COVID-19 US country-level time series

Problem Description:

The Covid-19 pandemic has been affecting people all over the world since early 2020. In the United States, the number of confirmed cases and deaths has been increasing rapidly, with different regions experiencing different levels of impact. Therefore, it is crucial to develop accurate prediction models to inform public health policies and resource allocation at a local level.

This project aims to predict the number of confirmed cases and deaths at the county level in the US using time series data. The dataset used for this project is provided by the John Hopkins University and includes daily county-level data on confirmed cases, deaths, and recovered cases from January 22, 2020, to present. The dataset also includes information on the population of each county, which can be used to calculate per capita rates.

Dataset Description:

The dataset used for this project contains the following features:

- **FIPS:** Federal Information Processing Standards code that uniquely identifies counties in the US
- **Admin2**: County name
- **Province_State:** State name
- **Country_Region:** Country name (always "US" for this dataset)

- **Last_Update:** Date and time in UTC when the data was last updated
- **Lat:** Latitude of the county centroid
- **Long_:** Longitude of the county centroid
- **Confirmed:** Cumulative number of confirmed cases up to the current date
- **Deaths:** Cumulative number of deaths up to the current date
- **Recovered**: Cumulative number of recovered cases up to the current date
- **Active:** Active cases (Confirmed – Deaths – Recovered)
- **Combined_Key:** County name and state name combined
- **Population:** Estimated population of the county in 2020

Approach

1. **Data Loading and Preprocessing:**

- Load the dataset from the given link
- Clean the data by removing unnecessary columns and rows, handling missing values and duplicates, and aggregating data at the county level

- Convert date columns to datetime format and set them as index
- Calculate per capita rates for confirmed cases and deaths

2. Exploratory Data Analysis:

- Visualize the data to identify trends, patterns, and outliers
- Conduct statistical tests to determine the significance of relationships between variables

3. Feature Engineering:

- Create lag features to incorporate temporal information
- Extract additional features such as day of the week, month, and season

4. Model Development:

- Split the data into training and testing sets
- Train various regression models such as Linear Regression, Random Forest, XGBoost, and LSTM
- Evaluate the models using metrics such as Mean Absolute Error, Mean Squared Error, and R-Squared

5. **Model Selection and Fine-tuning:**

- Select the best-performing model based on evaluation metrics

- Fine-tune the hyperparameters of the selected model using techniques such as Grid Search and Random Search

6. **Model Validation and Interpretation:**

- Validate the selected model using out-of-sample data and cross-validation techniques

- Interpret the model results and provide insights on the factors that influence the number of confirmed cases and deaths

Deliverables

The final deliverables for this project will include:

- Cleaned and preprocessed dataset ready for analysis

- Exploratory data analysis report with visualizations and statistical tests

- Trained regression models with evaluation metrics and interpretation of results

- Final report summarizing the project objectives, methodology, findings, limitations, and future work

Suggested Framework to Solve this problem :

1. **Problem Statement:** The goal of this project is to predict the number of confirmed cases and deaths at the county-level in the US.

2. **Dataset:** The dataset for this project is available on Kaggle and is provided by John Hopkins University. The dataset consists of multiple CSV files with the following features:

- **FIPS**: Federal Information Processing Standards code that uniquely identifies counties and county equivalents in the United States.

- **Admin2:** County name.

- **Province_State:** Province, state or dependency name.

- **Country_Region:** Country, region or sovereignty name.

- **Last_Update**: Date and time in UTC of the last update to the row.

- **Confirmed**: Confirmed cases of COVID-19.

- **Deaths:** Deaths attributable to COVID-19.

- **Recovered:** Recovered cases from COVID-19.

- **Active:** Active cases of COVID-19.

- **Combined_Key:** Combination of Admin2, Province_State, and Country_Region if the data is not at the county level.

3. **Approach**: We can approach this problem by using a time series forecasting model to predict the number of confirmed cases and deaths at the county-level in the US. The following approach can be used:

- **Data Cleaning:** The dataset needs to be cleaned and preprocessed before it can be used for modeling. This includes handling missing values, removing unnecessary columns, and converting the data into a time series format.

- **Exploratory Data Analysis (EDA):** We need to perform exploratory data analysis to understand the distribution of confirmed cases and deaths across different counties and states in the US. This will help us identify any patterns or trends in the data and also identify any outliers or anomalies.

- **Feature Engineering:** We can create additional features from the existing dataset, such as the rolling average of confirmed cases and deaths, the percentage change in cases and deaths, and the difference between the current day's cases and deaths and the previous day's cases and deaths.

