SPRAWOZDANIE

Grupa 3

Wykonali:

Szymon Ciura

Tomasz Jaśkowiec

Szymon Dziedzina

07.2025 Kraków

Wstęp

Celem projektu jest analiza zmienności zasięgu lodu morskiego wokół Antarktydy. Analiza obejmuje dane z lat 1978 do 2009. W ramach analizy wyznaczono kontur przedstawiający minimalny zasięg lodu morskiego w badanym okresie, dla wszystkich kątów, co pozwala na przybliżenie rzeczywistego kształtu obszaru okołobiegunowego.

Następnie dla każdego kąta z osobna opracowano model matematyczny opisujący zmienność zasięgu lodu w funkcji czasu, z uwzględnieniem specyfiki danych (również tego że danych może brakować) i nieregularności pojawiania się lodu.

Końcowym elementem projektu była wizualizacja zmian zasięgu lodu morskiego w formie animacji. Animacja przedstawia zarówno rzeczywisty zasięg lodu w kolejnych dniach, jak i jego modelowaną wartość, umożliwiając obserwację zmian sezonowych i długoterminowych. Zaproponowany został również model biorący pod uwagę wszystkie dane.

Dane

Dane użyte do projektu znajdują się w pliku daily_ice_edge.csv. Plik zawiera informacje dotyczące zasięgu lodu morskiego wokół Antarktydy, zebrane dla 360 kątów odpowiadających długości geograficznej. Informacje przedstawiają promień oddalenia od środka bieguna i obejmują długi okres czasowy, od 26 października 1978 roku do 16 maja 2009 roku, co pozwala na analizę zmian zasięgu lodu morskiego w ujęciu wieloletnim.

Wykonanie

Całość kodu użyta do zobrazowania danych oraz animacji znajduje się na repozytorium GitHub pod adresem:

Repozytorium

Część 0) Narysowanie konturu minimalnego zasięgu lodu

Wczytanie danych:

```
df = pd.read_csv('daily_ice_edge.csv')
```

Abyśmy mogli odpowiednio „narysować” kontur minimalnego zasięgu lodu, należało przekształcić podane dane tak by można było je zaprezentować w biegunowym układzie.

Model minimalnego zasięgu wymaga od nas użycia poniższego kodu:

```
18
19
20  minimum = [df[col].min() for col in df.columns[1:]]
```

Takie dane wymagają normalizacji, tak by kształt modelu oddał najbliższe przybliżenie rzeczywistego terenu.

Tworzymy do tego celu odpowiednią funkcje normalizującą:

```
def normalize_range(data, new_min, new_max):
    old_min = min(data)
    old_max = max(data)
    return [((x - old_min) / (old_max - old_min)) * (new_max - new_min) + new_min for x in data]
```

Która jest rozszerzoną wersją normalizacji min-max o formule:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)} \cdot (\text{new\_max} - \text{new\_min}) + \text{new\_min}$$

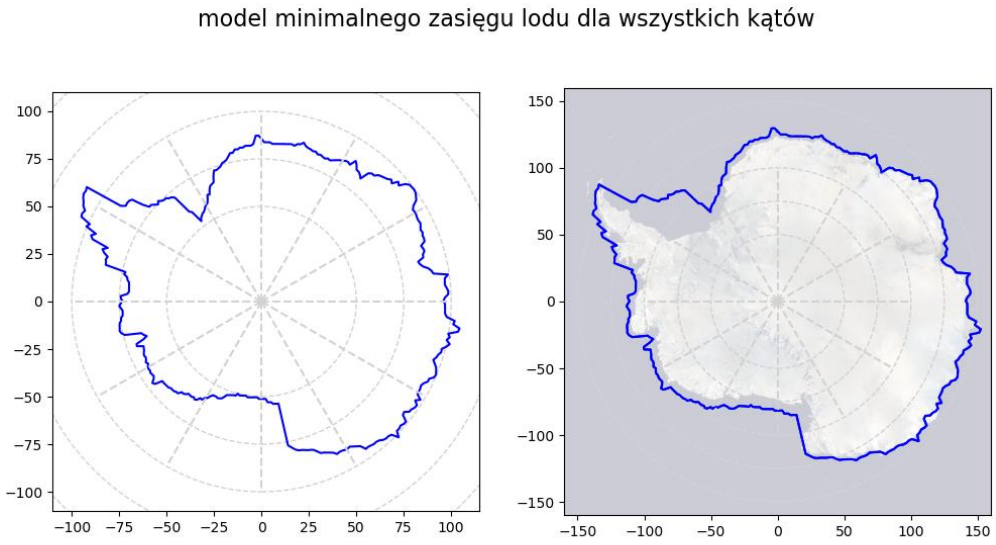
Generujemy kąty:

```
angles = np.linspace( start: 0,  stop: 360, len(minimum), endpoint=False)
angles=(angles)%360
```

