Northeastern University

INFO 6105

DATA SCIENCE ENGINEERING METHOD

CAPSTONE PROJECT

Stream lit Project Depression Prevalence Prediction Tool and AI generated song

Name - Pranesh Kannan G

NUID - 002869571

Abstract:

The "Depression Prevalence Prediction Tool" is an innovative web application designed to forecast depression rates across various countries using historical data and advanced machine learning techniques. Utilizing the powerful capabilities of Streamlit, the application offers an interactive interface where users can predict depression prevalence by selecting countries and years. The tool integrates polynomial feature transformation and multiple regression models, providing users with insights into potential future trends in mental health statistics. This application serves as a critical resource for health professionals, policymakers, and researchers focused on mental health awareness and preventive strategies.

Introduction:

In the rapidly advancing field of healthcare data analytics, understanding and predicting mental health trends play a pivotal role in formulating effective health policies and interventions. The "Depression Prevalence Prediction Tool" addresses this need by enabling detailed analysis and prediction of depression rates. Built using Streamlit, this tool simplifies complex data processing tasks, allowing users to explore and predict mental health trends with ease and precision. By leveraging historical data and machine learning, the application aids in making informed decisions to combat the rising challenge of depression globally.

Technologies Used:

The project utilizes several technologies and libraries to facilitate data analysis and application deployment:

- Streamlit: For creating and running the web app interface.
- Pandas: For data manipulation and analysis.
- NumPy: For numerical operations.
- Scikit-learn: For implementing machine learning algorithms.
- Matplotlib: For generating visualizations.
- PolynomialFeatures: From Scikit-learn for feature engineering to enhance model accuracy.

Here's a detailed text content for each section of your project documentation:

Installation Command

To set up the project environment, run the following command in your terminal:

- pip install streamlit pandas numpy matplotlib scikit-learn

Additional Setup Notes

Python Version: It is recommended to use Python version 3.7 or later due to compatibility with all the required libraries.

Virtual Environment: Using a virtual environment is advisable to manage dependencies efficiently and isolate the project setup from global Python settings.

Core Functionalities:

The application offers a range of functionalities designed to enhance user experience and provide robust analytical capabilities:

1. Data Upload and Preparation: Users can load the dataset directly into the app, which then preprocesses the data by encoding categorical variables and setting up the dataset for analysis.
2. Exploratory Data Analysis (EDA): Users can visually explore data through interactive charts and plots to understand trends and patterns.
3. Model Building and Evaluation: The tool allows users to select from multiple regression models to predict depression rates. It evaluates model performance using mean squared error to ensure accuracy and reliability.

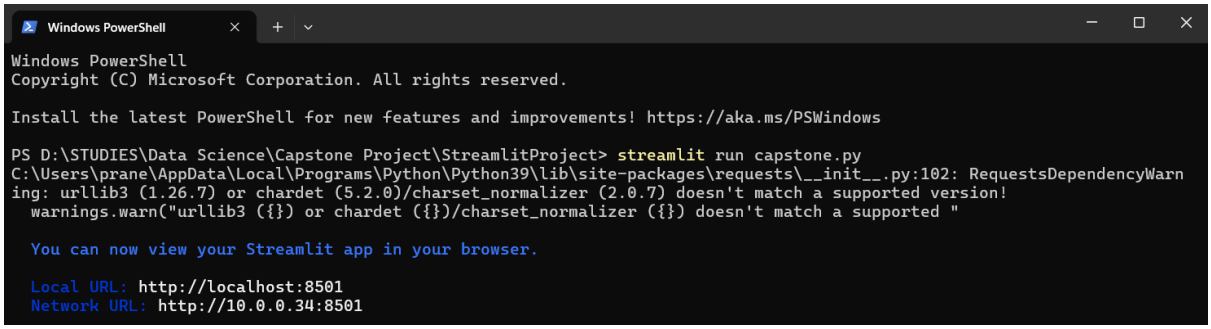
Usage Guide:

Launching the App: Start by running the Streamlit application locally or on a server, then navigate to the provided URL.