- **Model Selection:** We can use different time series forecasting models such as ARIMA, SARIMA, Prophet, and LSTM to predict the number of confirmed cases and deaths. We can also use ensemble methods such as stacking and blending to improve the performance of the models.

- **Model Evaluation:** We can evaluate the performance of the models using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared value. We can also use cross-validation techniques such as Time Series Split to validate the performance of the models.

- **Model Deployment:** Once we have selected the best performing model, we can deploy the model to a production environment and use it to predict the number of confirmed cases and deaths at the county-level in the US.

4. **Deliverables:** The following deliverables can be expected from this project:

- A cleaned and preprocessed dataset for the Covid-19 US county-level time series.
- An exploratory data analysis report.
- A feature engineering report.
- A time series forecasting model that predicts the number of confirmed cases and deaths at the county-level in the US.
- A model evaluation report.
- A model deployment guide.

5. **Possible Extensions:** The following extensions can be explored to extend this project:

- Use demographic data and socioeconomic factors to improve the accuracy of the model.
- Develop a dashboard that displays the predicted number of confirmed cases and deaths for each county in the US.

- Use deep learning models such as CNNs and RNNs to perform image analysis on chest X-rays to predict the severity of Covid-19 cases.

Code Explanation :

Here is the simple explanation for the code you can find at code.py file.

Section 1: Importing Required Libraries

This section involves importing all the necessary libraries such as pandas, numpy, and matplotlib for data processing and visualization.

Section 2: Data Collection and Preprocessing

This section involves loading the dataset using pandas read_csv() function, removing unnecessary columns, renaming some columns for better readability, and handling missing values. The cleaned data is then saved in a new dataframe.

Section 3: Data Visualization

This section involves plotting different types of charts such as line charts, bar charts, and heatmap using the matplotlib library. These charts help in visualizing the trends and patterns in the data.

Section 4: Feature Engineering

This section involves creating new features such as the number of active cases and the case fatality rate. These new features are calculated using the existing columns in the dataset.

Section 5: Model Training and Evaluation

This section involves splitting the dataset into training and testing sets, defining a regression model (RandomForestRegressor), and fitting the model on the training data. The model's accuracy is evaluated using the mean absolute error and the R-squared score.

Section 6: Model Prediction

This section involves using the trained model to predict the number of confirmed cases for the next 30 days for each county. The predictions are saved in a new dataframe and plotted on a line chart to visualize the forecasted trends.

The functions used in the code include:

- **Rename_columns():** This function renames some columns in the dataset for better readability.
- **Handle_missing_values():** This function handles missing values in the dataset by either filling them with the mean of the column or dropping the rows with missing values.
- **Calculate_active_cases():** This function calculates the number of active cases for each county by subtracting the number of recovered cases and deaths from the total confirmed cases.
- **Calculate_case_fatality_rate():** This function calculates the case fatality rate for each county by dividing the number of deaths by the total confirmed cases.
- **Split_train_test_data():** This function splits the dataset into training and testing sets for model training and evaluation.

- **Train_and_evaluate_model():** This function defines a regression model (RandomForestRegressor), fits the model on the training data, and evaluates the model's accuracy using the mean absolute error and the R-squared score.

- **Make_predictions():** This function uses the trained model to predict the number of confirmed cases for the next 30 days for each county. The predictions are saved in a new dataframe and plotted on a line chart to visualize the forecasted trends.

Overall, the code aims to predict the number of confirmed cases for the next 30 days for each county in the US using machine learning regression models. The data is first preprocessed and visualized, then new features are engineered, and finally, a regression model is trained and used for predictions. The predictions can help in making informed decisions for resource allocation and public health measures.

Future Work :

1. **Data augmentation:** Currently, we are using the original images for training the model. Data augmentation techniques can be applied to generate more training images, which can help improve the model's performance.

2. **Fine-tuning of pre-trained models**: Transfer learning using pre-trained models can be explored to improve the model's performance. The pre-trained models can be fine-tuned on our dataset to learn the specific features of COVID-19 chest X-rays.

3. **Model interpretability:** Interpretability techniques such as Grad-CAM can be applied to understand which regions of the X-ray image are contributing the most to the model's prediction. This can help in identifying specific patterns that are indicative of COVID-19.

4. **Comparison with other models:** In addition to the model we have implemented, other state-of-the-art models can be explored to compare their performance on the same dataset.

5. **Real-time prediction:** The current model is trained on static images, and it would be interesting to explore real-time prediction using video streams from X-ray machines.

