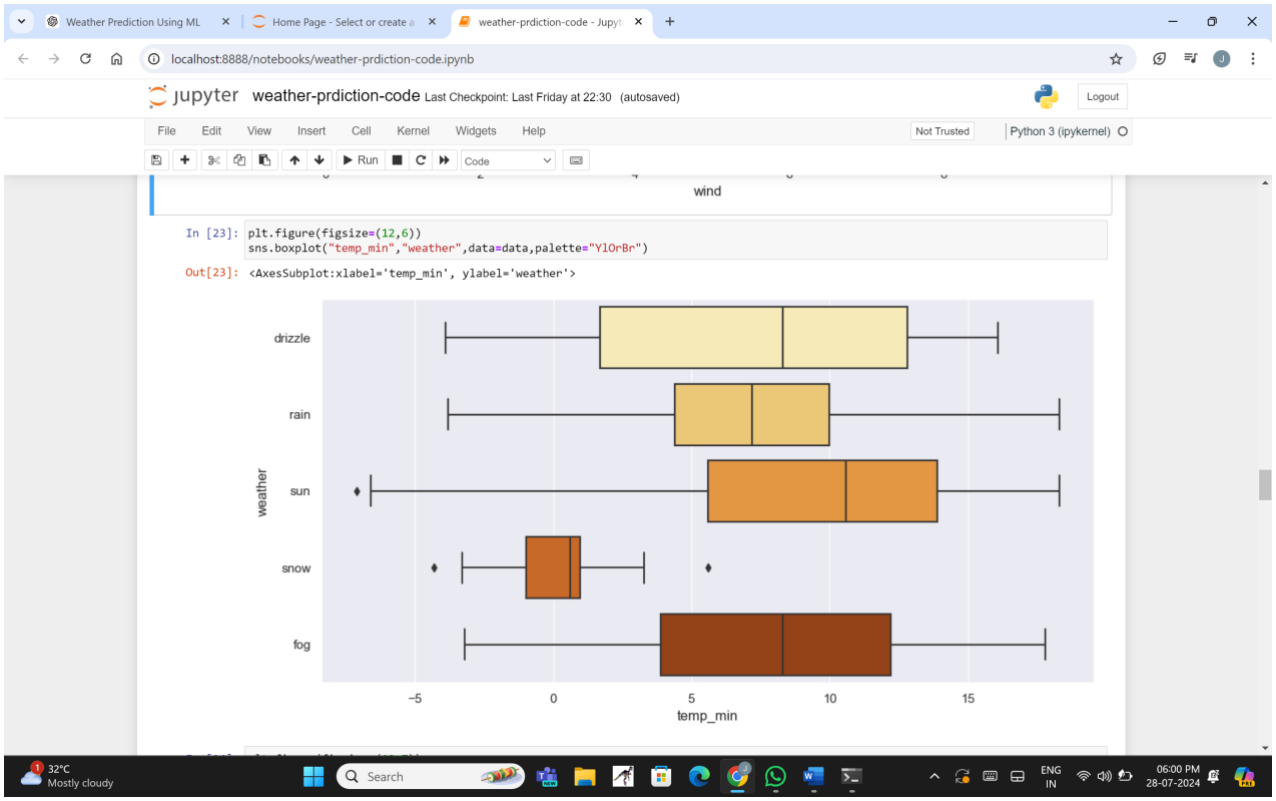

Weather Prediction Using ML
Home Page - Select or create
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Count
precipitation
temp_max
temp_min
wind
32°C
Mostly cloudy
Search
ENG
IN
05:57 PM
28-07-2024

