**Predicting Disease Outbreaks Using Time Series Analysis and Classification**

**Project Overview**

This study executed a predictive modeling approach to anticipate disease outbreaks, focusing on the novel coronavirus as a primary case study. We used time-series analysis techniques such as ARIMA, Prophet, and LSTM to analyze historical datasets sourced from authoritative global health databases after addressing discrepancies through meticulous data preprocessing. The model successfully classified countries into low, moderate, and high-risk categories based on their healthcare infrastructure capacity and socio-economic metrics, leveraging machine learning algorithms like decision trees, random forests, and neural networks. This approach is adaptable to other infectious disease outbreaks, demonstrating potential as a universal tool in global health crisis management and preparedness.

**Table of Contents**

1. [Introduction](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#introduction)
2. [Methods](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#methods)
   * [Data Collection](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#data-collection)
   * [Data Preprocessing and Visualizations](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#data-preprocessing-and-visualizations)
   * [Models Used and Implementation](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#models-used-and-implementation)
3. [Results](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#results)
4. [Discussion](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#discussion)
5. [Conclusion](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#conclusion)
6. [References](https://github.com/preetham315/Data_expedition/tree/main/Disease_Outbreak%20Forecasting_Classification#references)

**Introduction**

The COVID-19 pandemic has brought unforeseen challenges to global health systems, underscoring the importance of predictive analytics in managing public health crises. Rapid and accurate forecasting of disease outbreaks is crucial for effective resource allocation, policymaking, and implementation of public health interventions. This project aims to establish a predictive model using historical data to forecast COVID-19 case trends and classify potential risk levels for different regions.

Our approach involved harnessing data from credible sources like the World Health Organization and the Centers for Disease Control and Prevention. The primary dataset focused on new COVID-19 cases, providing a temporal snapshot of the pandemic's progression across various geographies. Additional datasets detailing medical infrastructure and socio-economic metrics were incorporated to provide a comprehensive picture of each country's capacity to handle the pandemic.

**Methods**

**Data Collection**

* **Outbreak Data**: Sourced from an open-access repository hosted by Figshare, including records from the World Health Organization's Disease Outbreak News and the Coronavirus Dashboard.
* **Medical Parameter Data**: Harvested from various repositories curated by Statista, including population density, ICU availability, health expenditure as a percentage of GDP, hospital bed density, and total number of hospitals.

**Data Preprocessing and Visualizations**

* **Loading and Initial Inspection**: Loaded several CSV datasets and an Excel file, conducted preliminary inspections using .info(), .describe(), and .head() functions.
* **Data Cleaning**: Converted 'Date\_reported' field to DateTime object, addressed null values, and rectified negative values in the 'New\_cases' column.
* **Visualizations**: Generated count plots, stacked bar charts, and time series plots to visualize historical outbreaks and COVID-19 case trends.
* **Aggregation and Sorting**: Aggregated new COVID-19 cases by date for comprehensive time series forecasting.

**Models Used and Implementation**

* **ARIMA (Autoregressive Integrated Moving Average)**: Applied ADF test for stationarity, differenced and log-transformed data, and fine-tuned using auto Arima function.
* **Prophet**: Adapted dataset to fit Prophet's requirements, extended dataset for future forecasting, and calculated MAPE.
* **LSTM (Long Short-Term Memory)**: Normalized and reshaped data, constructed LSTM network with dropout layers, and evaluated using RMSE.
* **Random Forest**: Categorized risk levels based on the number of cases, trained model using medical infrastructure and socio-economic metrics, and visualized results in a heatmap.
* **Decision Tree**: Classified daily data into risk categories, visualized results in a heatmap.
* **Neural Networks**: Utilized MLPClassifier, encoded categorical target variable, trained model, and visualized results in a heatmap.

**Results**

* **ARIMA Model**: Forecast of new COVID-19 cases with new cases column, predictions versus differenced and log-transformed differenced data.
* **Prophet Model**: Forecasting of new COVID-19 cases with log-transformed data.
* **LSTM Model**: Performance on test data, comparing actual versus predicted new COVID-19 cases.
* **Random Forest**: Classification of COVID-19 risk levels by country and day.
* **Decision Tree**: Analysis of COVID-19 risk levels by country and day.
* **Neural Network**: Predictions of country-specific COVID-19 risk levels over time.

**Discussion**

* **ARIMA Model**: Reasonably captured trend and fluctuations, improved accuracy with differenced data, but less so with log-transformed data.
* **Prophet Model**: Successfully captured seasonal patterns and trends, with log-transformed differences simplifying underlying data structure.
* **LSTM Model**: Showed strong performance in capturing temporal dependencies.
* **Random Forest, Decision Tree, Neural Networks**: Provided nuanced risk assessment, visualized in heatmaps, helping guide strategic interventions.

**Conclusion**

The project conducted a comprehensive analysis using time-series forecasting and classification techniques to predict COVID-19 outbreak risk levels across various countries. Among the time-series models, LSTM emerged as the superior approach, achieving an impressive 90.12% accuracy. However, classification models indicated room for improvement. Enhancing these models by integrating more granular data, applying more sophisticated algorithms, or leveraging ensemble methods is imperative. The insights gained point towards a promising direction for developing more nuanced and actionable predictive models.

