Projekt 1: Wskazanie optymalnej lokalizacji farmy fotowoltaicznej – analizy wielokryterialne (MCE)

Adrian Fabisiewicz (328935)

13 stycznia 2025

Spis treści

1 Wybór lokalizacji farmy fotowoltaicznej

co należy wziąć pod uwagę, wybierając lokalizację farmy fotowoltaicznej (rozważania teoretyczne, akty prawne wraz z cytowaniami źródeł / bibliografią)

2 Cel i analizowany obszar

Celem projektu było wskazanie optymalnej lokalizacji nowej farmy fotowoltaicznej dla obszaru gminy Świeradów-Zdrój (powiat lubański, województwo dolnośląskie).

3 Analizowane kryteria

Lp	**Kryterium**	**Parametry**	**Źródło danych do kryterium**
1	odległość od rzek i zbiorników wodnych	jak najbliżej; nieprzekraczalna 100-metrowa strefa ochronna	BDOT10k(SWRS, PTWP)
2	odległość od budynków mieszkalnych	jak najdalej, powyżej 150m	BDOT10k(BUBD)
3	pokrycie terenu	powyżej 15m od lasu, optymalnie powyżej 100m od lasu	BDOT10k(PTLZ)
4	dostęp do dróg utwardzonych	jak największe zagęszczenie	BDOT10k(SKDR)
5	nachylenie stoków	jak najbardziej płasko	NMT
6	dostęp światła słonecznego	optymalnie: stoki południowe (SW-SE)	NMT
7	dobry dojazd od istotnych drogowych węzłów komunikacyjnych	jak najkrótszy czas dojazdu	BDOT10k(SKDR)
Łączenie kryteriów			
8	ocena przydatności terenu (próg przydatności)	80% / 90% max. przydatności	
9	przydatne działki / grupy działek	min 60% działki na terenie przydatnym	EGIB
10	powierzchnia i min. szerokość obszaru	2ha / 50m	
11	koszt przyłącza do sieci SN (mapy kosztów)	jak najniższy	BDOT10k (wszystkie warstwy PT)

Tabela 1: Tabela z kryteriami lokalizacji

4 Realizacja

4.1 Ustalenie środowiska pracy i ścieżek do danych

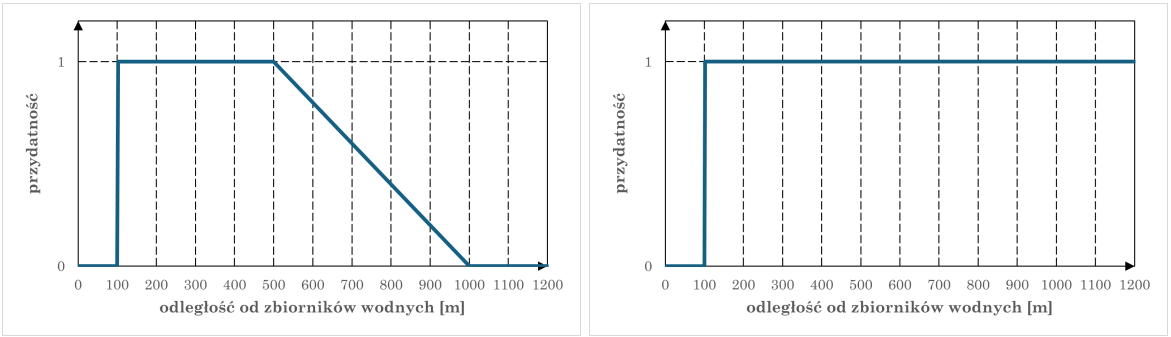
```
import arcpy.analysis
import arcpy.management
import arcpy.sa

geobaza = r"C:\Users\adria\Desktop\STUDIA_FOLDERY\analizy\MyProject12\MyProject12.gdb"
arcpy.env.workspace = "in_memory"
arcpy.env.outputCoordinateSystem = arcpy.SpatialReference("ETRS_1989_Poland_CS92")
arcpy.env.extent = f"{geobaza}\\gmina_buffer"
arcpy.env.mask = f"{geobaza}\\gmina_buffer"
arcpy.env.cellSize = 5
arcpy.env.overwriteOutput = True

swrs_0210_buffer = arcpy.analysis.Buffer(f'{geobaza}\\SWRS_L_0210', f'{geobaza}\\SWRS_L_0210_buffer', '1 Centimeter')
swrs_0212_buffer = arcpy.analysis.Buffer(f'{geobaza}\\SWRS_L_0212', f'{geobaza}\\SWRS_L_0212_buffer', '1 Centimeter')
water = arcpy.management.Merge([swrs_0210_buffer, swrs_0212_buffer, f'{geobaza}\\PTWP_A_0210', f'{geobaza}\\PTWP_A_0212'], 'water')
budynki = arcpy.management.Merge([f'{geobaza}\\BUBD_A_0210', f'{geobaza}\\BUBD_A_0212'], 'budynki')
ptlz = arcpy.management.Merge([f'{geobaza}\\PTLZ_A_0210', f'{geobaza}\\PTLZ_A_0212'], 'ptlz')
nmt = f'{geobaza}\\nmt'
drogi = arcpy.management.Merge([f'{geobaza}\\SKDR_L_0210', f'{geobaza}\\SKDR_L_0212'], 'drogi')
wezly = f'{geobaza}\\wezly_raster'
dzialki = f'{geobaza}\\dzialki'
pt_merged = f'{geobaza}\\PT_merged'
linie_elektroenergetyczne = arcpy.management.Merge([f'{geobaza}\\SULN_L_0210', f'{geobaza}\\SULN_L_0212'], 'linie_elektroenergetyczne')
```

