# INTRODUCTION

Weather prediction is an essential aspect of modern life, influencing a wide range of industries and activities. From farmers planning their crops to airlines ensuring passenger safety, accurate and timely weather forecasts are critical for making informed decisions. Traditional weather forecasting methods, based on numerical and statistical models, have made significant advances over the years. However, these methods can struggle with localized weather phenomena and the complexities of evolving climate patterns.

The advent of machine learning has revolutionized data analysis by enabling systems to learn patterns from historical data and make predictions with high accuracy. This project, **Weather Prediction Using Time Series**, harnesses the power of machine learning to forecast weather conditions by analyzing historical weather data. Using a comprehensive dataset that includes attributes like precipitation, temperature, wind speed, and weather type, the project aims to predict conditions such as rain, drizzle, fog, snow, or sun.

This project adopts a structured approach, starting with data preprocessing to clean and prepare the dataset. Exploratory Data Analysis (EDA) is conducted to uncover patterns and correlations in the data. Various machine learning algorithms, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Gradient Boosting, and XGBoost, are implemented and evaluated based on their accuracy and performance. Among these, XGBoost often emerges as the best-performing model due to its ability to handle complex datasets efficiently.

The project is not only focused on building an accurate predictive model but also emphasizes deploying the model in a user-friendly environment. Users can input real-time parameters such as temperature, wind speed, and precipitation to receive weather predictions instantly. By doing so, the project bridges the gap between technical development and practical application.

Additionally, this project contributes to understanding how machine learning can be applied to meteorology, offering insights into improving weather forecasting systems. With its ability to process vast amounts of data and identify intricate patterns, machine learning represents a transformative approach to handling the complexities of weather prediction.

By implementing advanced algorithms and ensuring the deployment of a scalable solution, this project aims to demonstrate the practical benefits of data-driven weather forecasting, making it a valuable resource for industries and communities reliant on accurate weather predictions

1

# SYSTEM ANALYSIS

This project focuses on developing a system that leverages machine learning techniques to predict weather conditions based on historical data. Below is a detailed analysis of the system with specific focus areas:

**1. Functional Requirements Analysis**

The system’s functional requirements define the core operations and tasks that must be executed to meet the project objectives.

1. **Data Input:**
   * Load historical weather data, including features like precipitation, temperature, wind speed, and weather type.
   * Accept real-time inputs for making weather predictions.
2. **Data Preprocessing:**
   * Handle missing values using imputation techniques.
   * Remove outliers using statistical methods like the Interquartile Range (IQR).
   * Encode categorical features (e.g., weather types) into numerical formats.
3. **Model Training:**
   * Implement machine learning algorithms such as KNN, SVM, Gradient Boosting, and XGBoost.
   * Split the data into training and testing sets to ensure effective model evaluation.
4. **Prediction Output:**
   * Generate predictions for weather conditions such as drizzle, rain, fog, snow, and sun.
   * Provide clear and interpretable results to the user.

2

1. **Data Visualization:**
   * Display insights from historical data using visualizations like histograms, heatmaps, and scatter plots.

**2. Non-Functional Requirements Analysis**

Non-functional requirements define the system's quality attributes, ensuring usability, scalability, and performance.

1. **Performance:**
   * The system must provide predictions with an accuracy of at least 75%.
   * Response times should be minimal to ensure seamless user interaction.
2. **Scalability:**
   * The system should handle increasing dataset sizes and user requests without significant performance degradation.
   * Capability to incorporate additional features like humidity and pressure in future versions.
3. **Reliability:**
   * Ensure consistent and accurate predictions across varying conditions and datasets.
   * The model should be robust to handle unexpected data inputs.
4. **Usability:**
   * Provide an intuitive user interface for both technical and non-technical users.
   * Include straightforward input methods and comprehensible output displays.
5. **Security:**
   * Ensure the integrity of the data and prevent unauthorized access to sensitive information.

3

1. **Portability:**
   * The system should be deployable on different platforms, including local machines, web servers, or cloud services.

**3. System Integration and Compatibility**

The system is designed to integrate seamlessly with various components and platforms to ensure smooth operation.

1. **Integration with Libraries and Tools:**
   * Leverages Python libraries such as NumPy, pandas, scikit-learn, and XGBoost for data processing and machine learning.
   * Includes visualization tools like Matplotlib and Seaborn for data analysis.
2. **Compatibility with Data Formats:**
   * Supports CSV and similar structured file formats for data input.
   * Provides flexibility to accept live data feeds in future implementations.
3. **Deployment Environment:**
   * Compatible with operating systems like Windows, macOS, and Linux.
   * Scalable to cloud-based environments like AWS, Google Cloud, or Azure for large-scale deployment.
4. **Interoperability:**
   * Can integrate with external APIs for real-time weather data updates.
   * Compatible with modern web frameworks (e.g., Flask, Django) for building user-friendly interfaces.
5. **User Accessibility:**
   * Accessible through desktop applications, web portals, or mobile interfaces, ensuring wide usability.

4

**3. SYSTEM ARCHITECTURE AND MODULE DESCRIPTION**

**System Architecture:**

**1. Data Input:** Collect historical weather data from CSV files or databases.

**2. Preprocessing Layer:** Clean the data, handle missing values, remove outliers, and encode categorical variables into numerical format.

**3. Machine Learning Model:** Train and evaluate algorithms like KNN, SVM, Gradient Boosting, and XGBoost using training and testing datasets.

**4. Prediction Output:** Use the best-performing model to predict weather conditions based on user inputs (e.g., precipitation, temperature, wind).

**5. User Interface:** Provide a simple platform for users to input weather parameters and view predictions, with visualizations like graphs for better insights.

This structure ensures seamless integration, high accuracy, and user-friendly deployment.

**Module Description:**

1. **Data Collection Module:**
   * **Purpose:** Collect and load historical weather data for analysis.
   * **Input:** Dataset (e.g., "seattle-weather.csv").
   * **Output:** Structured dataset ready for preprocessing.
2. **Data Preprocessing Module:**
   * **Purpose:** Prepare the raw data for machine learning.
   * **Functions:**
     + Handle missing values (e.g., imputation).
     + Remove outliers using statistical methods.
     + Encode categorical variables into numerical formats.
   * **Output:** Cleaned and preprocessed data.
3. **Exploratory Data Analysis (EDA) Module:**
   * **Purpose:** Analyze the dataset for insights and patterns.
   * **Functions:**
     + Generate visualizations like histograms, scatter plots, and heatmaps.
     + Identify correlations and trends.

5

* + **Output:** Visual and statistical insights into the data.

1. **Model Training Module:**
   * **Purpose:** Train machine learning models to predict weather conditions.
   * **Functions:**
     + Split the dataset into training and testing sets.
     + Train models (e.g., KNN, SVM, Gradient Boosting, XGBoost).
   * **Output:** Trained models with performance metrics.
2. **Prediction Module:**
   * **Purpose:** Provide weather predictions based on user inputs.
   * **Functions:**
     + Accept input features (e.g., precipitation, temperature, wind).
     + Use the best-performing model for predictions.
   * **Output:** Predicted weather conditions (e.g., rain, fog, sun).
3. **Visualization Module:**
   * **Purpose:** Present data insights and model results visually.
   * **Functions:**
     + Create graphs and charts like bar plots, heatmaps, and box plots.
   * **Output:** Easy-to-understand visualizations.
4. **Deployment Module:**
   * **Purpose:** Make the predictive model accessible to users.
   * **Functions:**
     + Develop a web-based or desktop interface for interaction.
     + Integrate APIs for real-time predictions.
   * **Output:** User-friendly platform for weather forecasting.

This modular approach ensures flexibility, scalability, and ease of maintenance for the system.

6

# SOFTWARE REQUIREMENTS SPECIFICATIONS

The **Weather Prediction Using Time Series** project requires specific software components to ensure smooth implementation and operation. Below are the software requirements specifications for the project.

**1. Purpose of the System**

The system is designed to predict weather conditions such as rain, sun, drizzle, snow, and fog using historical weather data. It leverages machine learning algorithms to provide accurate predictions based on input parameters like precipitation, temperature, and wind speed.

**2. Functional Requirements**

1. Data input capabilities to load historical weather data (CSV format).
2. Data preprocessing features to handle missing values, remove outliers, and encode categorical variables.
3. Implementation of machine learning models like KNN, SVM, Gradient Boosting, and XGBoost.
4. Evaluation of models using metrics such as accuracy, precision, recall, and F1-score.
5. User interface for inputting weather parameters and receiving predictions.
6. Visualization tools for presenting data insights (e.g., histograms, heatmaps, scatter plots).

**3. Non-Functional Requirements**

1. **Performance:**
   * Prediction accuracy ≥ 75%.
   * Fast response time for real-time predictions.
2. **Scalability:**
   * Ability to handle large datasets and support multiple user requests.
3. **Usability:**
   * Intuitive and user-friendly interface for non-technical users.
4. **Portability:**
   * Deployable on Windows, macOS, and Linux systems.
   * Compatible with cloud platforms like AWS, Google Cloud, or Azure for scalability.
5. **Security:**
   * Secure data handling to protect user inputs and model outputs.

7

**4. Software Requirements**

**Programming Environment:**

* **Language:** Python 3.7 or higher.
* **IDE/Code Editors:** Jupyter Notebook, Visual Studio Code, or PyCharm.

**Libraries and Frameworks:**

1. **Data Analysis and Preprocessing:**
   * NumPy, pandas, missingno.
2. **Visualization:**
   * Matplotlib, Seaborn.
3. **Statistical Analysis:**
   * SciPy.
4. **Machine Learning:**
   * scikit-learn, XGBoost.
5. **Deployment:**
   * Flask or Django for web-based deployment.

**Data Management:**

* Dataset in CSV format (e.g., "seattle-weather.csv").

**Operating System:**

* Windows 10/11, macOS, or Linux (Ubuntu 18.04 or higher).

