Project: Spatial KNN Regression for Water Permeability Prediction

Import necessary libraries

In [9]:
```
# In this cell, import all the libraries that you need. For example:
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
```

Read in the datasets

In [4]:
```
# In this cell, read the files input.csv, output.csv and coordinates.csv.
# Print out the dataset dimesions (i.e. number of rows and columns).

# Step 2: Read in the datasets
input_data = pd.read_csv("input.csv")
output_data = pd.read_csv("output.csv")
coordinates_data = pd.read_csv("coordinates.csv")

# Print dataset dimensions
print("Input Data Dimensions:", input_data.shape)
print("Output Data Dimensions:", output_data.shape)
print("Coordinates Data Dimensions:", coordinates_data.shape)
```

```
Input Data Dimensions: (1690, 75)
Output Data Dimensions: (1690, 1)
Coordinates Data Dimensions: (1690, 2)
```

In [40]:
```
input_data.head()
```

Out[40]:

	-5.319627000693968877e-02	-2.192960385319173422e-01	2.100203710608411767e-01	7.04424885783739!
0	-0.632098	-0.682804	-0.285522	-0.3
1	-0.674980	-0.597421	-0.568689	-0.9
2	-0.331927	-0.154649	1.307292	0.0
3	-0.610657	-0.673046	-0.161636	-0.5
4	-0.256884	-0.220516	1.360386	0.8

5 rows × 75 columns

In [41]:
```
output_data.head()
```

Out[41]:

	5.8359
0	6.2592
1	6.9041
2	6.2065
3	7.0642
4	6.8203

In [42]:
```
coordinates_data.head()
```

Out[42]:

	4.5914e+05	7.5242e+06
0	461590.0	7549000.0
1	462040.0	7549300.0
2	462040.0	7549300.0
3	462130.0	7549400.0
4	462200.0	7547400.0

Standardization of the predictor features (input.csv)

In [6]:
```
# Standardize the predictor features (input.csv) by removing the mean and scal
# In other words, z-score the predictor features. You are allowed to use third

# Step 3: Standardization of the predictor features (input.csv)
scaler = StandardScaler()
input_data_scaled = pd.DataFrame(scaler.fit_transform(input_data), columns=inp
```

Functions and analysis code

In [ ]:

```
# Include here all the functions and other relevant code that you need in orde

# Note! Utilize the following two functions in your implementation:

### Function for calculating C-index ###
# y: array containing true label values.
# yp: array containing the predicted label values.
def cindex(y, yp):
    n = 0
    h_num = 0
    for i in range(0, len(y)):
        t = y[i]
        p = yp[i]
        for j in range(i+1, len(y)):
            nt = y[j]
            np = yp[j]
            if (t != nt):
                n = n + 1
                if (p < np and t < nt) or (p > np and t > nt):
                    h_num += 1
                elif (p == np):
                    h_num += 0.5
    return h_num/n


### Function for calculating the pairwise spatial distances between the data p
# The function will return a n-by-n matrix of Euclidean distances. For example
# distance_matrix element at indices i,j will contain the spatial distance bet
# data point i and j. Note that the element value is 0 always when i==j.
# coordinate_array: n-by-2 array containing the coordinates of the exercise da
def cdists(coordinate_array):
    number_of_observations = coordinate_array.shape[0]
    distance_matrix = np.zeros((number_of_observations, number_of_observations
    for i in range(0, number_of_observations):
        distance_matrix[i, :] = np.sqrt(np.sum((coordinate_array - coordinate_
    return distance_matrix
```

In [16]:
```
def spatial_knn_cv(input_data, output_data, coordinates_data, k=7, distance_th
    distances = cdists(coordinates_data.values)
    cindex_results = []

    for d in distance_thresholds:
        y_true = []
        y_pred = []

        for i in range(len(output_data)):
            train_indices = np.where(distances[i] > d)[0]
            if len(train_indices) < k:
                continue

            knn = KNeighborsRegressor(n_neighbors=k)
            knn.fit(input_data.iloc[train_indices], output_data.iloc[train_inc
            pred = knn.predict(input_data.iloc[[i]])[0]

            y_true.append(output_data.iloc[i, 0])
            y_pred.append(pred)

        if len(y_true) > 1:
            cindex_results.append(cindex(np.array(y_true), np.array(y_pred)))
        else:
            cindex_results.append(None)

    return distance_thresholds, cindex_results
```

In [17]:
```
# Running the spatial KNN cross-validation
thresholds, cindex_values = spatial_knn_cv(input_data_scaled, output_data, coc
```