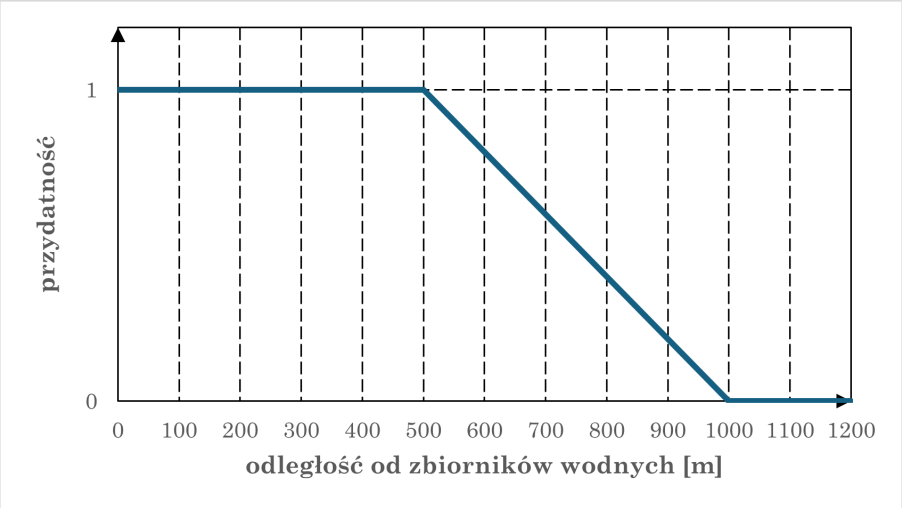
4.2 Kryterium 1: odległość od rzek i zbiorników wodnych

4.2.1 Kod

```
out_distance_accumulation_raster = arcpy.sa.DistanceAccumulation(in_source_data
    =water)
woda_rosnaca = arcpy.sa.FuzzyMembership(out_distance_accumulation_raster,
    fuzzy_function="LINEAR 100 102")
woda_malejaca = arcpy.sa.FuzzyMembership(out_distance_accumulation_raster,
    fuzzy_function="LINEAR 1000 500")
woda_mapa = arcpy.sa.FuzzyOverlay([woda_rosnaca, woda_malejaca], 'AND')
woda_mapa.save(f'{geobaza}\\kryterium_1')
```

4.2.2 Opis działania





Reklasyfikacja dla kryterium 1.

