Problem statement

To develop a machine learning model capable of predicting the groundwater potential of aquifers, quantified on a scale of 0 to 1, indicating the yield of an aquifer.

Background and Rationale

- Transmissivity: Defined as the measure of the ease of groundwater flow through an aquifer, calculated by multiplying the hydraulic conductivity by the thickness of the aquifer. This metric serves as a comprehensive indicator of water flow ease, rendering hydraulic conductivity as a feature is redundant when transmissivity is available.
- The following features are what are considered in predicting aquifer storage in this project.
    - Aquifer thickness
    - Transimisivity
    - hydraulic conductivy
    - Fracture contrast
    - Transverse resistance
    - longitudinal conductance
    - Overburden thickness
    - aquifer resistivity
    - Reflection co-efficient

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import RobustScaler
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pearsonr, spearmanr, kendalltau
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import  mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.tree import  DecisionTreeRegressor
from sklearn.ensemble import  RandomForestRegressor
from sklearn.svm import LinearSVR


np.random.seed(42)  # Ensure reproducibility for scipy.stats operations i
```

Section 1: Load up and describe data

In [ ]:
```
# Settings
pd.set_option('display.max_columns', None)
```

In [ ]:
```
# load up the datasets
df_1 = pd.read_excel("data/dataset_2.xlsx")
df_1.head()
```

Out[ ]:

	VES S/N	Easting (Min)	Northing (Min)	Elev(m)	No. of Layer	Curve Types	Thickness Topsoil h1(m)	Thickness Laterite h2(m)	Thickne weathere h3 (r
0	1	36.94	28.00	343.0	4	KH	1.1	4.9	23
1	2	36.73	28.80	359.0	4	KH	0.8	1.6	22
2	3	36.60	28.75	397.0	3	A	0.8	0.0	9
3	4	36.84	29.58	342.0	3	H	0.9	0.0	6
4	5	36.60	29.54	348.0	3	H	2.5	0.0	62

In [ ]:
```
df_1.columns
```

Out[ ]:
```
Index(['VES S/N', 'Easting (Min)', 'Northing (Min)', 'Elev(m)', 'No. of
Layer',
       'Curve Types', 'Thickness Topsoil h1(m)', 'Thickness Laterite h2
(m)',
       'Thickness weathered h3 (m)', 'Corrected Thick. weathered (m) H
3',
       'Thickness Fractured h4 (m)', 'Corrected Thick. Fractured (m) H
4',
       'Thick. Overb.\nB1=h1+h2', 'Total Aquifer Thick. B2=h3+h4',
       'Corrected Total Aquifer Thick', 'p1', 'p2', 'p3', 'p4', 'p5',
       'Res. Of Topsoil', 'Res. Of Laterite', 'Long. Cond. (mhos) Topsoi
l',
       'Long. Cond. (mhos) Laterite', 'Long. Cond. (mhos) OVERBURRDEN',
       'Logarithm of Topsoil', 'RESISTIVITY OF FRESH BASEMENT',
       'Logarithm Fresh Basement', 'AQUIFER RES of Weathered. \n(Ohm-
M)',
       'Logarithm Weathered', 'Hydraulic Conductivity (K)',
       'Transmissivity (T)', 'Wrong', 'Aquifer storage '],
      dtype='object')
```

In [ ]:
```
df_1.describe()
```

Out[ ]:

	VES S/N	Easting (Min)	Northing (Min)	Elev(m)	No. of Layer	Thickness Topsoil h1(m)
count	253.000000	253.000000	253.000000	253.000000	253.000000	253.000000
mean	127.000000	33.721700	30.040079	329.477075	3.379447	2.248221
std	73.179004	2.190428	1.293328	56.739089	0.510109	2.456100
min	1.000000	29.340000	27.600000	143.000000	3.000000	0.300000
25%	64.000000	32.060000	29.050000	305.000000	3.000000	1.000000
50%	127.000000	33.780000	29.950000	323.000000	3.000000	1.500000
75%	190.000000	35.520000	30.820000	343.000000	4.000000	2.200000
max	253.000000	37.080000	33.540000	930.000000	5.000000	12.700000

In [ ]:

```
df_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253 entries, 0 to 252
Data columns (total 34 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   VES S/N                            253 non-null    int64  
 1   Easting (Min)                      253 non-null    float64
 2   Northing (Min)                     253 non-null    float64
 3   Elev(m)                            253 non-null    float64
 4   No. of Layer                       253 non-null    int64  
 5   Curve Types                        252 non-null    object 
 6   Thickness Topsoil h1(m)            253 non-null    float64
 7   Thickness Laterite h2(m)           253 non-null    float64
 8   Thickness weathered h3 (m)         253 non-null    float64
 9   Corrected Thick. weathered (m) H3  253 non-null    float64
 10  Thickness Fractured h4 (m)         253 non-null    float64
 11  Corrected Thick. Fractured (m) H4  253 non-null    float64
 12  Thick. Overb. 
B1=h1+h2              253 non-null    float64
 13  Total Aquifer Thick. B2=h3+h4      253 non-null    float64
 14  Corrected Total Aquifer Thick      253 non-null    float64
 15  p1                                 253 non-null    float64
 16  p2                                 253 non-null    float64
 17  p3                                 253 non-null    float64
 18  p4                                 90 non-null     float64
 19  p5                                 4 non-null      float64
 20  Res. Of Topsoil                    253 non-null    float64
 21  Res. Of Laterite                   253 non-null    float64
 22  Long. Cond. (mhos) Topsoil         253 non-null    float64
 23  Long. Cond. (mhos) Laterite        253 non-null    float64
 24  Long. Cond. (mhos) OVERBURRDEN     253 non-null    float64
 25  Logarithm of Topsoil               253 non-null    float64
 26  RESISTIVITY OF FRESH BASEMENT      253 non-null    float64
 27  Logarithm Fresh Basement           253 non-null    float64
 28  AQUIFER RES of Weathered.
(Ohm-M)  253 non-null    float64
 29  Logarithm Weathered                253 non-null    float64
 30  Hydraulic Conductivity (K)         253 non-null    float64
 31  Transmissivity (T)                 253 non-null    float64
 32  Wrong                              253 non-null    float64
 33  Aquifer storage                    253 non-null    float64
dtypes: float64(31), int64(2), object(1)
memory usage: 67.3+ KB
```

Insights from description and info

- The average value for aquifer storage is found to be 22.4. According to the literature, it is recommended that an aquifer storage exceeding 60 percent is indicative of a good water-bearing capacity. This recommendation is from W.O. Raji and K.A. Abdulkadir in their 2020 study.
- Most of the values we want to use for the study are not null

Section 2: Feature Selection

In order to select the features that would be used in this project, plotting to visualize the data is essential. This will help visualize similarities between the features if any

Assumptions:

1. Easting: Represents longitude values. (X axis)
2. Northing: Represents latitude values. (Y axis)

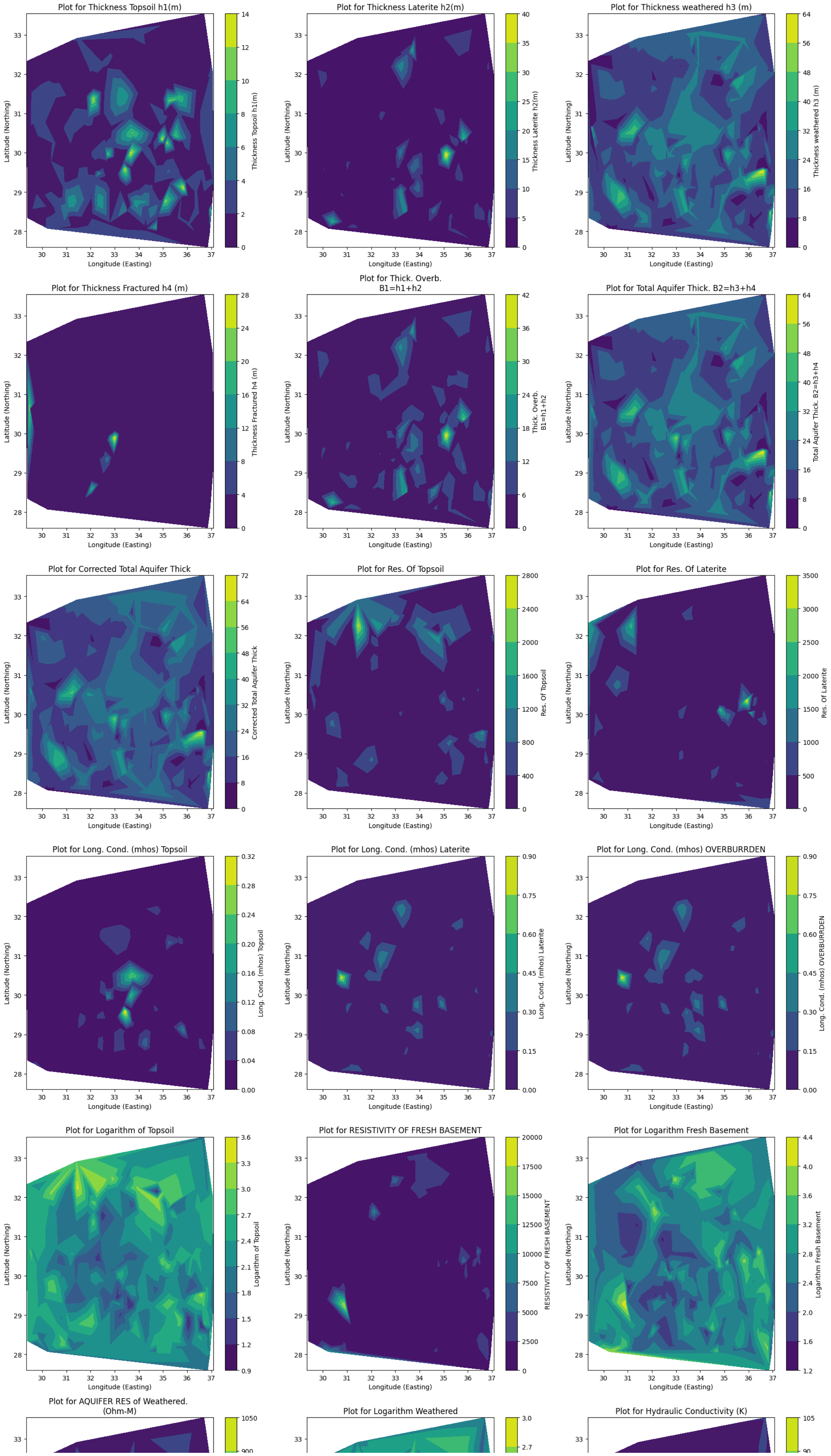
In [ ]:
```
cols_to_plot = [
      'Thickness Topsoil h1(m)', 'Thickness Laterite h2(m)',
       'Thickness weathered h3 (m)',
       'Thickness Fractured h4 (m)',
       'Thick. Overb.\nB1=h1+h2', 'Total Aquifer Thick. B2=h3+h4',
       'Corrected Total Aquifer Thick',
       'Res. Of Topsoil', 'Res. Of Laterite', 'Long. Cond. (mhos) Topsoil
       'Long. Cond. (mhos) Laterite', 'Long. Cond. (mhos) OVERBURRDEN',
       'Logarithm of Topsoil', 'RESISTIVITY OF FRESH BASEMENT',
       'Logarithm Fresh Basement', 'AQUIFER RES of Weathered. \n(Ohm-M)',
       'Logarithm Weathered', 'Hydraulic Conductivity (K)',
       'Transmissivity (T)', 'Aquifer storage '
]
```

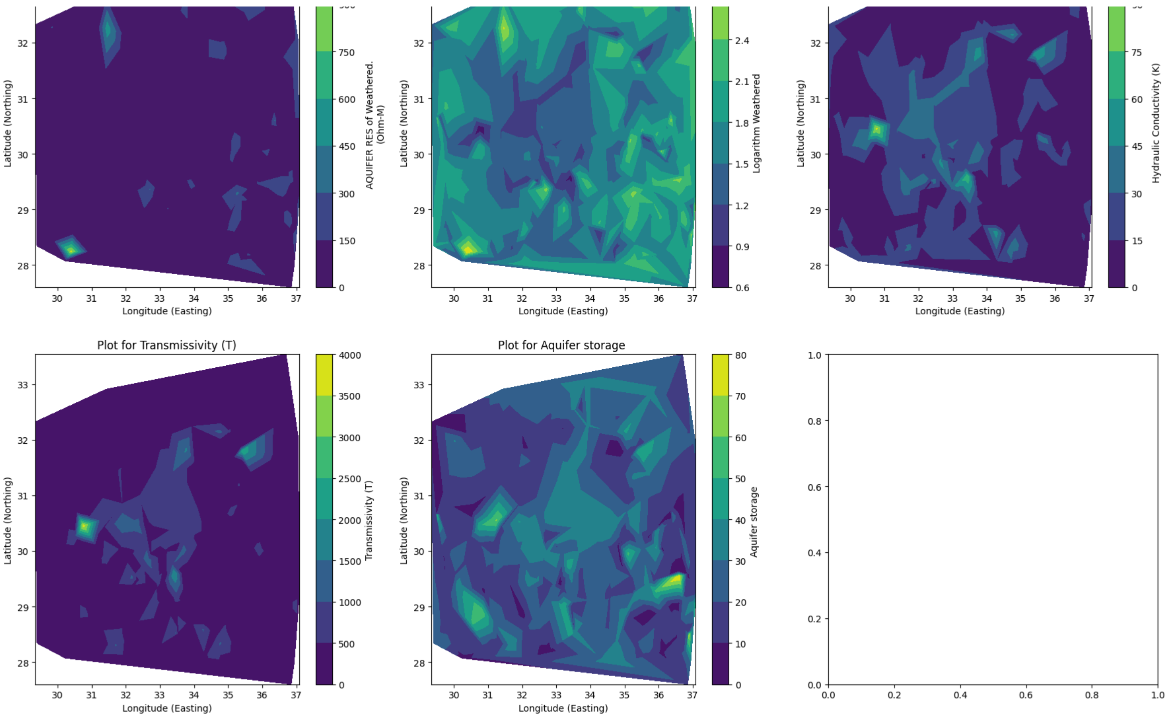
In [ ]:
```
# Create a function for making plots
def plot_features(df, cols_to_plot, scale_data=False, title=""):
    """
    Plot data for specified columns. Optionally scale the data before plo

