پیشنهاد پروژه: طبقه‌بندی درخت تصمیم برای تحلیل داده‌های آب و هوا

مقدمه

هدف این پروژه توسعه یک مدل یادگیری ماشین برای پیش‌بینی بارش باران در روز بعد بر اساس ویژگی‌های مختلف آب و هوا است. این فرآیند شامل جمع‌آوری داده‌ها، پیش‌پردازش، تحلیل اکتشافی داده‌ها، آموزش مدل و ارزیابی مدل می‌باشد. الگوریتم‌های کلیدی مورد استفاده در این پروژه شامل طبقه‌بند درخت تصمیم و طبقه‌بند نزدیک‌ترین-K برای مقابله با مشکلات عدم توازن کلاس‌ها مورد استفاده SMOTE است که با استفاده از تکنیک (KNN) همسایگان قرار می‌گیرد.

اهداف

1. **جمع‌آوری داده‌ها**: دریافت یک مجموعه داده شامل ویژگی‌های مختلف آب و هوا.
2. **پیش‌پردازش داده‌ها**: تمیز کردن و پیش‌پردازش داده‌ها برای مدیریت مقادیر گمشده، تبدیل داده‌های دسته‌ای و نرمال‌سازی داده‌های عددی.
3. **تحلیل اکتشافی داده‌ها (EDA)**: مصورسازی داده‌ها برای درک توزیع‌ها و روابط بین ویژگی‌ها.
4. **آموزش مدل**: آموزش یک طبقه‌بند درخت تصمیم و یک طبقه‌بند نزدیک‌ترین همسایگان-K.
5. **ارزیابی مدل**: ارزیابی عملکرد مدل‌ها با استفاده از معیارهایی مانند دقت، دقت مثبت، بازخوانی و امتیاز F1.

روش‌شناسی

1. جمع‌آوری داده‌ها

مجموعه داده از یک URL مشخص دانلود می‌شود. داده‌ها شامل ویژگی‌های مختلف آب و هوا است که در زمان‌های مختلف روز ثبت شده‌اند.

```
import pandas as pd
import requests
from io import StringIO

csv_url = "https://drive.google.com/uc?export=download&id=1GfVGnAA0FN6xpRg-clvQavWlh34M-NKH"
response = requests.get(csv_url)

if response.status_code == 200:
    csv_data = response.text
    df = pd.read_csv(StringIO(csv_data))
```

```
else:
    print("Failed to download the CSV file.")
```

2. پیش‌پردازش داده‌ها

مدیریت مقادیر گمشده

- حذف ستون‌ها و ردیف‌هایی که مقادیر گمشده زیادی دارند.
- درونیابی و پر کردن مقادیر گمشده برای متغیرهای پیوسته.
- استفاده از مد و رگرسیون خطی برای پر کردن مقادیر گمشده دسته‌ای و سایر مقادیر.

```
df = df.dropna(subset=['Date', 'Weather Station', 'Rain that day', 'Recorded Temperature at 9 AM'
df['Minimum Temperature'] = df['Minimum Temperature'].interpolate(method='linear')
# Additional interpolations and filling methods follow...
```

تبدیل داده‌های دسته‌ای

تبدیل داده‌های دسته‌ای به مقادیر عددی.

```
direction_mapping = {'N': 0, 'NNE': 22.5, 'NE': 45, 'ENE': 67.5, 'E': 90, 'ESE': 112.5, 'SE': 135
yes_no = {'Yes': 1, 'No': 0}
df = df.replace({'Gust Trajectory': direction_mapping, 'Rain the day after': yes_no, 'Rain that d
```

3. تحلیل اکتشافی داده‌ها (EDA)

مصورسازی همبستگی بین ویژگی‌ها با استفاده از نمودار حرارتی.

```
import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix = df.iloc[:, :-1].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

4. آموزش مدل

نمونه‌برداری مجدد داده‌ها

برای مقابله با مشکلات عدم توازن کلاس‌ها SMOTE استفاده از تکنیک.

```
from imblearn.over_sampling import SMOTE

X, y = df.iloc[:, :-1], df['Rain the day after']
smote = SMOTE(random_state=123)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

تقسیم داده به مجموعه‌های آموزش و تست

تقسیم داده‌ها به مجموعه‌های آموزشی و تست.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, rand
```

آموزش و پیش‌بینی مدل

طبقه‌بند درخت تصمیم

```
from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
predictions = classifier.predict(X_test)
```

طبقه‌بند K-نزدیک‌ترین همسایگان (KNN)

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)
```

5. ارزیابی مدل

و ماتریس اغتشاش F1 ارزیابی عملکرد مدل با استفاده از دقت، دقت مثبت، بازخوانی، امتیاز.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_ma

accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)
cm = confusion_matrix(y_test, predictions)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['No Rain', 'Rain'])
disp.plot()

print(f'Accuracy: {accuracy:.4f}\nPrecision: {precision:.4f}\nRecall: {recall:.4f}\nF1: {f1:.4f}'
```