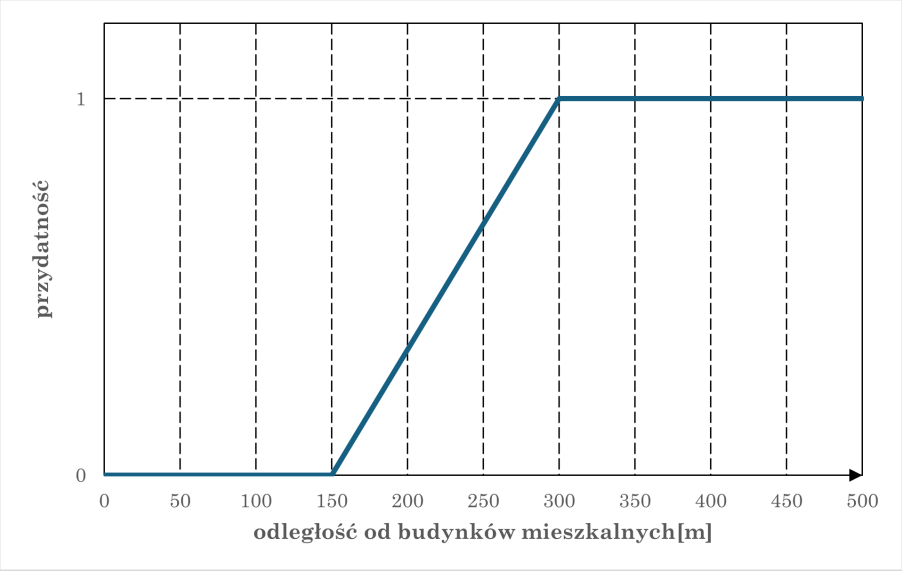
4.2.3 Wynik

4.3 Kryterium 2: odległość od budynków mieszkalnych

4.3.1 Kod

```
query = "FOBUD = 'budynki mieszkalne'"
arcpy.analysis.Select(budynki, 'budynki_mieszkalne', query)
out_distance_accumulation_buildings = arcpy.sa.DistanceAccumulation(
    in_source_data='budynki_mieszkalne')
budynki_mieszkalne = arcpy.sa.FuzzyMembership(
    out_distance_accumulation_buildings, fuzzy_function="LINEAR 150 300")
budynki_mieszkalne.save(f'{geobaza}\\kryterium_2')
```

4.3.2 Opis działania



Reklasyfikacja dla kryterium 2.

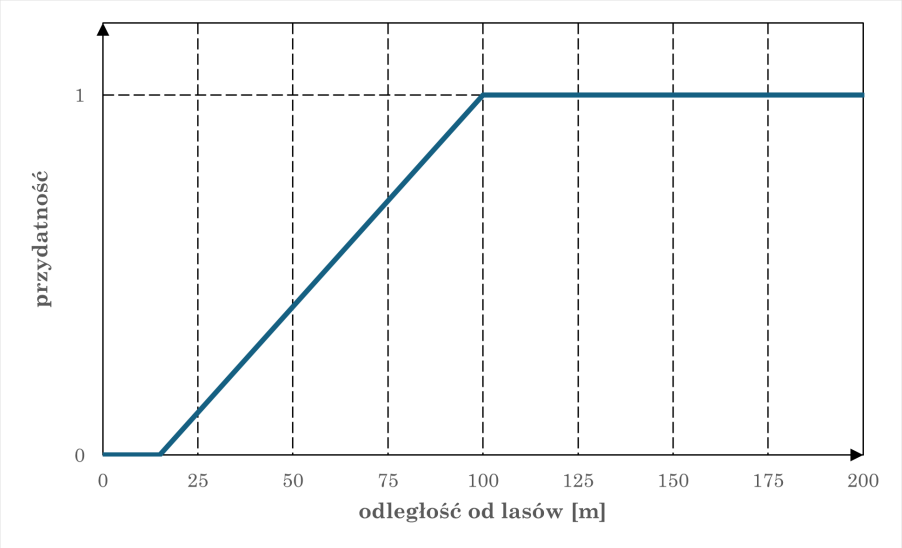
4.3.3 Wynik

4.4 Kryterium 3: pokrycie terenu

4.4.1 Kod

```
query = "RODZAJ = 'las'"
arcpy.analysis.Select(ptlz, 'ptlz_las', query)
out_distance_accumulation_ptlz = arcpy.sa.DistanceAccumulation(in_source_data="
    ptlz_las")
lasy_fuzzy = arcpy.sa.FuzzyMembership(out_distance_accumulation_ptlz,
    fuzzy_function="LINEAR 15 100")
lasy_fuzzy.save(f'{geobaza}\\kryterium_3')
```

4.4.2 Opis działania



Reklasyfikacja dla kryterium 3.