    Parameters:
    - df: DataFrame containing the data
    - cols_to_plot: list of columns to plot
    - scale_data: boolean, if True, scale the data in cols_to_plot before
    """
    # Copy the DataFrame to avoid modifying the original
    df_to_plot = df.copy()

    if scale_data:
        # Initialize StandardScaler
        scaler = RobustScaler()
        # Apply StandardScaler only to the specified columns
        df_to_plot[cols_to_plot] = scaler.fit_transform(df[cols_to_plot])

    n_plots = len(cols_to_plot)
    max_plots_per_row = 3

    # Calculate nrows and ncols for subplots
    ncols = min(n_plots, max_plots_per_row)
    nrows = (n_plots + max_plots_per_row - 1) // max_plots_per_row

    # Plotting the data on the same image
    fig, axs = plt.subplots(nrows=nrows, ncols=ncols, figsize=(ncols*6, n

    # Add a general title for the entire plot
    if title:
        fig.suptitle(title, fontsize=16)


    for i, col in enumerate(cols_to_plot):
        # Calculate row and column index for subplot
        row = i // ncols
        col_idx = i % ncols
```

```
        ax = axs[row, col_idx]
        # Plot using original or scaled data as per the scale_data flag
        contour = ax.tricontourf(df['Easting (Min)'], df['Northing (Min)'
        fig.colorbar(contour, ax=ax, label=col)
        ax.set_title(f'Plot for {col}' + (' (Scaled)' if scale_data else
        ax.set_xlabel('Longitude (Easting)')
        ax.set_ylabel('Latitude (Northing)')

    plt.tight_layout()
    plt.show()
```

In [ ]:

```
# Plot the raw features
plot_features(df_1, cols_to_plot, False)
```

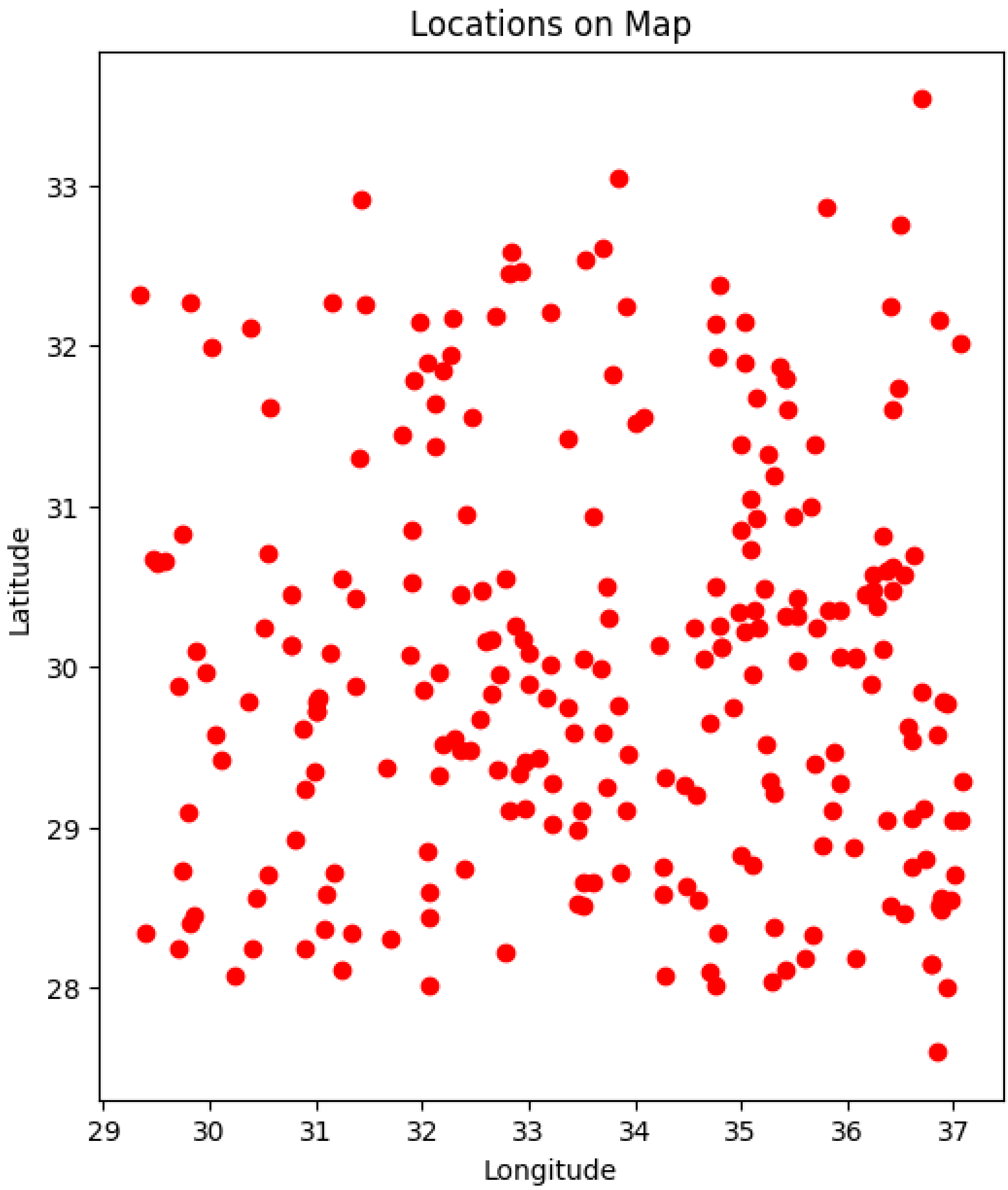

Plot for Thickness Topsoil h1(m)
Thickness Topsoil h1(m)
Plot for Thickness Laterite h2(m)
Thickness Laterite h2(m)
Plot for Thickness weathered h3 (m)
Thickness weathered h3 (m)
Plot for Thickness Fractured h4 (m)
Thickness Fractured h4 (m)
Plot for Thick. Overb. B1=h1+h2
Thick. Overb. B1=h1+h2
Plot for Total Aquifer Thick. B2=h3+h4
Total Aquifer Thick. B2=h3+h4
Plot for Corrected Total Aquifer Thick
Corrected Total Aquifer Thick
Plot for Res. Of Topsoil
Res. Of Topsoil
Plot for Res. Of Laterite
Res. Of Laterite
Plot for Long. Cond. (mhos) Topsoil
Long. Cond. (mhos) Topsoil
Plot for Long. Cond. (mhos) Laterite
Long. Cond. (mhos) Laterite
Plot for Long. Cond. (mhos) OVERBURRDEN
Long. Cond. (mhos) OVERBURRDEN
Plot for Logarithm of Topsoil
Logarithm of Topsoil
Plot for RESISTIVITY OF FRESH BASEMENT
RESISTIVITY OF FRESH BASEMENT
Plot for Logarithm Fresh Basement
Logarithm Fresh Basement
Plot for AQUIFER RES of Weathered. (Ohm-M)
Plot for Logarithm Weathered
Plot for Hydraulic Conductivity (K)
Latitude (Northing)
Longitude (Easting)



Plot db on the map to visualize VES points

In [ ]:
```python
def plot_locations(df):
    # Create a scatter plot
    fig, ax = plt.subplots(figsize=(6,7))

