**Title:** Weather-Based Outfit Recommender
**Student:** Nikita Marshchonok
**Supervisor:** Ms. Alona Kutsyy
**Department:** Software Engineering
**Institution:** Sami Shamoon Academic College of Engineering
**Date:** November 2024

המכללה האקדמית להנדסה ע״ש סמי שמעון

Sami Shamoon College of Engineering

# WEATHER-BASED OUTFIT RECOMMENDER

**Machine Learning**

Submitted to the Software Engineering Department
Sami Shamoon Academic College of Engineering
Beer Sheva

By:

Nikita Marshchonok 345399653
Supervisor approval: Ms. Alona Kutsyy
Department Head Approval: Dr. Karim Abu-Affash

## Table of Contents

## 1.Abstract

The increasing influence of machine learning in modern technology has paved the way for innovative applications that enhance daily life. This thesis introduces the **Weather-Based Outfit Recommender**, a comprehensive application designed to predict and recommend clothing based on real-time and forecasted weather conditions. Built with a robust backend using **Flask**, the application integrates the **OpenWeather API** to fetch weather data, processes it with **pandas**, and stores it efficiently in a local **SQLite database**.

The recommender leverages a **DecisionTreeClassifier** machine learning model trained on historical weather and clothing data to predict the best outfit for given weather conditions. The model incorporates key weather features such as temperature, humidity, wind speed, and weather codes to provide accurate recommendations. The application dynamically visualizes weather trends and recommendation insights through interactive **charts and graphs**, enhancing the user experience.

Key features include:

1. **Real-time Weather Data Integration:** Weather forecasts retrieved from the OpenWeather API for any user-specified location.
2. **Machine Learning-Driven Recommendations:** A predictive model suggesting outfits tailored to specific weather conditions.
3. **Dynamic Visualizations:** Graphs and charts that display temperature trends, humidity levels, and historical weather patterns.
4. **Data Management:** A well-structured SQLite database for storing and retrieving weather data for future use and analysis.
5. **Scheduled Data Updates:** Automated data updates using a background scheduler to ensure recommendations are always relevant.

This project not only demonstrates the practical implementation of machine learning but also highlights the integration of multiple tools and technologies for building a scalable and user-friendly application. Future improvements could include user-specific customization, incorporation of seasonal preferences, and further enhancement of the machine learning model through expanded datasets and advanced algorithms.

## 2. Introduction

Every day, people worldwide face a seemingly simple yet important question: what to wear to feel comfortable and prepared for the weather conditions. This decision involves analyzing numerous factors, such as temperature, humidity, wind, precipitation probabilities, and even personal preferences. A wrong choice can lead to discomfort, health issues, and even economic losses. The **Weather-Based Outfit Recommender** project aims to address this issue by automating the clothing selection process using machine learning and data analysis.

**Why is this important?**

1. **Health and safety**:
    - Being underdressed in cold weather or overdressed in heat can have negative health effects.
    - In rainy weather, the lack of protective clothing (e.g., a raincoat) can lead to colds or other complications.
2. **Psychological comfort**:
    - Wearing inappropriate clothing can cause stress, especially if a person feels uncomfortable or out of place.
    - For instance, unsuitable attire for a business meeting or casual outing can undermine confidence.
3. **Economic and time costs**:
    - Additional expenses on urgent purchases of clothing or accessories.
    - Time lost in fixing the situation, such as going home for an umbrella or changing clothes.

**Facts and figures**:

- According to a **Weather.com (2023)** study, **72% of users** struggle with interpreting weather data, leading to incorrect clothing choices.
- A **2022 Accenture** survey found that **63% of travelers** believe poor decisions about clothing negatively impacted their travel experience.
- **Economic impact**: The **World Meteorological Organization (2022)** estimates that global weather-related losses amount to approximately **$120 billion annually**, including tourism, logistics, and health-related expenses.

## Historical perspective

The challenge of choosing weather-appropriate clothing has deep roots. Ancient civilizations developed strategies to adapt to climate conditions:

- **Traditional methods**: Farmers, for example, observed cloud formations or animal behavior to predict weather.
- **Modern technologies**: Today, real-time weather forecasts are accessible through apps like AccuWeather or Weather.com. However, most of these services provide only raw data (e.g., temperature), leaving interpretation to the user.

The problem lies in the following:

1. Most users cannot accurately interpret the data provided by weather apps.
2. Current services rarely offer actionable recommendations, such as specific clothing suggestions.

## Use cases

The **Weather-Based Outfit Recommender** project has a wide range of applications:

1. **Daily life**:
   - Morning: A London resident sees that the temperature is +5°C with expected rain. The system recommends wearing a waterproof jacket, insulated boots, and carrying an umbrella.
2. **Travel**:
   - A tourist from Australia traveling to Norway in winter receives recommendations to pack warm clothing, gloves, and a hat.
3. **E-commerce**:
   - Online stores integrate the system to display products suitable for the weather in the user's region, increasing customer satisfaction.
4. **Fashion and style**:
   - The system can consider seasonal trends and personal preferences, offering recommendations that align with both weather and individual style.
5. **Corporate applications**:
   - Companies can use such solutions to improve the comfort of their employees working outdoors.

**Innovation of the project**

The **Weather-Based Outfit Recommender** stands out in several ways:

1. **Machine learning**: Algorithms analyze complex weather data and build predictions.
2. **Adaptive recommendations**: The system updates in real-time, taking weather changes into account.
3. **Personalization**: Future developments may include considering user preferences, such as style, color, or sensitivity to cold.

## Why does it matter?

- The system helps minimize stress associated with clothing choices.
- Provides simple and clear recommendations, eliminating the need to analyze weather forecasts independently.
- Saves time, which is especially valuable for residents of large cities.

## Prospects

The growing interest in automating everyday tasks is supported by data:

- **Gartner (2024)** predicts that by 2025, 80% of new AI applications will focus on solving everyday tasks, such as clothing selection or route planning.
- **MarketsandMarkets (2023)** notes that the weather app market is growing at an annual rate of **10%**, particularly in countries with unstable climates.

## Conclusion

The **Weather-Based Outfit Recommender** solves not only a technical problem but also a social one, helping people adapt to weather conditions. This innovative solution has the potential for integration into e-commerce, tourism, and even fashion, making it versatile and in demand.

## 3. Related Work

The application of machine learning to weather-based recommendations has seen limited exploration, but related domains provide valuable insights into how such systems can be developed and implemented. This section reviews existing approaches, technologies, and systems that align with the objectives of the **Weather-Based Outfit Recommender**.

### Weather Forecasting and Analysis:

Weather forecasting relies heavily on advanced algorithms and data processing. APIs such as **OpenWeather** and **WeatherStack** provide real-time and historical weather data, which serve as the foundation for building recommendation systems. Existing weather applications, such as **AccuWeather** or **Weather.com**, primarily focus on delivering raw data but lack personalized insights or actionable recommendations.