In [20]:
```
print(cindex_values)
```

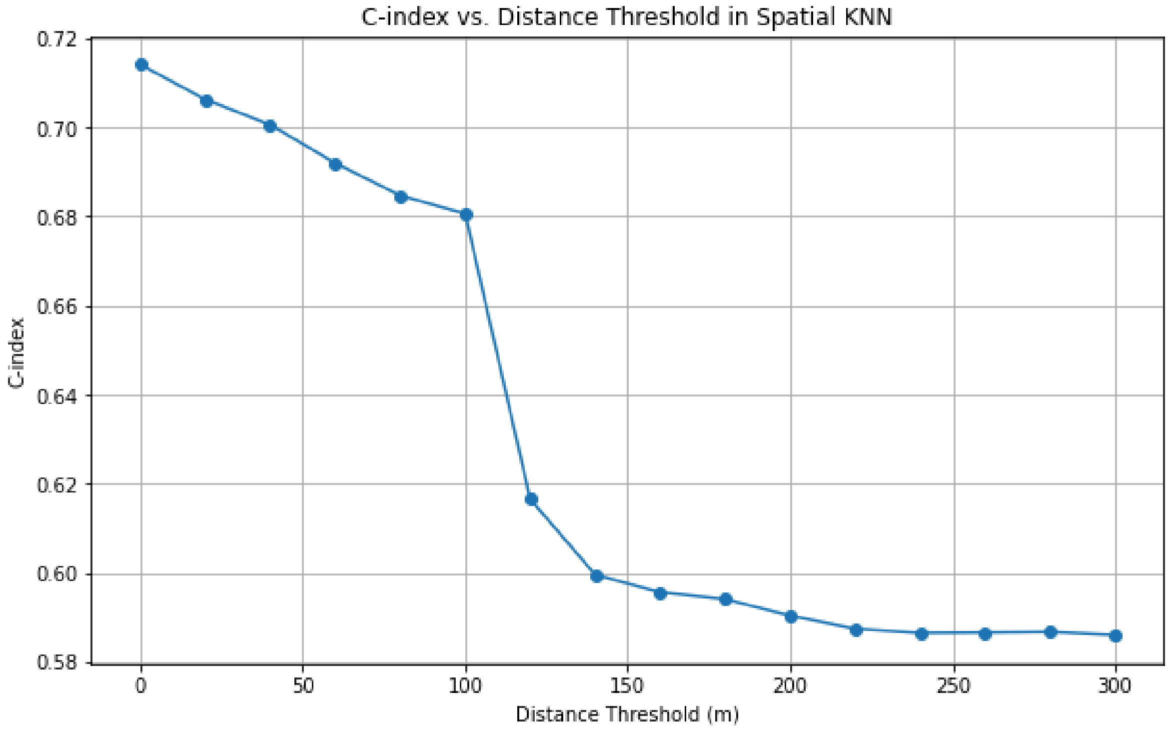
```
[0.7141470586586385, 0.7063399459553402, 0.7006271338950838, 0.6920109631049
89, 0.6847410033021063, 0.6806631262929512, 0.616538352647011, 0.59955408951
00661, 0.5956293343648912, 0.5940536157783496, 0.590367478457821, 0.58742908
03998408, 0.5864644476960941, 0.5865709672536454, 0.5867139277124644, 0.5860
043813175909]
```