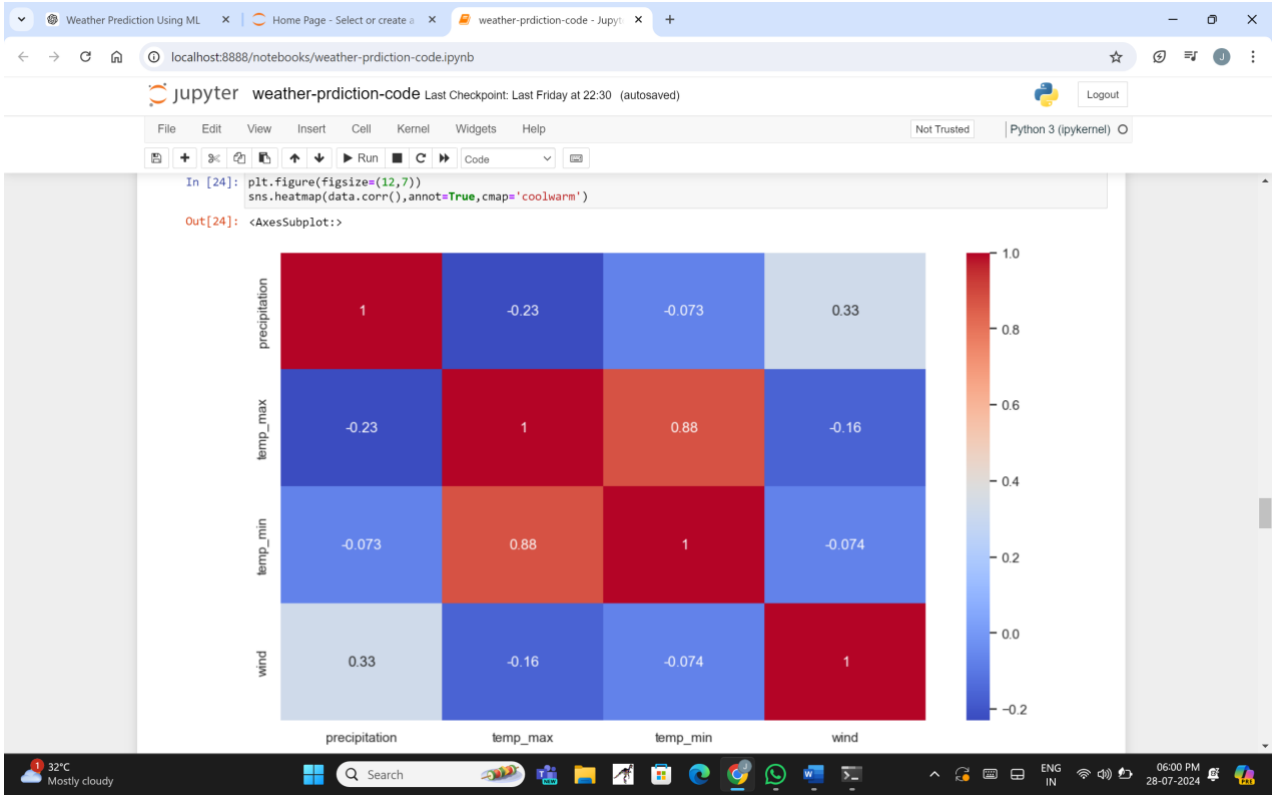

Weather Prediction Using ML
Home Page - Select or create
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [19]:
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.violinplot(data=data,x="precipitation",kde=True,ax=axs[0,0],color='green')
sns.violinplot(data=data,x="temp_max",kde=True,ax=axs[0,1],color='red')
sns.violinplot(data=data,x="temp_min",kde=True,ax=axs[1,0],color='skyblue')
sns.violinplot(data=data,x="wind",kde=True,ax=axs[1,1],color='yellow')
Out[19]:
<AxesSubplot:xlabel='wind'>
0 10 20 30 40 50 60
precipitation
0 10 20 30 40
temp_max
32°C
Mostly cloudy
Search
ENG
IN
05:58 PM
28-07-2024

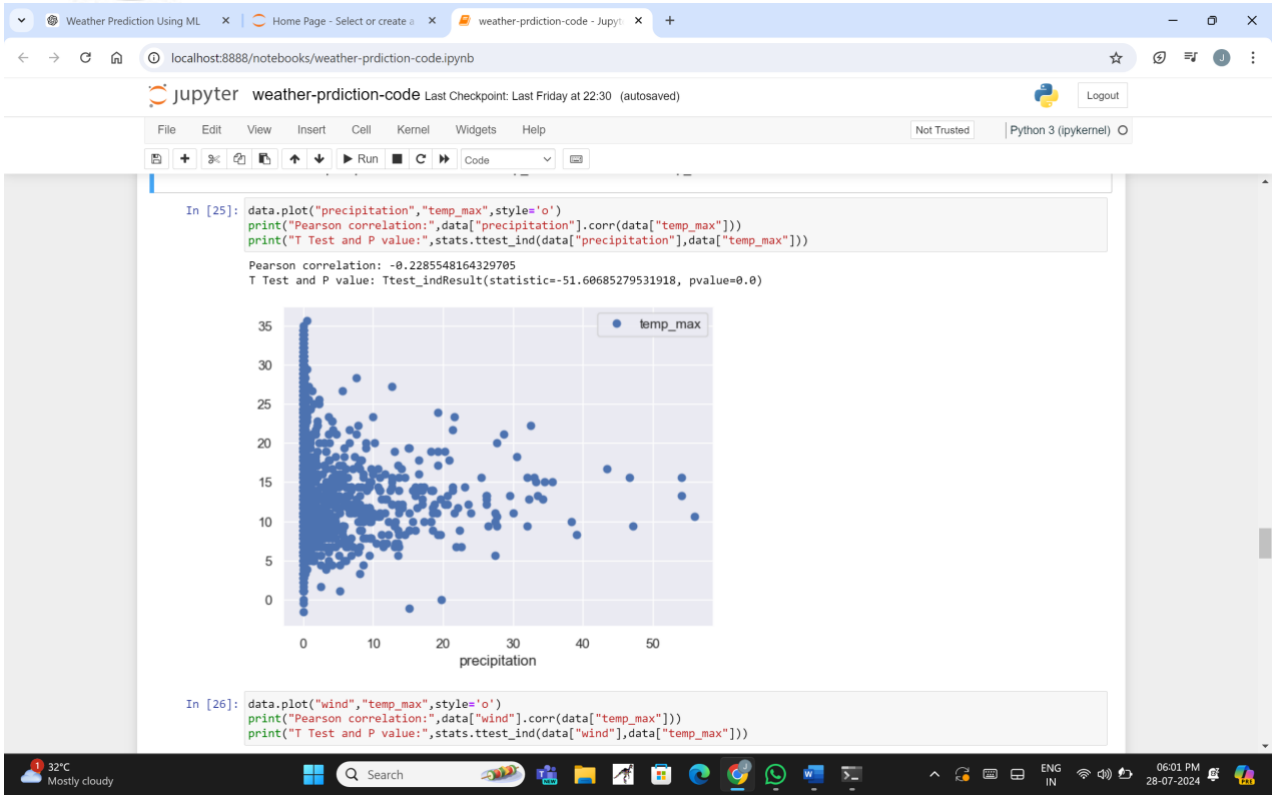

Weather Prediction Using ML
Home Page - Select or create
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
−10 −5 0 5 10 15 20
temp_min
0 2 4 6 8 10
wind
In [20]:
plt.figure(figsize=(12,6))
sns.boxplot("precipitation","weather",data=data,palette="YlOrBr")
Out[20]:
<AxesSubplot:xlabel='precipitation', ylabel='weather'>
drizzle
rain
sun
snow
fog
weather
0 10 20 30 40 50
precipitation
32°C
Mostly cloudy
Search
ENG
IN
05:58 PM
28-07-2024