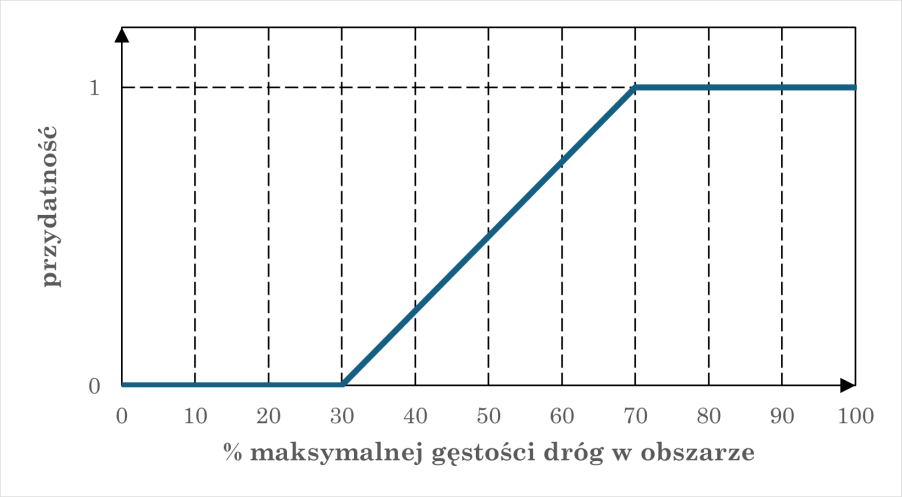
4.4.3 Wynik

4.5 Kryterium 4: dostęp do dróg utwardzonych

4.5.1 Kod

```
query = "MATE\_NAWIE IN ('beton', 'bruk', 'kostka kamienna', 'kostka
    prefabrykowana', 'masa bitumiczna', 'plyty betonowe')"
drogi_utwardzone = arcpy.analysis.Select(drogi, 'drogi_utwardzone', query)

density = arcpy.sa.LineDensity(
    in_polyline_features=drogi_utwardzone,
    population_field=None,
    cell_size=5,
    search_radius=1000,
    area_unit_scale_factor="SQUARE_KILOMETERS",
)
density.save(f'{geobaza}\\drogi_density')

arcpy.management.CalculateStatistics(density)
max_value = density.maximum

kryterium_4 = arcpy.sa.RescaleByFunction(
    in_raster=density,
    transformation_function=f"LINEAR {0.3 * max_value} {0.7 * max_value}",
    from_scale=0,
    to_scale=1
)
kryterium_4.save(f'{geobaza}\\kryterium_4')
```

4.5.2 Opis działania



Reklasyfikacja dla kryterium 4.

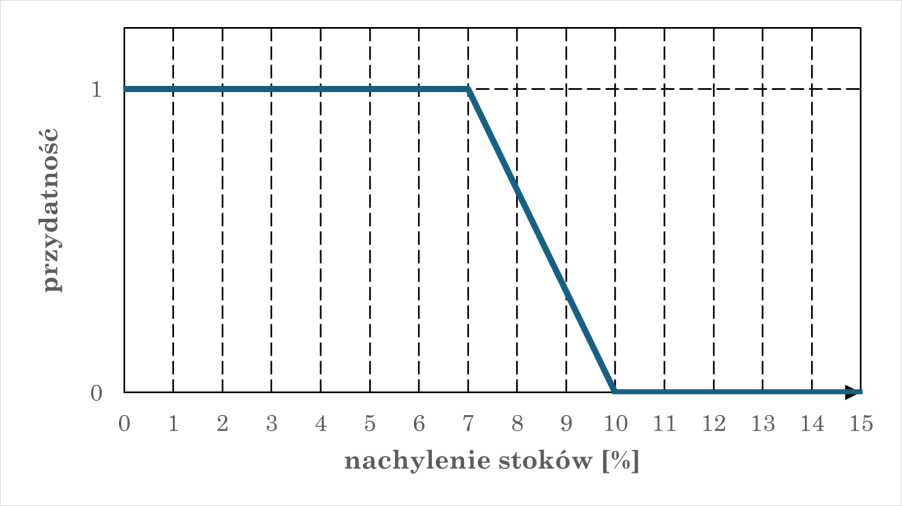
4.5.3 Wynik

4.6 Kryterium 5: nachylenie stoków

4.6.1 Kod

```
arcpy.ddd.Slope(nmt, "slope", "PERCENT_RISE", 1)
slope_fuzzy = arcpy.sa.FuzzyMembership("slope", fuzzy_function="LINEAR 10 7")
slope_fuzzy.save(f'{geobaza}\\kryterium_5')
```

4.6.2 Opis działania



Reklasyfikacja dla kryterium 5.

