PROJECT 1

Weather Data Classification - Determining Humidity

Artificial Intelligence & Machine Learning

Table Of Content

Submitted By

PRAKASH N

E-mail ID:*prakashpoint2005@gmail.com*

1. Aim

This project aims to develop a predictive model that accurately forecasts the relative humidity at **3 pm** based on weather measurements taken at **9 am**. By leveraging machine learning techniques, specifically a Decision Tree classifier, the project seeks to provide an efficient and reliable method for predicting humidity levels. This can aid in various applications such as agricultural planning, weather forecasting, and environmental monitoring.

2. Introduction

Accurate weather forecasting is crucial for various sectors, with relative humidity being a key factor influencing agriculture, health, and daily activities.

This project aims to predict whether the relative humidity at 3 pm will exceed 24.99% using weather data collected at 9am.

The study involves importing and exploring the dataset to understand variable distributions and relationships, followed by preprocessing to handle missing values and engineer relevant features.

The data is then split into training and testing sets to build and evaluate a Decision Tree Classifier. Finally, the model is prepared for deployment to enable real-time predictions of afternoon humidity based on morning weather data.

By improving humidity forecasts, this study seeks to enhance decision-making processes in weather-dependent activities.

3. Technical Requirements

Hardware Specifications

- **Processor:** Intel Core i5 or higher
- **Memory:** 8 GB RAM or higher
- **Storage:** 256 GB SSD or higher
- **Graphics:** Integrated graphics card sufficient for data visualization tasks

Software Requirements and Environment Setup

- **Operating System:** Windows 10, macOS, or Linux
- **Programming Language:** Python 3.x
- **Development Environment:**
  - **Primary IDE:** Visual Studio Code (VS Code) with Jupyter Notebook integration
  - **Alternative:** Jupyter Notebook standalone, google colab

- **Libraries:**
  - **Data Handling and Analysis:** pandas, numpy
  - **Data Visualization:** matplotlib, seaborn
  - **Machine Learning:** scikit-learn
  - **Training Module:** DecisionTreeClassifier from the scikit-learn library.

- **VS Code Extensions:**

  - **Python Extension for Visual Studio Code:** Microsoft's official extension for Python development

  - **Jupyter Extension for Visual Studio Code:** Provides support for Jupyter Notebooks inside VS Code (Microsoft's official extension for Jupyter Notebooks like Jupiter, Jupiter keymap, jupyter power toys)

- **Additional Tools:**

  - **Version Control:** Git (optional, for version control and collaboration)
  - **Package Management:** pip or conda for managing Python packages

Suggested Tools / Tech Stacks

1. Python Programming Language
2. Jupyter Notebook for interactive data analysis
3. Visual Studio Code (VS Code) as a versatile IDE with Jupyter support
4. Essential libraries for data handling, visualization, and machine learning
5. Git for version control and collaboration

4. Modeling Approach

Model Selection

The Decision Tree Classifier was chosen as the model for predicting relative humidity at 3 pm based on morning weather data for several reasons:

1. **Interpretability:** Decision trees provide a clear and interpretable structure, making it easier to understand how each feature contributes to the prediction.
2. **Non-linearity Handling:** Decision trees can capture non-linear relationships between features and the target variable, which is beneficial when dealing with complex interactions in weather data.
3. **Feature Importance:** They automatically select the most important features for prediction, which helps in understanding which weather parameters are critical for determining high humidity levels.

4. **Robustness to Outliers:** Decision trees are robust to outliers and can handle data with different scales without requiring normalization.
5. **Easy to Implement:** Implementing and visualizing decision trees is straightforward, making them suitable for initial model exploration and analysis.

Training and Evaluation

1. **Training:** Split the dataset into training (70%) and testing (30%) sets using `train_test_split`, then train the Decision Tree Classifier on the training data.
2. **Evaluation:** Assess the model's accuracy in predicting high humidity levels using metrics like `accuracy_score`. Aim for high accuracy to ensure reliable predictions.

This approach ensures a clear and effective method for predicting high relative humidity based on morning weather conditions, providing actionable insights for decision-making.