Przekształcamy dane z formatu biegunowego na współrzędne kartezjańskie:

```
punkty = []
for angle, radius in zip(angles, minimum_normalized):
    x = math.sin(math.radians(angle)) * radius
    y = math.cos(math.radians(angle)) * radius
    punkty.append((x, y))
```

Następnie tworzymy wykres, który przedstawia minimalny zasięg lodu:



Wymodelowany zasięg lodu z prawej strony został nałożony na mapę w projekcji ortograficznej, opartą na zestawie danych Blue Marble opracowanym przez NASA
https://en.wikipedia.org/wiki/File:Antarctica_6400px_from_Blue_Marble.jpg

Część 1) Model matematyczny zasięgu lodu w funkcji czasu

By zautomatyzować znalezienie funkcji optymalnej dla każdego wierzchołka użyjemy funkcji **curve_fit,** która przy podanych współczynnikach znajduje najlepiej dopasowane parametry funkcji podanej w modelu

```
def sinusoid(t, A, omega, phi, C):
    return A * np.sin(omega * t + phi) + C
```

Dla każdego kąta podstawiane są odpowiednie dane: t (czas, z którego pochodzi kąt) oraz y (wartość danego kąta). Proces ten realizowany jest w pętli, co umożliwia obliczenia dla kolejnych kątów.

Znalezienie funkcji, opiera się na zgadnięciu początkowych wartości zawartych w liście **guess**

```
for col in tqdm(df_numeric.columns):
    y = df_numeric[col].values

    try:
        guess = [np.std(y), 2 * np.pi / len(y), 0, np.mean(y)]
        popt, _ = curve_fit(sinusoid, t, y, p0=guess, maxfev=10000)
        A, omega, phi, C = popt

        A_ = round(A, 8)
        omega_ = round(omega, 8)
        phi_ = round(phi, 8)
        C_ = round(C, 8)

        funkcja_str = f"lambda x: {A_} * sin({omega_} * x + {phi_}) + {C_}"
    except Exception:
        funkcja_str = "None"
```

Która kolejno zawiera:

- Amplitudę (parametr A) – wyliczana na podstawie odchylenia standardowego (wielkości zmian)
- Omega- obliczona na podstawie założenia że cykl trwa całą długość danych
- Phi – przesunięcie w czasie(w tym przypadku0)
- C – średnia wartość danych (liczona na podstawie średniej)

Na podstawie tych danych funkcja curve_fit dopasowuje najlepsze możliwe parametry, tak aby wynikowa funkcja sinusoidalna zależna od t i y najlepiej odwzorowywała dane. Parametr p0 wskazuje, że dopasowanie powinno rozpocząć się od wartości z listy guess, natomiast maxfev określa maksymalną liczbę prób dopasowania. Wynikiem działania funkcji jest zmienna popt.

Wynik ten przypisywany jest jako tekst (string) do funkcji lambda, co umożliwia późniejszy eksport do osobnego DataFrame'a. Finalnie, DataFrame ten zawiera — dla każdego kąta od 0 do 360 stopni — odpowiednio dopasowaną funkcję wyznaczoną na podstawie powyższych obliczeń.

```
popt, _ = curve_fit(sinusoid, t, y, p0=guess, maxfev=10000)
A, omega, phi, C = popt

A_ = round(A, 8)
omega_ = round(omega, 8)
phi_ = round(phi, 8)
C_ = round(C, 8)
```

Oto przykładowy wynik:

wierzcholek	lambda_funkcja
longitude_0E	lambda x: -0.29129447 * sin(0.00047897 * x + 0.90424355) + -62.50054478
longitude_1E	lambda x: -0.28349888 * sin(0.00047276 * x + 0.84410672) + -62.58995801
longitude_2E	lambda x: -0.28825549 * sin(0.00045919 * x + 0.77888505) + -62.58431987
longitude_3E	lambda x: -0.27761473 * sin(0.00046156 * x + 0.73627581) + -62.56683344
longitude_4E	lambda x: -0.2620355 * sin(0.00045654 * x + 0.76953585) + -62.58166678
longitude_5E	lambda x: -0.25839012 * sin(0.00045458 * x + 0.78396742) + -62.54431808
longitude_6E	lambda x: -0.25429779 * sin(0.00048417 * x + 0.62650072) + -62.50588588
longitude_7E	lambda x: -0.24494354 * sin(0.00048886 * x + 0.56635734) + -62.44656687
longitude_8E	lambda x: -0.23670056 * sin(0.00052472 * x + 0.46103368) + -62.40224811
longitude_9E	lambda x: -0.25958268 * sin(0.00051781 * x + 0.55348207) + -62.33048228
longitude_10E	lambda x: -0.24693171 * sin(0.00049334 * x + 0.58817137) + -62.23803966
longitude_11E	lambda x: -0.25174867 * sin(0.00044658 * x + 0.78440864) + -62.15348098
longitude_12E	lambda x: -0.2443874 * sin(0.00041228 * x + 0.90518902) + -62.10871006
longitude_13E	lambda x: -0.28075989 * sin(0.00029638 * x + 1.29667801) + -62.06270938
longitude_14E	lambda x: -3.12535305 * sin(5.34e-05 * x + 1.70913584) + -59.34984094
longitude_15E	lambda x: -1.3531636 * sin(0.00010192 * x + 1.53320864) + -61.08940132

