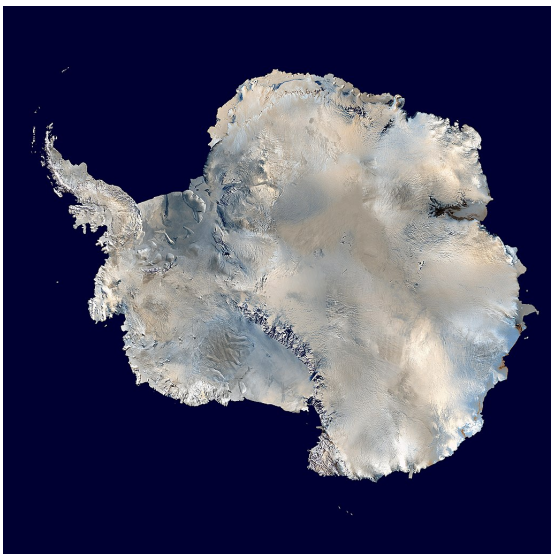
Sprawozdanie - projekt zaliczeniowy. Analiza Zasięgu Lodu Morskiego Wokół Antarktydy.

Radosław Beta, Teresa Holerek

czerwiec 2024

1 Wstęp

Niniejsze sprawozdanie stanowi podsumowanie projektu, którego celem była analiza zasięgu lodu morskiego wokół Antarktydy na przestrzeni wielu lat. Plik danych, który posłużył jako podstawa do analizy, nosi nazwę daily-ice-edge.csv i zawiera szczegółowe informacje dotyczące zasięgu lodu morskiego dla wszystkich długości geograficznych (w formacie długość-szerokość). Dane te obejmują długi okres, co pozwala na przeprowadzenie dogłębnej analizy zmian w zasięgu lodu morskiego.
Na podstawie analizy danych należało narysować kontur przedstawiający minimalny zasięg lodu morskiego wokół Antarktydy w analizowanym okresie. Dla wszystkich długości geograficznych (dla każdego kąta z osobna) należało znaleźć odpowiedni model matematyczny opisujący zasięg lodu jako funkcję czasu. Kolejnym celem było stworzenie animacji przedstawiającej zmianę zasięgu lodu morskiego w czasie, zarówno rzeczywistego, jak i modelowanego. Ostatnim etapem projektu było zaproponowanie i, w miarę możliwości, obliczenie modelu zasięgu lodu, który uwzględniałby wszystkie dostępne dane jednocześnie.



Rysunek 1: Antarktyda. Odwzorowanie prostokątne opublikowanego w 2002 roku przez NASA zestawu danych Blue Marble.

2 Zastosowane metody i uzyskane wyniki

Nad kodem pracowaliśmy na platformie jupyter. Ostatecznie całość kodu napisana została w języku Python. Korzystaliśmy z poniżej wczytanych bibliotek i danych.

```
import pandas as pd
import plotly.express as px
from chart_studio import plotly as py
import plotly.graph_objects as go
import matplotlib.pyplot as plt

df = pd.read_csv("daily_ice_edge.csv")
df.columns = [(float(index)-1.0)for index in range(len(df.columns))]
df = df.drop(df.columns[0], axis=1)
lons = list(df)
headers = lons
df.columns = headers
lats = df.loc[0].tolist()
print(df)
```

Rysunek 2: Początek kodu.

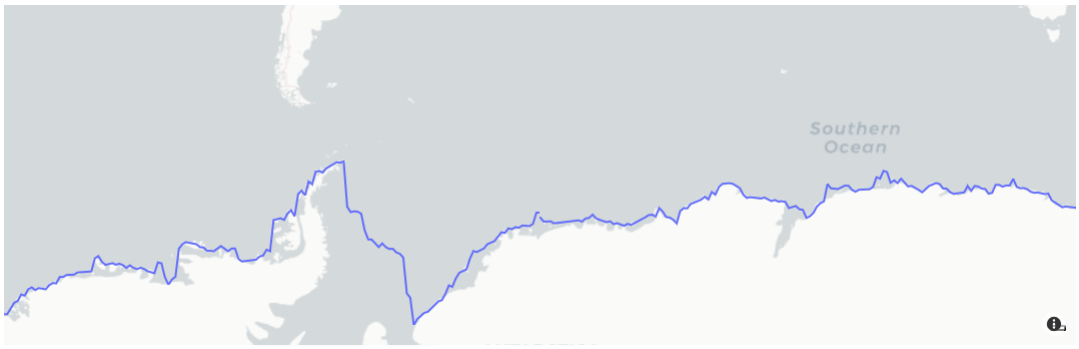
Poniżej umieszczamy nasze rozwiązania poszczególnych poleceń.

