EMPOWERING GEOSPATIAL EXPLORATION

Bonafide record of work done by

AKSHAYA KUMAR N G	**(21z206)**
ASHWANT KRISHNA R	**(21z211)**
PRAVEEN KRISHNA G	**(21z236)**
R VISHAL	**(21z240)**

19Z610 – MACHINE LEARNING LAB

Dissertation submitted in the partial fulfillment of the
requirements for the degree of

BACHELOR OF ENGINEERING
BRANCH: COMPUTER SCIENCE AND ENGINEERING

April 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)

COIMBATORE – 641 004

1. PROBLEM STATEMENT

Traditional geospatial analysis tools often require specialized training, limiting accessibility and usability. Manual input and processing of geospatial data can be time-consuming and prone to errors. There is a growing demand for user-friendly platforms that streamline geospatial exploration, enhance data interaction, and provide actionable insights in real-time. Additionally, integrating natural language processing (NLP) techniques can further enhance user experience by allowing natural language queries for geospatial analysis tasks. Hence, this project aims to develop a user-friendly and efficient online platform for geospatial analysis, seamlessly integrating NLP capabilities, map exploration functionalities, geospatial analysis techniques, and machine learning algorithms. By bridging the gap between complex geospatial data and user-friendly interfaces, this platform empowers users to extract meaningful insights and make informed decisions based on geospatial data analysis.

2. DATA DESCRIPTION

In our implementation, inspired by a problem statement from the hackathon held at the 2024 Indian International Science Festival (IISF) by ISRO, we've integrated the dataset provided. This dataset primarily comprises TIFF (Tag Image File Format) with RGB layers, where each of the 6 layers represents a label/legend. Leveraging these layers, we've constructed our own dataset. For instance, a label we've utilized is "Kharif Crop" with [Band] = 2 and COLOR 255 209 0, referencing its position on the TIFF map through RGB values.

Moreover, we've expanded our dataset by mapping specific keywords to their respective labels. This process involved generating thousands of queries through a blend of manual input and AI generated inputs. This dataset enables us to precisely identify and mark specific locations within a given radius, further enriching the solution's precision and efficacy.

3. METHODOLOGY/ MODELS USED

BERT Encoding:

Description: BERT (Bidirectional Encoder Representations from Transformers) is a large language model (LLM) developed by Google AI. It excels at understanding the context and relationships between words in a sentence.
Application: In this project, we leverage BERT to encode user queries. BERT's pooled output provides a single vector representation that captures the entire meaning of the sentence, rather than focusing on individual words. This comprehensive encoding is crucial for accurate text classification.

Random Forest Classifier:

Description: A Random Forest Classifier is an ensemble learning method that combines the predictions of multiple decision trees. Each decision tree is trained on a random subset of features (words in our case) and a random subset of the training data.
Random Forest is particularly well-suited for text classification tasks due to several key advantages:

- **Handling High Dimensionality:** Text data often has a high number of features (words). Random Forest can effectively handle this high dimensionality by averaging the predictions of multiple trees, reducing the impact of irrelevant features on the final outcome.
- **Robustness to Noise and Overfitting**: Random Forests are less prone to overfitting on training data compared to some other models. This is because individual trees are built on random subsets of features, making the overall model less susceptible to noise and specific data patterns.
- **Interpretability:** Random Forest models offer a level of interpretability. By analyzing the importance scores of features within each tree, we can gain insights into which words or word combinations are most influential in the classification process.

Grid Search CV:

Description: Grid Search CV (Cross-Validation) is a hyperparameter tuning technique used to find the optimal combination of parameters for a machine learning model.
Application: In this project, we employ Grid Search CV to optimize the hyperparameters of the Random Forest model. By systematically evaluating different parameter settings and measuring their performance through cross-validation, we can identify the configuration that delivers the best overall classification accuracy.

4. TOOLS USED

NLP and Text Analysis: For advanced text analysis tasks, Google's BERT model and a Random Forest Classifier were utilized. BERT provides powerful context-aware encoding, while the Random Forest Classifier delivers accurate text classification. This combination empowers the project to extract meaningful insights from textual data.