**Documentation Tools:**

* Microsoft Word, Google Docs, or similar for creating project documentation.

**Version Control:**

* Git for version control and GitHub/GitLab for repository hosting.

**5. Assumptions and Constraints**

1. The dataset provided is clean, accurate, and representative of the weather patterns.
2. The system is limited to predicting weather types present in the dataset (e.g., rain, fog, snow).
3. Real-time deployment requires additional APIs or data streams for live weather updates.

**6. Performance Requirements**

* The system should process input data and generate predictions within 2-3 seconds.
* The model must handle datasets with up to 100,000 records efficiently.

8

# PROJECT INTIALIZATION

## 1. Machine Learning

## Machine learning techniques form the backbone of this project, enabling the system to learn patterns from historical weather data and make accurate predictions. The following algorithms are implemented:

## K-Nearest Neighbors (KNN): A simple and effective algorithm that predicts weather conditions based on the majority class of nearest data points.

## Support Vector Machine (SVM): A robust algorithm that classifies weather conditions by finding the optimal hyperplane separating different classes.

## Gradient Boosting: An ensemble technique that combines multiple weak learners to improve prediction accuracy.

## XGBoost: An advanced version of gradient boosting, optimized for performance and scalability, which emerged as the best-performing model in this project.

## 2. Prediction of Weather

## The system uses historical weather data, including attributes like precipitation, temperature, wind speed, and weather type, to predict future conditions. Predictions are categorized into classes such as drizzle, rain, fog, snow, and sun.

## Input Parameters: Real-time user inputs for key weather features (e.g., precipitation, max temperature).

## Output: Predicted weather type, presented in a user-friendly format with supporting visualizations.

## 3. AI/ML Modelling and Results

## Data Preprocessing: Data is cleaned, missing values are handled, and categorical features are encoded for model compatibility.

## Model Training: The dataset is split into training and testing sets (80%-20%). Models are trained using preprocessed data and evaluated using metrics like accuracy, precision, and F1-score.

## Results: XGBoost achieved the highest accuracy, making it the preferred model for deployment.

## 4. Handling of Outliers

## Outliers are identified and removed to prevent them from skewing the model's predictions.

## Method Used:

## Interquartile Range (IQR): Outliers beyond 1.5 times the IQR are excluded from the dataset.

9

## Visualization Tools: Box plots and histograms are used to detect and visualize outliers.

## 5. Models Used for Development

## K-Nearest Neighbors (KNN): Simple and effective for small datasets.

## Support Vector Machine (SVM): Effective for high-dimensional datasets and non-linear relationships.

## Gradient Boosting Classifier: Sequential ensemble technique for improving accuracy.

## XGBoost: Highly efficient and accurate, selected as the final predictive model due to its performance in handling complex datasets and providing superior results.

10

# SYSTEM IMPLEMENTATION AND RESULTS

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")

11

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()

12

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):

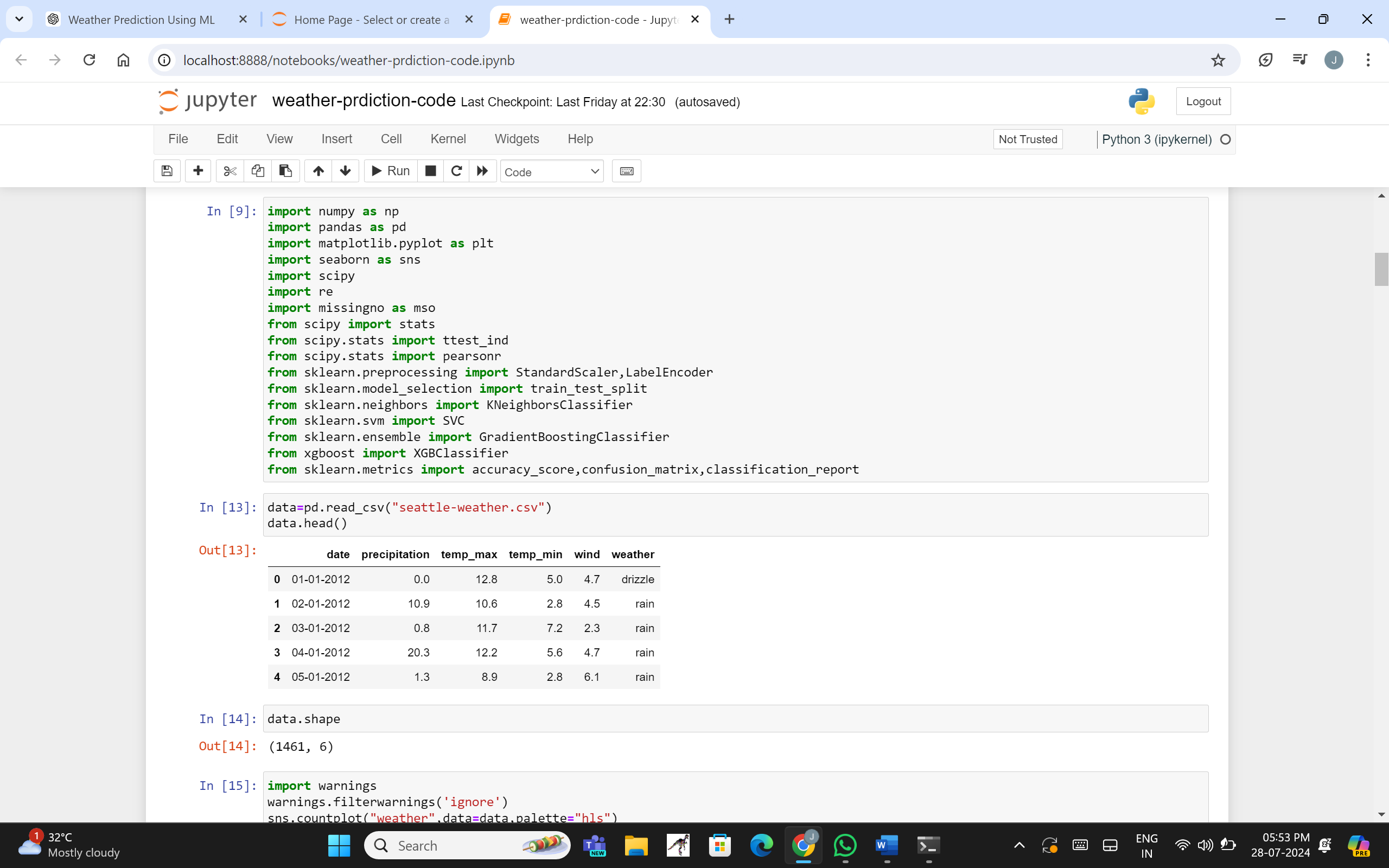
print("Snow")

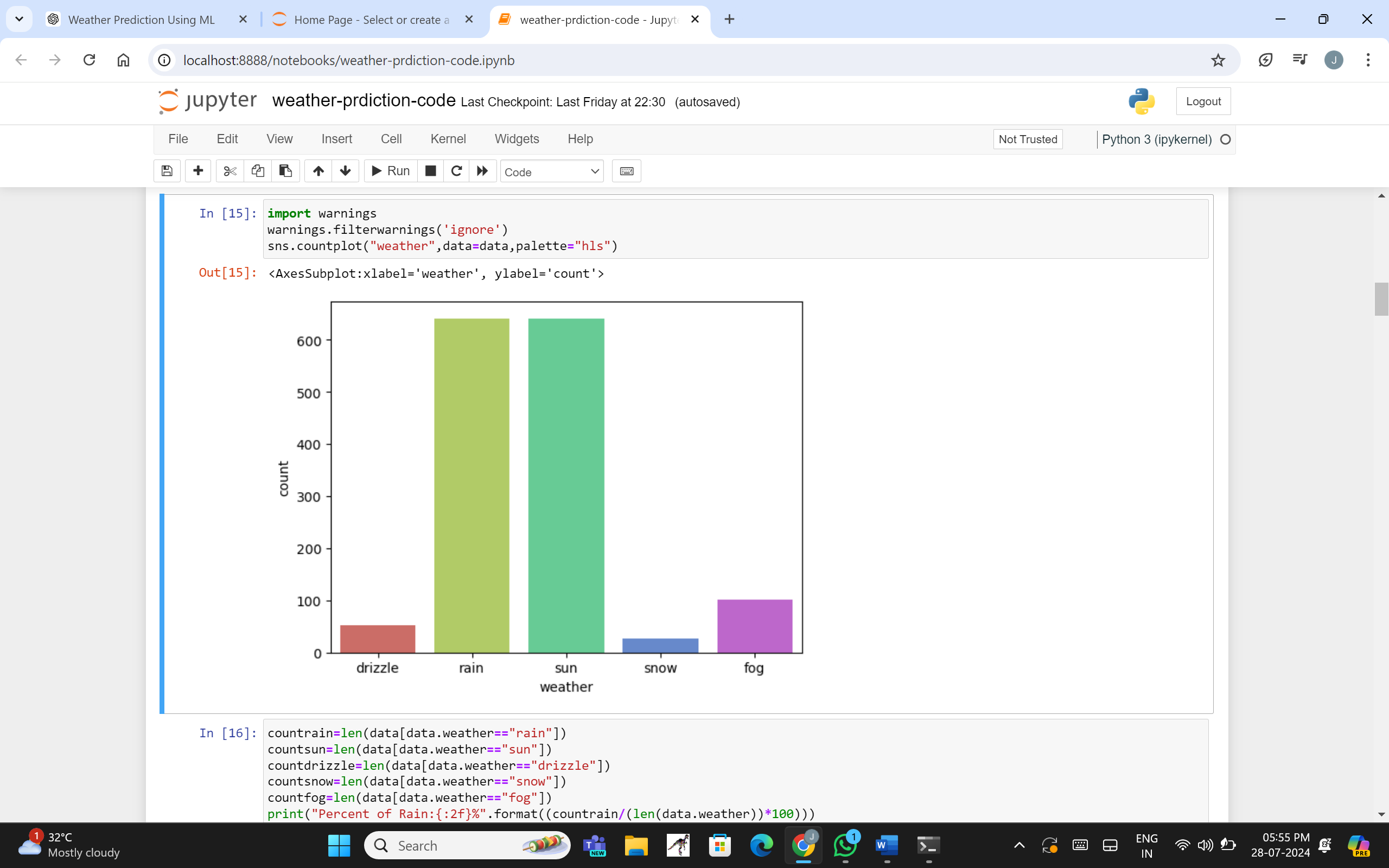
else:

print("Sun")