2.1 Minimalny zasięg lodu

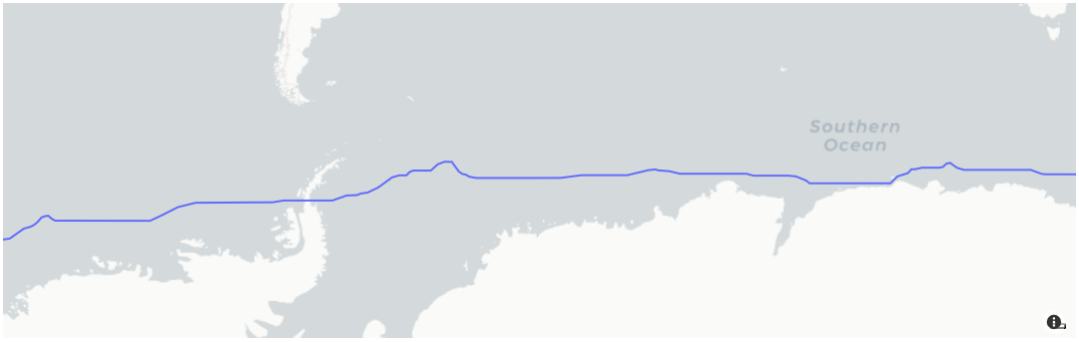
```
min_ice_extent = df.min()
lon_min = list(min_ice_extent.index)
lat_min = min_ice_extent.values
print(min_ice_extent)
fig = go.Figure(go.Scattermapbox(
    mode="lines",
    lon=lon_min,
    lat=lat_min,
    marker={'size': 12}
))

fig.update_layout(
    margin={'l': 200, 't': 20, 'b': 0, 'r': 200},
    mapbox={
        'style': "carto-positron",
        'center': {'lon': 0, 'lat': -65},
        'zoom': 1.3
    }
)
```

Rysunek 3: Minimalny zasięg lodu na Antarktydzie w badanych latach - kod.

Wynik zastosowania kodu widoczny poniżej jest przedstawiony w odwzorowaniu Merkatora.



Rysunek 4: Minimalny zasięg lodu na Antarktydzie w badanych latach - wynik dla danych rzeczywistych.



Rysunek 5: Minimalny zasięg lodu na Antarktydzie w badanych latach - wynik dla danych wymodelowanych.

2.2 Model matematyczny zasięgu lodu w funkcji czasu

W celu osiągnięcia modelu matematycznego postanowiono skorzystać z regresji wielomianowej, będącej jedną z podstawowych technik uczenia maszynowego. Nasz model zależny jest od 3 zmiennych - szerokości geograficznej, długości geograficznej oraz czasu. Ponieważ dopasowanie odpowiedniej funkcji dla danych z przeciągu blisko 30 lat byłoby ciężkie i wiązałoby się z liczeniem wielomianów bardzo wysokich stopni, postanowiliśmy rozbić nasze dane i przypasować odpowiednią funkcję dla każdego roku. Za odpowiednią funkcję do wymodelowania uznaliśmy wielomian stopnia dziewiątego. Porównanie wykresu funkcji modelowanej z zasięgiem rzeczywistym umieściliśmy na rysunku 12. Wadą przyjętego przez nas rozwiązania jest niestety brak ciągłości animacji między poszczególnymi latami. Drugą przyczyną takich "skoków"na animacji jest także niespójność danych wejściowych, zaprezentowana na przykładzie poniżej.