5. Implementation Steps

1. Data Collection and Import

The analysis starts by importing the `daily_weather.csv` dataset, which includes essential weather variables like air pressure, temperature, wind direction and speed, rain accumulation, and humidity levels at different times.

```
import pandas as pd


# Load the dataset
data = pd.read_csv('daily_weather.csv')
```

My csv file

number	air_pressure_9am	air_temp_9am	avg_wind_direction_9am	avg_wind_speed_9am	max_wind_direction_9am	max_wind_speed_9am	rain_accumulation_9am	rain_duration_9am	relative_humidity_9am	relative_humidity_3pm
0	918.06	74.822	271.1	2.0803542	295.4	2.8632832	0	0	42.42	36.16
1	917.3476881	71.40384263	101.9351794	2.443009216	140.4715485	3.533323602	0	0	24.32869729	19.4265968
2	923.04	60.638	51	17.0678522	63.7	22.1009672	0	20	8.9	14.46
3	920.5027512	70.13889487	198.8321327	4.337363056	211.2033412	5.19004536	0	0	12.18910187	12.74254735
4	921.16	44.294	277.8	1.8566602	136.5	2.8632832	8.9	14730	92.41	76.74
5	915.3	78.404	182.8	9.9320136	189	10.9833754	0.02	170	35.13	33.93
6	915.5988675	70.04330432	177.8754072	3.745586538	186.6066959	4.589632429	0	0	10.65742166	21.38565673
7	918.07	51.71	242.4	2.5277422	271.6	3.6462122	0	0	80.47	74.92
8	920.08	80.582	40.7	4.5186188	63	5.8831522	0	0	29.58	24.03
9	915.01	47.498	163.1	4.9436374	195.9	6.5766036	0	0	88.6	68.05
10	919.65	77.036	70.6	3.8251674	85.5	4.7646822	0	0	22.07	32.13
11	915.64	45.716	241.6	5.8607828	265.8	8.0306146	0.55	1770	90.56	79.09
12	917.39	49.784	204.1	1.2750558	211.8	2.013246	0	0	73.15	58.43
13	920.82	62.438	213.6	2.6172198	165.7	3.3106712	0	0	43.64	27.99
14	911	86.432	202.9	1.2079476	162.9	1.677705	0	0	15.19	24.37
15	922.3831312	70.86526349	36.17417478	1.847278084	58.42863249	2.529142074	0	0	12.11088934	14.80170596
16	917.89		169.2	2.1922012	196.8	2.9303914	0	0	48.99	51.19
17	916.9152554	77.01896058	234.539345	2.274725297	229.4741987	2.906513109	0	0	21.03146177	20.75568332
18	918.8	67.082	176.1	4.8765292	183.4	5.5699806	0	0	18.9	45.87
19	922.04	68.576	58.3	9.5517338	81.9	12.5716028	0	0	7.54	7.74
20	919.9922622	62.96438312	54.79909362	12.68043574	74.25422349	15.45230581	0	0	18.80951786	14.64990936
21	917.23	67.676	177.8	2.460634	93.2	3.2883018	0	0	40.64	41.34
22	921.125626	68.81877189	71.79909228	2.576537767	95.47233356	3.487443587	0	0	11.74220123	17.28116117
23	920.35	47.57	192.1	6.263432	205.7	7.605596	0	0	54.55	66
24	921.7882289	71.65957221	217.4055201	1.946446506	253.758003	2.719711659	0	0	11.19424977	16.33171563
25	918.03	50.666	128.9	2.5277422	117.4	4.0041226	0	0	76.88	47.03

This includes importing the necessary libraries like,

```
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

→ **pandas (import pandas as pd)**:
- Handles data manipulation with DataFrames.

→ **accuracy_score (from sklearn.metrics import accuracy_score)**:
- Computes model accuracy.

→ **train_test_split (from sklearn.model_selection import train_test_split)**:
- Splits data for model training and evaluation.

→ **DecisionTreeClassifier (from sklearn.tree import DecisionTreeClassifier)**:
- Builds decision tree models for classification.

2. Data Cleaning and Preprocessing

Data cleaning and preprocessing are crucial to ensure the dataset's quality and suitability for modeling. This phase typically involves handling missing values, addressing outliers, and preparing features for analysis.

- **Handling Missing Values:** Missing data can introduce bias into our analysis. Therefore, we removed rows with any missing values to maintain the integrity of our dataset.

```
# Remove rows with any missing values
data.dropna(inplace=True)
```

- **Creating Labels:** To facilitate our prediction task, we created a binary label (High_humidity_label) based on whether the relative humidity at 3pm exceeded 24.99%. This label serves as our target variable for training the model.

```
# Create a binary label for high humidity (> 24.99%) at 3pm
data['High_humidity_label'] = (data['relative_humidity_3pm'] > 24.99).astype(int)
```

3. Model Selection and Training

DecisionTreeClassifier

- **Features and Target**:
  - Features include air pressure, temperature, wind directions and speeds, rain metrics, and relative humidity.

- Target is `High_humidity_label`, indicating whether relative humidity exceeds 24.99% at 3pm.

- **Data Splitting**:
  - **Purpose**: Split the data into training and testing sets.
  - **Details**: Uses `train_test_split` from sklearn to divide `X` and `y` into `X_train`, `X_test`, `y_train`, and `y_test`, with a test size of 33% for evaluation.

- **Model Initialization**:
  - DecisionTreeClassifier set with `max_leaf_nodes=5` and `random_state=0` for reproducibility.

- **Model Training**:
  - Fit `humidity_classifier` to `X_train` and `y_train` using `fit()` method.