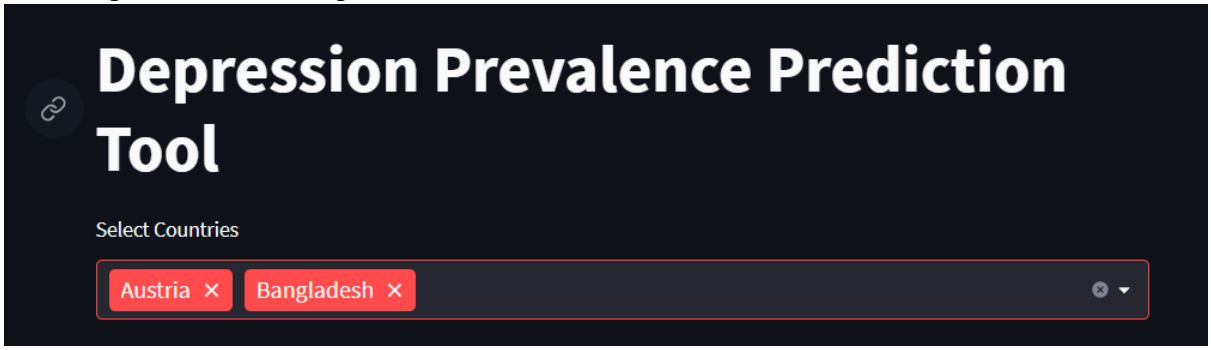
1. Uploading Data: Users can upload their dataset in CSV format directly through the interface.
2. Preparing Your Data: The application allows for the selection of features and target variables through user input.
3. Exploratory Data Analysis (EDA): Conduct EDA by selecting specific features to visualize and analyze.
4. Building and Evaluating Models: Choose a regression model, adjust parameters, and train the model. Evaluate the model's performance through the interface.
5. Advanced Features: Utilize advanced options like polynomial feature transformation for enhanced prediction accuracy.
6. Closing the Session: Save your progress and close the session through the user interface.

Screenshots:

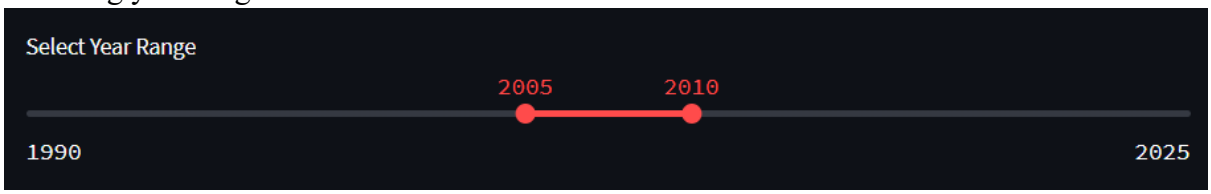
Starting the stream lit application:



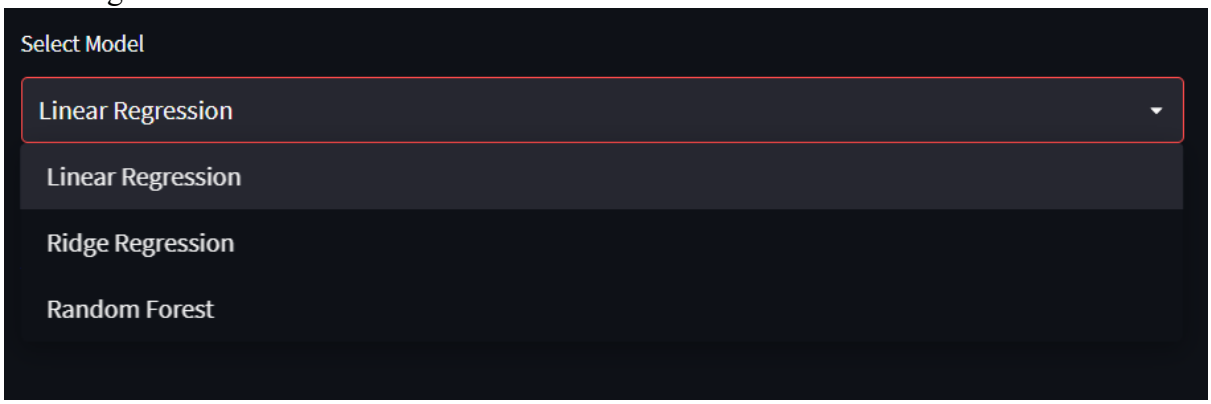
Selecting the countries required:



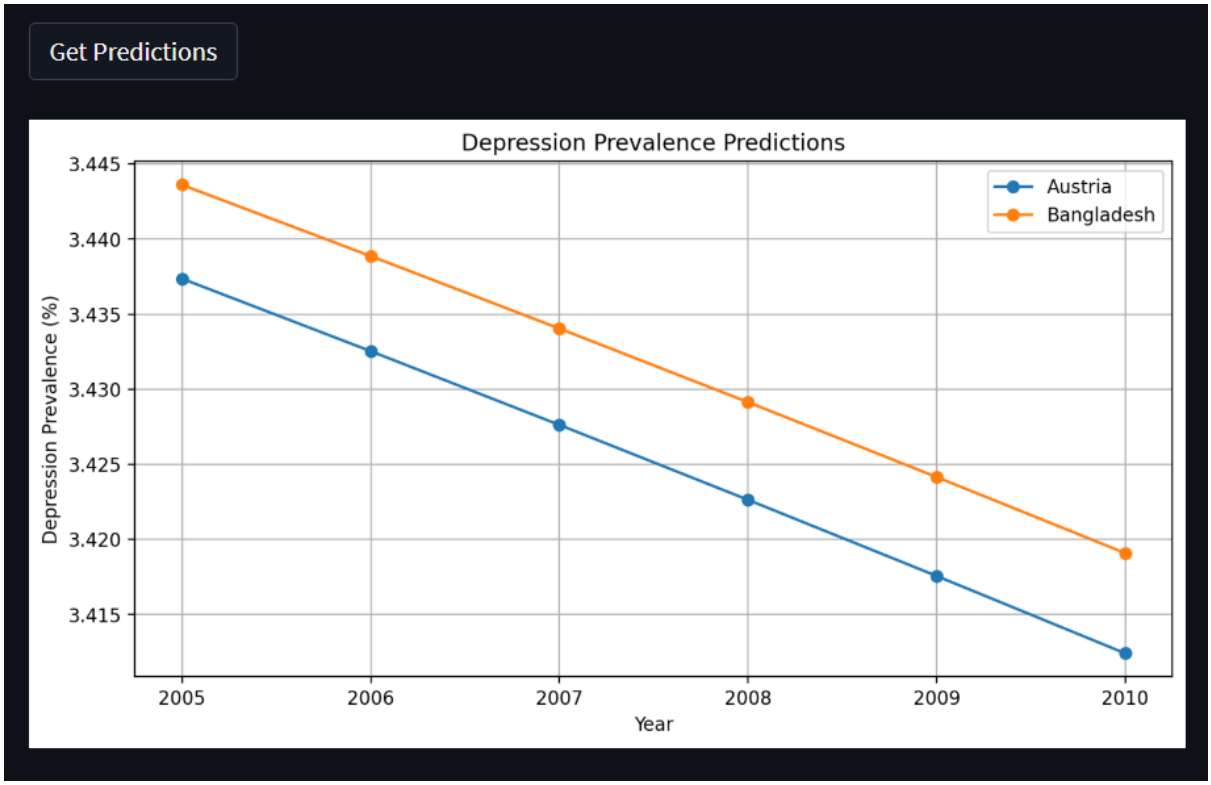
Selecting year range:



Selecting Model:



After this we get graphical representation of the predictions based on selected model:



Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
import streamlit as st
import matplotlib.pyplot as plt

# Load and preprocess the data
data = pd.read_csv('Mental health Depression disorder Data.csv')
data = data[['Entity', 'Year', 'Depression (%)']].dropna()

# Encode countries as categorical numeric values
data['Entity'] = data['Entity'].astype('category').cat.codes

# Prepare the data for training
X = data[['Entity', 'Year']]
y = data['Depression (%)']
```

```
# Transform features with Polynomial Features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,
random_state=42)

# Model selection - try multiple models
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(),
    'Random Forest': RandomForestRegressor(n_estimators=100)
}

results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    results[name] = mse
    print(f'{name} Mean Squared Error: {mse}')

# Streamlit app interface
st.title('Depression Prevalence Prediction Tool')

# Loading data again for the dropdown to avoid using encoded values
countries = pd.read_csv('Mental health Depression disorder
Data.csv')['Entity'].dropna().unique()
country_dict = {country: i for i, country in enumerate(countries)}