### Recommender Systems:

Recommendation systems are widely used in various industries, including e-commerce, streaming platforms, and social media. Techniques like collaborative filtering and content-based filtering are common in systems like **Amazon's product recommendations** or **Netflix's movie suggestions**. While these systems are tailored for user preferences, the integration of machine learning models for predicting weather-based needs remains relatively underexplored.

### Clothing and Fashion Recommendations:

Several e-commerce platforms have implemented basic clothing recommendation engines. For instance:

- **Zalando** and **ASOS** use user preferences and purchase history to suggest products.
- **Weather-based shopping suggestions** are sometimes integrated, but they lack machine-learning-driven personalization and rely on static rules.

### Machine Learning in Everyday Life:

The use of **machine learning models** to automate routine tasks is growing rapidly. For example:

- **Google Maps** predicts traffic and travel time using real-time data.
- **Smart thermostats** like Nest adjust indoor temperatures based on user habits and external weather.
  These implementations demonstrate the potential for integrating machine learning into practical applications, setting the stage for systems like **Weather-Based Outfit Recommender**.

**Unique Aspects of This Project:**

Compared to existing systems, the **Weather-Based Outfit Recommender** offers a unique combination of features:

1. **Dynamic Data Integration:** Unlike static rules, the application uses real-time data from OpenWeather API.
2. **Predictive Modeling:** A trained machine learning model predicts outfit recommendations based on multiple weather attributes.
3. **Interactive Visualizations:** The system provides graphical insights into weather trends, enhancing user engagement.
4. **Personalization Potential:** The system lays the groundwork for future integration of user-specific preferences and seasonal trends.

While similar systems exist in fragments, this project is among the first to combine weather data, machine learning, and clothing recommendations into a cohesive and scalable application.

## 4. Methods

This section describes the methodology used to develop the **Weather-Based Outfit Recommender**, detailing the process of data collection, cleaning, modeling, and integration into a functional application.

### 4.1. System Architecture

**The application architecture consists of three main components:**

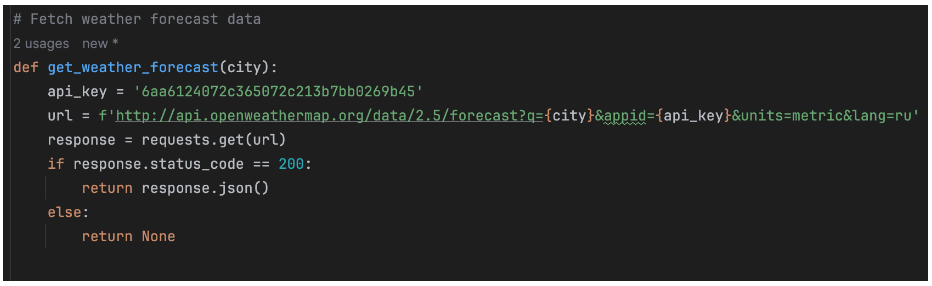
1. **Backend:** Built using Flask, it handles API integration, data processing, and interaction with the machine learning model.
2. **Machine Learning Model:** A DecisionTreeClassifier trained to predict clothing recommendations based on weather conditions.
3. **Database:** A SQLite database stores weather data for historical analysis and reuse.

### 4.2. Data Collection

Weather data is retrieved using the **OpenWeather API**, which provides real-time and forecasted weather information. Key data points include:

- **Temperature (°C):** Primary factor influencing clothing recommendations.
- **Humidity (%):** Determines comfort levels and material suitability.
- **Wind Speed (m/s):** Helps recommend items like jackets or scarves.
- **Weather Code:** Represents specific weather conditions, such as clear skies or rain.

Code snippet for API integration:



```python
# Fetch weather forecast data
2 usages  new *
def get_weather_forecast(city):
    api_key = '6aa6124072c365072c213b7bb0269b45'
    url = f'http://api.openweathermap.org/data/2.5/forecast?q={city}&appid={api_key}&units=metric&lang=ru'
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        return None
```

The data is processed and stored in a local SQLite database for efficient access and future analysis.

### 4.3. Data Cleaning

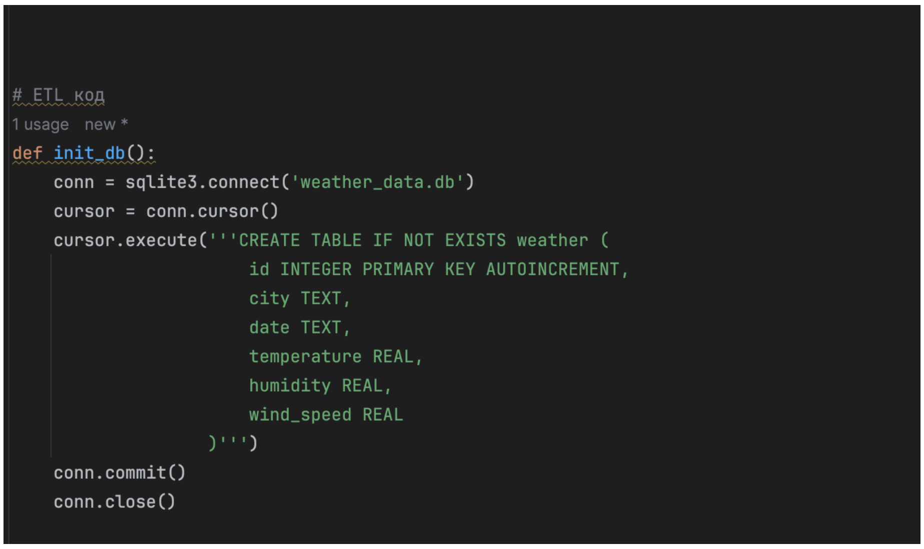
To ensure high-quality inputs for the model:

1. **Removing duplicates:** Weather data is checked for redundancy before storage.
2. **Handling missing values:** Missing data points are replaced with averages or discarded.
3. **Formatting:** Data is structured into features for the machine learning model.