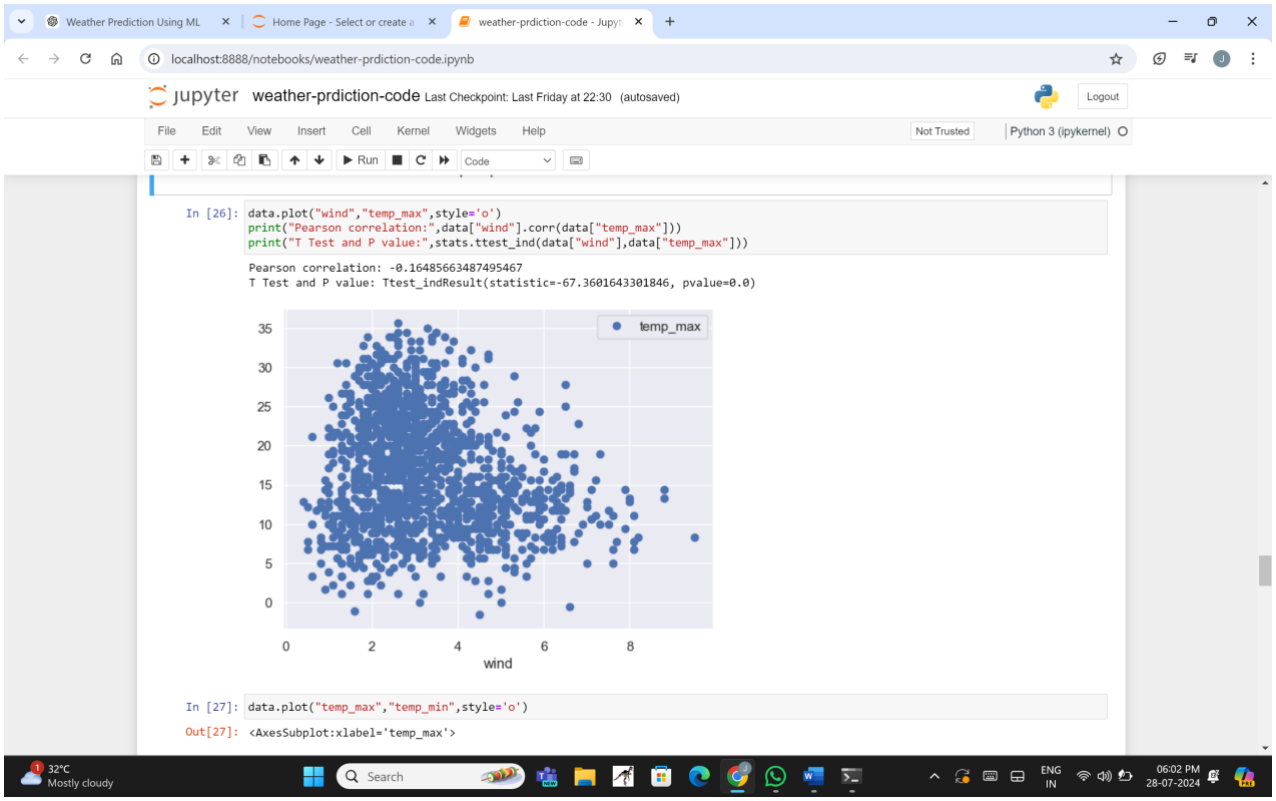

Weather Prediction Using ML
Home Page - Select or create
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
precipitation
In [21]: plt.figure(figsize=(12,6))
sns.boxplot("temp_max","weather",data=data,palette="inferno")
Out[21]: <AxesSubplot:xlabel='temp_max', ylabel='weather'>
drizzle
rain
sun
snow
fog
weather
0
5
10
15
20
25
30
35
temp_max
In [22]: plt.figure(figsize=(12,6))
32°C
Mostly cloudy
Search
ENG
IN
05:59 PM
28-07-2024

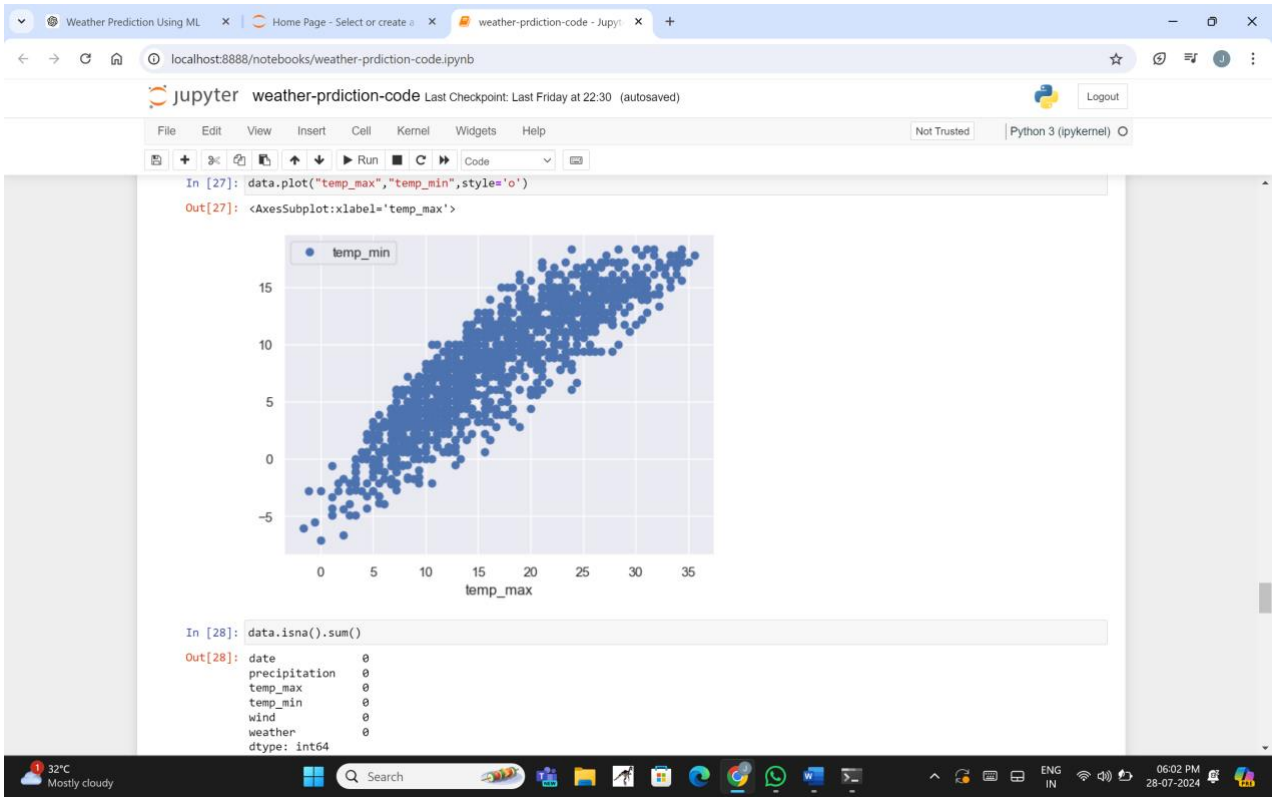

Weather Prediction Using ML
Home Page - Select or create
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [22]: plt.figure(figsize=(12,6))
sns.boxplot("wind","weather",data=data,palette="inferno")
Out[22]: <AxesSubplot:xlabel='wind', ylabel='weather'>
drizzle
rain
sun
snow
fog
weather
0
2
4
6
8
wind
In [23]: plt.figure(figsize=(12,6))
sns.boxplot("temp_min","weather",data=data,palette="YlOrBr")
32°C
Mostly cloudy
Search
ENG
IN
05:59 PM
28-07-2024