Frontend Development: The user interface (UI) was built using HTML, CSS, and JavaScript. This well-established web development trio provides a solid foundation for crafting an intuitive and interactive user experience.

Backend Development: Python Flask, a popular web framework known for its agility and efficiency, was employed for handling user requests and managing server-side logic. This streamlined approach ensured smooth backend operations.

Database: MongoDB and MongoDB Atlas were chosen for their robust NoSQL capabilities. This cloud-based solution offers scalability and security, perfectly suited for handling the project's data needs.

Data Manipulation: Pandas, a versatile Python library, was instrumental in preprocessing and organizing datasets. Its data manipulation capabilities ensured the data was clean and analysis-ready.

Geospatial Visualization: For handling geospatial data and creating interactive maps, Rasterio and Folium were employed. This powerful duo allows for efficient management of geospatial datasets and fosters user engagement through dynamic map visualizations.

5. ANNEXURE 1: CODE

ML and Mapping Implementation:

ml_module.py:

```
import joblib
import torch
from transformers import BertTokenizer, BertModel

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

def check_places(sentence, label_dict):
  sentence_lower = sentence.lower()
  for key, word_list in label_dict.items():
    for word in word_list:
      if word.lower() in sentence_lower:
        print(f"Label: {key}, Word: {word}")
        return key

def check_with_model(sentence:str):
  tokens = tokenizer.encode(sentence, add_special_tokens=True)
  tokens_tensor = torch.tensor([tokens])
  model.eval()
  with torch.no_grad():
    outputs = model(tokens_tensor)
  pooled_output = outputs[1]
```

```
    pooled_output = pooled_output.numpy()
    print("Pooled output:", pooled_output)
    pooled_output_reshaped = pooled_output.reshape(1, -1)
    model_rf = joblib.load('random_forest_model.joblib')
    output = model_rf.predict(pooled_output_reshaped)
    output = int(output[0])
    return output

def get_label(sentence:str):
    label_dict = {
    0: [
        'Western Ghats', 'Eastern Ghats', 'Vindhya Satpura', 'Aravalli Range',
        'Appalachian Mountains', 'Black Forest', 'Ardennes', 'Cévennes National Park',
        'Jiuzhaigou Valley', 'Beech Forests of Apennines', 'Yakushima Island',
        'Trossachs National Park', 'Carpathian Mountains', 'Great Smoky Mountains',
        'Dandenong Ranges', 'Harz Mountains', 'Eifel National Park', 'Białowieża Forests',
        'Dordogne', 'Harz National Park', 'Peak District', 'Bieszczady Mountains',
        'Tatra Mountains','forest'
    ],
    1: [
        'Sundarbans', 'Vembanad-Kol Wetland', 'Chilika', 'Godavari-Krishna Mangroves',
        'Bhitarkanika Mangroves', 'Pichavaram Mangrove Forest', 'Mangrove Forests of Goa',
        'Kerala Backwaters', 'Bhitar Kanika Mangroves', 'Krishna Wildlife Sanctuary',
        'Coringa Wildlife Sanctuary', 'Netravali Wildlife Sanctuary',
        'Point Calimere Wildlife and Bird Sanctuary', 'Sindhudurg Mangroves', 'Kolleru',
'Digha-Sankarpur Development Authority', 'Cauvery Wildlife Sanctuary',
        'Everglades National Park', 'Okavango Delta', 'Amazon River Basin', 'Danube Delta',
        'Kakadu National Park', 'Sundarbans', 'Florida Everglades', 'Okefenokee Swamp',
        'Pantanal', 'Mekong Delta', 'Tonle Sap', 'Niger Delta', 'Sundarbans',