31-Dec-08	-59.660967
01-Jan-09	-67.103929

Rysunek 6: Niespójność danych między poszczególnymi latami.

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from array import array
import pandas as pd
import plotly.express as px
from chart_studio import plotly as py
import plotly.graph_objects as go
arrays=[]
days1=0
days=365
for m in range(0,25):
    for i in range(0,361):
        y1=df[i]
        x=[]
        for j in range(0,366):
            x.append(j)
        x1 = pd.DataFrame(data=x)
        y1=y1[0:366]
        poly_reg = PolynomialFeatures(degree = 9)
        X_poly = poly_reg.fit_transform(x1)
        lin_reg_2 = LinearRegression()
        lin_reg_2.fit(X_poly, y1)
        model = LinearRegression().fit(x1, y1)
        r_sq = model.score(x1, y1)
        y_pred = model.predict(x1)
        arrays.append(y_pred)
    days+=365
    days1+=365
df1 = pd.DataFrame(data=arrays)
df1=df1.transpose()
print(df1)
```

Rysunek 7: Model matematyczny zasięgu lodu - kod cz. 1.

```
d=[]
r=360
p=360
e=366
b=0
for i in range(0,25):
    d.append(df1.iloc[:,r-360:r+1])
    r+=360
for j in range (0,25):
    for k in range(0,361):
        d[j].index=range(b-365,e-365)
        d[j].rename(columns={j*360+k:k},inplace = True)
    b+=365
    e+=365
df_temp =pd.concat([d[1],d[2],d[3],d[4],d[5],d[6],d[7],d[8],d[9],d[10],d[11],d[12],d[13],d[14],d[15],d[16],d[17],d[18],d[19],d[20],d[21],d[22],d[23],d[24]])
df2=pd.DataFrame(data=df_temp)
```

Rysunek 8: Model matematyczny zasięgu lodu - kod cz. 2.

Rysunek 7 i 8 przedstawia obróbkę danych oraz ich przygotowanie do użycia w animacji.

2.3 Animacja przedstawiająca zmianę w czasie rzeczywistego zasięgu lodu morskiego.

Animacja wynikowa została wysłana wraz ze sprawozdaniem.

```
fig = go.Figure(go.Scattermapbox(
    mode = "lines",
    lon = lons,
    lat = lats,
    marker = {'size': 12}))
fig.update_layout(
     margin ={'l':200,'t':20,'b':0,'r':200},
    mapbox = {
        'style': "carto-positron",
        'center': {'lon': 0, 'lat': -65},
        'zoom': 1.3})
frames = [go.Frame(data= [go.Scattermapbox(
                                    lat=df.loc[k].tolist(),
                                    lon=lons)],
                   traces= [0],
                   name=f'frame{k}'
                  )for k  in  range(0, 9301, 15)]
fig.update(frames=frames);
sliders = [dict(steps= [dict(method= 'animate',
                             args= [[ f'frame{k}'],
                                    dict(mode= 'immediate',
                                    frame= dict(duration=150, redraw= True ),
                                                transition=dict( duration= 0))
                                   ],
                             label='{:.0f}'.format(k/300+1978.0)
                              ) for k in range(0,9301, 15)],
                transition= dict(duration= 0 ),
                x=0,#slider starting position
                y=0,
                currentvalue=dict(font=dict(size=12),
                                  prefix='Rok: ',
                                  visible=True,
                                  xanchor= 'center'),
                len=1.0)
           ]
```

Rysunek 9: Animacja zasięgu lodu - kod cz. 1.