    ax.scatter(df['Easting (Min)'], df['Northing (Min)'], marker='o', col

    # Set labels and title
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    ax.set_title('Locations on Map')

    # Show plot
    plt.show()
```

In [ ]:
```python
plot_locations(df_1)
```



Plot Insights

- The plot for the thickness of the weathered layer is very similar to the plot for the aquifer storage. The goal of the final model is to predict the aquifer storage. This is insightful because this project can investigate how the final model would be improved or impaired by that feature.
- This same distinction can be seen when comparing the total aquifer thickness with the aquifer storage (this is logical as the larger the aquifer, the larger the expected yield of the aquifer.)
- From the plots some features could be removed, as they seem to be duplicates from a prediction point of view. e.g, corrected thicknesses, fracture thickness

Rank Features

Rank the features with different methods, in order to decide which features would be the most important for the study. Three correlation methods are considered for this;

- Pearson
- Spearman
- Kendall

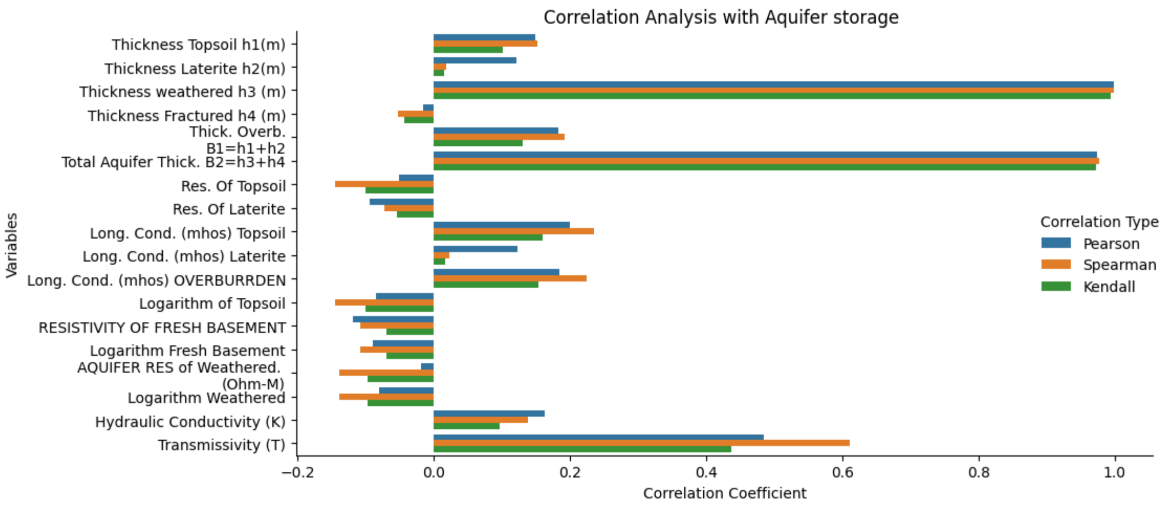
In [ ]:
```
# Create a function to plot correlation analysis
def plot_correlation_analysis(df, columns_to_check, column_to_correlate_a
    '''
    Perform and plot correlation analysis between specified columns and a

    Parameters:
    - df (pandas.DataFrame): The DataFrame containing the data.
    - columns_to_check (list of str): A list of column names to calculate
    - column_to_correlate_against (str): The name of the column to correl

    '''
    # Initialize a DataFrame to store the correlation results
    correlation_results = []

    # Loop through each column to calculate correlation coefficients
    for column in columns_to_check:
        pearson_corr, _ = pearsonr(df[column], df[column_to_correlate_aga
        spearman_corr, _ = spearmanr(df[column], df[column_to_correlate_a
        kendall_corr, _ = kendalltau(df[column], df[column_to_correlate_a

        # Append the results to the list as a dictionary
        correlation_results.append({
            'Variable': column,
            'Pearson': pearson_corr,
            'Spearman': spearman_corr,
            'Kendall': kendall_corr
        })

    # Convert the list to a DataFrame
    correlation_results_df = pd.DataFrame(correlation_results)
    correlation_results_df = correlation_results_df.melt(id_vars=['Variab
    print(correlation_results_df)

    # Plotting
    sns.catplot(x='Coefficient', y='Variable', hue='Correlation Type', da
    plt.title(f'Correlation Analysis with {column_to_correlate_against}')
    plt.xlabel('Correlation Coefficient')
    plt.ylabel('Variables')
    plt.tight_layout()
    plt.show()
```

In [ ]:
```
features_to_corr = [
      'Thickness Topsoil h1(m)', 'Thickness Laterite h2(m)',
       'Thickness weathered h3 (m)',
       'Thickness Fractured h4 (m)',
       'Thick. Overb.\nB1=h1+h2', 'Total Aquifer Thick. B2=h3+h4',
       'Res. Of Topsoil', 'Res. Of Laterite', 'Long. Cond. (mhos) Topsoil
       'Long. Cond. (mhos) Laterite', 'Long. Cond. (mhos) OVERBURRDEN',
       'Logarithm of Topsoil', 'RESISTIVITY OF FRESH BASEMENT',
       'Logarithm Fresh Basement', 'AQUIFER RES of Weathered. \n(Ohm-M)',
       'Logarithm Weathered', 'Hydraulic Conductivity (K)',
```

```
        'Transmissivity (T)',
]
```

In [ ]:
```
plot_correlation_analysis(df_1, features_to_corr, 'Aquifer storage ')
```

	Variable	Correlation Type	Coefficient
0	Thickness Topsoil h1(m)	Pearson	0.149565
1	Thickness Laterite h2(m)	Pearson	0.121427
2	Thickness weathered h3 (m)	Pearson	0.999225
3	Thickness Fractured h4 (m)	Pearson	-0.016601
4	Thick. Overb.\nB1=h1+h2	Pearson	0.182830
5	Total Aquifer Thick. B2=h3+h4	Pearson	0.974173
6	Res. Of Topsoil	Pearson	-0.051762
7	Res. Of Laterite	Pearson	-0.093737
8	Long. Cond. (mhos) Topsoil	Pearson	0.199653
9	Long. Cond. (mhos) Laterite	Pearson	0.121976
10	Long. Cond. (mhos) OVERBURRDEN	Pearson	0.184247
11	Logarithm of Topsoil	Pearson	-0.084771
12	RESISTIVITY OF FRESH BASEMENT	Pearson	-0.119565
13	Logarithm Fresh Basement	Pearson	-0.089655
14	AQUIFER RES of Weathered. \n(Ohm-M)	Pearson	-0.019119
15	Logarithm Weathered	Pearson	-0.080903
16	Hydraulic Conductivity (K)	Pearson	0.162177
17	Transmissivity (T)	Pearson	0.484136
18	Thickness Topsoil h1(m)	Spearman	0.152381
19	Thickness Laterite h2(m)	Spearman	0.017991
20	Thickness weathered h3 (m)	Spearman	0.999122
21	Thickness Fractured h4 (m)	Spearman	-0.052515
22	Thick. Overb.\nB1=h1+h2	Spearman	0.192251
23	Total Aquifer Thick. B2=h3+h4	Spearman	0.977808
24	Res. Of Topsoil	Spearman	-0.145283
25	Res. Of Laterite	Spearman	-0.072456
26	Long. Cond. (mhos) Topsoil	Spearman	0.234784
27	Long. Cond. (mhos) Laterite	Spearman	0.022671
28	Long. Cond. (mhos) OVERBURRDEN	Spearman	0.224567
29	Logarithm of Topsoil	Spearman	-0.145283
30	RESISTIVITY OF FRESH BASEMENT	Spearman	-0.108986
31	Logarithm Fresh Basement	Spearman	-0.108986
32	AQUIFER RES of Weathered. \n(Ohm-M)	Spearman	-0.138672
33	Logarithm Weathered	Spearman	-0.138672
34	Hydraulic Conductivity (K)	Spearman	0.138672
35	Transmissivity (T)	Spearman	0.610228
36	Thickness Topsoil h1(m)	Kendall	0.101673
37	Thickness Laterite h2(m)	Kendall	0.015624
38	Thickness weathered h3 (m)	Kendall	0.994315
39	Thickness Fractured h4 (m)	Kendall	-0.043257
40	Thick. Overb.\nB1=h1+h2	Kendall	0.129861
41	Total Aquifer Thick. B2=h3+h4	Kendall	0.972838
42	Res. Of Topsoil	Kendall	-0.100154
43	Res. Of Laterite	Kendall	-0.054790
44	Long. Cond. (mhos) Topsoil	Kendall	0.159451
45	Long. Cond. (mhos) Laterite	Kendall	0.017110
46	Long. Cond. (mhos) OVERBURRDEN	Kendall	0.153257
47	Logarithm of Topsoil	Kendall	-0.100154
48	RESISTIVITY OF FRESH BASEMENT	Kendall	-0.069306
49	Logarithm Fresh Basement	Kendall	-0.069306
50	AQUIFER RES of Weathered. \n(Ohm-M)	Kendall	-0.097077
51	Logarithm Weathered	Kendall	-0.097077
52	Hydraulic Conductivity (K)	Kendall	0.097077
53	Transmissivity (T)	Kendall	0.436888



Insights from the plot

Two features, weathered thickness and aquifer thickness seem to be very highly correlated with aquifer storage. The following features will be selected for training from the plots; the features selected generally have correlation greater than or equal to 0.2

1. Thickness Topsoil h1(m)
2. Thickness weathered h3 (m) (high correlation)
3. Thick. Overb.\nB1=h1+h2
4. Total Aquifer Thick. B2=h3+h4 (high correlation)
5. Long. Cond. (mhos) OVERBURRDEN
6. Long. Cond. (mhos) Topsoil
7. Transmissivity (T)

In [ ]:

```
# The following features will be used for the rest of the analysis.
all_selected_features = [
        'Thickness Topsoil h1(m)',
        'Thickness weathered h3 (m)',
        'Thick. Overb.\nB1=h1+h2',
        'Total Aquifer Thick. B2=h3+h4',
        'Long. Cond. (mhos) OVERBURRDEN',
        'Long. Cond. (mhos) Topsoil',
        'Transmissivity (T)',
]

features_without_high_corr = [
        'Thickness Topsoil h1(m)',
        'Thick. Overb.\nB1=h1+h2',
        'Long. Cond. (mhos) Topsoil',
        'Long. Cond. (mhos) OVERBURRDEN',
        'Transmissivity (T)',
]


features_without_weathered_thick = [
        'Thickness Topsoil h1(m)',
        'Thick. Overb.\nB1=h1+h2',
        'Total Aquifer Thick. B2=h3+h4',
        'Long. Cond. (mhos) OVERBURRDEN',
```

```
        'Long. Cond. (mhos) Topsoil',
        'Transmissivity (T)',

]



features_without_aquifer_thick = [
        'Thickness Topsoil h1(m)',
        'Thickness weathered h3 (m)',
        'Thick. Overb.\nB1=h1+h2',
        'Long. Cond. (mhos) OVERBURRDEN',
        'Long. Cond. (mhos) Topsoil',
        'Transmissivity (T)',
]
```

Section 3: Preprocessing data

The selected features have been broken down into categories. In order to pre-process the data, two major step taken is to transform the features.

Transform features

It is important to do some level of feature transformation in any machine learning project. The idea is to modify the features in the dataset to improve the performance and accuracy of the machine learning models.

In [ ]:

```
# function to scale features as needed
def scale_features(df, feature_columns, label):
    """
    Scales the specified features in the DataFrame using StandardScaler.

    Parameters:
    - df: pandas DataFrame containing the data.
    - feature_columns: List of column names to be scaled.
    - label: String name of the label column

    """
    scaler = StandardScaler()

    # Copy the original DataFrame to avoid modifying it directly
    df_scaled = df.copy()

    df_scaled = df_scaled[feature_columns]

    # Scale the specified features
    df_scaled[feature_columns] = scaler.fit_transform(df_scaled[feature_c

    df_scaled['Aquifer Storage'] = df[label]
    df_scaled[["Easting (Min)", "Northing (Min)"]]  = df[["Easting (Min)"
    return df_scaled
```

In [ ]:

```
# feature scaling for all feature groups
# creating scaled version of all  feature groups
df_all_predicted = scale_features(df_1, all_selected_features, 'Aquifer s
```

```
df_without_high_corr  = scale_features(df_1, features_without_high_corr,
df_without_weathered_thick  = scale_features(df_1, features_without_weath
df_without_aquifer_thick = scale_features(df_1, features_without_aquifer_
```

In [ ]:
```
df_all_predicted.head()
```

Out[ ]:

	Thickness Topsoil h1(m)	Thickness weathered h3 (m)	Thick. Overb.\nB1=h1+h2	Total Aquifer Thick. B2=h3+h4	Long. Cond. (mhos) OVERBURRDEN	Long Cond (mhos Topsoi
0	-0.468424	0.641042	0.273002	0.596086	-0.234145	-0.30943
1	-0.590811	0.565512	-0.407114	0.522118	-0.505891	-0.51993
2	-0.590811	-0.858771	-0.709387	-0.872713	0.098037	1.36583
3	-0.550016	-1.160891	-0.690495	-1.168587	-0.523319	-0.47918
4	0.102715	4.827571	-0.388222	4.696045	-0.533278	-0.50875

In [ ]:
```
df_without_high_corr.head()
```

Out[ ]:

	Thickness Topsoil h1(m)	Thick. Overb.\nB1=h1+h2	Long. Cond. (mhos) Topsoil	Long. Cond. (mhos) OVERBURRDEN	Transmissivity (T)	Aqui Stora
0	-0.468424	0.273002	-0.309435	-0.234145	-0.179934	30.017
1	-0.590811	-0.407114	-0.519935	-0.505891	-0.657226	29.119
2	-0.590811	-0.709387	1.365838	0.098037	-0.598868	12.186
3	-0.550016	-0.690495	-0.479188	-0.523319	-0.738212	8.594
4	0.102715	-0.388222	-0.508759	-0.533278	-0.465206	79.79(

Key insights

- Features are now scaled as seen in the printed out values of the data
- Aquifer storage has been renamed to label just for clarity and to make it easy to work with

Split data and train ML model

The scaled data will be split at a ratio of 70:30.

- 70% of the data will be used for training the model
- 30% of the data will be used for testing the data

Multiple ML models will be tested to evaluate which works best This is a regression problem so the major regression models will e evaluated on effectiveness;

1. Linear Regression
2. Decision Tree Regression

3. Random Forest Regressor
4. Linear SVR

In [ ]:
```
def evaluate_models(df, features, target, models=None, title=""):
    # Split the data into features (X) and target (y)
    X = df[features]
    y = df[target]

    default_models = [
        ('Linear Regression', LinearRegression()),
        ('Decision Tree Regressor', DecisionTreeRegressor()),
        ('Random Forest Regressor', RandomForestRegressor()),
        ("Linear SVR", LinearSVR())
    ]

    # If no models are provided, use the default models
    if models is None:
        models = default_models

    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

    results = []
    best_score = float('-inf')
    best_model = None

    for name, model in models:
        # Perform cross-validation
        cv_scores = cross_val_score(model, X_train, y_train, cv=5)
        cv_mean = np.mean(cv_scores)

        # Fit the model on the full training data and evaluate on the tes
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        test_score = mean_squared_error(y_test, y_pred, squared=False) #

        results.append({
            'Model': name,
            'Average Accuracy (CV Mean Score)': cv_mean,
            'Test Score': test_score
        })

        # Update the best model if this model has a better score
        if cv_mean > best_score:
            best_score = cv_mean
            best_model = model

    results_df = pd.DataFrame(results)

    # Predict with the best model for the test dataset
    print(f"{best_model} is the best model with a CV mean score / accurac
    predicted_col_name = f'Predicted {target}'

    # Only predict for X_test and append to the original df
    test_predictions = best_model.predict(X_test)
    test_df = df.loc[X_test.index].copy()
    test_df[predicted_col_name] = test_predictions
```

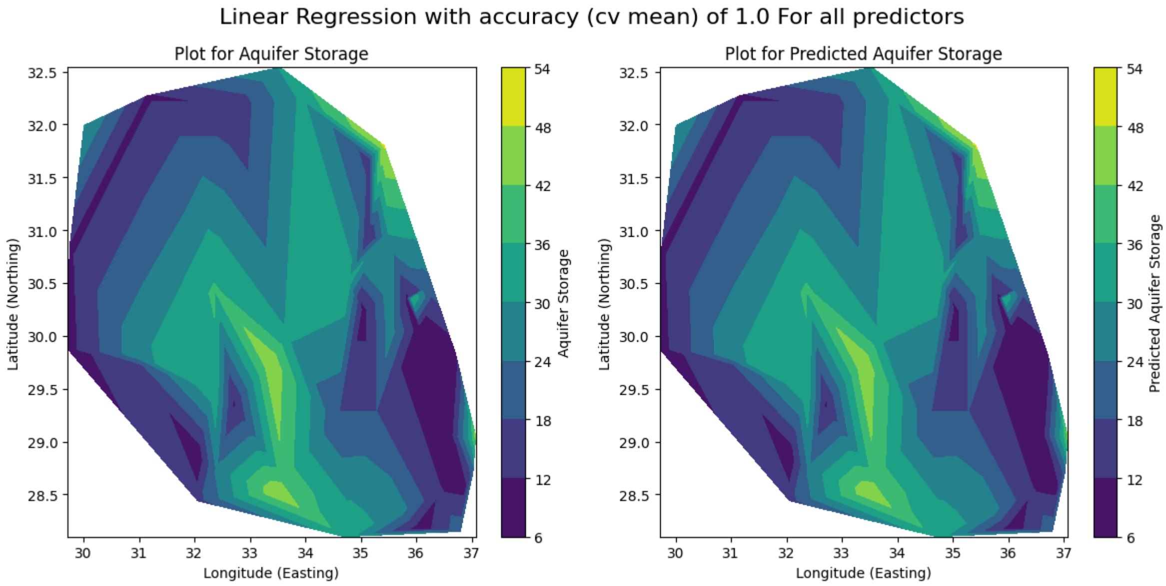
```
    # Plotting or additional operations can be added here
    # For simplicity, this part is omitted
    plot_features(test_df, [target, predicted_col_name], title=title)