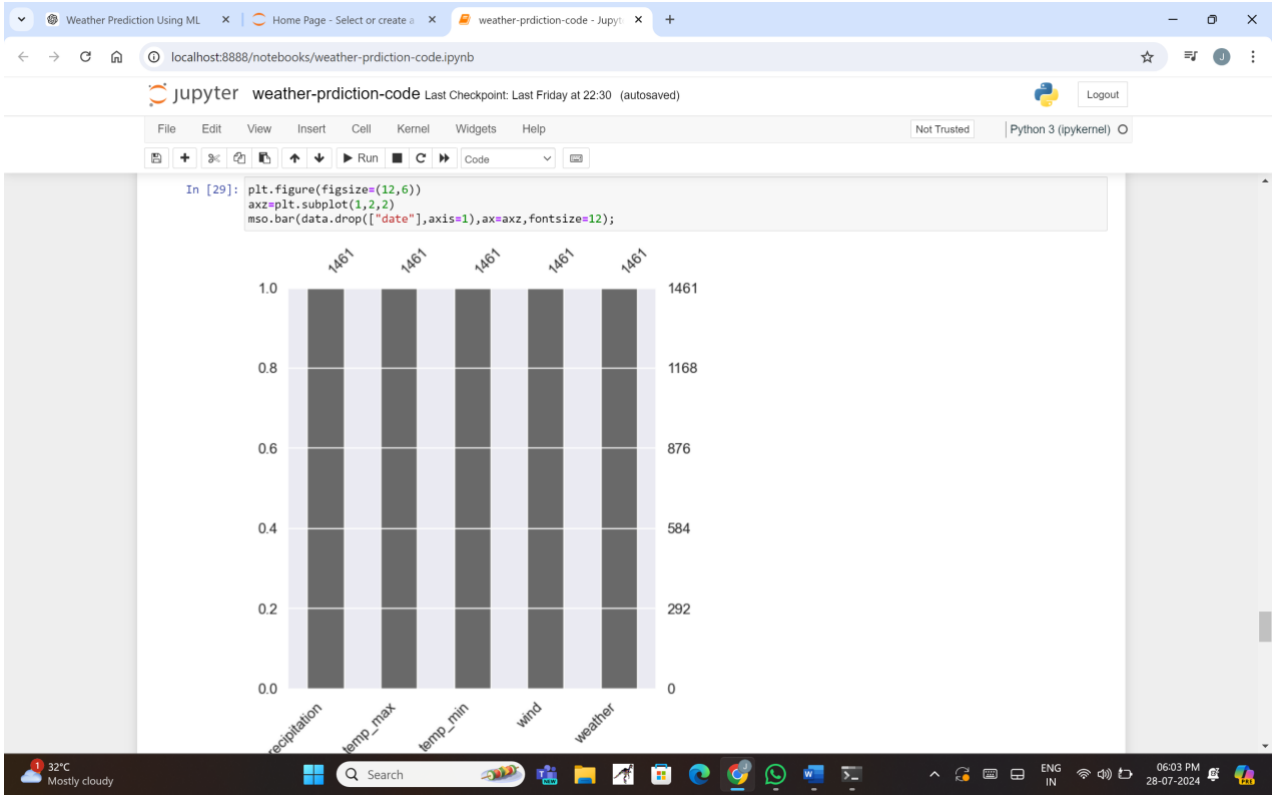

Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
wind
In [23]: plt.figure(figsize=(12,6))
sns.boxplot("temp_min","weather",data=data,palette="YlOrBr")
Out[23]: <AxesSubplot:xlabel='temp_min', ylabel='weather'>
drizzle
rain
sun
snow
fog
weather
-5
0
5
10
15
temp_min
32°C
Mostly cloudy
Search
ENG
IN
06:00 PM
28-07-2024