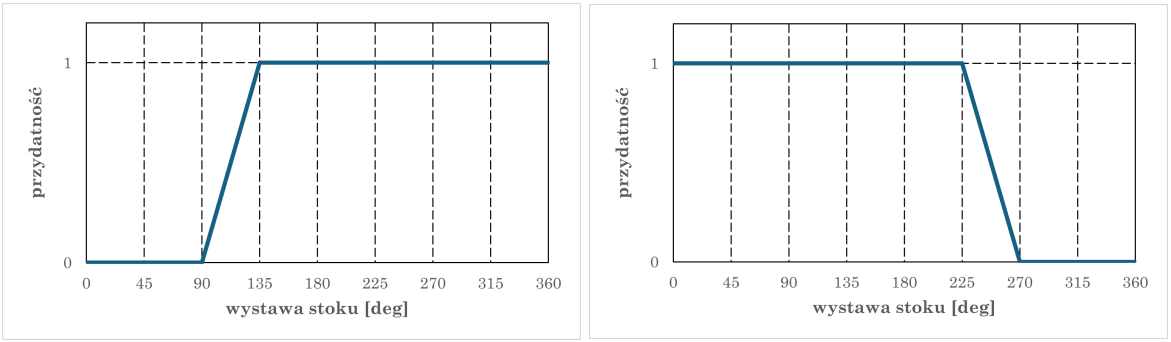
4.6.3 Wynik

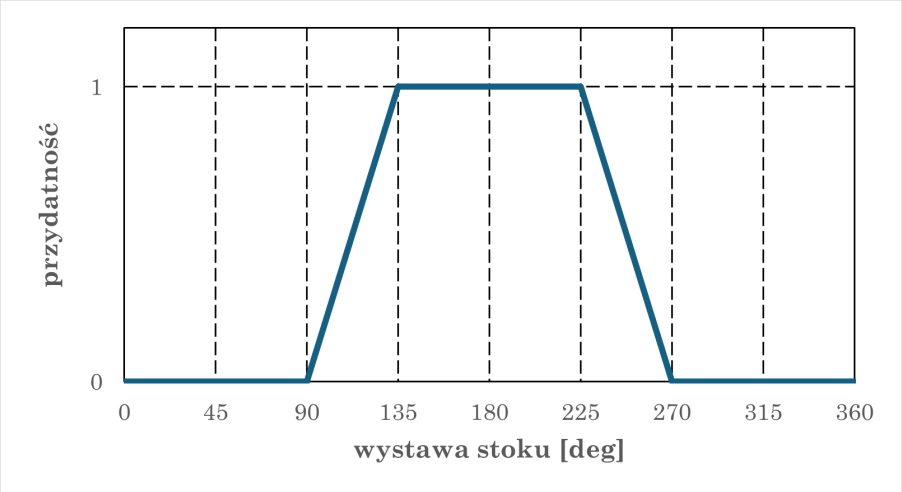
4.7 Kryterium 6: dostęp do światła słonecznego

4.7.1 Kod

```
aspect = arcpy.ddd.Aspect(nmt)
aspect_fuzzy = arcpy.sa.FuzzyMembership(aspect, fuzzy_function="LINEAR 90 135")
aspect_fuzzy_1 = arcpy.sa.FuzzyMembership(aspect, fuzzy_function="LINEAR 270
    225")
aspect_overlay = arcpy.sa.FuzzyOverlay([aspect_fuzzy, aspect_fuzzy_1], 'AND')
aspect_overlay.save(f'{geobaza}\\kryterium_6')
```

4.7.2 Opis działania





Reklasyfikacja dla kryterium 6.

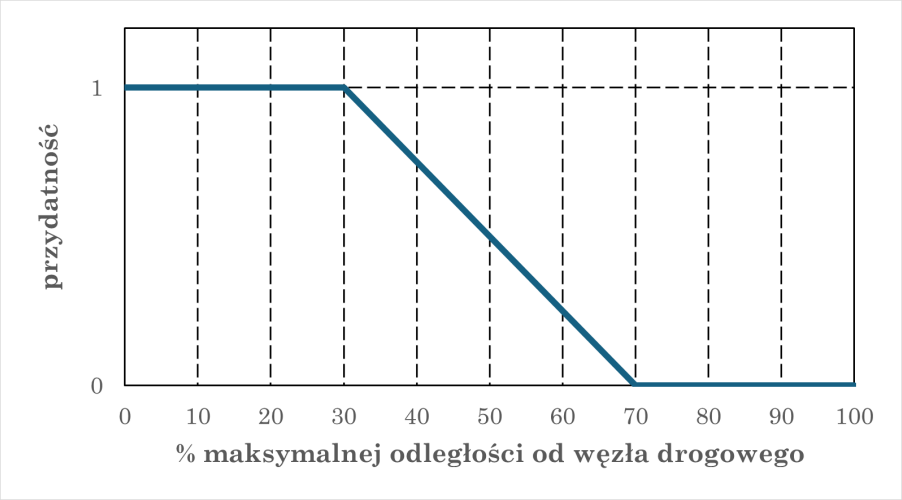
4.7.3 Wynik

4.8 Kryterium 7: dojazd do istotnych drogowych węzłów komunikacyjnych

4.8.1 Kod

```
wezly_max = float(arcpy.management.GetRasterProperties(wezly, "MAXIMUM")[0].
    replace(',', '.'))
wezly_fuzzy = arcpy.sa.FuzzyMembership(wezly, fuzzy_function=f"LINEAR {0.7 *
    wezly_max} {0.3 * wezly_max}")
wezly_fuzzy.save(f'{geobaza}\\kryterium_7')
```

4.8.2 Opis działania



Reklasyfikacja dla kryterium 7.