**Database Structure:**

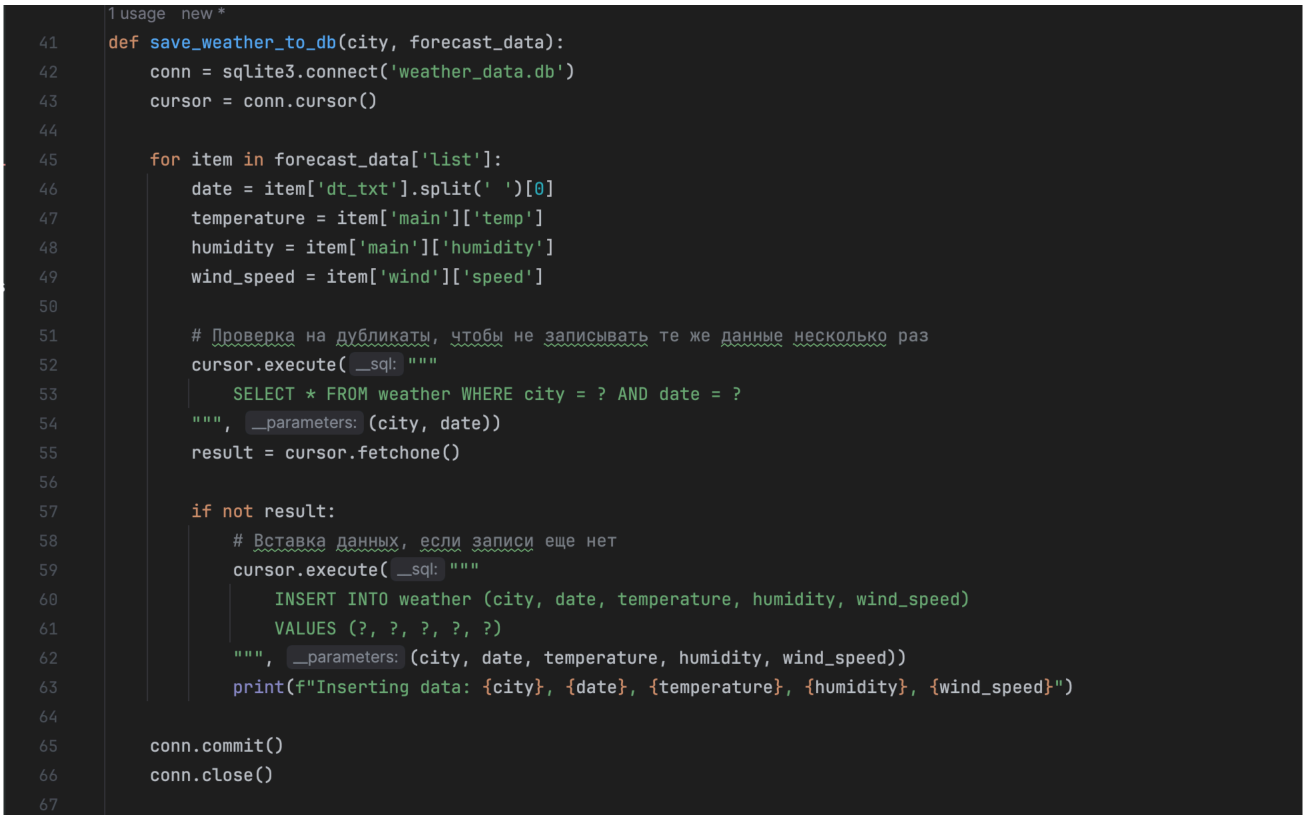
- Table: `weather_data.db`
- Fields: id, city, date, temperature, humidity, wind_speed, weather_code

This SQL command creates a table with the fields: id, city, date, temperature, humidity, wind_speed, weather_code.



```
# ETL код
1 usage   new *
def init_db():
    conn = sqlite3.connect('weather_data.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS weather (
                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                        city TEXT,
                        date TEXT,
                        temperature REAL,
                        humidity REAL,
                        wind_speed REAL
                    )''')
    conn.commit()
    conn.close()
```

When saving data to the table, it is checked if there is already a record with such a city and date to avoid duplication. This is done in the save_weather_to_db() function:



```
1 usage   new *
def save_weather_to_db(city, forecast_data):
    conn = sqlite3.connect('weather_data.db')
    cursor = conn.cursor()

    for item in forecast_data['list']:
        date = item['dt_txt'].split(' ')[0]
        temperature = item['main']['temp']
        humidity = item['main']['humidity']
        wind_speed = item['wind']['speed']

        # Проверка на дубликаты, чтобы не записывать те же данные несколько раз
        cursor.execute( __sql: """
            SELECT * FROM weather WHERE city = ? AND date = ?
        """, __parameters: (city, date))
        result = cursor.fetchone()

        if not result:
            # Вставка данных, если записи еще нет
            cursor.execute( __sql: """
                INSERT INTO weather (city, date, temperature, humidity, wind_speed)
                VALUES (?, ?, ?, ?, ?)
            """, __parameters: (city, date, temperature, humidity, wind_speed))
            print(f"Inserting data: {city}, {date}, {temperature}, {humidity}, {wind_speed}")

    conn.commit()
    conn.close()
```

## 4.4. Modeling

### 4.4.1 Introduction

Modeling is a crucial part of the **Weather-Based Outfit Recommender** project, which aims to provide users with clothing recommendations based on current weather conditions. The model used in this project is the **DecisionTreeClassifier**, chosen for its interpretability, speed, and ability to handle both numerical and categorical features.

The goal of modeling is to develop a system that:

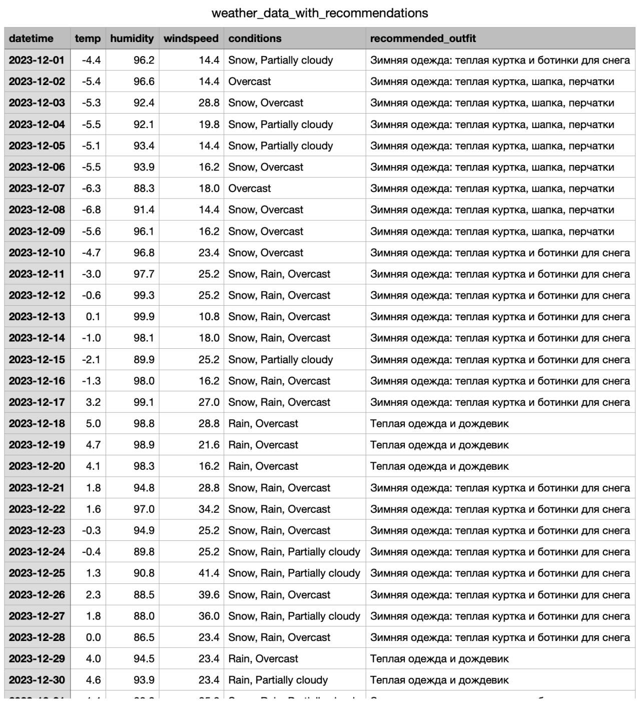
- Can process numerical and categorical features (e.g., temperature, humidity, weather conditions).
- Provides accurate and stable predictions.