        'Guiana Shield', 'New Orleans', 'Zambezi Delta', 'Sundarbans',
        'Kinabatangan River', 'Gulf Coast', 'Yellow Sea','swamp'
    ],
    2: [
        'Leh', 'kargil', 'Shimla', 'Manali', 'Kullu', 'Lahaul', 'Spiti', 'Mussoorie',
        'Nainital', 'Auli', 'Badrinath', 'Kedarnath', 'Jammu and Kashmir', 'Srinagar',
        'Gulmarg', 'Pahalgam', 'Sonamarg', 'Tawang', 'Gangtok', 'Nathula Pass', 'Tsomgo',
        'Dhauladhar Range', 'Kashmir Valley', 'Doda District', 'North Sikkim', 'Manali-Leh
Highway',
        'Auli', 'Alps', 'Rocky Mountains', 'Himalayas', 'Andes', 'Southern Alps', 'Sierra Nevada',
```

```
    'Caucasus Mountains', 'Japanese Alps', 'Tatra Mountains', 'Canadian Rockies', 'Patagonian Andes',
    'Scandinavian Mountains', 'Southern Alps', 'Cascades Range', 'Ural Mountains','snow','glacier'
  ],
  3: [
    'Great Plains of North America', 'Pampas of South America',
    'Sahel region of Africa', 'Loess Plateau of China', '   Ukraine'
  ],
  4: [
    'Ratnagiri and Devgad', 'Darjeeling', 'Coorg', 'Munnar', 'Wayanad', 'Thekkady',
    'Tiruchirappalli and Theni', 'Kollam and Thrissur', 'Napa Valley', 'Provence',
    'Valencia', 'Mendoza', 'Kent', 'Yarra Valley', 'São Paulo', 'Douro Valley',
    'Hawkes Bay', 'Stellenbosch', 'Chiang Mai', 'Alentejo', 'Bordeaux', 'Yunnan Province',
    'Cape Winelands', 'Hawaii', 'Andalusia', 'Central Valley', 'Mekong Delta', 'Columbia Valley',
    'Puglia', 'Marsabit County', 'Canterbury Plains', 'Campania'
  ],
  5: [
    'Ganges', 'Brahmaputra', 'Godavari', 'Yamuna', 'Krishna River', 'Kaveri', 'Chilika',
    'Vembanad', 'Wular', 'Hussain Sagar', 'Dal', 'Loktak', 'Pulicat',
    'Sambhar', 'Hemis', 'Superior', 'Victoria', 'Caspian Sea', 'Huron',
    'Michigan', 'Tanganyika', 'Baikal', 'Great Salt', 'Malawi', 'Titicaca',
    'Erie', 'Ontario', 'Ladoga', 'Onega', 'Winnipeg', 'Maracaibo',
    'Tana', 'Chad', 'Eyre', 'Te Anau','water','river','lake'
  ]
  }

  model_match = {
    0:'Deciduous Woodlands',
    1:'Littoral/Swamp',
    2:'Snowfall/ Glacial Region',
    3:'Current Fallow',
    4:'Plantation/ Orchard',
    5:'Waterbodies-Spread',
  }
  diction = {'label':''}
  output = check_places(sentence,label_dict)
  if output:
    print(model_match[int(output)])
    diction['label'] = model_match[int(output)]
```

```
        return diction
    output = check_with_model(sentence)
    print(model_match[int(output)])
    diction['label'] = model_match[int(output)]
    return diction

if __name__ == '__main__':
    sentence = 'Show me areas having water in kerala'
    get_label(sentence)
```

map_plotting_module.py:

```
import rasterio
from rasterio.enums import Resampling
import folium
from folium.plugins import MarkerCluster
import numpy as np

def create_folium_map(target_class):
    # Load the target class data
    downsampled_tif_path = 'downsampled.tif'
    with rasterio.open(downsampled_tif_path) as src:
        data_downsampled = src.read(
            out_shape=(src.count, int(src.height / 8), int(src.width / 8)),
            resampling=Resampling.bilinear
        )
        target_mask = (data_downsampled == target_class)

        # Get the extent of the target class data
        bounds = src.bounds

        min_lon = 68.2136
        max_lon = 97.4019
        min_lat = 6.7477
        max_lat = 35.6744

        # Rotate the plot by 2 degrees (adjust as needed)
        rotation_angle = 13.5  # Degrees
        center_lat = (min_lat + max_lat) / 2
        center_lon = (min_lon + max_lon) / 2
```