```
fig.update_layout(updatemenus=[dict(type='buttons', showactive=False,
                                    y=1,
                                    x=0.75,
                                    xanchor='right',
                                    yanchor='top',
                                    pad=dict(t=0, r=30),
                                    buttons=[dict(label='Play',
                                                  method='animate',
                                                  args=[None,
                                                        dict(frame=dict(duration=100,
                                                                        redraw=True),
                                                             transition=dict(duration=0),
                                                             fromcurrent=True,
                                                             mode='immediate'
                                                            )
                                                       ]
                                                 )
                                            ]
                                   )
                              ],
                  sliders=sliders);
fig.show()
```

Rysunek 10: Animacja zasięgu lodu - kod cz. 2.

2.4 Animacja przedstawiająca zmianę w czasie wymodelowanego zasięgu lodu morskiego.

Animacja została wysłana wraz ze sprawozdaniem.

```
lons1 = list(df2)
headers1 = lons1
df2.columns = headers1
lats1 = df2.loc[0].tolist()
fig1 = go.Figure(go.Scattermapbox(
    mode = "lines",
    lon = lons1,
    lat = lats1,
    marker = {'size': 12}))
fig1.update_layout(
     margin ={'l':200,'t':20,'b':0,'r':200},
    mapbox = {
        'style': "carto-positron",
        'center': {'lon': 0, 'lat': -65},
        'zoom': 1.3})
frames1 = [go.Frame(data= [go.Scattermapbox(
                                    lat=df2.loc[k].values.tolist(),
                                    lon=lons1)],
                  traces= [0],
                  name=f'frame{k}'
                 )for k  in  range(0, 8701,15)]
fig1.update(frames=frames1);
sliders1 = [dict(steps= [dict(method= 'animate',
                           args= [[ f'frame{k}'],
                                  dict(mode= 'immediate',
                                  frame= dict(duration=150, redraw= True ),
                                              transition=dict( duration= 0))
                                 ],
                            label='{:.0f}'.format(k/365+1978.0)
                            ) for k in range(0,8701,30)],
                transition= dict(duration= 0 ),
                x=0,
                y=0,
                currentvalue=dict(font=dict(size=12),
                                  prefix='Rok: ',
                                  visible=True,
                                  xanchor= 'center'),
                len=1.0)
           ]
```

Rysunek 11: Animacja zasięgu lodu wymodelowanego - kod cz. 1.

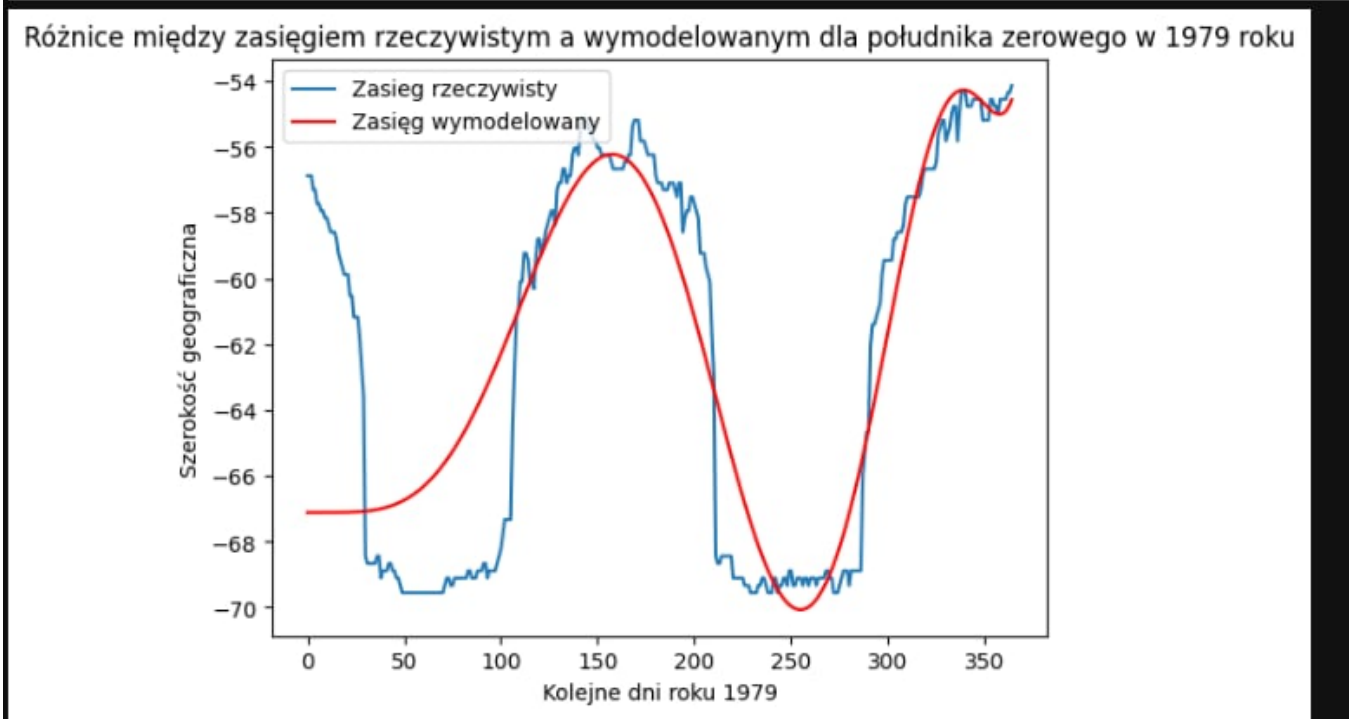
```
fig1.update_layout(updatemenus=[dict(type='buttons', showactive=False,
                                y=1,
                                x=0.75,
                                xanchor='right',
                                yanchor='top',
                                pad=dict(t=0, r=15),
                                buttons=[dict(label='Play',
                                              method='animate',
                                              args=[None,
                                                    dict(frame=dict(duration=150,
                                                                    redraw=True),
                                                         transition=dict(duration=0),
                                                         fromcurrent=True,
                                                         mode='immediate'
                                                        )
                                                   ]
                                             )
                                        ]
                               )
                          ],
                 sliders=sliders1);
fig1.show()
```

Rysunek 12: Animacja zasięgu lodu wymodelowanego - kod cz. 2.

Rysunki 9, 10, 11 oraz 12 przedstawiają kod użyty w celu wygenerowania animacji, zarówno dla zasięgu rzeczywistego, jak i wymodelowanego. W celu optymalizacji wyników postanowiliśmy przedstawić zmiany co 15 dni (zauważyliśmy, że biblioteka plotly znacząco obciąża procesor). Zaletą przyjętego przez nas rozwiązania jest jego interaktywność - można animację cofać, przybliżać, a także swobodnie poruszać się po mapie.

2.5 Różnice między zasięgiem rzeczywistym a wymodelowanym