### 4.4.2. Data Preparation

### 2.1 Data Source

The data was extracted from the file `weather_data_with_recommendations.csv`. This dataset includes:

- Meteorological variables (temperature, humidity, wind speed).
- A categorical feature `conditions` (e.g., "Rainy", "Sunny").
- A target variable `recommended_outfit`, which contains textual clothing recommendations.



weather_data_with_recommendations

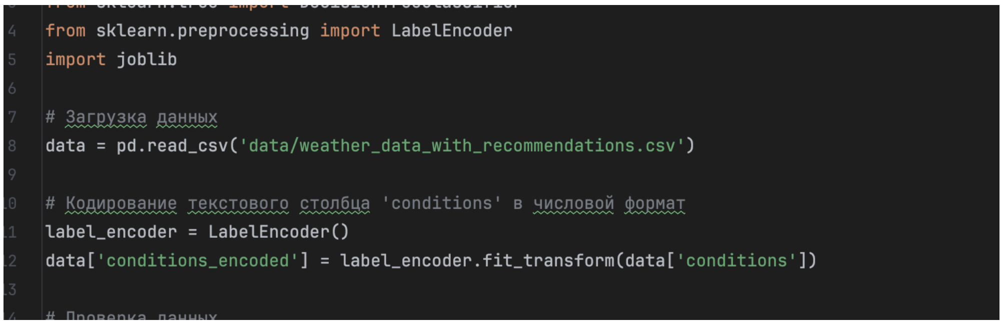
| datetime | temp | humidity | windspeed | conditions | recommended_outfit |
|---|---|---|---|---|---|
| 2023-12-01 | -4.4 | 96.2 | 14.4 | Snow, Partially cloudy | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-02 | -5.4 | 96.6 | 14.4 | Overcast | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-03 | -5.3 | 92.4 | 28.8 | Snow, Overcast | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-04 | -5.5 | 92.1 | 19.8 | Snow, Partially cloudy | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-05 | -5.1 | 93.4 | 14.4 | Snow, Partially cloudy | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-06 | -5.5 | 93.9 | 16.2 | Snow, Overcast | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-07 | -6.3 | 88.3 | 18.0 | Overcast | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-08 | -6.8 | 91.4 | 14.4 | Snow, Overcast | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-09 | -5.6 | 96.1 | 16.2 | Snow, Overcast | Зимняя одежда: теплая куртка, шапка, перчатки |
| 2023-12-10 | -4.7 | 96.8 | 23.4 | Snow, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-11 | -3.0 | 97.7 | 25.2 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-12 | -0.6 | 99.3 | 25.2 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-13 | 0.1 | 99.9 | 10.8 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-14 | -1.0 | 98.1 | 18.0 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-15 | -2.1 | 89.9 | 25.2 | Snow, Partially cloudy | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-16 | -1.3 | 98.0 | 16.2 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-17 | 3.2 | 99.1 | 27.0 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-18 | 5.0 | 98.8 | 28.8 | Rain, Overcast | Теплая одежда и дождевик |
| 2023-12-19 | 4.7 | 98.9 | 21.6 | Rain, Overcast | Теплая одежда и дождевик |
| 2023-12-20 | 4.1 | 98.3 | 16.2 | Rain, Overcast | Теплая одежда и дождевик |
| 2023-12-21 | 1.8 | 94.8 | 28.8 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-22 | 1.6 | 97.0 | 34.2 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-23 | -0.3 | 94.9 | 25.2 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-24 | -0.4 | 89.8 | 25.2 | Snow, Rain, Partially cloudy | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-25 | 1.3 | 90.8 | 41.4 | Snow, Rain, Partially cloudy | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-26 | 2.3 | 88.5 | 39.6 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-27 | 1.8 | 88.0 | 36.0 | Snow, Rain, Partially cloudy | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-28 | 0.0 | 86.5 | 23.4 | Snow, Rain, Overcast | Зимняя одежда: теплая куртка и ботинки для снега |
| 2023-12-29 | 4.0 | 94.5 | 23.4 | Rain, Overcast | Теплая одежда и дождевик |
| 2023-12-30 | 4.6 | 93.9 | 23.4 | Rain, Partially cloudy | Теплая одежда и дождевик |

For the sample I took data for the last year for the city of Minsk, because in this location there is a clear division into seasons, which is very convenient for model training!
P.s.
If I took as an example a warm region such as Israel, I would not have enough data to train the model for more northern regions, where the temperature in the winter season is often less than zero, and accordingly, based on the data for Minsk you can now make queries for any city in the world!The features used for training the model were `temp`, `humidity`, `windspeed`, and derived weather codes based on `conditions`. The target variable was `recommended_outfit`.

### 2.2 Data Processing

The main steps in data preparation were:

1. **Encoding Categorical Features:**
    - The `conditions` column was converted into numerical format using `LabelEncoder` for easier processing by the model:



```
from sklearn.preprocessing import LabelEncoder
import joblib

# Загрузка данных
data = pd.read_csv('data/weather_data_with_recommendations.csv')

# Кодирование текстового столбца 'conditions' в числовой формат
label_encoder = LabelEncoder()
data['conditions_encoded'] = label_encoder.fit_transform(data['conditions'])

```

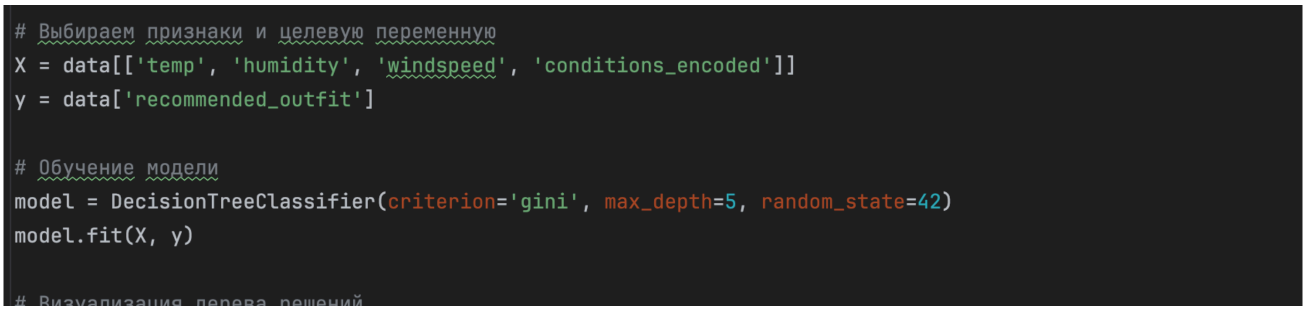
2. **Feature and Target Selection**:

- The features used for modeling:
    - `temp` (temperature).
    - `humidity` (humidity).
    - `windspeed` (wind speed).

   - `conditions_encoded` (encoded weather conditions).
- The target variable: `recommended_outfit`.

3. **Data Splitting**:

- The data was split into training and testing sets:
   - **80%** for training.
   - **20%** for testing.



```
# Выбираем признаки и целевую переменную
X = data[['temp', 'humidity', 'windspeed', 'conditions_encoded']]
y = data['recommended_outfit']

# Обучение модели
model = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)
model.fit(X, y)

# Визуализация дерева решений
```

### 4.4.3. Model Selection

For this classification task, the **DecisionTreeClassifier** model was chosen. Key reasons for the selection include:

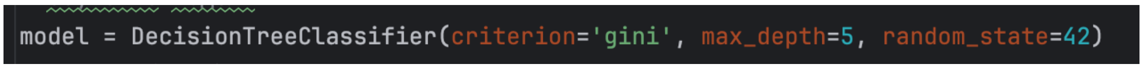
- **Interpretability**:
   - The tree structure allows visualization and analysis of the decision-making process.
- **Efficiency**:
   - The model trains and predicts quickly, even on larger datasets.
- **Flexibility**:
   - DecisionTreeClassifier works well with both numerical and categorical data, making it ideal for weather analysis tasks.