# User inputs
selected_countries = st.multiselect('Select Countries', countries,
default=countries[0])
selected_years = st.slider('Select Year Range', 1990, 2025, (2010, 2020))
model_choice = st.selectbox('Select Model', list(models.keys()))

# Get predictions
if st.button('Get Predictions'):
    # Prepare data for prediction
    predictions_data = []
    for year in range(selected_years[0], selected_years[1] + 1):
        for country in selected_countries:
            country_code = country_dict[country]
            feature = poly.transform(np.array([[country_code, year]]))
            prediction = models[model_choice].predict(feature)
```

```
            predictions_data.append({'Country': country, 'Year': year,
'Prediction': prediction[0]})

    # Convert to DataFrame
    df_predictions = pd.DataFrame(predictions_data)

    # Visualization
    fig, ax = plt.subplots(figsize=(10, 5))
    for country in selected_countries:
        subset = df_predictions[df_predictions['Country'] == country]
        ax.plot(subset['Year'], subset['Prediction'], marker='o',
label=country)
    ax.set_title('Depression Prevalence Predictions')
    ax.set_xlabel('Year')
    ax.set_ylabel('Depression Prevalence (%)')
    ax.legend()
    ax.grid(True)
    st.pyplot(fig)

# Explain the output
st.markdown("""
This tool allows you to compare the predicted depression rates across
different countries over a selected range of years.
You can choose between different regression models to see how predictions
vary.
""")
```

Code Explanation:

Main Function

- Purpose: The main function serves as the entry point for the Streamlit app. This function orchestrates the entire user interface and integrates various functionalities of the application, including file uploads, data preprocessing, and navigation between different sections of the app.
- Key Components:
    - File Uploader: This component uses Streamlit's st.file_uploader to allow users to upload their data in CSV format. Once uploaded, the data is automatically read into a Pandas DataFrame.
    - Data Display: After uploading the data, users can view a snippet of the dataset directly in the interface, which helps in verifying the data and ensuring that it has been loaded correctly.
    - Navigation: The main function also sets up navigation controls, allowing users to move between different functionalities such as data preparation, EDA, and model training.

Perform EDA Function

- Purpose: The perform_eda function is designed to facilitate exploratory data analysis, providing users with insights into the dataset through visualizations. This function helps in understanding underlying patterns, identifying outliers, and exploring the distributions of various features.
- Key Components:
    - Feature Selection: Users can select one or more numerical or categorical features to analyze. Streamlit widgets, such as multiselect dropdowns, are used for this selection process.
    - Visualization: Depending on the type of data selected (numeric or categorical), different types of plots are generated. For numeric data, histograms or scatter plots can be displayed to explore distributions or relationships between variables. For categorical data, bar charts are used to visualize the frequency of categories.
    - Interactivity: The EDA function incorporates interactive elements, allowing users to dynamically change the features they are analyzing without needing to reload the page or rerun the app from the start.

Run Models Function

- Purpose: This function allows users to select, configure, and train various machine learning models. It also provides mechanisms for evaluating the performance of these models, thereby assisting users in making informed decisions about the best models for their specific datasets.
- Key Components:
    - Model Selection: Users can choose from a predefined list of models (e.g., Linear Regression, Random Forest, Ridge Regression). This selection is facilitated by a dropdown menu powered by Streamlit.
    - Model Configuration: After selecting a model, users can specify parameters and options, which can include the degree of polynomial features to be used or the complexity of the model.
    - Training and Evaluation: Once the model is configured, it is trained on the provided dataset. Key performance metrics such as Mean Squared Error, R-squared, or classification accuracy are computed to evaluate the model. This feedback is crucial for tuning the model and making adjustments to the feature selection or model parameters.
    - Visualization of Results: The function also includes capabilities to visually display the results of the model training, such as plotting actual vs. predicted values or showing error distributions, which can help in diagnosing model performance issues.

Future Improvements:

Future enhancements could include integrating more machine learning models, adding more interactive visual elements, and expanding the app's capabilities to include real-time data analysis and broader geographic and demographic data.

Conclusion:

The "Depression Prevalence Prediction Tool" significantly enhances the capability to analyse and predict mental health trends. This tool is designed to assist researchers, healthcare professionals, and policymakers in making data-driven decisions to address mental health challenges effectively. Through continuous improvements and user feedback, this application aims to remain an asset in the field of healthcare analytics.