Step-by-Step Guide:

1. **Data augmentation:** Various data augmentation techniques such as rotation, scaling, cropping, and flipping can be applied to the training data. This can be achieved using libraries such as Keras ImageDataGenerator.

2. **Fine-tuning of pre-trained models:** A pre-trained model such as VGG-16 or ResNet can be loaded using a library such as Keras. The pre-trained model can then be fine-tuned on our dataset by freezing the initial layers and training the remaining layers using our dataset.

3. **Model interpretability:** Grad-CAM can be implemented using libraries such as keras-vis to visualize the important regions of the X-ray image.

4. **Comparison with other models:** Various state-of-the-art models can be implemented using libraries such as Keras or PyTorch. The performance of these models can be compared using metrics such as accuracy, precision, recall, and F1-score.

5. **Real-time prediction:** Real-time prediction can be achieved using OpenCV, which can capture video streams from X-ray machines. The X-ray frames can be fed into the trained model to predict the severity of COVID-19.

Exercise Questions :

1. **What is the difference between Ridge and Lasso regression, and when would you use each one for Covid-19 county-level time series analysis?**

   Answer: Ridge and Lasso regression are both used for regularization in linear regression models. The main difference between the two is the type of penalty term used. Ridge regression adds a penalty term equal to the squared magnitude of the coefficients, while Lasso regression adds a penalty term equal to the absolute value of the coefficients. Ridge regression can be useful when there are many predictors with small to moderate effect sizes, while Lasso regression can be useful when there are many predictors and some are thought to have no effect on the outcome.

2. **How would you handle missing data in the Covid-19 US county-level time series dataset?**
   Answer: There are several approaches to handling missing data, depending on the extent and nature of the missing data. One approach is to simply remove observations with missing values, but this can lead to loss of information and bias in the analysis. Another approach is to impute missing values using various methods such as mean imputation, regression imputation, or multiple imputation. However, the choice of imputation method should depend on the underlying distribution of the data and the amount of missing data.

3. **What evaluation metrics would you use to assess the performance of your Covid-19 county-level time series models?**
   Answer: There are several evaluation metrics that can be used to assess the performance of regression models, including mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), R-squared, and adjusted R-squared. These metrics provide information on the accuracy, precision, and goodness of fit of the model. Additionally, it is important to perform cross-validation to ensure the model's performance is consistent across different subsets of the data.

4. **How would you handle multicollinearity in the Covid-19 county-level time series dataset**? Answer: Multicollinearity occurs when there are high correlations between predictor variables in a regression model, which can cause instability in the estimates of

the regression coefficients. One approach to handling multicollinearity is to remove one or more of the highly correlated predictors from the model. Another approach is to use regularization methods such as Ridge or Lasso regression, which can reduce the impact of multicollinearity by shrinking the regression coefficients towards zero.

5. **How would you interpret the coefficients of the regression models for the Covid-19 county-level time series analysis?**
   Answer: The coefficients in a regression model represent the effect of each predictor variable on the outcome variable, holding all other predictors constant. A positive coefficient indicates a positive association between the predictor and the outcome, while a negative coefficient indicates a negative association. The magnitude of the coefficient represents the strength of the association, and can be used to compare the relative importance of different predictors. However, it is important to keep in mind that correlation does not imply causation, and that the interpretation of the coefficients should be done in the context of the research question and underlying causal mechanisms.

Concept Explanation :

Random Forest is a supervised learning algorithm that can be used for both classification and regression tasks. It works by constructing a multitude of decision trees at training time and outputting the class or mean prediction of the individual trees.

Now, let’s say you’re a detective and you want to catch a criminal. You have a bunch of clues, but you’re not sure which one is important. You could try to analyze each clue individually, but that would take too much time. Instead, you decide to ask a group of people for their opinions – this is essentially what Random Forest does.

In Random Forest, the “group of people” are the decision trees. Each tree is built using a subset of the data and a random selection of features. Once all the trees are built, they “vote” on the class or prediction for a given input. The majority vote is then taken as the final prediction.

To give an example, let’s say you want to predict whether a person is happy or sad based on their facial expression, tone of voice, and body language. You have a dataset with thousands of samples, each with a different combination of these features. Instead of analyzing each feature individually, you build a group of decision trees using a subset of the data and a random selection of features. Each tree decides whether the person is happy or sad based on the features it was given. Then, you let all the trees vote on the final prediction for a new person. The majority vote is taken as the final prediction – whether the person is happy or sad.

Random Forest is a popular algorithm because it is relatively easy to use, can handle large datasets with many features, and is less prone to overfitting compared to other algorithms. It’s like having a team of detectives working together to solve a case, each with their own perspective and insights.