```
# Features and target
features = ['air_pressure_9am', 'air_temp_9am', 'avg_wind_direction_9am', 'avg_wind_speed_9am',
            'max_wind_direction_9am', 'max_wind_speed_9am', 'rain_accumulation_9am',
            'rain_duration_9am', 'relative_humidity_9am']
X = data[features]
y = data['High_humidity_label']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Initialize Decision Tree Classifier
humidity_classifier = DecisionTreeClassifier(max_leaf_nodes=5, random_state=0)

# Train the classifier
humidity_classifier.fit(X_train, y_train)
```

4. Testing the Model

1. **Evaluation**: Assess model performance on test data (X_test, y_test).
2. **Accuracy Score**: Measure how accurately the model predicts outcomes using accuracy_score from sklearn.metrics.
3. **Prediction**: Use trained humidity_classifier to predict outcomes (y_predicted).

```
from sklearn.metrics import accuracy_score

# Predict outcomes on test data
y_predicted = humidity_classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_predicted) * 100
```

This process verifies how well the model predicts high humidity levels based on morning weather data.

5. Model Deployment

Deploying the model involves preparing it for real-time or batch predictions:

- **Saving the Model:** The trained Decision Tree Classifier was saved using joblib for future deployment.

- **Integration:** Integration into deployment pipelines or applications allows for seamless utilization in predicting afternoon humidity based on morning weather conditions.

6. Expected Output

This project aims to predict the humidity level at 3pm based on morning weather data. The expected output includes:

1. **Model Accuracy and Performance Metrics:**
    - Evaluation of the model's accuracy using metrics such as accuracy score, precision, recall, and confusion matrix.
    - **Accuracy Score**: Quantitative assessment of the model's overall correctness in predicting whether the relative humidity at 3pm will exceed 24.99%.
    - Insight into how well the model predicts whether the relative humidity at 3pm will exceed 24.99%.

2. **Predictive Insights:**
    - Example predictions showcasing scenarios where the model forecasts high or low humidity levels.
    - Interpretation of these predictions to understand their implications for various applications, such as agriculture, energy management, and outdoor activities planning.

3. **User Interaction and Application:**
    - Demonstration of how users can input morning weather parameters to obtain predictions for afternoon humidity.
    - Explanation of the model's feedback mechanism and how it supports decision-making processes.

7. Conclusion

In conclusion, this project utilized a Decision Tree Classifier to analyze morning weather data and predict afternoon humidity levels. By focusing on key meteorological features such as air pressure, temperature, wind direction, and humidity, the model was trained to forecast whether humidity would exceed 24.99% at 3pm. Through rigorous data cleaning and feature selection, we ensured the model's readiness for training and testing.

Evaluating its performance on unseen data revealed an accuracy rate of approximately 80%, demonstrating its effectiveness in predicting high humidity conditions. This predictive capability holds practical implications for weather monitoring and planning, offering insights that can aid in proactive decision-making across various sectors. Looking forward, further enhancements and exploration of alternative algorithms could potentially refine accuracy and broaden the model's application in meteorology and related fields.

8. Reference

Daily Weather Observations Dataset

Link: https://drive.google.com/file/d/1W7xQyN0wa2jdTNQtThBTIIzqn8bRiy-u/view?usp=sharing

My project Location

Link: https://drive.google.com/file/d/1fn3qqxKVJcoRr79Kdk9kz1MuJHLmdLAr/view?usp=sharing

Jupyter Notebook

Link 1:

Name: Jupyter
Id: ms-toolsai.jupyter
Description: Jupyter notebook support, interactive programming and computing that supports Intellisense, debugging and more.
Version: 2024.5.0
Publisher: Microsoft
VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter

Link 2: https://jupyter.org

Visual Studio Code

Link: https://code.visualstudio.com/download

Libraries

Scikit-learn:

Link: https://scikit-learn.org/stable/index.html

Pandas

Link: https://pandas.pydata.org/docs/

Decision Trees:

Ref: DOI: 10.1109/TSMCC.2010.2101770

Accuracy Score in Scikit-learn

Link: https://scikitlearn.org/stable/modules/generated/sklearn.metrics.accuracy_scope.html