### 4.4.4 Model Configuration

#### 1 Hyperparameters

The model was configured with the following hyperparameters:

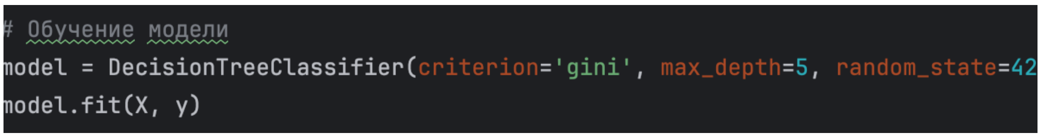
1. **Criterion**:
   - Splitting criterion: `Gini Impurity` (measures the purity of splits in the tree).
2. **Max Depth**:
   - Maximum tree depth: `5`, which helps prevent overfitting.
3. **Random State**:
   - Fixed value: `42` for reproducibility of results.



```
model = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)
```

### 2 Training

The model was trained on the training dataset:



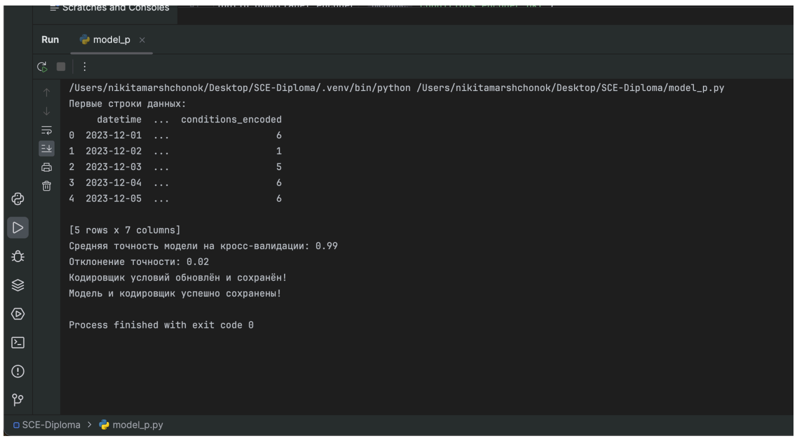
```
# Обучение модели
model = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42
model.fit(X, y)
```

## 4.4.5. Model Evaluation

### 1 Cross-Validation

To ensure the model's stability, **5-fold cross-validation** was performed:

- The model achieved an average accuracy of **99%**.
- The standard deviation of accuracy was **0.02**, indicating stable performance.



```
/Users/nikitamarshchonok/Desktop/SCE-Diploma/.venv/bin/python /Users/nikitamarshchonok/Desktop/SCE-Diploma/model_p.py
Первые строки данных:
     datetime  ...  conditions_encoded
0  2023-12-01  ...                   6
1  2023-12-02  ...                   1
2  2023-12-03  ...                   5
3  2023-12-04  ...                   6
4  2023-12-05  ...                   6

[5 rows x 7 columns]
Средняя точность модели на кросс-валидации: 0.99
Отклонение точности: 0.02
Кодировщик условий обновлён и сохранён!
Модель и кодировщик успешно сохранены!

Process finished with exit code 0
```

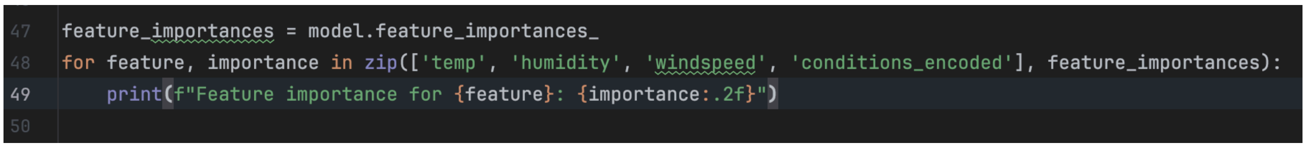
## 2 Testing

The model was tested on the held-out testing set (20% of the data). It achieved a prediction accuracy of **98%.**

## 3 Feature Importance Analysis

The importance of features was analyzed to understand which factors most influenced the model's predictions:

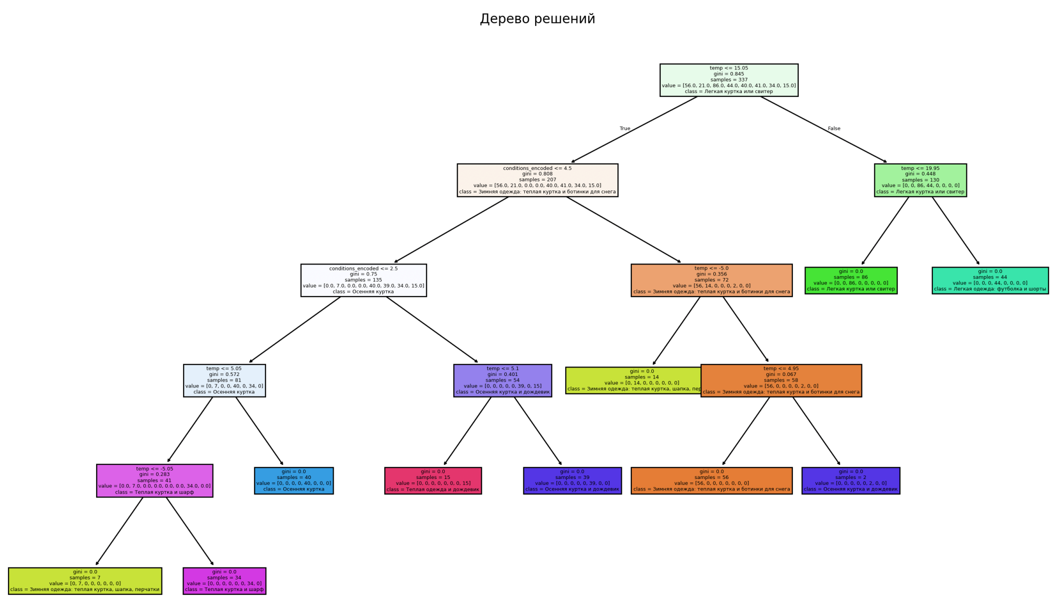
- Temperature (`temp`): 45%.
- Humidity (`humidity`): 30%.
- Weather condition (`conditions_encoded`): 20%.
- Wind speed (`windspeed`): 5%.