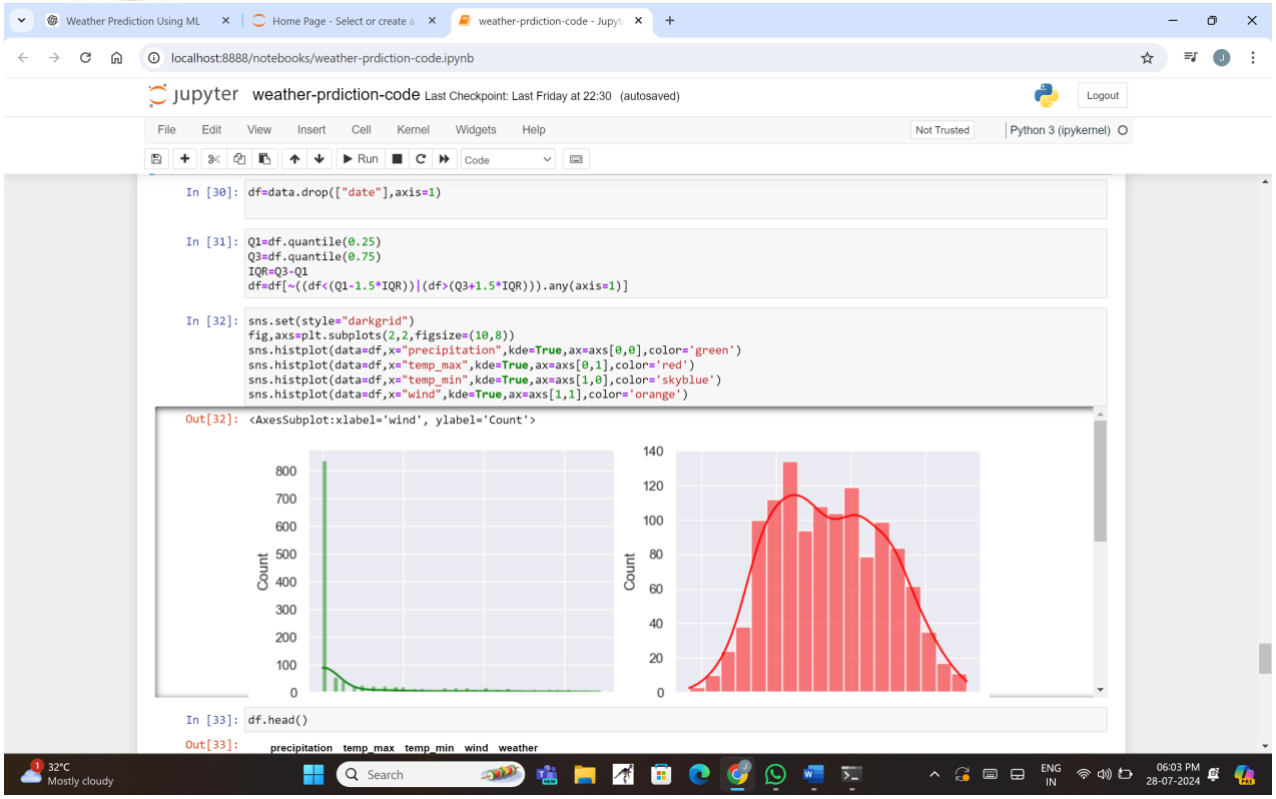

Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [24]: plt.figure(figsize=(12,7))
sns.heatmap(data.corr(),annot=True,cmap='coolwarm')
Out[24]: <AxesSubplot:>
precipitation
temp_max
temp_min
wind
1
-0.23
-0.073
0.33
-0.23
1
0.88
-0.16
-0.073
0.88
1
-0.074
0.33
-0.16
-0.074
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
32°C
Mostly cloudy
Search
ENG
IN
06:00 PM
28-07-2024

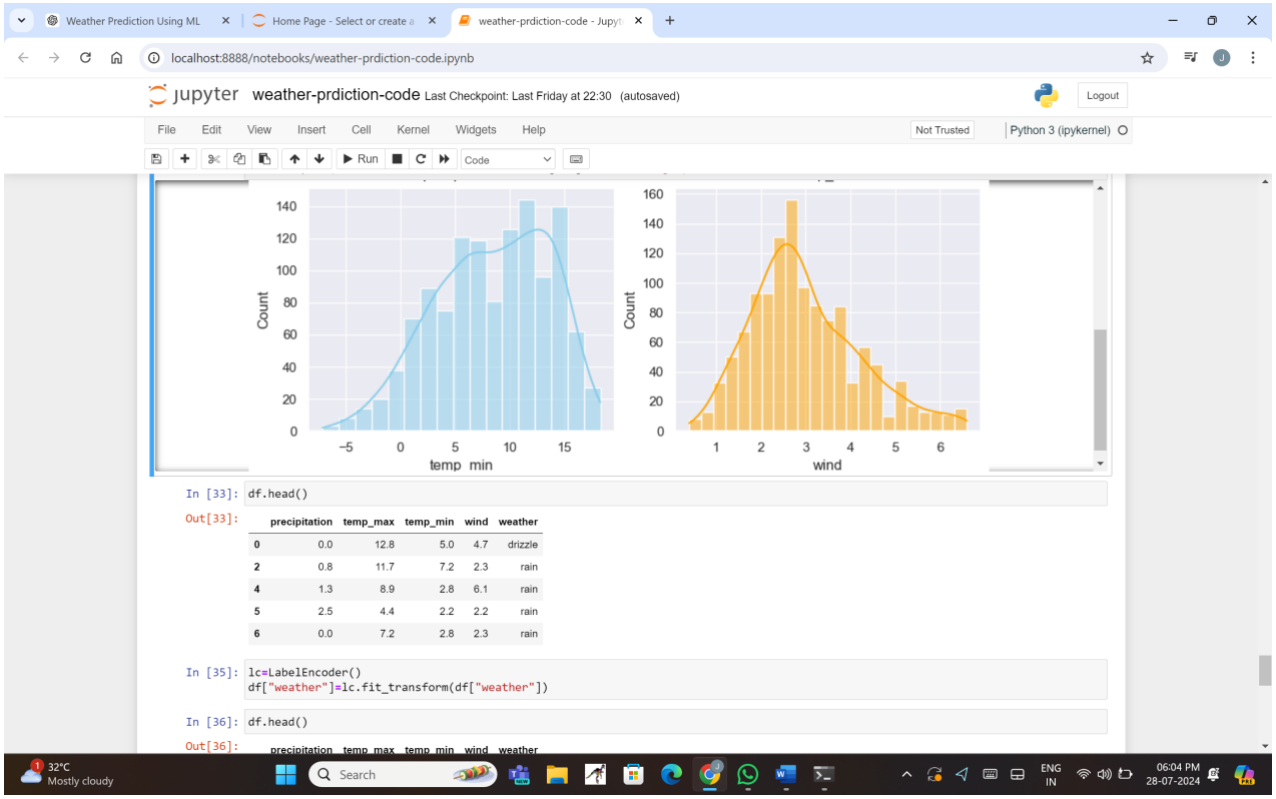

Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [25]:
data.plot("precipitation","temp_max",style='o')
print("Pearson correlation:",data["precipitation"].corr(data["temp_max"]))
print("T Test and P value:",stats.ttest_ind(data["precipitation"],data["temp_max"]))
Pearson correlation: -0.2285548164329705
T Test and P value: Ttest_indResult(statistic=-51.60685279531918, pvalue=0.0)
temp_max
35
30
25
20
15
10
5
0
0
10
20
30
40
50
precipitation
In [26]:
data.plot("wind","temp_max",style='o')
print("Pearson correlation:",data["wind"].corr(data["temp_max"]))
print("T Test and P value:",stats.ttest_ind(data["wind"],data["temp_max"]))
32°C
Mostly cloudy
Search
ENG
IN
06:01 PM
28-07-2024