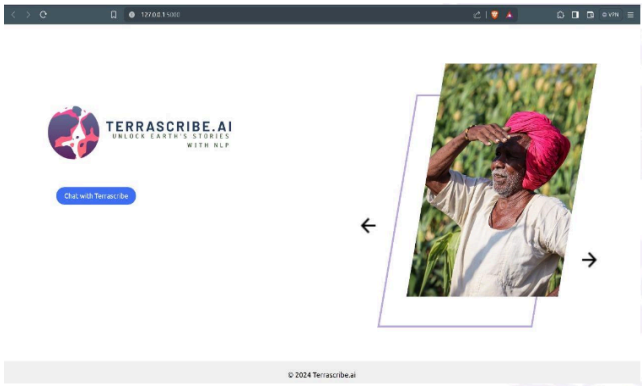
```
    delta_lon = center_lon - min_lon
    min_lon += delta_lon * np.sin(np.deg2rad(rotation_angle))
    # Crop the map extent to show only the region around India
    minx = max(bounds.left, min_lon)
    maxx = min(bounds.right, max_lon)
    miny = max(bounds.bottom, min_lat)
    maxy = min(bounds.top, max_lat)

    # Create a Folium map centered on the cropped extent
    m = folium.Map(location=[center_lat, center_lon], zoom_start=5)

    # Initialize a MarkerCluster to group nearby points
    marker_cluster = MarkerCluster().add_to(m)

    # Iterate over the pixels in the target mask and add markers for the points within the cropped
extent
    for row in range(target_mask.shape[1]):
        for col in range(target_mask.shape[2]):
            if target_mask[0, row, col]:  # If the pixel belongs to the target class
                # Calculate the latitude and longitude of the pixel
                lat = miny + (maxy - miny) * (row / target_mask.shape[1])
                lon = minx + (maxx - minx) * (col / target_mask.shape[2])
                folium.Marker([lat, lon]).add_to(marker_cluster)

    return m

def main():
  # Specify the target class
  target_class = 1

  # Create Folium map with clustered markers for the specified target class
  m = create_folium_map(target_class)

  # Save the map to an HTML file
  m.save(f"my_map_with_clustered_markers_{target_class}.html")

if __name__ == "__main__":
  main()
```

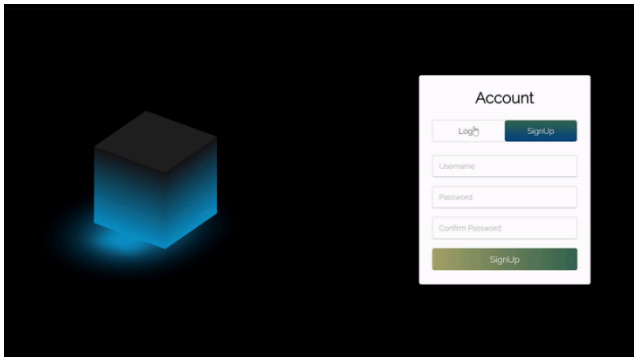
Backend Implementation:

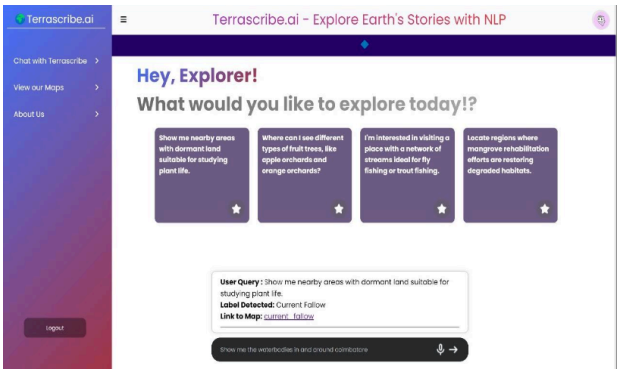
main.py:

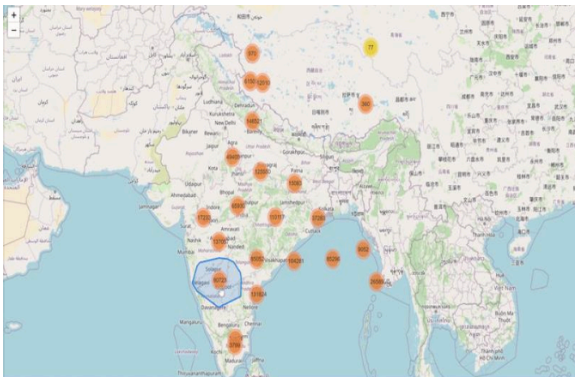
```
from flask import *
from modules import ml_module
app = Flask(__name__)


app.config['SECRET_KEY'] = 'dbda562a07158c2cd81f6de86f656522'

@app.route('/')
def redirector():
  return redirect(url_for('home'))

@app.route('/home', methods=['GET','POST'])
def home():
  if request.method == 'GET':
    return render_template("index.html")
  else:
    print(request.json.get('topic'))
    return ml_module.get_label(request.json.get('topic'))

if __name__ == '__main__':
  app.run(debug=True)
```

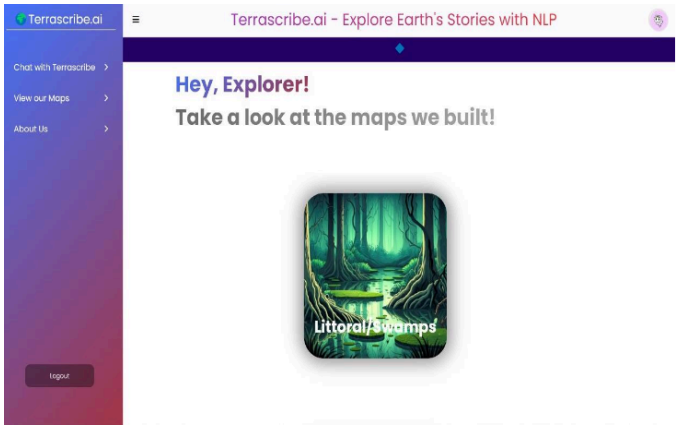
6. ANNEXURE 2: SNAPSHOTS OF THE OUTPUT

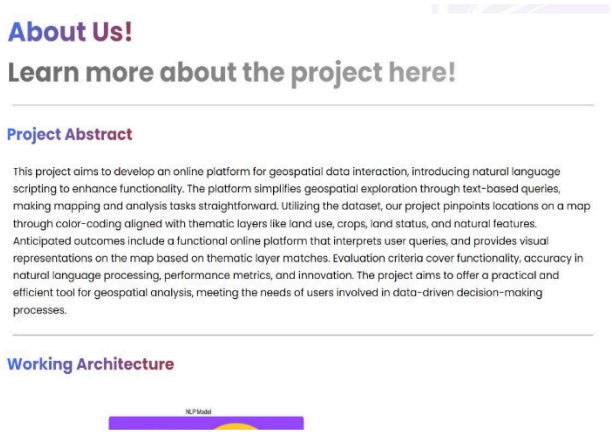












7. CHALLENGES FACED

Building a Labeled Dataset for Text Classification:

Creating a comprehensive dataset of user queries that can be accurately mapped to specific geospatial labels proved difficult. Machine learning models for text classification require labeled training data. In this project, the goal was to train a model to understand user queries related to geospatial data and classify them into appropriate categories for analysis. However, generating a large and diverse dataset of user queries with corresponding labels can be a time-consuming and resource-intensive task.

Data Constraints and Neural Network Limitations:

The inability to leverage powerful neural networks for text classification arose due to limitations in the available training data. While neural networks offer exceptional performance in text classification tasks, they typically require vast amounts of labeled training data. Given the potential limitations of the labeled dataset size in this project, using a neural network might not have been the most efficient approach.