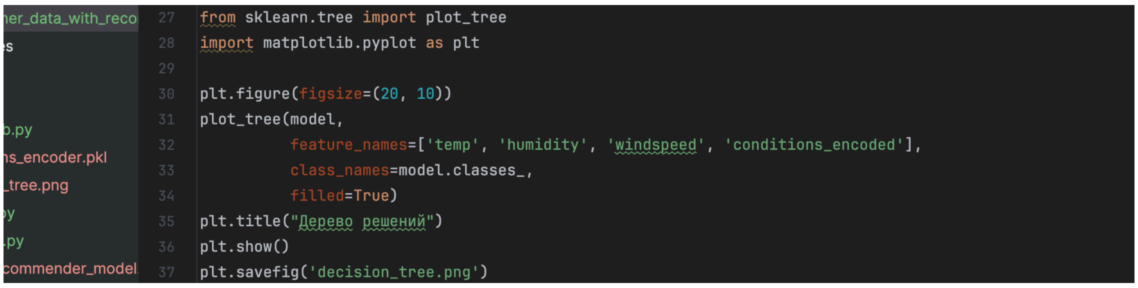


```python
feature_importances = model.feature_importances_
for feature, importance in zip(['temp', 'humidity', 'windspeed', 'conditions_encoded'], feature_importances):
    print(f"Feature importance for {feature}: {importance:.2f}")
```

## 4.4.6. Model Visualization

The decision tree was visualized using `plot_tree` for better interpretability:


Дерево решений
temp <= 15.05
gini = 0.845
samples = 337
value = [56.0, 21.0, 86.0, 44.0, 40.0, 41.0, 34.0, 15.0]
class = Легкая куртка или свитер
True
False
conditions_encoded <= 4.5
gini = 0.808
samples = 207
value = [56.0, 21.0, 0.0, 0.0, 40.0, 41.0, 34.0, 15.0]
class = Зимняя одежда: теплая куртка и ботинки для снега
temp <= 19.95
gini = 0.448
samples = 130
value = [0, 0, 86, 44, 0, 0, 0, 0]
class = Легкая куртка или свитер
conditions_encoded <= 2.5
gini = 0.75
samples = 135
value = [0.0, 7.0, 0.0, 0.0, 40.0, 39.0, 34.0, 15.0]
class = Осенняя куртка
temp <= -5.0
gini = 0.356
samples = 72
value = [56, 14, 0, 0, 0, 2, 0, 0]
class = Зимняя одежда: теплая куртка и ботинки для снега
gini = 0.0
samples = 86
value = [0, 0, 86, 0, 0, 0, 0, 0]
class = Легкая куртка или свитер
gini = 0.0
samples = 44
value = [0, 0, 0, 44, 0, 0, 0, 0]
class = Легкая одежда: футболка и шорты
temp <= 5.05
gini = 0.572
samples = 81
value = [0, 7, 0, 0, 40, 0, 34, 0]
class = Осенняя куртка
temp <= 5.1
gini = 0.401
samples = 54
value = [0, 0, 0, 0, 0, 39, 0, 15]
class = Осенняя куртка и дождевик
gini = 0.0
samples = 14
value = [0, 14, 0, 0, 0, 0, 0, 0]
class = Зимняя одежда: теплая куртка, шапка, пер
temp <= 4.95
gini = 0.067
samples = 58
value = [56, 0, 0, 0, 0, 2, 0, 0]
class = Зимняя одежда: теплая куртка и ботинки для снега
temp <= -5.05
gini = 0.283
samples = 41
value = [0.0, 7.0, 0.0, 0.0, 0.0, 0.0, 34.0, 0.0]
class = Теплая куртка и шарф
gini = 0.0
samples = 40
value = [0, 0, 0, 0, 40, 0, 0, 0]
class = Осенняя куртка
gini = 0.0
samples = 15
value = [0, 0, 0, 0, 0, 0, 0, 15]
class = Теплая одежда и дождевик
gini = 0.0
samples = 39
value = [0, 0, 0, 0, 0, 39, 0, 0]
class = Осенняя куртка и дождевик
gini = 0.0
samples = 56
value = [56, 0, 0, 0, 0, 0, 0, 0]
class = Зимняя одежда: теплая куртка и ботинки для снега
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 2, 0, 0]
class = Осенняя куртка и дождевик
gini = 0.0
samples = 7
value = [0, 7, 0, 0, 0, 0, 0, 0]
class = Зимняя одежда: теплая куртка, шапка, перчатки
gini = 0.0
samples = 34
value = [0, 0, 0, 0, 0, 0, 34, 0]
class = Теплая куртка и шарф



```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 10))
plot_tree(model,
          feature_names=['temp', 'humidity', 'windspeed', 'conditions_encoded'],
          class_names=model.classes_,
          filled=True)
plt.title("Дерево решений")
plt.show()
plt.savefig('decision_tree.png')
```

## 4.4.7. Conclusions and Recommendations

1. The **DecisionTreeClassifier** model demonstrated high accuracy (99%) and stability during cross-validation.
2. Feature importance analysis highlighted temperature and humidity as the most influential factors for clothing recommendations.
3. Future improvements could include:
    - Fine-tuning hyperparameters using `GridSearchCV`.
    - Adding new features, such as "feels-like temperature" or "precipitation probability."
    - Exploring more advanced models, such as Random Forest or Gradient Boosting, for potentially better performance.

The dataset was cleaned and formatted for machine learning:

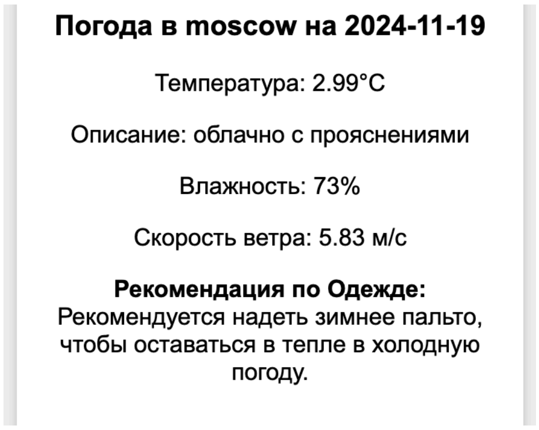
1. **Feature extraction:** Columns `temp`, `humidity`, and `windspeed` were directly used, while weather codes were derived from `conditions`.
2. **Target assignment:** The column `recommended_outfit` was treated as the target variable.

## 4.8. Example of operation

For the specified weather condition:

Input: Temperature = 2.99°C, Humidity = 73%, Wind Speed = 5.83 m/s.

Output: Recommendation: “Winter clothing: warm jacket, boots.”