4.8.3 Wynik

4.9 Ocena przydatności terenu

4.9.1 Kod

```
tabela_kryteriow = arcpy.sa.WSTable([[f'{geobaza}\\kryterium_1', "VALUE",
    waga_woda], [f'{geobaza}\\kryterium_2', "VALUE", waga_budynki], [f'{geobaza
    }\\kryterium_3', "VALUE", waga_lasy], [f'{geobaza}\\kryterium_4', "VALUE",
    waga_drogi], [f'{geobaza}\\kryterium_5', "VALUE", waga_wysokosc], [f'{
    geobaza}\\kryterium_6', "VALUE", waga_aspect], [f'{geobaza}\\kryterium_7', "
    VALUE", waga_wezly]])
weighted_sum = arcpy.sa.WeightedSum(tabela_kryteriow)
weighted_sum.save(f'{geobaza}\\{wariant}_suma_rozmyte')

kryteria_ostre = arcpy.sa.FuzzyOverlay([woda_rosnaca, f"{geobaza}\\kryterium_2"
    , f"{geobaza}\\kryterium_3"], 'AND')
kryteria_ostre.save(f'{geobaza}\\kryteria_ostre')

suma = arcpy.sa.FuzzyOverlay([kryteria_ostre, weighted_sum], 'AND')
suma.save(f'{geobaza}\\{wariant}_wynik')

wynik_reclassified = arcpy.sa.Reclassify(suma, "VALUE", arcpy.sa.RemapRange
    ([[0, prog_przydatnosci, 0], [prog_przydatnosci, 1, 1]]))
wynik_reclassified.save(f'{geobaza}\\{wariant}_wynik_reclassified')
```

4.9.2 Opis działania

Powyższy kod najpierw tworzy tabelę zawierającą wagi dla każdego z kryteriów, zmienne w zależności od wariantu. Następnie tworzy sumę ważoną dla wszystkich z kryteriów.
Później brane są pod uwagę kryteria ostre, tj. 100-metrowa strefa ochronna od wód, 150-metrowa odległość od budynków mieszkalnych oraz 15-metrowa odległość od lasów. Zostaje utworzony raster, który w pikselu, dla którego przydatność dla któregokolwiek z kryterium wyniosła 0, przyjmuje również przydatność równą 0. Dzięki temu miejsca wyeliminowane przez którekolwiek z kryteriów ostrych nie będą brane pod uwagę przy wyborze lokalizacji farmy.
Następnie zostaje utworzona suma warstwy kryteriów ostrych i rozmytych. Piksele z przydatnością powyżej progu przydatności zostały zremapowane na 1, pozostałe na 0.