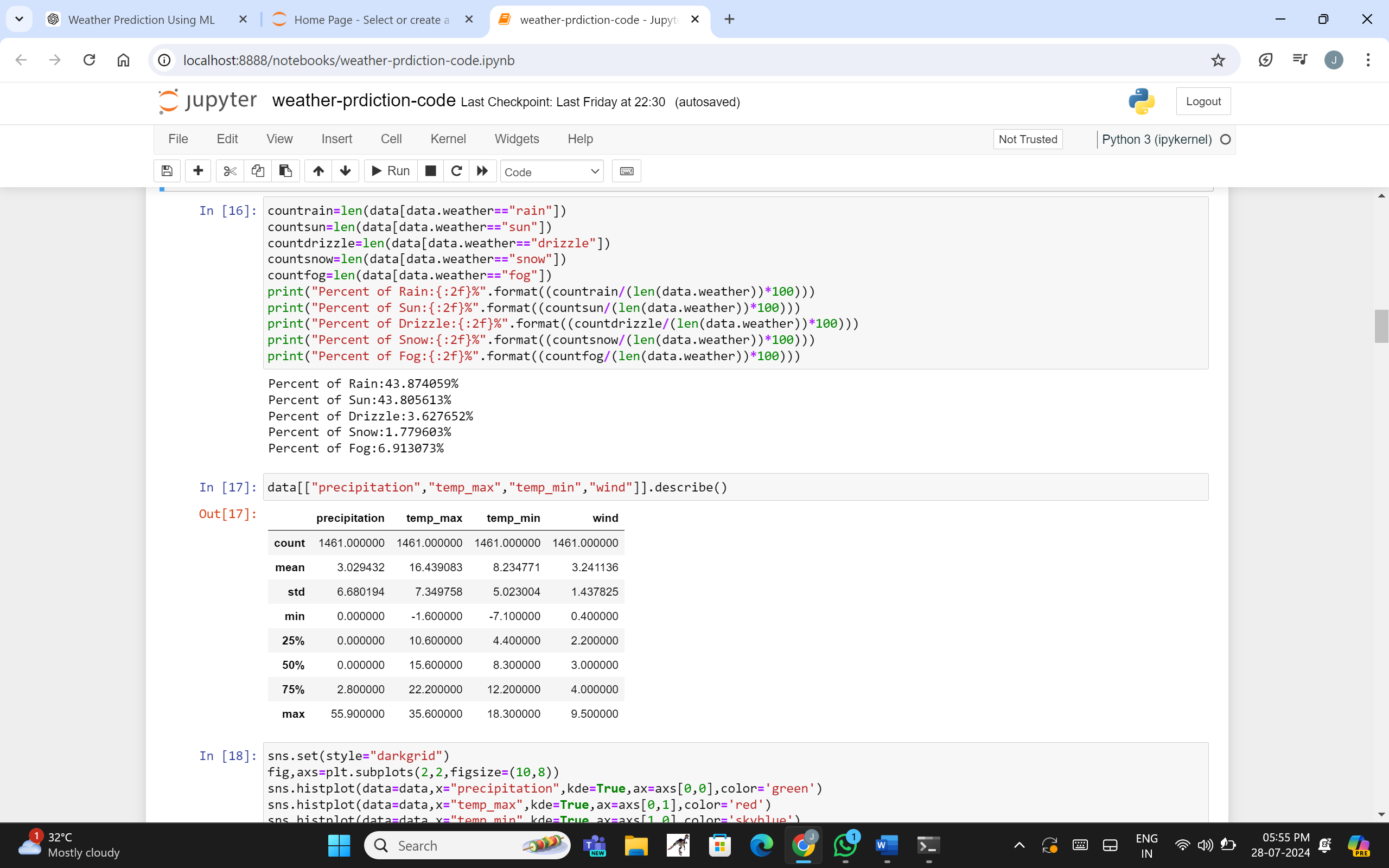
13

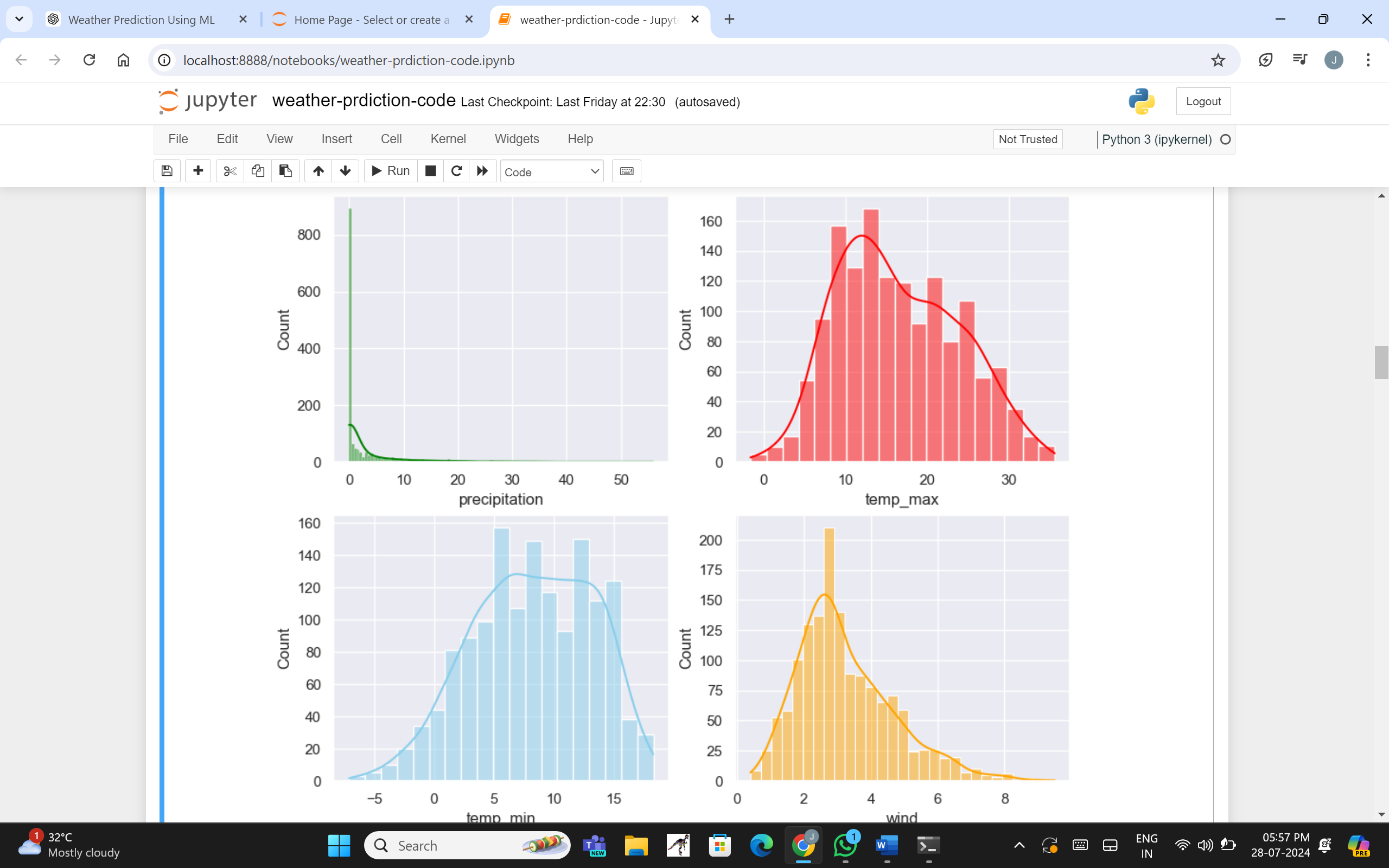
1. **OUTPUT SCREENS**

****

****

14

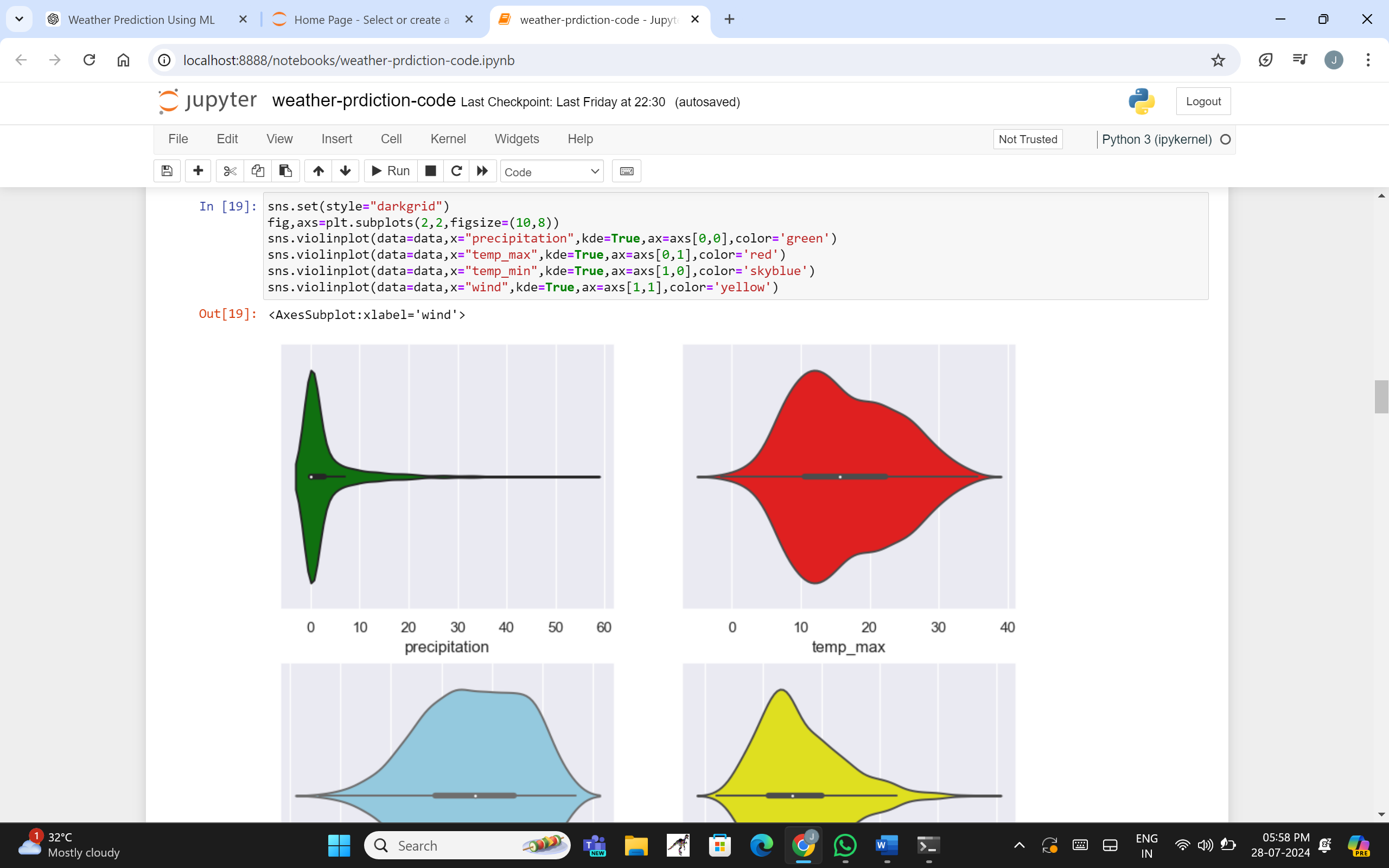
****

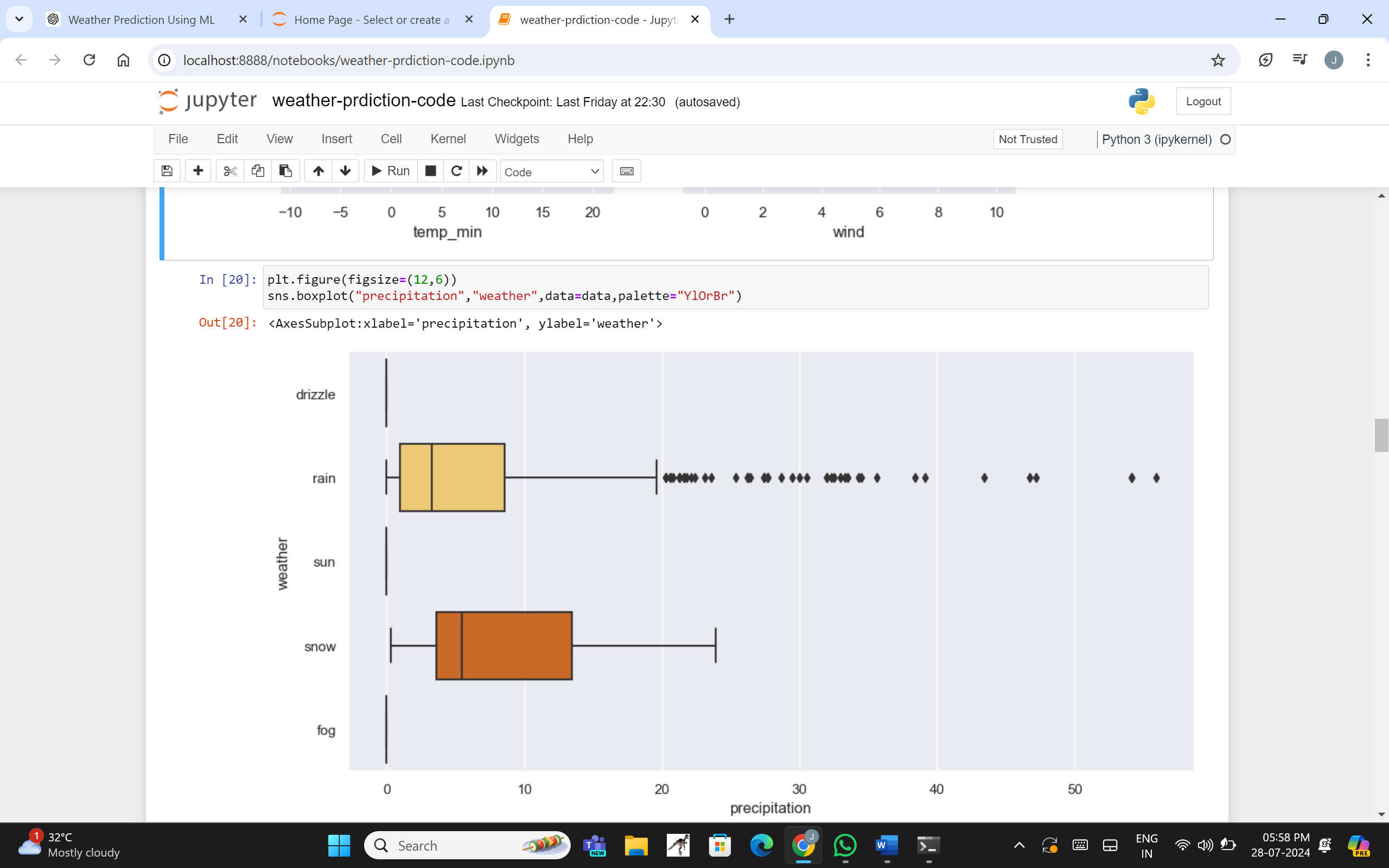
****

an empty form) and POST requests (validating and saving submitted data).