    # Return the evaluation results and the DataFrame with predictions on
    return results_df, test_df
```

In [ ]:
```
def evaluate_models(df, features, target, models=None, title=""):
    # Split the data into features (X) and target (y)
    X = df[features]
    y = df[target]

    default_models = [
        ('Linear Regression', LinearRegression()),
        ('Decision Tree Regressor', DecisionTreeRegressor()),
        ('Random Forest Regressor', RandomForestRegressor()),
        ("Linear SVR", LinearSVR())
    ]

    # If no models are provided, use the default models
    if models is None:
        models = default_models

    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

    results = []
    best_score = float('-inf')
    best_model = None

    for name, model in models:
        # Perform cross-validation
        cv_scores = cross_val_score(model, X_train, y_train, cv=5)
        cv_mean = np.mean(cv_scores)

        # Fit the model on the full training data and evaluate on the tes
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        test_df = df.loc[X_test.index].copy()
        predicted_col_name = f'Predicted {target}'
        test_df[predicted_col_name] = y_pred
        plot_features(test_df, [target, predicted_col_name], title=f"{nam

        test_score = mean_squared_error(y_test, y_pred, squared=False) #

        results.append({
            'Model': name,
            'Average Accuracy (CV Mean Score)': cv_mean,
            'Test Score': test_score
        })

        # Update the best model if this model has a better score
        if cv_mean > best_score:
            best_score = cv_mean
            best_model = model

    results_df = pd.DataFrame(results)
```

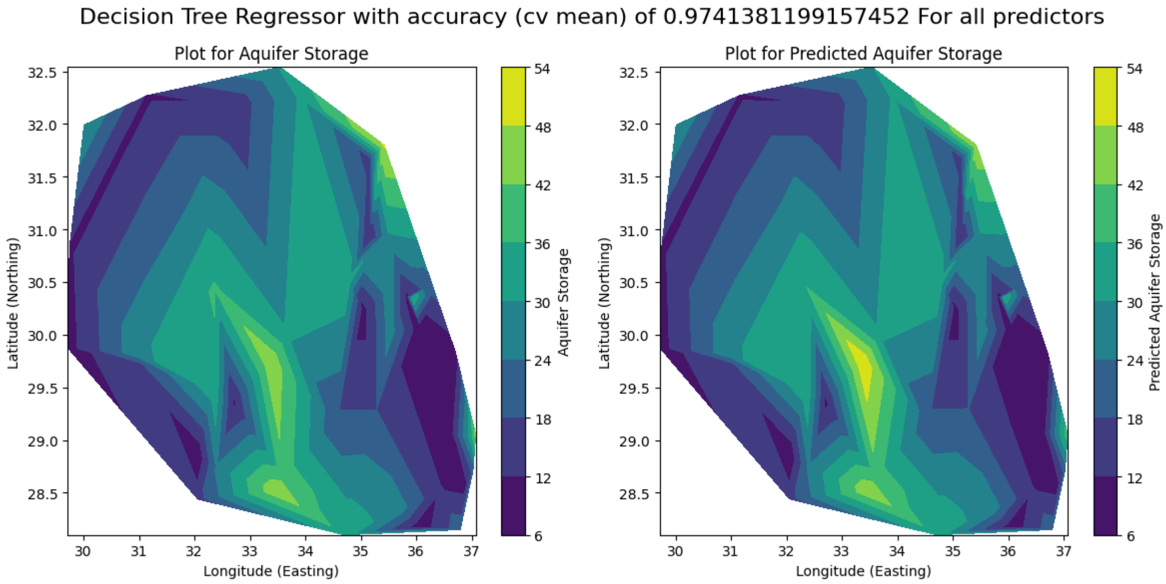
```
    # Print the best model
    print(f"{best_model} is the best model with a CV mean score / accurac
    # Predict with the best model for the test dataset
    predicted_col_name = f'Predicted {target}'
    test_predictions = best_model.predict(X_test)
    test_df = df.loc[X_test.index].copy()
    test_df[predicted_col_name] = test_predictions

    # Return the evaluation results and the DataFrame with predictions on
    return results_df, test_df
```

In [ ]:

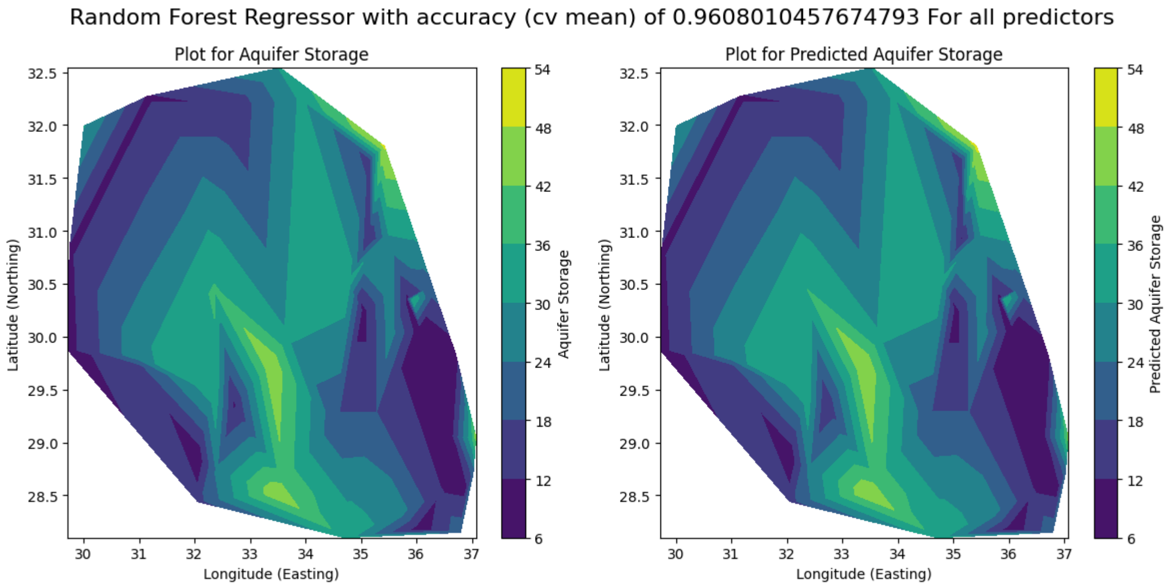
```
results, df_pred = evaluate_models(df_all_predicted, all_selected_feature
```



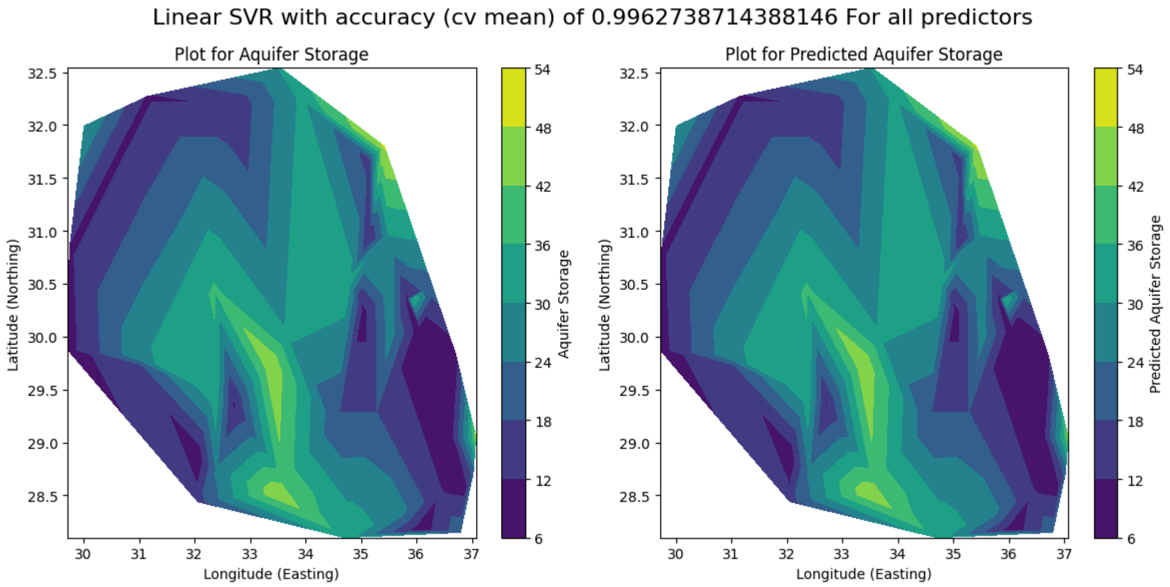

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```




```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```




```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
```