4.10 Wybór przydatnych działek

4.10.1 Kod

```
poligon_przydatnosci = "poligon_przydatnosci"
arcpy.conversion.RasterToPolygon(f'{geobaza}\\{wariant}_wynik_reclassified',
    poligon_przydatnosci, "NO_SIMPLIFY", "VALUE")

poligon_przydatnosci_layer = "poligon_przydatnosci_layer"
arcpy.management.MakeFeatureLayer(poligon_przydatnosci,
    poligon_przydatnosci_layer)
arcpy.management.SelectLayerByAttribute(poligon_przydatnosci_layer, "
    NEW_SELECTION", "gridcode = 1")
arcpy.management.CopyFeatures(poligon_przydatnosci_layer, f"{geobaza}\\{wariant
    }_poligon_przydatnosci")

dzialki_przeciete = "dzialki_przeciete"
arcpy.analysis.Intersect([dzialki, poligon_przydatnosci], dzialki_przeciete, "
    ALL")
arcpy.management.CopyFeatures(dzialki_przeciete, f"{geobaza}\\{wariant}
    _dzialki_przeciete_przez_poligony")

arcpy.management.AddField(dzialki_przeciete, "pole", "DOUBLE")
arcpy.management.CalculateField(dzialki_przeciete, "pole", "!shape.
    area@SQUAREMETERS!", "PYTHON3")

arcpy.management.AddField(dzialki, "pole", "DOUBLE")
arcpy.management.CalculateField(dzialki, "pole", "!shape.area@SQUAREMETERS!", "
    PYTHON3")

arcpy.management.AddField(dzialki, "pole_przydatne", "DOUBLE")
arcpy.management.CalculateField(dzialki, "pole_przydatne", 0, "PYTHON3")

with arcpy.da.UpdateCursor(dzialki_przeciete, ["ID_DZIALKI", "pole", "gridcode"
    ]) as cursor:
    for row in cursor:
        if row[2] == 1:
            with arcpy.da.UpdateCursor(dzialki, ["ID_DZIALKI", "pole", "
                pole_przydatne"]) as cursor2:
                for row2 in cursor2:
                    if row[0] == row2[0]:
                        row2[2] += row[1]
                        cursor2.updateRow(row2)

arcpy.management.AddField(dzialki, "przydatnosc", "DOUBLE")
arcpy.management.CalculateField(dzialki, "przydatnosc", "(!pole_przydatne! / !
    pole!) * 100", "PYTHON3")

dzialki_layer = "dzialki_layer"
arcpy.management.MakeFeatureLayer(dzialki, dzialki_layer)
```

```
arcpy.management.SelectLayerByAttribute(dzialki_layer, "NEW_SELECTION", f"
    przydatnosc > {prog_przydatnosci} AND shape_area > 20000 ")
arcpy.management.CopyFeatures(dzialki_layer, f"{geobaza}\\{wariant}
    _dzialki_przydatne_powyzej_{prog_przydatnosci}")
```

4.10.2 Opis działania

Kod najpierw zamienia uzyskany wcześniej raster na warstwę poligonową. Następnie zostają wybrane tylko te poligony, które stanowią teren przydatny.
Później warstwa z działkami zostaje przecięta powstałymi poligonami. Dzięki temu działka zostaje podzielona na części przydatne i nieprzydatne. Kolejno, aby uzyskać procentową przydatność każdej z działek, pętla przechodzi przez warstwę z działkami przeciętymi oraz pierwotną warstwę z działkami. Jeżeli pętla natrafi na część przydatną działki, zostaje znaleziona odpowiednia działka w warstwie pierwotnej, i pole części przydatnej działki zostaje dodane do pola pole_przydatne.
Dzięki temu możemy obliczyć przydatność działki - stosunek powierzchni przydatnej do powierzchni całkowitej. Wybieramy jedynie te działki, których przydatność jest wyższa od wybranego progu przydatności oraz takie, które mają powierzchnię co najmniej 2ha.

4.11 Koszt przyłącza do sieci SN

4.11.1 Opis działania

Przed rozpoczęciem analiz należało stworzyć tzw. mapę kosztów względnych(jednostkowych), która przedstawia faktyczną lub umowną wartość kosztu zbudowania przyłącza przez dany obszar. Dla mapy kosztów względnych w postaci rastrowej, będzie to względny koszt budowy przyłącza na obszarze o powierzchni odpowiadającej rozmiarowi piksela. W ćwiczeniu przyjęto, że koszt = 1 jest odniesiony do obszarów rolniczych (najmniejszy koszt prac ziemnych / budowlanych dla 1 piksela). Koszty względne dla innych obszarów są obliczane jako wielokrotność kosztów dla terenów rolniczych. Przypisane koszty dla wszystkich kategorii użytkowania terenu dla doprowadzenia do farmy przyłącza przedstawia Tabela 2.
Następnie, dla każdego obiektu połączonej warstwy zawierającej pokrycie terenu gminy dodano odpowiedni koszt, na podstawie pola x_kod, mówiącego o typie terenu, jaki stanowi. Warstwę nastepnie zamieniamy na raster, wykorzystując do tego nowo dodane pole z kosztem. Tworzymy mapę kosztów skumulowanych (cost map) oraz mapę kierunków (backlink raster) z użyciem narzędzia Cost Distance. Wykorzystując utworzone mapy, tworzymy ścieżkę przyłącza o najmniejszym koszcie (cost path). Rastrowa ścieżka zostaje przekonwertowana na warstwę wektorową.