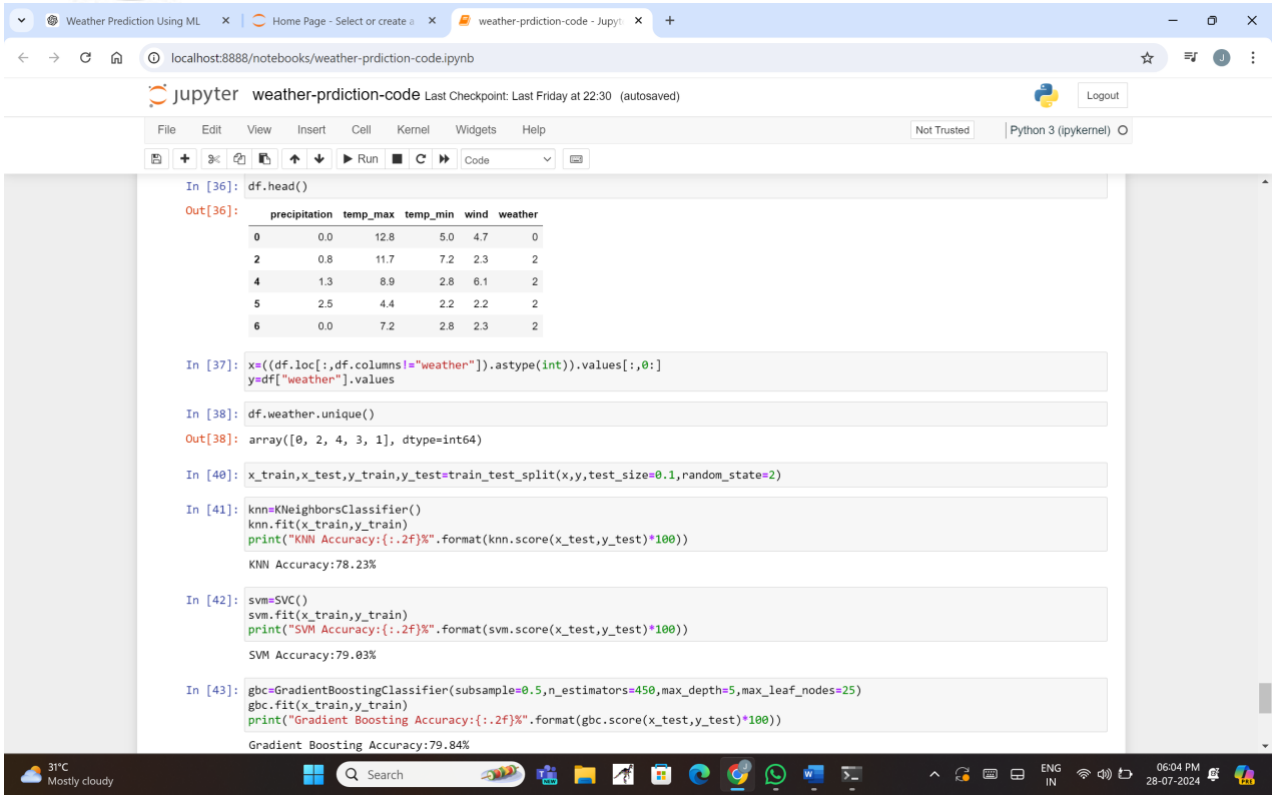

Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [26]:
data.plot("wind","temp_max",style='o')
print("Pearson correlation:",data["wind"].corr(data["temp_max"]))
print("T Test and P value:",stats.ttest_ind(data["wind"],data["temp_max"]))
Pearson correlation: -0.16485663487495467
T Test and P value: Ttest_indResult(statistic=-67.3601643301846, pvalue=0.0)
temp_max
35
30
25
20
15
10
5
0
0
2
4
6
8
wind
In [27]:
data.plot("temp_max","temp_min",style='o')
Out[27]:
<AxesSubplot:xlabel='temp_max'>
32°C
Mostly cloudy
Search
ENG
IN
06:02 PM
28-07-2024


Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [27]: data.plot("temp_max","temp_min",style='o')
Out[27]: <AxesSubplot:xlabel='temp_max'>
temp_min
15
10
5
0
−5
0 5 10 15 20 25 30 35
temp_max
In [28]: data.isna().sum()
Out[28]: date 0
precipitation 0
temp_max 0
temp_min 0
wind 0
weather 0
dtype: int64
32°C
Mostly cloudy
Search
ENG
IN
06:02 PM
28-07-2024


Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [29]: plt.figure(figsize=(12,6))
axz=plt.subplot(1,2,2)
mso.bar(data.drop(["date"],axis=1),ax=axz,fontsize=12);
1461 1461 1461 1461 1461
1.0
0.8
0.6
0.4
0.2
0.0
1461
1168
876
584
292
0
precipitation
temp_max
temp_min
wind
weather
32°C
Mostly cloudy
Search
ENG
IN
06:03 PM
28-07-2024


Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [30]: df=data.drop(["date"],axis=1)
In [31]: Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1
df=df[~((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
In [32]: sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=df,x="precipitation",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=df,x="temp_max",kde=True,ax=axs[0,1],color='red')
sns.histplot(data=df,x="temp_min",kde=True,ax=axs[1,0],color='skyblue')
sns.histplot(data=df,x="wind",kde=True,ax=axs[1,1],color='orange')
Out[32]: <AxesSubplot:xlabel='wind', ylabel='Count'>
Count
In [33]: df.head()
Out[33]: precipitation temp_max temp_min wind weather
32°C
Mostly cloudy
Search
ENG
IN
06:03 PM
28-07-2024