The register view function uses Django’s CustomUserCreationForm to collect user data, including username, password, and email.

15

****

****

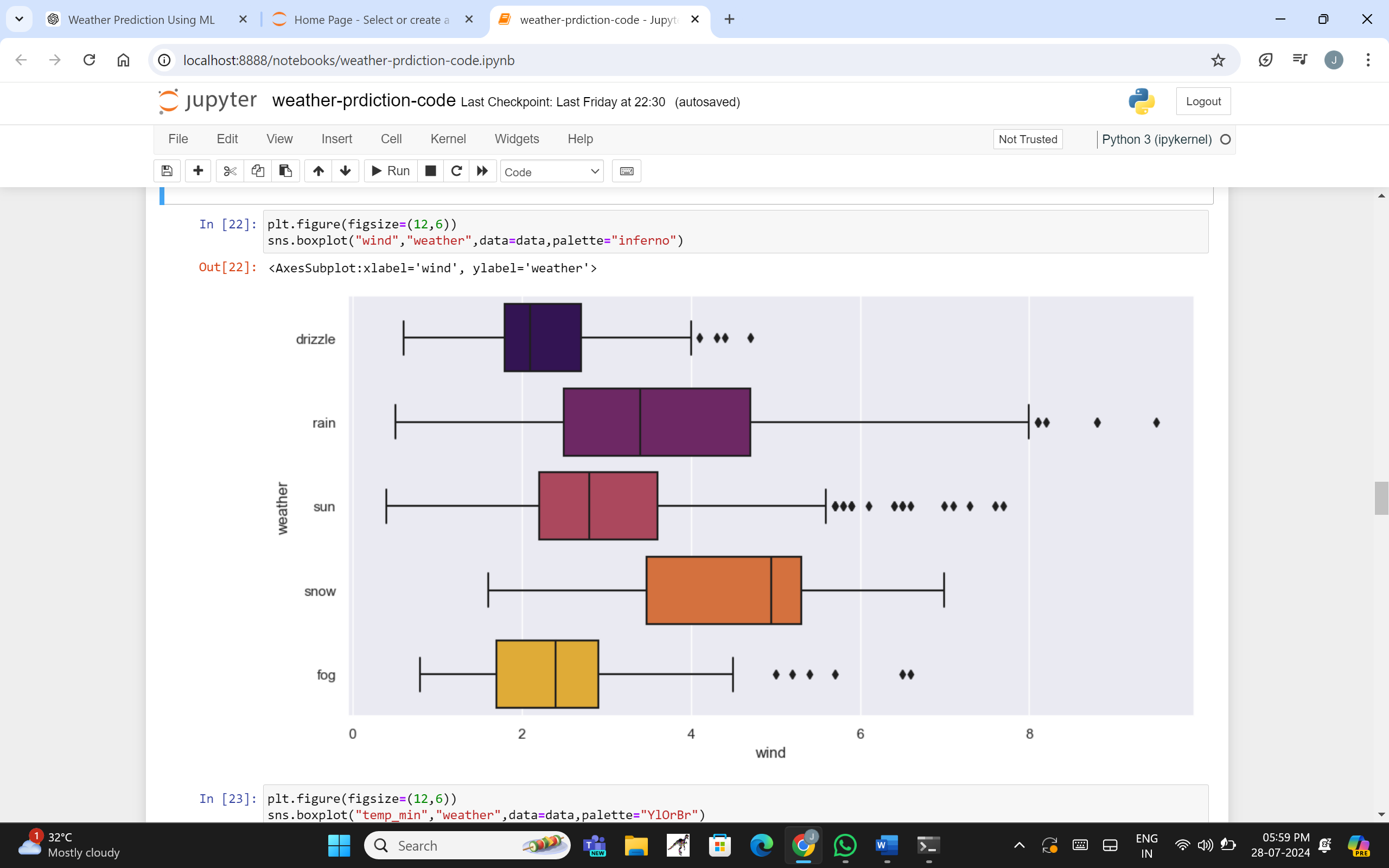
from django.shortcuts import render, redirect