```
y1=df[0]
x=[]
for j in range(0,365):
    x.append(j)
x1 = pd.DataFrame(data=x)
y1=y1[0:365]
poly_reg = PolynomialFeatures(degree = 9)
X_poly = poly_reg.fit_transform(x1)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y1)
plt.title("Różnice między zasięgiem rzeczywistym a wymodelowanym dla południka zerowego w 1979 roku")
plt.plot(x1,y1)
plt.plot(x1, lin_reg_2.predict(poly_reg.fit_transform(x1)), color = 'red')
plt.legend(['Zasieg rzeczywisty', 'Zasięg wymodelowany'])
plt.ylabel("Szerokość geograficzna")
plt.xlabel("Kolejne dni roku 1979")
plt.show()
```



Rysunek 13: Funkcja zasięgu zależna od czasu dla południka 0.

W celu dokładniejszej analizy różnic między zasięgiem rzeczywistym a wymodelowanym postanowiliśmy przenieść na wykres wyniki naszej pracy. Jak widać na załączonej grafice, zasięg rzeczywisty charakteryzuje się dużymi spadkami w zależność od dnia. Wynika to najpewniej z błędów pomiarowych bądź niekompletności danych. Zasięg wymodelowany jest z kolei bardziej "gładki", bardziej ciągły, a także odporny na wahania.

3 Materiały, z których korzystaliśmy

- Prezentacje z zajęć
- https://aswed.gitbooks.io/uczenie-maszynowe/content/przyklad-w-python.html
- https://plotly.com/python/animations/