Część 2) Animacja

W ramach tego podpunktu przygotowano animację obrazującą zmiany zasięgu lodu morskiego w czasie na podstawie danych z pliku. Działając na kodzie bazowym z podpunktu 0 uzyskano podstawę do wykonania animacji.

W pierwszej kolejności za pomocą list dla współrzędnych x i y pochodzącej z wcześniej przedstawionej listy punktów tworzymy obiekt **Polygon** i za pomocą funkcji .**add_patch** dodajemy wcześniej już uzyskany kształt minimalnego zasięgu lodu jako stały element.

```
points = list(zip(xm, ym))
granica = Polygon(points, closed=True, facecolor='none', edgecolor='red', alpha=0.5)
ax.add_patch(granica)
```

Następnie tworzymy funkcję update w której definiujemy zasady tworzenia kolejnych klatek obrazujących zmierzony zasięg lodu w kolejnych dniach. Dodatkowo dla dopełnienia danych przedstawianych na animacji dodajemy tytuł informujący o dniu w którym dana klatka -przedstawiająca zasięg lodu został zmierzony.

```
def update(frame):
    row = df.iloc[frame, 1:]
    normalized = normalize_range(row, radii_min, radii_max)
    xs = [math.sin(math.radians(a)) * r for a, r in zip(angles, normalized)]
    ys = [math.cos(math.radians(a)) * r for a, r in zip(angles, normalized)]
    xs.append(xs[0])
    ys.append(ys[0])
    line.set_data(xs, ys)
    title.set_text(f"Zasięg lodu - dzień {df.iloc[frame, 0]}")
    return line, title

# Stwórz animację
ani = animation.FuncAnimation(fig, update, frames=len(df), interval=100, blit=False)

plt.show()
```

Na koniec tworzymy animację i wywołujemy całość. Końcową wersję animacji można zobaczyć w załączonym pliku. Niebieska linia przedstawia granicę zmierzonego zasięgu lodu danego dnia a czerwona jego minimalną wartość.

Część 2) Model zasięgu lodu biorący pod uwagę wszystkie dane na raz

Aby stworzyć model zasięgu biorący pod uwagę wszystkie dane czyli f(Lat,Lon,t) – szerokość geograficzną (Lat), długość geograficzną (Lon) oraz czas (t), możemy założyć, że w trakcie każdego roku występuje podobny cykl zasięgu występowania. Rozkład lodu po długości geograficznej i szerokości jest mniej więcej regularny a więc moglibyśmy to modelować za pomocą sinusoidy. Należy do tego dodatkowo uwzględnić dzień dla którego modelujemy dane, czyli również jest to wartość dla której możemy brać pod uwagę dane z lat poprzednich. Pozwoliło by to nam otrzymać zasięg lodu dla podanych parametrów, a więc również przewidywać ich wartości.

Podsumowanie

Opracowanych dane zasięgu lodu morskiego na Antarktydzie na przestrzeni lat obrazują zmienność tego zjawiska. Opracowane dane pozwoliły otrzymać pełny obraz zmienności tego zasięgu w czasie. Przygotowano kontur minimalnego zasięgu lodu, który pozwolił ukazać przybliżony kształt rzeczywistego obszaru polarnego. Dla każdego kąta geograficznego wyznaczono model matematyczny opisujący zmiany zasięgu lodu w funkcji czasu, czego wyniki zostały zapisane do pliku. Modele te uwzględniają sezonowe fluktuacje oraz nieciągłości w danych wynikające z braku lodu w niektórych okresach. Przygotowano animację, która w przejrzysty sposób ilustruje zmiany zasięgu lodu. Zaproponowano również model biorący pod uwagę wszystkie dane. Przedstawiona analiza pozwala na lepsze zrozumienie dynamiki zjawisk zachodzących w rejonie Antarktydy oraz stanowi dobrą podstawę do dalszych, bardziej zaawansowanych badań w tym obszarze, na przykład wpływowi globalnego ocieplania na występowanie lodu.