from django.contrib.auth import login, authenticate

from .forms import CustomUserCreationForm

16

****

****

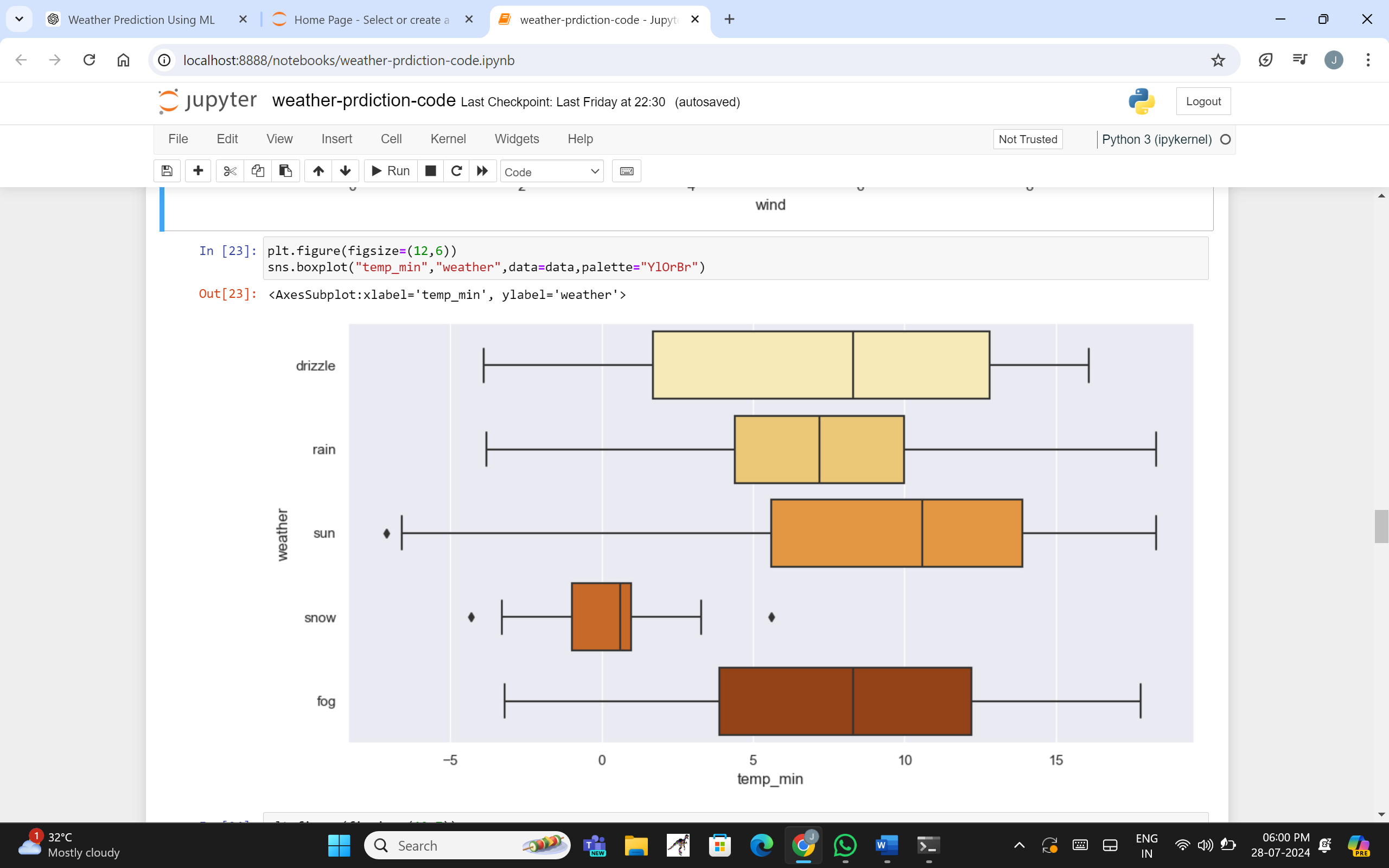
def register(request):

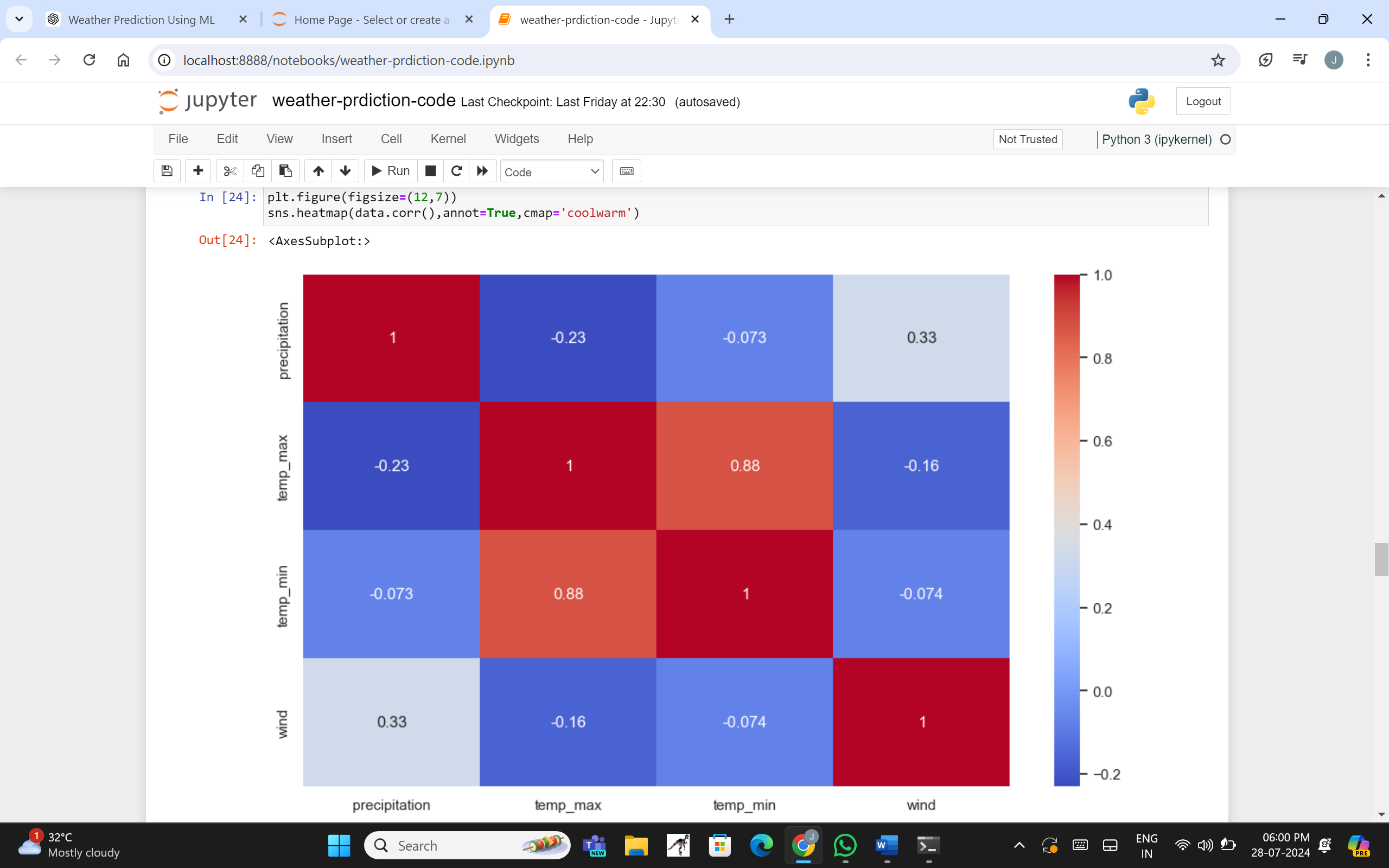
if request.method == 'POST': # Check if the form is submitted