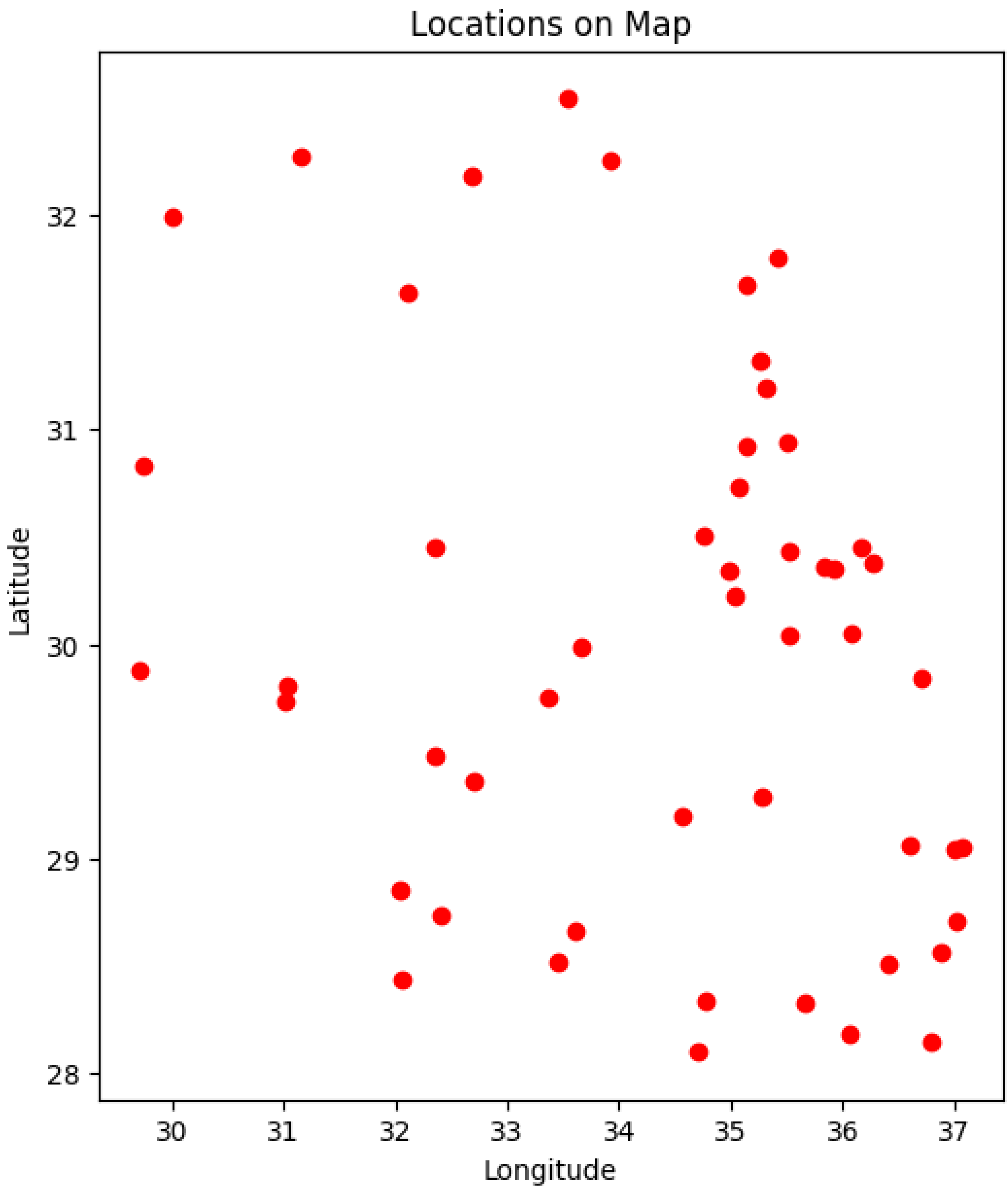



```
LinearRegression() is the best model with a CV mean score / accuracy score of 1.0.
```

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/sklearn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

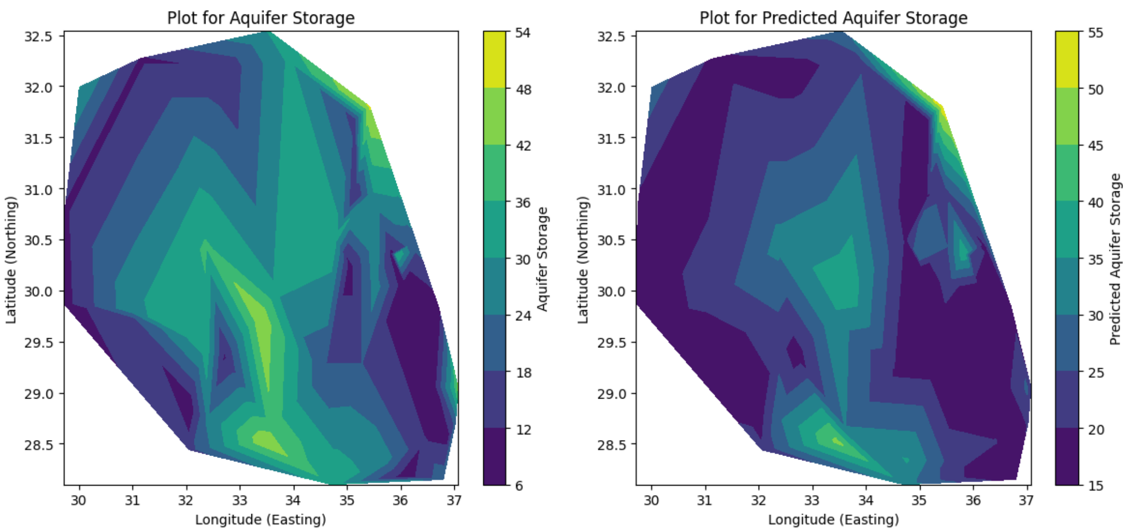
Plot data points of predicted values

In [ ]:
```
plot_locations(df_pred)
```



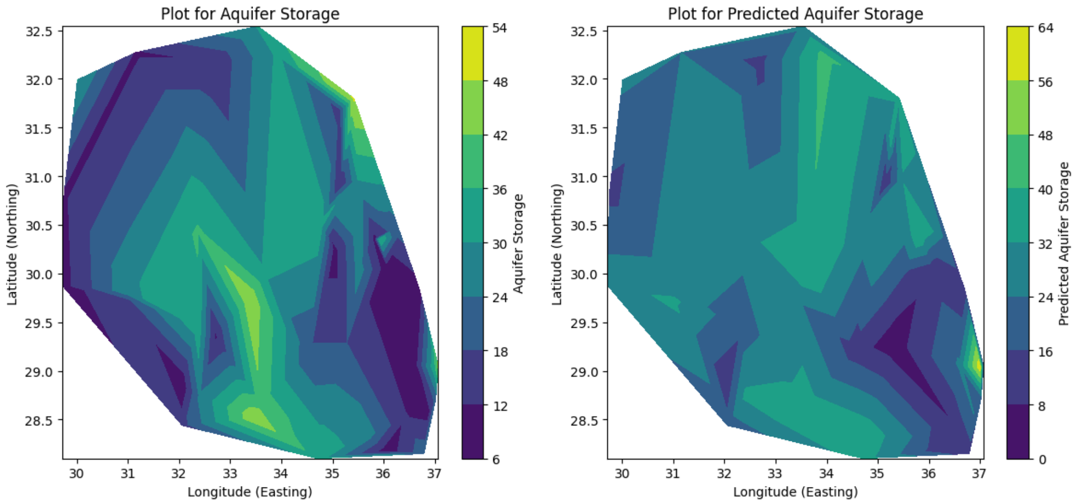
In [ ]:
```
results_whc, df_pred_whc  = evaluate_models(df_without_high_corr, feature
```

Linear Regression with accuracy (cv mean) of 0.18017291862228846 for predictors without high correlator



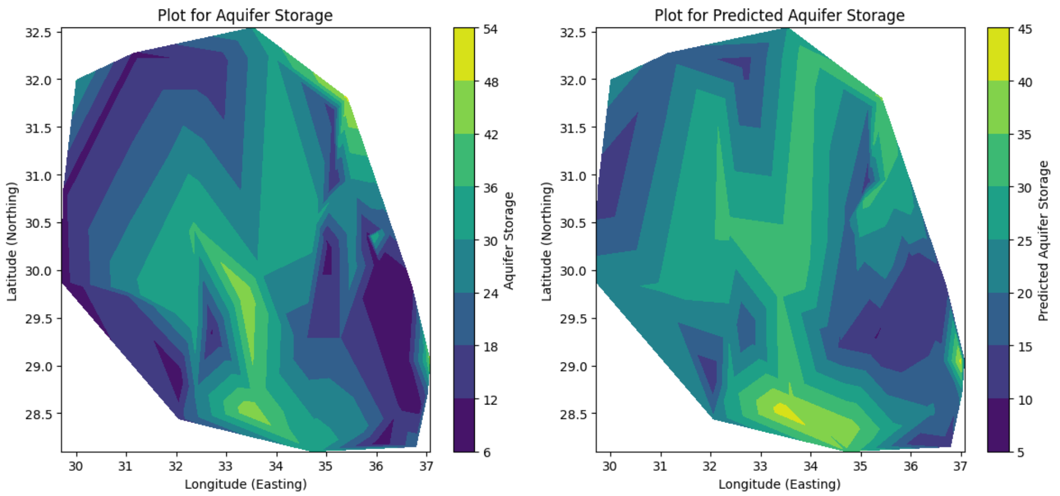
```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/sklearn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Decision Tree Regressor with accuracy (cv mean) of -0.4176076224031825 for predictors without high correlator

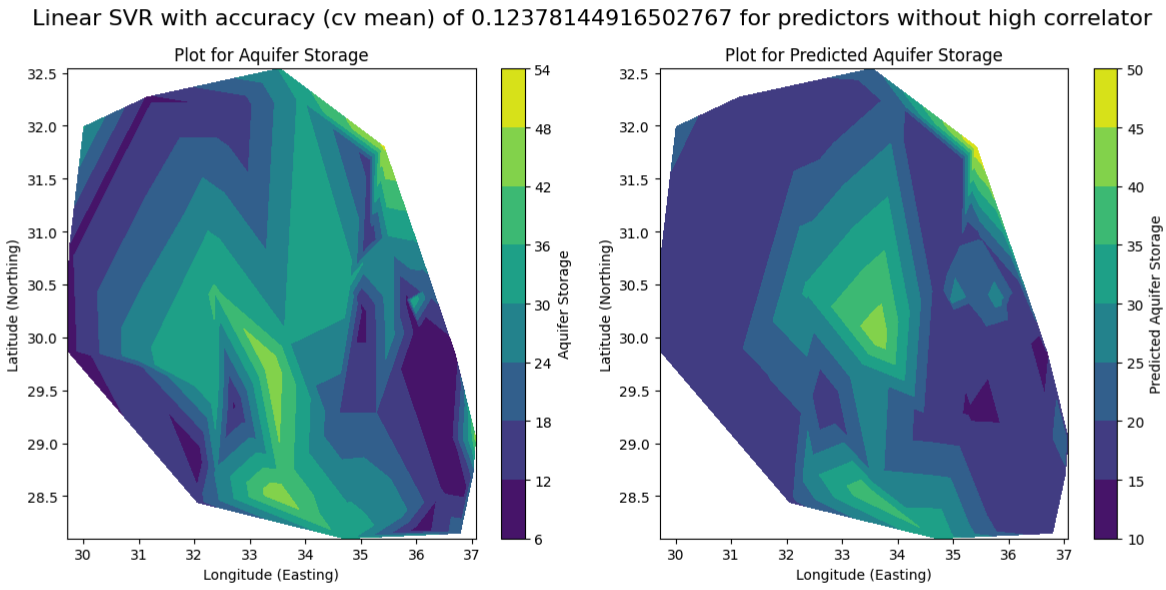