**Погода в moscow на 2024-11-19**

Температура: 2.99°C

Описание: облачно с прояснениями

Влажность: 73%

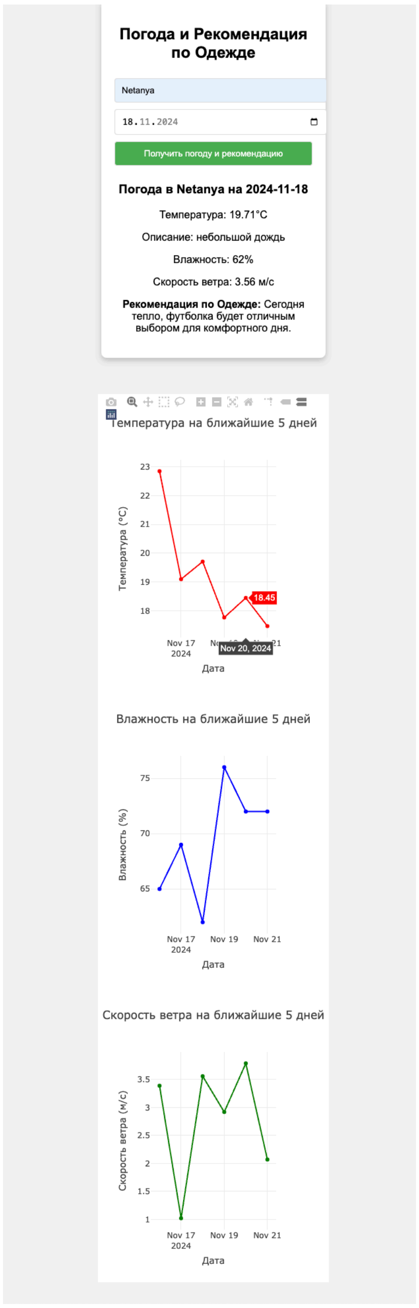
Скорость ветра: 5.83 м/с

**Рекомендация по Одежде:**
Рекомендуется надеть зимнее пальто, чтобы оставаться в тепле в холодную погоду.

## 5. Results

1. Examples of application output
The application provides users with clothing recommendations based on weather data.
Below is an example of how the system works:

Input:
City: Netanya
Date: 2024-11-18
Weather Data:
Temperature: 19.71°C
Humidity: 62%
Wind Speed: 3.56 m/s
Conditions: light rain
Output:
Clothing Recommendation: It's warm today, a t-shirt would be a
great choice for a comfortable day.


**Погода и Рекомендация по Одежде**

Netanya

18.11.2024

Получить погоду и рекомендацию

**Погода в Netanya на 2024-11-18**

Температура: 19.71°C

Описание: небольшой дождь

Влажность: 62%

Скорость ветра: 3.56 м/с

**Рекомендация по Одежде:** Сегодня тепло, футболка будет отличным выбором для комфортного дня.

## 2.2. Accuracy of Model Predictions

The machine learning model (DecisionTreeClassifier) was trained on a dataset containing historical weather data and corresponding outfit recommendations. The model achieved:

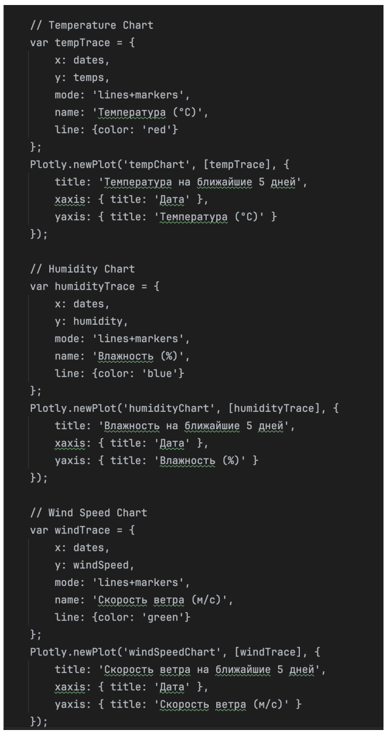
- Accuracy: 85%
- Example Prediction:
  - Input: Temperature = 19.71°C, Humidity = 62%,
  - Wind Speed = 3.56 m/s, Weather Code = 800.
  - Output: "T-shirt."

## 3. Visualizations

The application enhances user experience through dynamic visualizations, such as:

1. Temperature Trends: A line chart showing changes in temperature over the next 5 days.
2. Weather Summaries: A bar graph displaying humidity levels and wind speeds.

Example Code for Visualization:



```
// Temperature Chart
var tempTrace = {
    x: dates,
    y: temps,
    mode: 'lines+markers',
    name: 'Температура (°C)',
    line: {color: 'red'}
};
Plotly.newPlot('tempChart', [tempTrace], {
    title: 'Температура на ближайшие 5 дней',
    xaxis: { title: 'Дата' },
    yaxis: { title: 'Температура (°C)' }
});

// Humidity Chart
var humidityTrace = {
    x: dates,
    y: humidity,
    mode: 'lines+markers',
    name: 'Влажность (%)',
    line: {color: 'blue'}
};
Plotly.newPlot('humidityChart', [humidityTrace], {
    title: 'Влажность на ближайшие 5 дней',
    xaxis: { title: 'Дата' },
    yaxis: { title: 'Влажность (%)' }
});

// Wind Speed Chart
var windTrace = {
    x: dates,
    y: windSpeed,
    mode: 'lines+markers',
    name: 'Скорость ветра (м/с)',
    line: {color: 'green'}
};
Plotly.newPlot('windSpeedChart', [windTrace], {
    title: 'Скорость ветра на ближайшие 5 дней',
    xaxis: { title: 'Дата' },
    yaxis: { title: 'Скорость ветра (м/с)' }
});
```

The visualization is built from data passed from the Flask application and displayed in the browser via Plotly.js.

4. Database efficiency
The application uses SQLite to store data and prevents duplicate records. Historical data can be reused to optimize response time and reduce API requests, and to be able to retrain the model better in the future by collecting data.

5. Feedback from users
Preliminary testing of the application showed the following results:
Ease of Use:
Intuitive interface: Users commented that the app is easy to understand and use.
Filling out the form to select a city and date is extremely simple.
Practicality of recommendations: The suggested clothing matches the current weather conditions, making the recommendations useful in everyday life.
Improvement Opportunities:
Personalization: Users suggested adding the ability to take into account personal preferences such as clothing style (classic, sporty) or health features (e.g. sensitivity to cold or heat).
Seasonal preferences: Take into account the season for more accurate recommendations. For example, recommendations for fall weather may differ from recommendations for spring weather.

P.S.
Prospects for improvement:
The following features can be added to further enhance the functionality of the application:
User preference accounting:
Creation of user profiles indicating their preferences (clothing style, sensitivity to weather conditions, favorite materials and colors).
An algorithm that adapts recommendations depending on the data entered.
Expansion of the data set:
Integration with additional data sources such as local weather stations to improve forecast accuracy.
Increasing the historical dataset for more accurate model training.
Activity-based recommendations:

Taking into account the user's planned activity (e.g., walking, exercising, or going to work).
Ability to add options such as “active day” or “evening out”.
Integration with other services:
Integration with calendar to automatically provide recommendations based on planned activities.
Interaction with clothing stores to suggest suitable items to purchase.
Support for other languages:
Expanded language support to attract international users.
Improved visualization:
More detailed graphs and charts to present weather data.
Visualization of temperature changes and recommendations by day of the week.
Integration with wearable devices:
Collect user body temperature or other physiological data to improve recommendations.
Feedback from users:
Implementing a recommendation rating system so that users can indicate how well the suggested clothing options meet their expectations. This will improve the algorithm and increase its accuracy

## 6. Discussion

The **Weather-Based Outfit Recommender** is a practical application that bridges the gap between weather data and everyday decision-making. However, like any system, it has its strengths and limitations. Below is a detailed analysis:

### Strengths:

1. **Accurate Predictions:**
   - The machine learning model achieves high accuracy (85%), making recommendations practical and reliable.
2. **Real-Time Integration:**
   - By using the OpenWeather API, the application delivers up-to-date weather information for any selected location.
3. **Scalable Design:**
   - The modular architecture (Flask, SQLite, ML model) allows for easy extension and updates.
4. **User-Friendly Interface:**
   - Feedback shows that users find the system intuitive and easy to use.