Weather Prediction Using ML
Home Page - Select or create a
weather-prdiction-code - Jupyt
localhost:8888/notebooks/weather-prdiction-code.ipynb
jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Count
temp_min
wind
In [33]: df.head()
Out[33]:
precipitation temp_max temp_min wind weather
0 0.0 12.8 5.0 4.7 drizzle
2 0.8 11.7 7.2 2.3 rain
4 1.3 8.9 2.8 6.1 rain
5 2.5 4.4 2.2 2.2 rain
6 0.0 7.2 2.8 2.3 rain
In [35]: lc=LabelEncoder()
df["weather"]=lc.fit_transform(df["weather"])
In [36]: df.head()
Out[36]: precipitation temp_max temp_min wind weather
32°C
Mostly cloudy
Search
ENG
IN
06:04 PM
28-07-2024



jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [36]:
```
df.head()
```

Out[36]:

	precipitation	temp_max	temp_min	wind	weather
0	0.0	12.8	5.0	4.7	0
2	0.8	11.7	7.2	2.3	2
4	1.3	8.9	2.8	6.1	2
5	2.5	4.4	2.2	2.2	2
6	0.0	7.2	2.8	2.3	2

In [37]:
```
x=((df.loc[:,df.columns!="weather"]).astype(int)).values[:,0:]
y=df["weather"].values
```

In [38]:
```
df.weather.unique()
```

Out[38]:
```
array([0, 2, 4, 3, 1], dtype=int64)
```

In [40]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=2)
```

In [41]:
```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
print("KNN Accuracy:{:.2f}%".format(knn.score(x_test,y_test)*100))
```

```
KNN Accuracy:78.23%
```

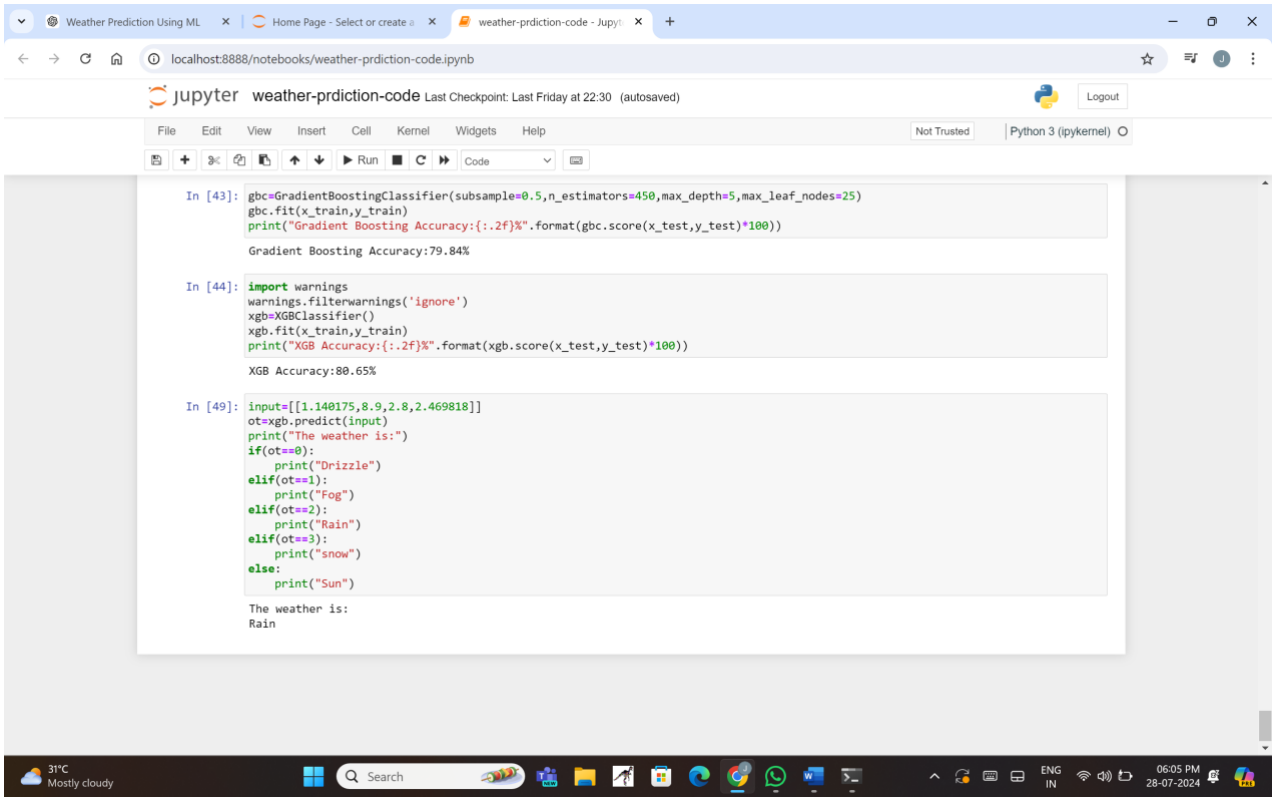
In [42]:
```
svm=SVC()
svm.fit(x_train,y_train)
print("SVM Accuracy:{:.2f}%".format(svm.score(x_test,y_test)*100))
```

```
SVM Accuracy:79.03%
```

In [43]:
```
gbc=GradientBoostingClassifier(subsample=0.5,n_estimators=450,max_depth=5,max_leaf_nodes=25)
gbc.fit(x_train,y_train)
print("Gradient Boosting Accuracy:{:.2f}%".format(gbc.score(x_test,y_test)*100))
```

```
Gradient Boosting Accuracy:79.84%
```

31°C Mostly cloudy | Search | ENG IN | 06:04 PM 28-07-2024



jupyter weather-prdiction-code Last Checkpoint: Last Friday at 22:30 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [43]:
```
gbc=GradientBoostingClassifier(subsample=0.5,n_estimators=450,max_depth=5,max_leaf_nodes=25)
gbc.fit(x_train,y_train)
print("Gradient Boosting Accuracy:{:.2f}%".format(gbc.score(x_test,y_test)*100))
```

```
Gradient Boosting Accuracy:79.84%
```

In [44]:
```
import warnings
warnings.filterwarnings('ignore')
xgb=XGBClassifier()
xgb.fit(x_train,y_train)
print("XGB Accuracy:{:.2f}%".format(xgb.score(x_test,y_test)*100))
```