In [43]:
```
print(thresholds)
```

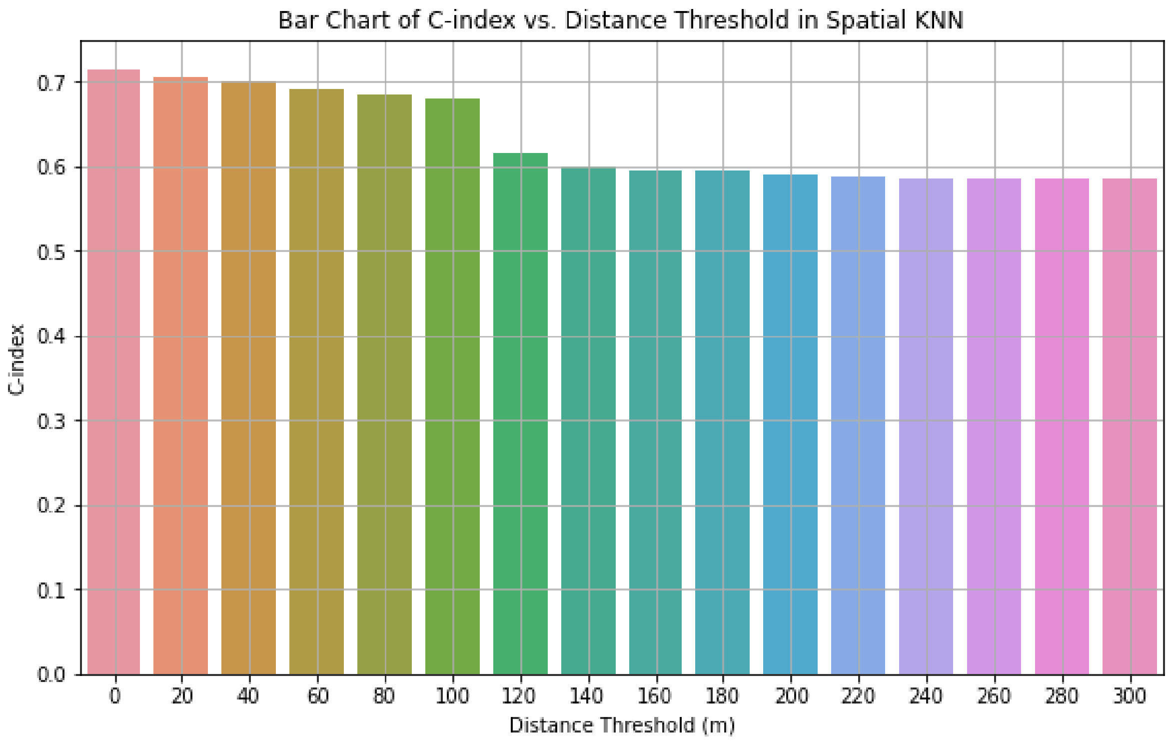
```
range(0, 301, 20)
```

Results for spatial leave-one-out cross-validation with 7-nearest neighbor regression model

In [22]:
```
# In this cell, run your script for the Spatial leave-One-Out cross-validation
# with 7-nearest neighbor regression model and visualize the results as
# requested in the task assignment.

plt.figure(figsize=(10,6))
plt.plot(thresholds, cindex_values, marker='o')
plt.xlabel("Distance Threshold (m)")
plt.ylabel("C-index")
plt.title("C-index vs. Distance Threshold in Spatial KNN")
plt.grid()
plt.show()
```



In [44]:
```
# Convert None values to NaN
cindex_values_clean = [val if val is not None else np.nan for val in cindex_va

# Remove NaN values for plotting
thresholds_clean = np.array(thresholds)[~np.isnan(cindex_values_clean)]
cindex_values_clean = np.array(cindex_values_clean)[~np.isnan(cindex_values_cl

# Plot the cleaned data
plt.figure(figsize=(10,6))
sns.barplot(x=thresholds_clean, y=cindex_values_clean)
plt.xlabel("Distance Threshold (m)")
plt.ylabel("C-index")
plt.title("Bar Chart of C-index vs. Distance Threshold in Spatial KNN")
plt.grid()
plt.show()
```



Analysis of the results

In this cell, we will try to answer the client's questions:

1. What happens to the 7NN performance as the prediction distance increases?
2. Do you think the results behave as was somewhat expected? Do they make sense, why?
3. If we require that the 7NN must have at least C-index performance of 0.68, then up to what distance should we trust the 7NN predictions, based on the results?

1. What happens to the 7NN performance as the prediction distance increases?

Answer I think as the geographical distance between known and unknown data points increases, the performance of the 7NN regression model declines as seen from the plot. This cab be seen from the C-index values, which start high at 0.714 when using nearby data points but gradually decrease as the distance threshold increases. I can see the model maintains relatively good predictive accuracy up to 100m, with a C-index above 0.68. However, beyond this threshold, there is a sharp decline in performance that can be seen, with the C-index dropping below 0.60 after 140m. At distances of 200m and beyond, the performance stabilizes around 0.586, which indicates a significant reduction in predictive accuracy. This trend suggests that the spatial correlation between data points weakens as the distance increases, which leads to less reliable predictions.

2. Do the results behave as expected? Do they make sense? Why?

Answer Yes, i think the results behave as expected and are consistent with spatial autocorrelation principles in geostatistical modeling. In spatial analysis, the closer data points mostly tend to have stronger relationships due to underlying spatial patterns. Therefore, it makes sense that the 7NN model performs well when using neighbors within 100m but it loses accuracy when distant points are used for prediction. I think the observed sharp decline beyond 100m indicates that after this distance, the spatial influence of nearby observations diminishes, which in turn reduces the model's predictive power. The stabilization of the C-index at around 0.586 for large distances seems to suggests that at these distances, the predictive relationship becomes weaker, which makes spatial proximity less relevant. These observations align with general expectations for spatial models, which reinforces that geographical proximity plays a key role in predictive accuracy.

3. If we require that the 7NN must have at least C-index performance of 0.68, up to what distance should we trust the 7NN predictions?

Answer From the plot, the C-index drops below 0.68 around 100 meters. I thik this means that: Up to ~100m, the model provides reliable predictions based on C-index (C-index ≥ 0.68). Beyond 100m, predictions become less trustworthy. Thus, from this what I can conclude is that for practical use, the 7NN model should be trusted for distances up to approximately 100 meters for ensuring reliable water permeability predictions.