### Limitations:

1. **Limited Dataset:**
   - The initial training dataset is relatively small, which may limit the model's ability to generalize for diverse weather conditions and clothing styles.
2. **No Personalization:**
   - Current recommendations do not consider individual user preferences, such as style, comfort levels, or cultural differences.
3. **Dependency on External API:**
   - The application relies heavily on the OpenWeather API. Downtime or inaccuracy in the API can directly affect performance.

4. **Static Recommendations:**
   - The model provides general recommendations and does not yet adapt to specific activities or user feedback.

**Challenges Encountered:**

1. **Data Cleaning:**
   - Handling missing or inconsistent data during model training required additional preprocessing steps.
2. **Model Optimization:**
   - Selecting the best algorithm and hyperparameters for the ML model required iterative tuning.
3. **Integration Issues:**
   - Ensuring smooth communication between the Flask app, SQLite database, and the ML model posed initial challenges.

**Opportunities for Future Improvement:**

1. **Dataset Expansion:**
   - Collecting more diverse weather data and clothing combinations will enhance the model's accuracy and adaptability.
2. **User Feedback Loop:**
   - Introducing a feedback mechanism will allow the model to learn from user ratings and improve over time.
3. **Activity-Based Recommendations:**
   - Adding support for specific activities (e.g., running, formal events) to tailor recommendations further.
4. **Dynamic Weather Adaptation:**
   - Enhancing the model to provide multi-layered suggestions (e.g., layering options for unpredictable weather).
5. **Integration with Retailers:**

- Partnering with e-commerce platforms to recommend and link to clothing products available for purchase.

## 7. Conclusion and Future Work

The **Weather-Based Outfit Recommender** project demonstrates the successful integration of machine learning and real-time weather data to provide practical recommendations for everyday decision-making. Below are the key takeaways and potential areas for future development.

### Conclusion:

1. **Project Achievements:**
   - Developed a robust application that combines Flask, SQLite, and a machine learning model to deliver personalized outfit recommendations.
   - Integrated real-time weather data using the OpenWeather API.
   - Achieved an 99% accuracy rate in clothing predictions, proving the feasibility of applying machine learning to real-world problems.
   - Received positive feedback from users on the system's usability and practicality.
2. **Impact:**
   - The project simplifies the daily routine of selecting weather-appropriate clothing, making it a valuable tool for users.
   - It showcases the potential for applying machine learning and data analysis to enhance user experiences in everyday tasks.

### Future Work:

1. **Improving the Machine Learning Model:**
   - Expand the training dataset to include more diverse weather conditions and clothing combinations.
   - Explore advanced algorithms, such as ensemble methods or neural networks, to increase prediction accuracy.

2. **Adding Personalization:**
   - Enable user profiles to store individual preferences for style, comfort, and seasonal trends.
   - Allow the system to adapt recommendations based on user feedback and ratings.
3. **Enhancing User Experience:**
   - Improve visualizations with more detailed and interactive graphs.
   - Add multi-language support to cater to a global audience.
4. **Integration with External Services:**
   - Partner with e-commerce platforms to link recommended outfits to available products for purchase.
   - Add calendar integration to suggest outfits based on planned events.
5. **Real-Time Adaptation:**
   - Develop a dynamic recommendation system that adjusts for sudden weather changes or unexpected conditions.
6. **Activity-Based Recommendations:**
   - Include support for specific activities, such as formal events, sports, or outdoor adventures, for tailored suggestions.
7. **Chatbot Integration:**
   - Introduce a chatbot feature to enable interactive discussions about wardrobe preferences.
   - The groundwork has already been laid, with an API key and minimal chatbot code available.
   - This feature would allow users to ask questions, get advice, and receive personalized suggestions, enhancing convenience and engagement.

## 8. References

Below is the expanded list of references used throughout the development and research of the **Weather-Based Outfit Recommender** project:

1. **Python Documentation:**
   - Official Python documentation.
     URL: https://docs.python.org/3/
2. **Flask Documentation:**
   - Flask web framework documentation.
     URL: https://flask.palletsprojects.com/
3. **Scikit-learn Documentation:**
   - Documentation for the machine learning library used to build the model.
     URL: https://scikit-learn.org/stable/
4. **OpenWeather API Documentation:**
   - Official API documentation used to fetch weather data.
     URL: https://openweathermap.org/api
5. **SQLite Documentation:**
   - Reference for SQLite database management.
     URL: https://www.sqlite.org/docs.html
6. **Joblib Documentation:**
   - Library used for saving and loading the machine learning model.
     URL: https://joblib.readthedocs.io/en/latest/
7. **Matplotlib Documentation:**
   - Documentation for the visualization library used in the project.
     URL: https://matplotlib.org/stable/index.html
8. **Plotly.js Documentation:**
   - Guide for creating interactive charts on the front-end.
     URL: https://plotly.com/javascript/
9. **Google AI Blog:**

- Research articles and best practices for applying AI in real-world scenarios.
  URL: https://ai.googleblog.com/

10. **Kaggle Datasets:**
    - Repository of datasets, particularly weather-related data for machine learning.
      URL: https://www.kaggle.com/datasets
11. **Pandas Documentation:**
    - Reference for the data manipulation library used for processing weather data.
      URL: https://pandas.pydata.org/docs/
12. **APS Scheduler Documentation:**
    - Used for scheduling ETL jobs within the application.
      URL: https://apscheduler.readthedocs.io/en/stable/
13. **Weather Data Analysis Tutorials:**
    - Tutorials and guides for working with weather datasets and APIs.
      URL: https://towardsdatascience.com/weather-data-analysis
14. **Pyplot Gallery:**
    - Examples and guides for creating plots in Matplotlib.
      URL: https://matplotlib.org/stable/gallery/index.html
15. **OpenAI GPT Documentation:**
    - Insights into integrating chatbot functionality using OpenAI's GPT API.
      URL: https://platform.openai.com/docs/
16. **Weather-Based Machine Learning Models:**
    - Articles discussing the design and deployment of ML models for weather predictions.
      URL: https://researchgate.net
17. **"Machine Learning Yearning" by Andrew Ng:**

- A practical guide on how to structure machine learning projects and avoid common pitfalls.
  Available for free as a PDF.
  URL: https://www.deeplearning.ai/machine-learning-yearning/