form = CustomUserCreationForm(request.POST) # Populate the form with POST data

if form.is\_valid(): # Validate the form

17

****

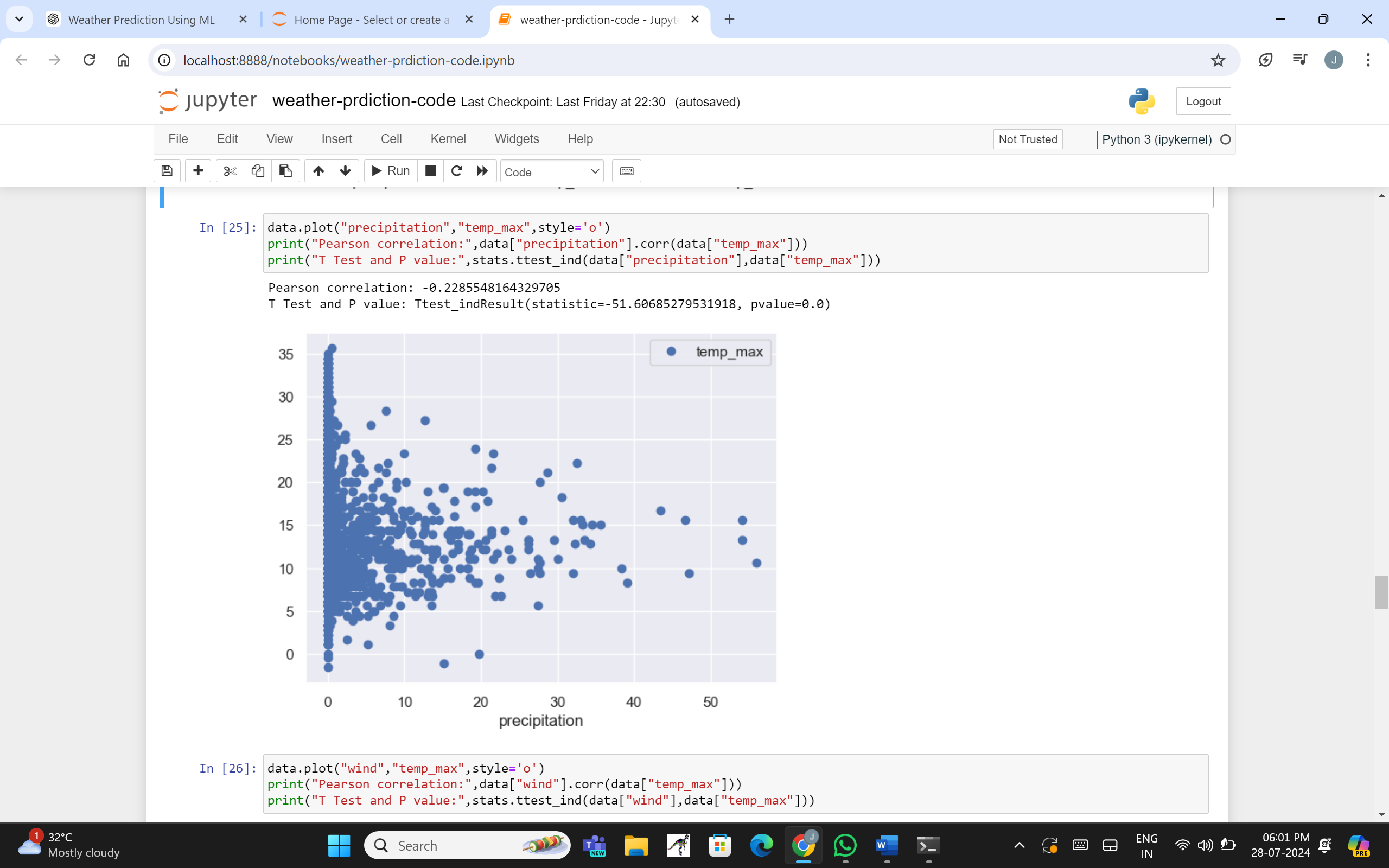
****

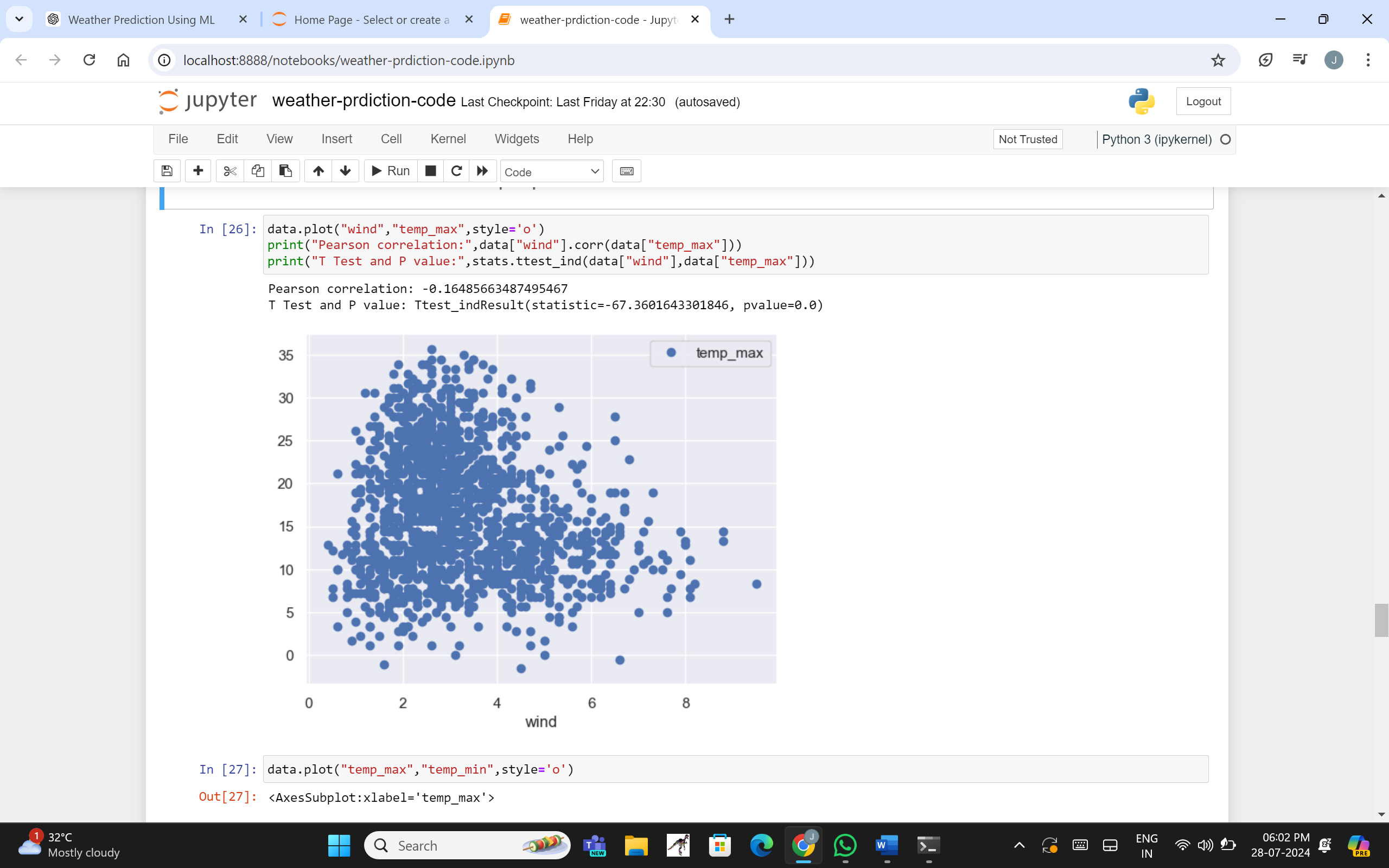
user = form.save() # Save the user to the database

raw\_password = form.cleaned\_data.get('password1') # Retrieve the raw password

user = authenticate(username=user.username, password=raw\_password) #Authenticate user