4.11.2 Kod

```
pt_merged_layer = "pt_merged_layer"
arcpy.management.MakeFeatureLayer(pt_merged, pt_merged_layer)
arcpy.management.AddField(pt_merged_layer, "cost", "FLOAT")
arcpy.management.CalculateField(
    in_table=pt_merged_layer,
    field="cost",
    expression="costs.get(!x_kod!, 0)",
    expression_type="PYTHON3",
    code_block="""costs = {
    "PTWP01": 0,
    "PTWP02": 200,
    "PTWP03": 0,
    "PTZB02": 100,
    "PTZB01": 200,
    "PTZB05": 50,
    "PTZB04": 200,
    "PTZB03": 200,
    "PTLZ01": 100,
    "PTLZ02": 50,
    "PTLZ03": 50,
    "PTRK01": 15,
```

```
    "PTRK02": 15,
    "PTUT03": 100,
    "PTUT02": 90,
    "PTUT04": 20,
    "PTUT05": 20,
    "PTUT01": 0,
    "PTTR02": 1,
    "PTTR01": 20,
    "PTKM02": 200,
    "PTKM01": 100,
    "PTKM03": 200,
    "PTKM04": 0,
    "PTGN01": 1,
    "PTGN02": 1,
    "PTGN03": 1,
    "PTGN04": 1,
    "PTPL01": 50,
    "PTSO01": 0,
    "PTSO02": 0,
    "PTWZ01": 0,
    "PTWZ02": 0,
    "PTNZ01": 150,
    "PTNZ02": 150
    }"""
)

out_cost = arcpy.conversion.PolygonToRaster(
    in_features=pt_merged_layer,
    value_field="cost",
    out_rasterdataset=f"{geobaza}\\{wariant}_cost_raster",
    cell_assignment="CELL_CENTER",
    cellsize=5
)

out_distance = arcpy.sa.CostDistance(
    in_source_data=f"{geobaza}\\{wariant}_dzialki_przydatne_powyzej_{
        prog_przydatnosci}",
    in_cost_raster=f"{geobaza}\\{wariant}_cost_raster",
    maximum_distance=None,
    out_backlink_raster=f"{geobaza}\\{wariant}_cost_backlink",
    source_cost_multiplier=None,
    source_start_cost=None,
    source_resistance_rate=None,
    source_capacity=None,
    source_direction=""
)
out_distance.save(f"{geobaza}\\{wariant}_cost_distance")

out_path = arcpy.sa.CostPath(
    in_destination_data=linie_elektroenergetyczne,
    in_cost_distance_raster=f"{geobaza}\\{wariant}_cost_distance",
    in_cost_backlink_raster=f"{geobaza}\\{wariant}_cost_backlink",
    path_type="BEST_SINGLE",
    force_flow_direction_convention="INPUT_RANGE"
)
out_path.save(f"{geobaza}\\{wariant}_cost_path")

path_vector = arcpy.conversion.RasterToPolyline(in_raster=out_path,
    out_polyline_features=f"{geobaza}\\{wariant}_cost_path_vector")
```

Kod klasy obiektów BDOT	**Nazwa klasy obiektów BDOT**	**X_kod**	**Typ obiektu**	**Koszt względny**
PTWP	woda powierzchniowa	PTWP01	woda morska	0 → NoData
		PTWP02	woda płynąca	200
		PTWP03	woda stojąca	0 → NoData
PTZB	zabudowa	PTZB02	jednorodzinna	100
		PTZB01	wielorodzinna	200
		PTZB05	pozostała zabudowa	50
		PTZB04	handlowo-usługowa	200
		PTZB03	przemysłowo-składowa	200
PTLZ	teren leśny i zadrzewiony	PTLZ01	las	100
		PTLZ02	zagajnik	50
		PTLZ03	zadrzewienie	50
PTRK	roślinność krzewiasta	PTRK01	kępy krzewów	20
		PTRK02	krzewy	15
PTUT	uprawa trwała	PTUT03	sad	100
		PTUT02	plantacja	90
		PTUT04, PTUT05	inne	20
		PTUT01	ogród działkowy	0 → NoData
PTTR	roślinność trawiasta i uprawa rolna	PTTR02	grunt orny	1
		PTTR01	roślinność trawiasta	20
PTKM	teren pod drogami kołowymi, szynowymi i lotniskowymi	PTKM02	torowisko	200
		PTKM01	droga kołowa	200
		PTKM03	teren pod drogą kołową i torowiskiem	200
		PTKM04	teren pod drogą lotniskową	0 → NoData
PTGN	grunt nieużytkowany	PTGN01, PTGN02, PTGN03, PTGN04	-	1
PTPL	plac	PTPL01	-	50
PTSO	składowisko odpadów	PTSO01, PTSO02	-	0 → NoData
PTWZ	wyrobisko i zwałowisko	PTWZ01, PTWZ02	-	0 → NoData
PTNZ	pozostały teren niezabudowany	PTNZ01, PTNZ02	-	150

Tabela 2: Tabela kosztów względnych dla różnych typów obiektów BDOT.