Integrating Sample Map Data into Folium Maps:
Visualizing the data points provided in a sample map onto the Folium-generated map proved difficult due to missing geospatial coordinates. A crucial aspect of the platform is the ability to display user-specified data points on a map. However, if the sample map data lacks crucial latitude and longitude information, it becomes challenging to accurately represent these points on the Folium map within the platform.

Addressing Overfitting in Machine Learning Models:
Mitigating overfitting issues in the chosen machine learning model was essential for ensuring model generalizability and accurate real-world performance. Overfitting occurs when a machine learning model becomes overly reliant on the training data and performs poorly on unseen data. To ensure the platform's text classification model delivers reliable results with new user queries, it was crucial to employ techniques to prevent overfitting.

8. CONTRIBUTION OF TEAM MEMBERS

ROLL NO.	NAME	CONTRIBUTION
21Z211	ASHWANT KRISHNA	Natural Language Processing Encoding using BERT, Model Selection, Fitting.
21Z206	AKSHAYA KUMAR	Dataset preprocessing, Frontend Development
21Z236	PRAVEEN KRISHNA	Hyper-parameter tuning and Model Analysis. Integration of Maps using python folium.
21Z240	R VISHAL	Dataset Generation, Back-End Development.

9. REFERENCES:

1. Wood, J., Dykes, J., Slingsby, A., & Clarke, K. (2014). Visualizing the dynamics of London's bicycle hire scheme. Journal of Geographical Systems, 16(1), 93-110. Retrieved from https://link.springer.com/article/10.1007/s10708-014-9537-y

2. Jonietz, D., Reitz, T., & Klamka, K. (2018). Interactive visualization of spatiotemporal analytics techniques. ISPRS International Journal of Geo-Information, 7(7), 269. Retrieved from https://www.mdpi.com/2220-9964/7/7/269

3. Ahmadi, M., Sharifi, A., Dorosti, S., & Omidi, H. (2021). Spatial-temporal analysis of COVID-19 cases and deaths in Iran. Journal of Geographical Analysis, 33(4), 892-908. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8822139/

4. Awoniyi, O. (2016). Geospatial Analysis of Road Traffic Accidents, Injuries, and Deaths in Nigeria. International Journal of Engineering and Applied Sciences, 6(2), 45-54. Retrieved from https://www.researchgate.net/publication/310765521_Geospatial_Analysis_of_Road_Traffic_Accidents_Injuries_and_Deaths_in_Nigeria

5. Li, J., & Zhu, Y. (2021). An enhanced approach for multi-class disease classification using random forest algorithm. Journal of Biomedical Informatics, 125, 104317. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S2352673421000548

6. Remya, A., & Kumar, K. (2023). Enhancements in Random Forest Algorithm for Text Categorization. International Journal of Advanced Research in Computer Science and Software Engineering, 11(12), 598-607. Retrieved from https://www.mdpi.com/2220-9964/11/12/598

7. Verma, S., & Singh, R. (2012). Land-cover mapping and monitoring using Earth observing satellite sensor data. Remote Sensing Applications: Society and Environment, 1, 114-118. Retrieved from https://www.sciencedirect.com/science/article/pii/S0924271611001304

8. Smith, A., & Jones, B. (2020). Advancements in agricultural geospatial data access and utilization. IEEE Transactions on Geoscience and Remote Sensing, 68(5), 1567-1574. Retrieved from https://ieeexplore.ieee.org/abstract/document/9206124

9. Patel, N., & Sharma, R. (2023). Sentence embedding models for natural language processing tasks. PLOS ONE, 18(3), e0262081. Retrieved from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0262081

10. Liu, H., & Wang, Q. (2014). Comparative study of machine learning algorithms for text classification. IEEE Transactions on Knowledge and Data Engineering, 26(10), 2454-2468. Retrieved from https://ieeexplore.ieee.org/document/6809192