```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/sklearn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Random Forest Regressor with accuracy (cv mean) of 0.24115725302049845 for predictors without high correlator



```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
```

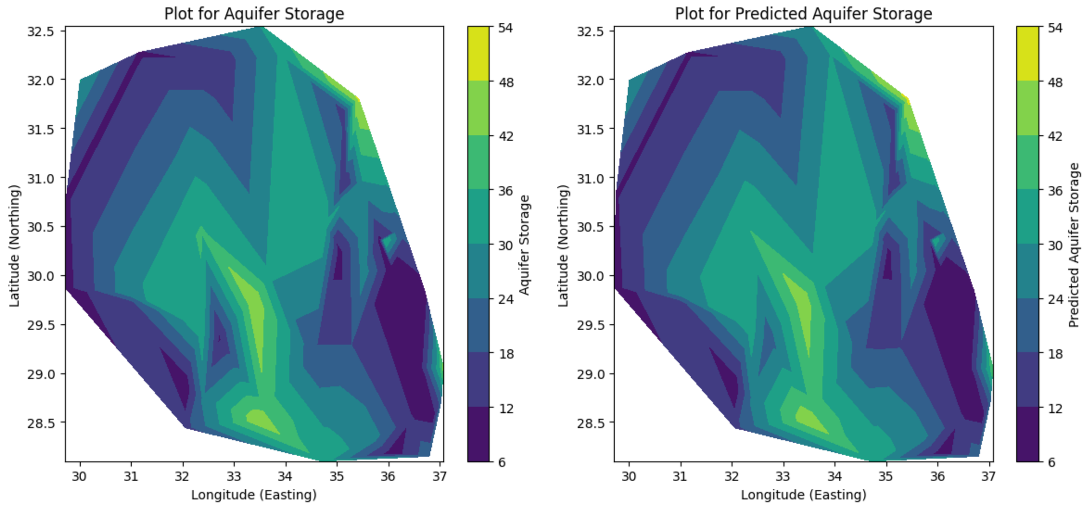



```
RandomForestRegressor() is the best model with a CV mean score / accuracy
score of 0.24115725302049845.
```

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

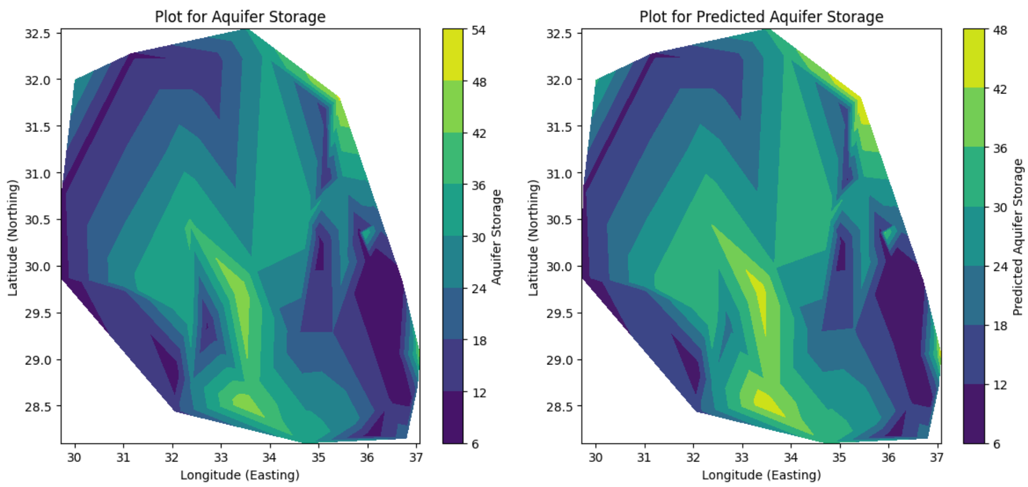
```
In [ ]: results_wwt, df_pred_wwt  = evaluate_models(df_without_weathered_thick, f
```

Linear Regression with accuracy (cv mean) of 0.9109362093015712 for prediction without weathered thickness



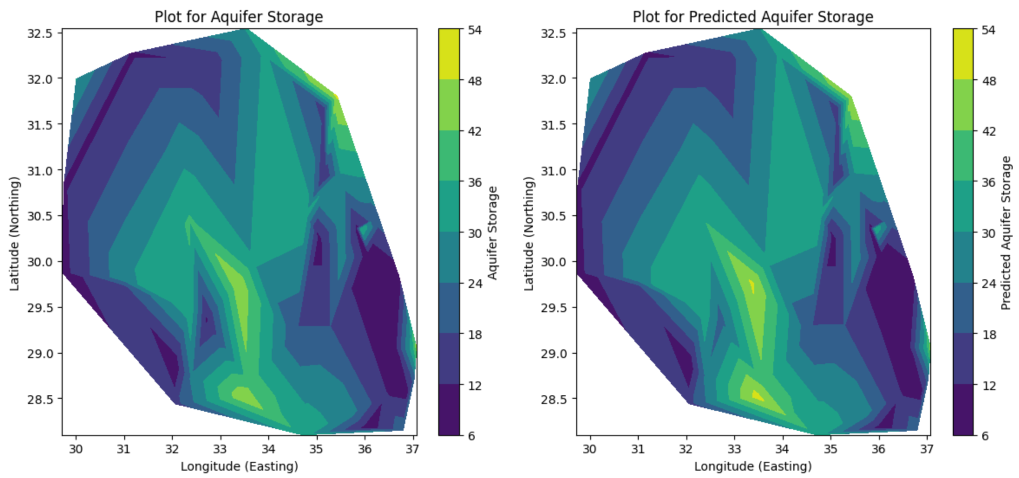

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Decision Tree Regressor with accuracy (cv mean) of 0.8454960928539178 for prediction without weathered thickness



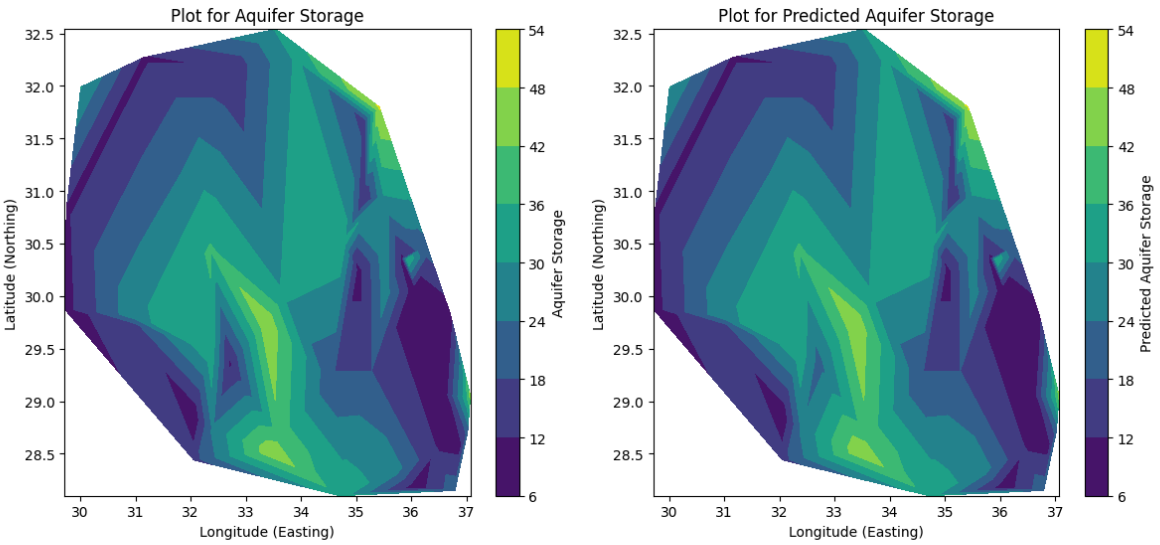

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Random Forest Regressor with accuracy (cv mean) of 0.8700824810439286 for prediction without weathered thickness



```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
```

Linear SVR with accuracy (cv mean) of 0.909417366942862 for prediction without weathered thickness

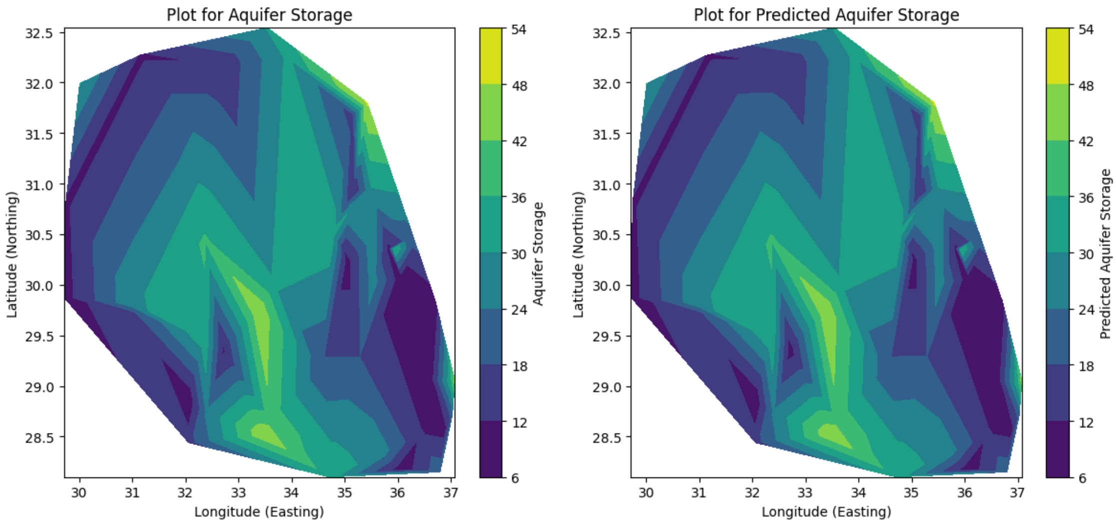



```
LinearRegression() is the best model with a CV mean score / accuracy score
of 0.9109362093015712.
```