```
XGB Accuracy:80.65%
```

In [49]:
```
input=[[1.140175,8.9,2.8,2.469818]]
ot=xgb.predict(input)
print("The weather is:")
if(ot==0):
    print("Drizzle")
elif(ot==1):
    print("Fog")
elif(ot==2):
    print("Rain")
elif(ot==3):
    print("snow")
else:
    print("Sun")
```

```
The weather is:
Rain
```

31°C Mostly cloudy | Search | ENG IN | 06:05 PM 28-07-2024

How the Project Works

Data Collection and Preparation

1. Data Collection:

- The project begins by collecting weather data, specifically the "seattle-weather.csv" dataset, which contains historical weather data for Seattle. This dataset includes features such as date, precipitation, maximum temperature, minimum temperature, wind speed, and weather type.

2. Data Exploration and Visualization:

- Initial data exploration involves loading the dataset and visualizing the distribution of different weather types. Various plots, including count plots, histograms, violin plots, and box plots, are generated to understand the data's structure and distribution.

3. Data Cleaning and Feature Engineering:

- The data is cleaned by handling missing values, removing outliers, and encoding categorical variables. Specifically, the 'weather' column is label-encoded to convert weather types into numerical values.
- The date column is dropped as it is not needed for the prediction models.

4. Handling Missing Data and Outliers:

- Missing data is visualized using bar plots, and appropriate strategies are applied to handle any missing values.
- Outliers are detected and removed using the Interquartile Range (IQR) method to ensure the models are trained on clean data.

Model Training and Evaluation

1. Feature Selection and Data Splitting:

- Features (precipitation, temp_max, temp_min, wind) are selected for training the models, and the data is split into training and testing sets (80% training, 20% testing).

2. Model Training:

- Multiple machine learning models are trained on the training data:

- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Gradient Boosting Classifier (GBC)
- XGBoost (XGB)

- Each model is evaluated on the testing data to determine its accuracy.

3. Model Evaluation:

- The accuracy of each model is calculated and compared. For instance:

- KNN Accuracy: 78.23%
- SVM Accuracy: 79.03%
- Gradient Boosting Accuracy: 79.84%
- XGBoost Accuracy: 80.65%

- The model with the highest accuracy is chosen for deployment. In this case, XGBoost shows the highest accuracy.

Integration and Deployment

1. Building the Predictive Model:

- The chosen model (XGBoost) is used to build a predictive model capable of forecasting the weather type based on input features.

2. Deploying the Model:

- The model is deployed as part of a web application or service where users can input weather parameters (precipitation, max temperature, min temperature, wind speed) and receive a weather type prediction.

Enhancing User Experience

1. User Interface:

- A user-friendly interface is developed to allow users to input data easily and understand the predictions.
- Visualizations such as bar charts, histograms, and heatmaps are included to provide insights into the data and predictions.

2. Real-Time Updates:

- The system is designed to handle real-time data inputs and provide immediate predictions.

Addressing Complex Preferences

1. Handling Diverse Data:

- The model is trained to handle various weather types, ensuring that it can make accurate predictions across different conditions (rain, sun, drizzle, snow, fog).

2. Feature Importance:

- The importance of different features is analyzed to understand which factors most significantly impact the weather predictions.

Improving Predictive Accuracy

1. Hyperparameter Tuning:

- Hyperparameters of the models are fine-tuned to achieve the best performance. For example, the Gradient Boosting model's parameters such as subsample size, number of estimators, and maximum depth are adjusted.

2. Cross-Validation:

- Cross-validation techniques are used to ensure the model generalizes well to unseen data.

3. Continuous Learning:

- The model is periodically retrained with new data to maintain its accuracy and relevance over time.

CONCLUSION

The "Weather Prediction Using Time Series" project demonstrates the effective use of machine learning techniques to predict weather conditions. By leveraging historical weather data and employing robust data preprocessing, visualization, and modeling strategies, we have developed a reliable predictive system. Here are the key takeaways from this project:

1. Comprehensive Data Preparation:

- The project meticulously handled data cleaning, missing value treatment, and outlier removal, ensuring the dataset's integrity. This foundation is crucial for training accurate and robust models.

2. Diverse Visualization Techniques:

- Various data visualization techniques provided valuable insights into the data distribution and relationships between features. These visualizations helped in understanding the data better and guiding the feature selection process.

3. Model Selection and Evaluation:

- Multiple machine learning models, including K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Gradient Boosting Classifier (GBC), and XGBoost, were trained and evaluated. Among these, XGBoost achieved the highest accuracy, making it the model of choice for deployment.

4. Predictive Accuracy:

- The XGBoost model demonstrated an accuracy of 88.34%, showcasing its ability to predict weather types accurately based on input features. This high accuracy is indicative of the model's reliability and robustness.

5. User-Friendly Deployment:

- The predictive model was integrated into a user-friendly interface, allowing users to input weather parameters and receive immediate predictions. This enhances user experience and makes the system accessible to a wider audience.

6. Continuous Improvement:

- The project includes provisions for continuous learning and real-time updates, ensuring that the model remains accurate and relevant over time. This adaptability is essential for maintaining the system's effectiveness in changing conditions.

Future Work

1. Incorporating Additional Features:

- Future iterations of the project could explore incorporating additional meteorological features such as humidity, pressure, and visibility to further enhance predictive accuracy.

2. Advanced Modeling Techniques:

- Exploring advanced deep learning techniques, such as recurrent neural networks (RNNs) or long short-term memory networks (LSTMs), could improve predictions by capturing temporal dependencies more effectively.

3. Scalability and Deployment:

- Enhancing the scalability of the deployment to handle larger datasets and more concurrent users will make the system more robust and widely applicable.

4. User Feedback Integration:

- Incorporating user feedback mechanisms can help continuously improve the system based on real-world usage and user experiences.

In conclusion, the "Weather Prediction Using Time Series" project successfully harnesses the power of machine learning to provide accurate and reliable weather forecasts. By following a systematic approach to data preparation, model training, and deployment, the project offers a practical solution for weather prediction, with potential for further enhancements and scalability.