18

****

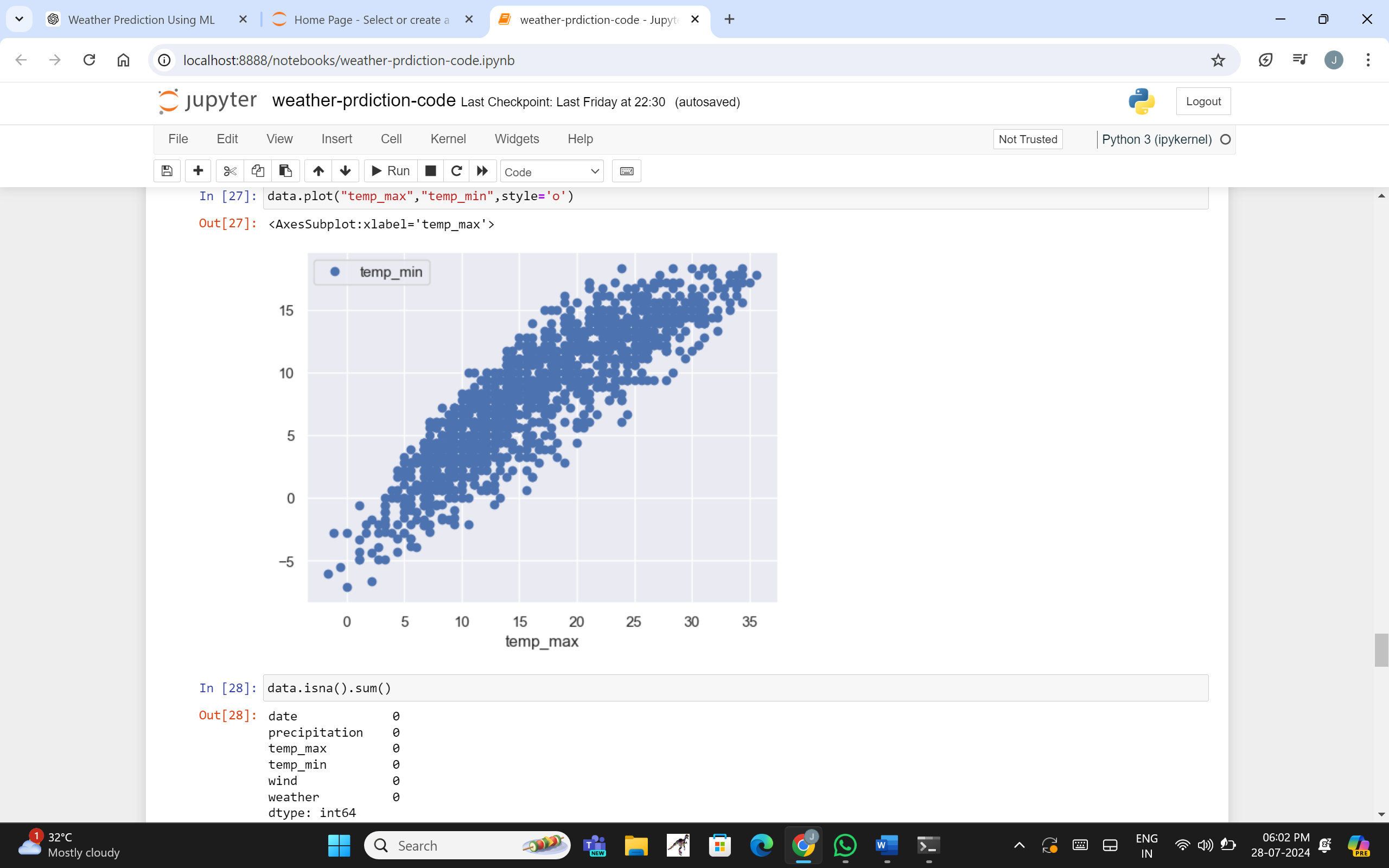
****

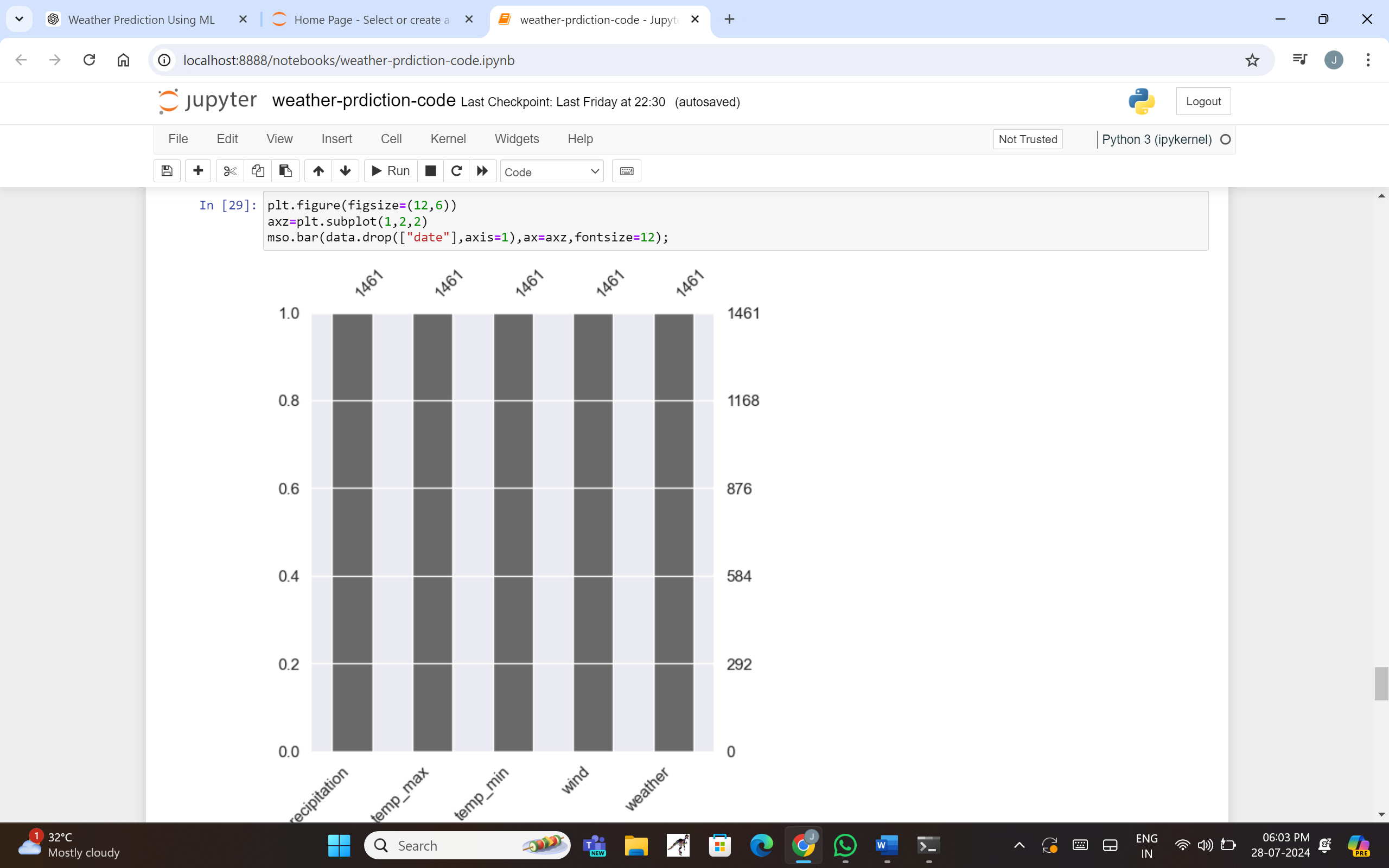
return redirect('home') # Redirect to homepage upon successful registration

else:

form = CustomUserCreationForm() # Display an empty form if the request is GET

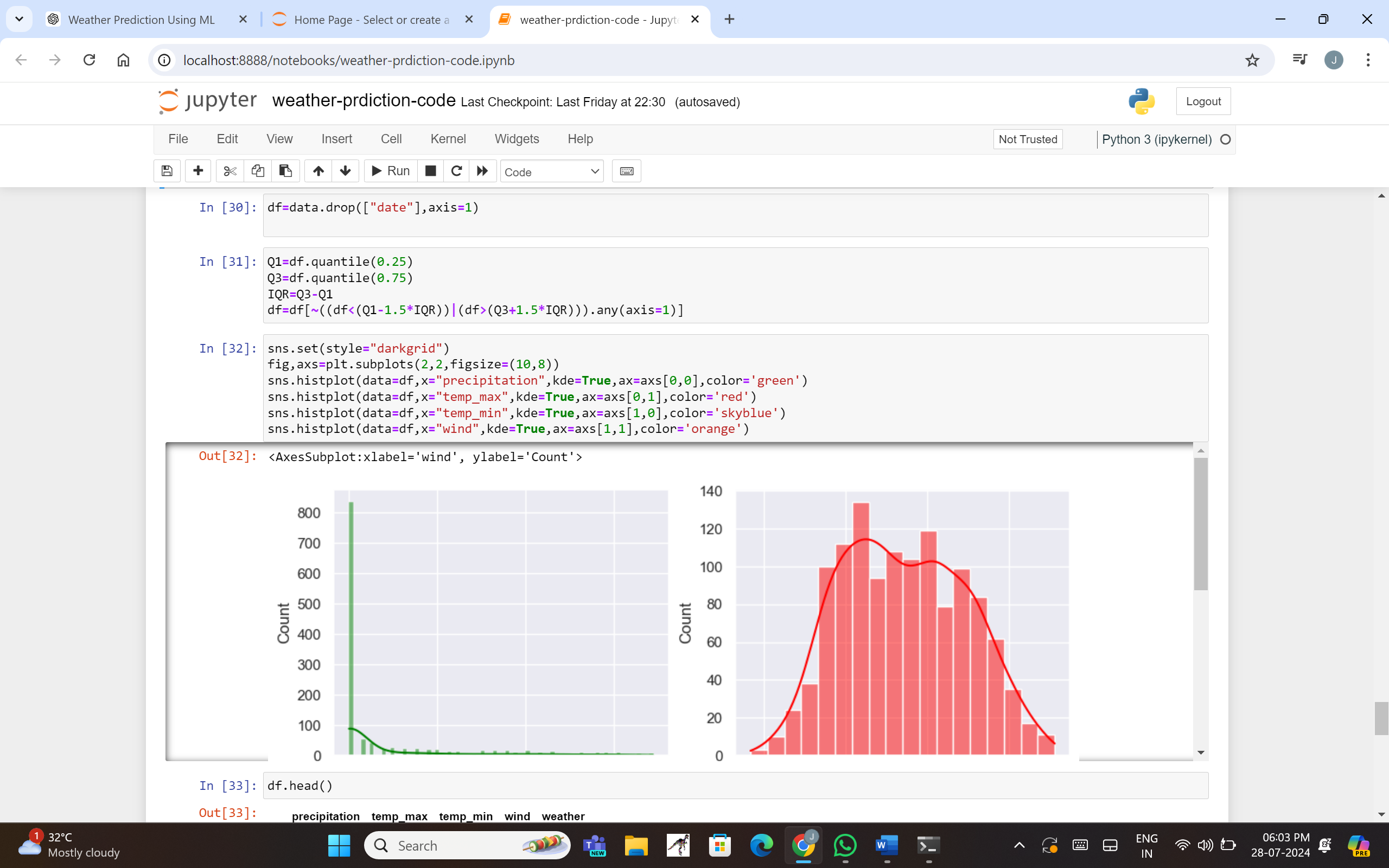
19

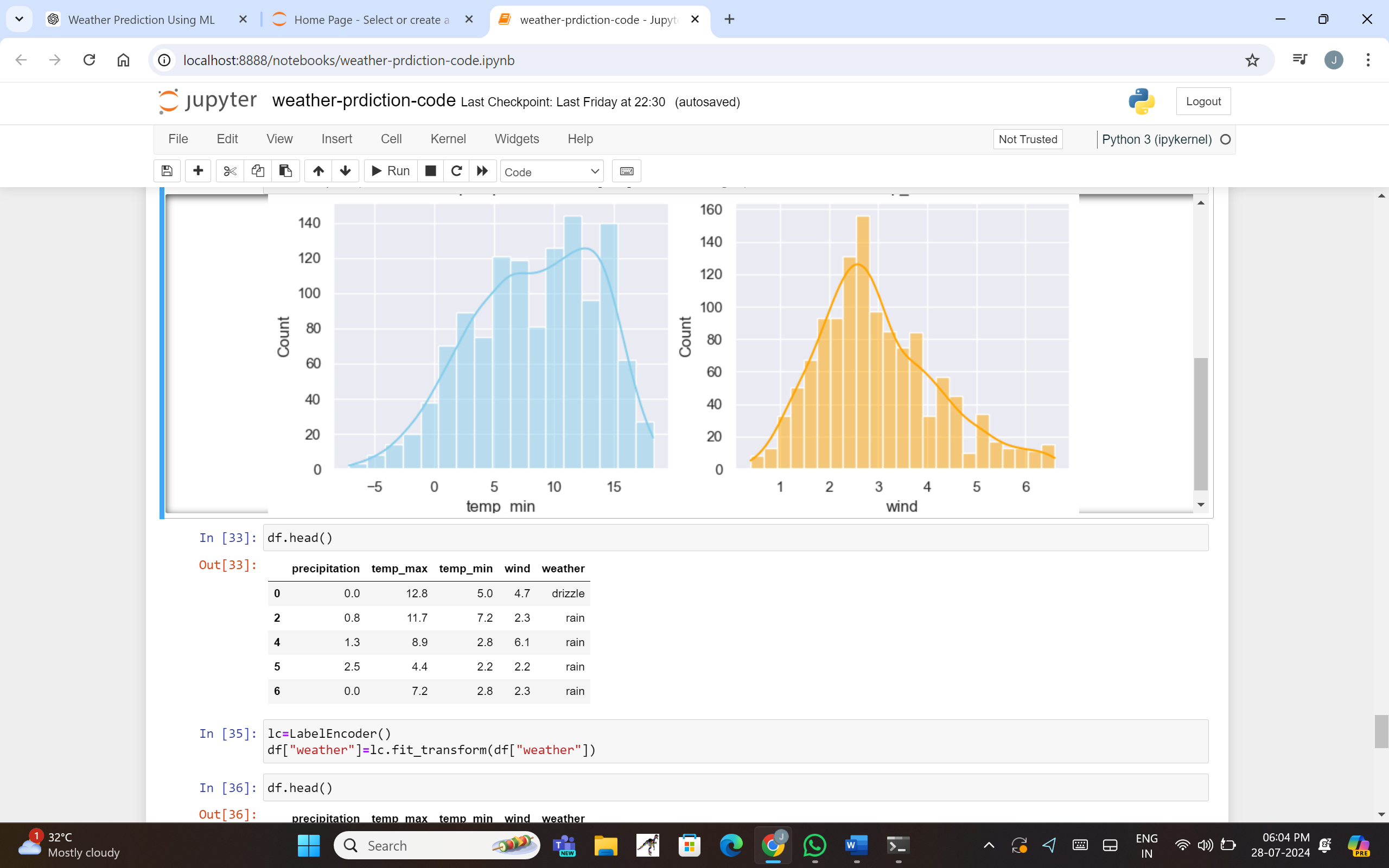
****

****

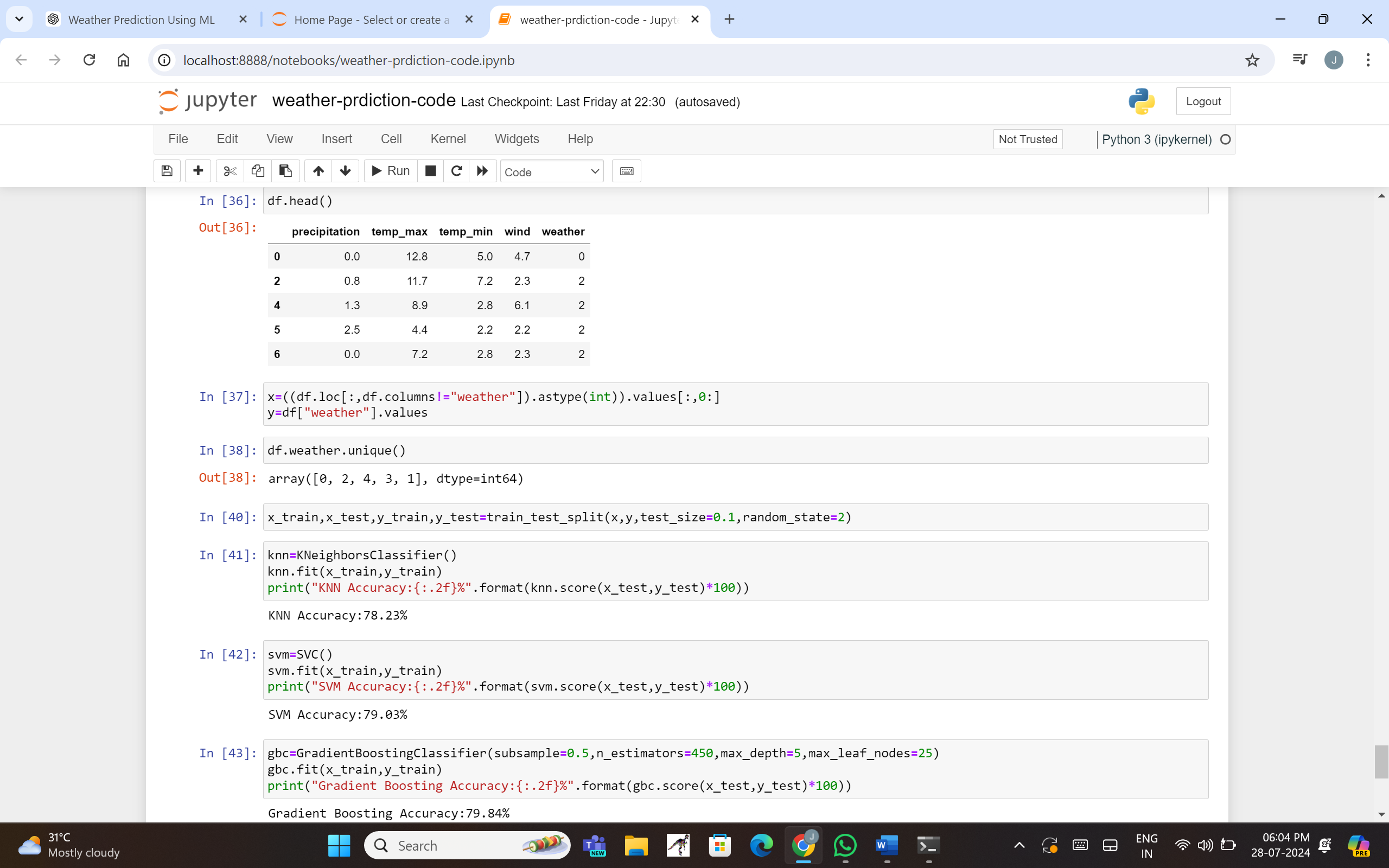
return render(request, 'accounts/register.html', {'form': form}) # Render the registration t

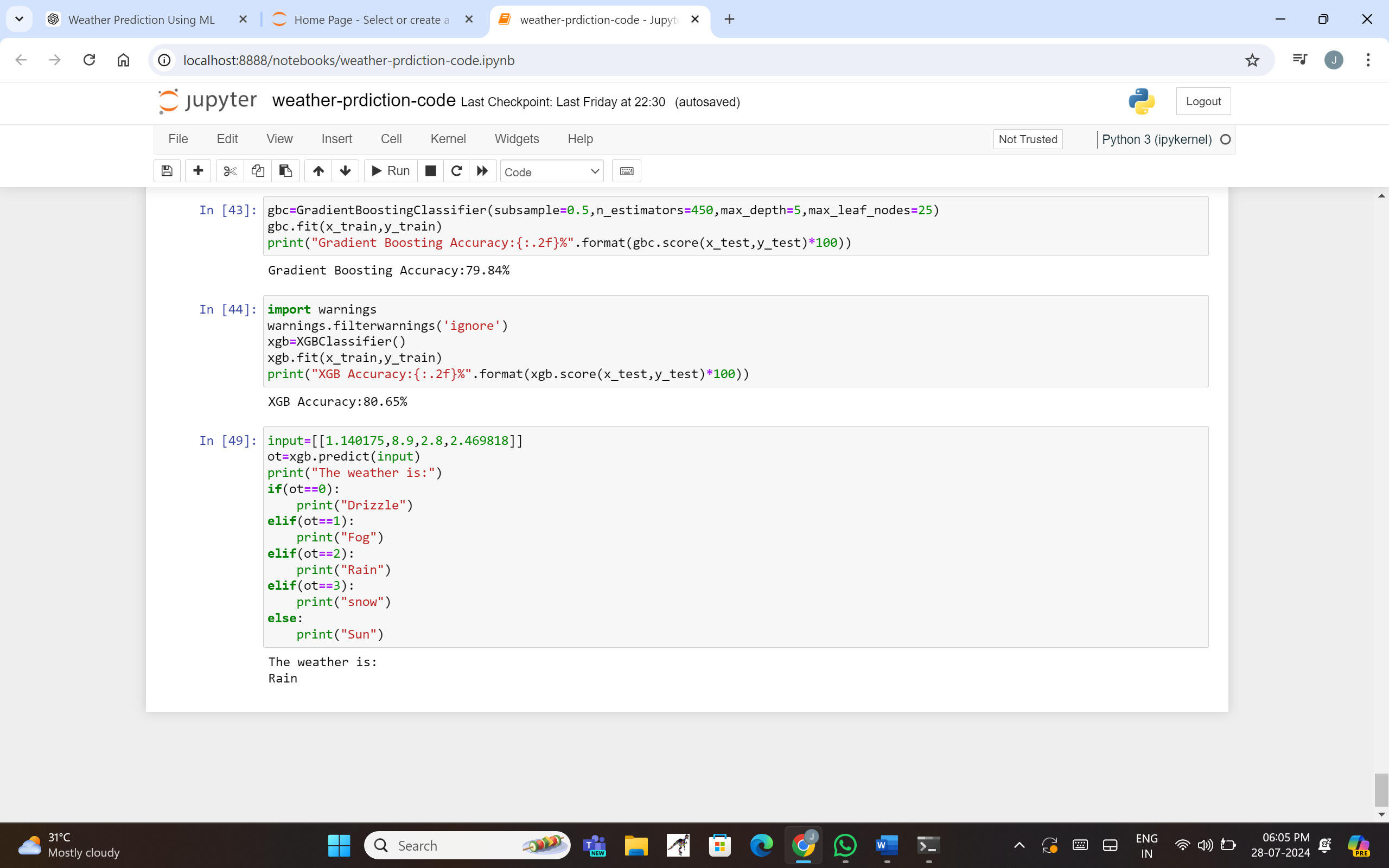
20

****

****

21

****

****

22

1. **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.

23

**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.

24