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

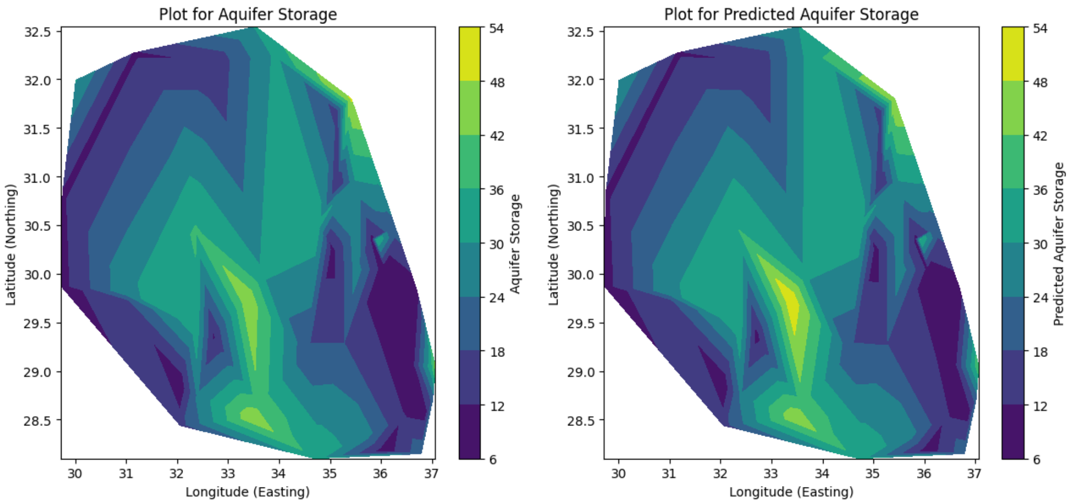
In [ ]:
```
results_wat, df_pred_wat  = evaluate_models(df_without_aquifer_thick, fea
```

Linear Regression with accuracy (cv mean) of 0.9973361885316774 for prediction without aquifer thickness



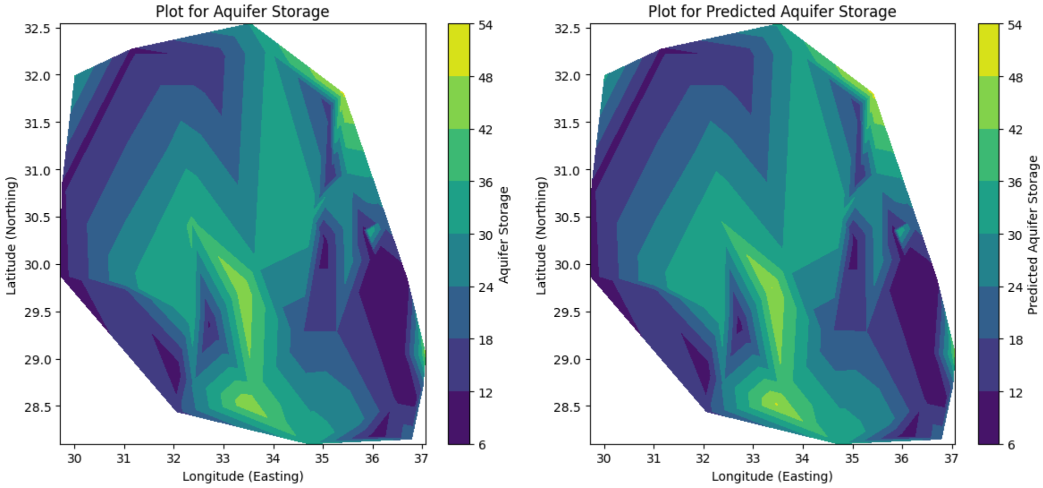

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Decision Tree Regressor with accuracy (cv mean) of 0.964061044687471 for prediction without aquifer thickness

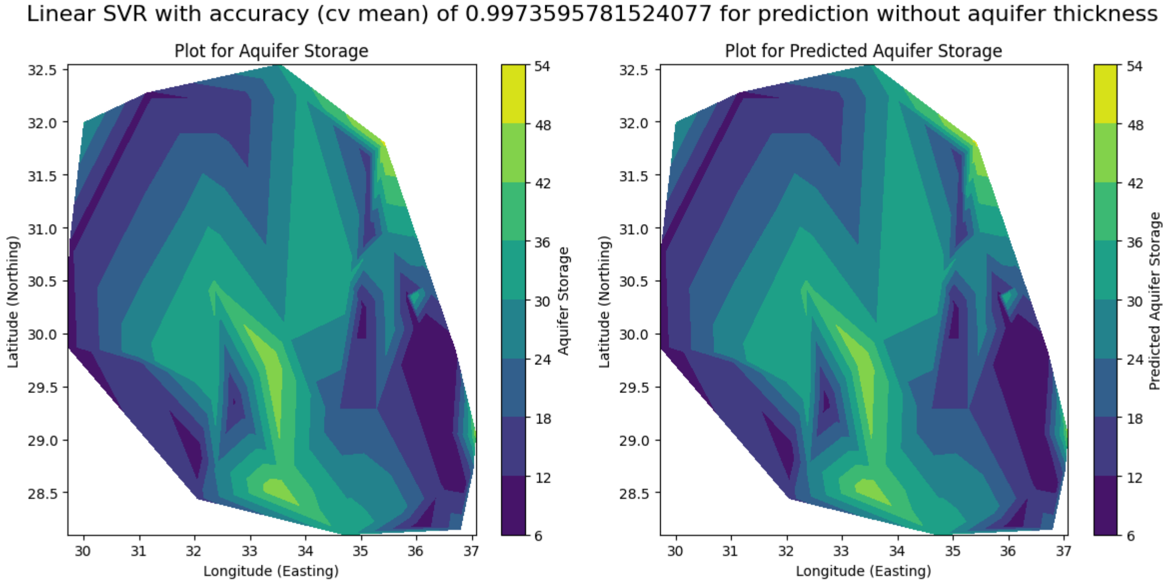



```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Random Forest Regressor with accuracy (cv mean) of 0.9795176464347648 for prediction without aquifer thickness




```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/svm/_classes.py:31: FutureWarning: The default value of `dual` will ch
ange from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to
suppress the warning.
  warnings.warn(
```




```
LinearSVR() is the best model with a CV mean score / accuracy score of 0.9
973595781524077.
```

```
/Users/ayomide/Work/ML/AquiferStorage/env/lib/python3.9/site-packages/skle
arn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Key Insights and Conclusion

The evaluation of the project on developing a machine learning model to predict aquifer storage capacity has yielded several important insights. A data split of 30:70 (test:training) was employed, and multiple models were trained to identify the one best suited for predicting aquifer storage. The models were evaluated across different feature groups, highlighting the significance of feature selection in machine learning. Below are the key insights derived from testing these feature groups:

- **All Selected Features**:
  - **Features**: Thickness Topsoil h1(m), Thickness weathered h3 (m), Thick. Overb.\nB1=h1+h2, Total Aquifer Thick. B2=h3+h4, Long. Cond. (mhos) OVERBURRDEN, Long. Cond. (mhos) Topsoil, Transmissivity (T).
  - **Result**: Linear Regression emerged as the best model with a cross-validation (CV) mean score of 1.0. This indicates a potentially perfect prediction, suggesting a linear relationship between these features and the target variable.
- **Features Without High Correlation**:
  - **Features**: Excluding the two highest correlators (with correlation scores of 0.98 and above), the features used were Thickness Topsoil h1(m), Thick. Overb.\nB1=h1+h2, Long. Cond. (mhos) Topsoil, Long. Cond. (mhos) OVERBURRDEN, Transmissivity (T).
  - **Result**: RandomForestRegressor was the best model with a CV mean score of 0.239. The drop in CV score compared to the all features model suggests that the excluded features were significant predictors of aquifer storage.

- **Features Without Weathered Thickness**:
  - **Features**: After removing one of the highest correlators, the selected features included Thickness Topsoil h1(m), Thick. Overb.\nB1=h1+h2, Total Aquifer Thick. B2=h3+h4, Long. Cond. (mhos) OVERBURRDEN, Long. Cond. (mhos) Topsoil, Transmissivity (T).
  - **Result**: Linear Regression again proved to be the best model with a CV mean score of 0.9109, highlighting its robustness in capturing the relationship between the selected features and the target variable even after the exclusion of a significant predictor.
- **Features Without Aquifer Thickness**:
  - **Features**: By excluding one of the highest correlators, the remaining features were Thickness Topsoil h1(m), Thickness weathered h3 (m), Thick. Overb.\nB1=h1+h2, Long. Cond. (mhos) OVERBURRDEN, Long. Cond. (mhos) Topsoil, Transmissivity (T).
  - **Result**: LinearSVR emerged as the best model with a CV mean score of 0.9974, indicating excellent performance and suggesting that the remaining features still capture significant predictive power.

Conclusion

The evaluation across different feature groups has shown that careful feature selection is paramount in developing predictive models. While Linear Regression showed outstanding performance in two of the feature groupings, suggesting a strong linear relationship, RandomForestRegressor and LinearSVR's best performances in other groupings indicate the complexity and non-linear nature of the relationships in groundwater potential prediction. This analysis underscores the necessity of experimenting with both feature selection and various modeling approaches to identify the best predictor of aquifer storage.

References

- Raji, W.O., Abdulkadir, K.A. Evaluation of groundwater potential of bedrock aquifers in Geological Sheet 223 Ilorin, Nigeria, using geo-electric sounding. Appl Water Sci 10, 220 (2020). https://doi.org/10.1007/s13201-020-01303-2