**References**

1. Alassafi, M. O., Jarrah, M., & Alotaibi, R. (2021). Time series predicting of COVID-19 based on deep learning. [Link](https://www.sciencedirect.com/science/article/pii/S0925231221015150?casa_token=ZX_ZpQ-LGVEAAAAA:nVKdGmFyShoSNJLIxML5PVtx9sCVMuRZr4zjwaFKRgstx2Wh-YN89BDCDLCl_UDXdezo7mPt)
2. Jiménez, F., Palma, J., Sánchez, G., Marín, D., Palacios, M. D. F., & López, L. (2020). Feature selection based multivariate time series forecasting: An application to antibiotic resistance outbreaks prediction. [Link](https://www.sciencedirect.com/science/article/abs/pii/S0933365719306608)
3. Satrio, C. B. A., Darmawan, W., Nadia, B. U., & Hanafiah, N. (2021). Time series analysis and forecasting of coronavirus disease in Indonesia using ARIMA model and PROPHET. [Link](https://www.sciencedirect.com/science/article/pii/S1877050921000417)
4. Thompson, R. N., & Brooks-Pollock, E. (2019). Detection, forecasting and control of infectious disease epidemics: modelling outbreaks in humans, animals and plants. [Link](https://royalsocietypublishing.org/doi/full/10.1098/rstb.2019.0038)
5. Didi, Y., Walha, A., Ben Halima, M., & Wali, A. (2022). COVID-19 Outbreak Forecasting Based on Vaccine Rates and Tweets Classification. [Link](https://downloads.hindawi.com/journals/cin/2022/4535541.pdf)
6. Desai, A. N., Kraemer, M. U. G., Bhatia, S., Cori, A., Nouvellet, P., Herringer, M., Cohn, E. L., Carrion, M., Brownstein, J. S., Madoff, L. C., & Lassmann, B. (2019). Real-time Epidemic Forecasting: Challenges and Opportunities. [Link](https://www.liebertpub.com/doi/abs/10.1089/hs.2019.0022)
7. Leopord, H., Cheruiyot, W. K., & Kimani, S. (2016). A Survey and Analysis on Classification and Regression Data Mining Techniques for Diseases Outbreak Prediction in Datasets. [Link](https://www.theijes.com/papers/v5-i9/A050901011.pdf)
8. Nsoesie, E. O., Leman, S. C., & Marathe, M. V. (2014). A Dirichlet process model for classifying and forecasting epidemic curves. [Link](https://bmcinfectdis.biomedcentral.com/articles/10.1186/1471-2334-14-12)
9. Medium article on Temperature Forecasting with ARIMA Model in Python. [Link](https://medium.com/swlh/temperature-forecasting-with-arima-model-in-python-427b2d3bcb53)
10. Figshare dataset on pandemic and epidemic-prone disease outbreaks. [Link](https://figshare.com/articles/dataset/A_global_dataset_of_pandemic-_and_epidemic-prone_disease_outbreaks/17207183/2)

**Problem 1 - Birds, heuristics, and A\***

Here our problem is sort the given 5 integers in 10 seconds. The given code works only if some of the numbers are misplaced. But when the list is in reverse order the exisiting code takes 20 seconds to run the code. So here to make this code efficient, we have make our fringe a priority queue and update our heuristic function.

In this problem, we can obtain the solution in mayy ways. But my heuristic function is based on Manhattan distance. So here, we calculate the absolute difference for current state and our goal state. Our goal state can be obtained by adding 1 to the index where the number is placed.

I make the fringe to be priority queue so that, instead of popping the intermediate states in order we can get the states which have more priority. So here distance would be our priority factor. Which ever distance comes out to be less, it would be popped out of the firnge first and next states are calculated based on that state and push all the next states to fringe.

We will continue the same procedure until the fringe is empty an we reach the goal state. Once we reach the goal state, we return all the intermidate states that we came across to attain the goal state.

**Problem 2 - The 2022 Puzzle**

The goal of the problem is to find the less number of board moves that will give us a board that is ordered.

Search Algorithm

This problem's search algorithm is an A\* search, which starts by exploring the most promising state in the fringe. The states in the fringe are stored using a data structure called a min-heap.

Initial State - 25 tiles board with initial board condition

State-Space - All board after valid moves (move entire row left/right, entire column up/down and outer and inner circle in clockwise and counter clockwise)

Successor Function - A function that after the change returns all legitimate board states.

Goal State - Ordered 2d board

For this search problem-

Cost function - cost of one board move. It accumulates cost of board move through the search.

Heuristic Function - Count of number of misplaced tiles in the board is considered as this is admissible. Preferred option for heuristic function was the sum of the number of moves of each tile to bring it to its correct place.

Total cost which is the sum of these cost functions is used as priority for the states in the fringe.

Question-1: The search tree's branching factor is 24. 5 possibilities for row right, 5 options for row left, 5 options for column up, 5 options for column down, 2 moves around the outside ring (clockwise and counterclockwise), and 2 moves around the inner ring (clockwise and counter clockwise).

Question-2: If the answer can be achieved in 7 moves, then if we use breadth-first search (BFS) instead of A\* search, then a total of 7+72+73+74+75+76+77, approx 77 states. Jotting down the thought process -

This search algorithm uses best cost function g(s) to keep track of cost from the start state to current intermediate state and heuristic cost function h(s) that calculates cost of reaching goal state from current intermediate step.

I tried to write the manhattan distance in the heuristic to find the optimal distance and used the same priority queue class in the solve function which i used in problem 1

* [Reference](https://docs.python.org/3/library/queue.html)

**Raichu**

**Problem Statement**

The Board game consists of three pieces. Pichu, Pickachu and Raichu.

Pichu: Pichu’s possible moves are that it can move diagonally one square forward only if that square is empty. Pichu can also jump over another pichu but it has to be of opposite color. The piece that was jumped over will be removed from the board.

Pikachu: Pikachu’s possible moves are that it can move 1 or 2 squares forward or left or right only if it’s a empty square. A pikachu can also jump over a pichu or a pikachu as long as it is of opposite color and it can move 2 or 3 squares in this case . Also there are some constraints to this. First of all the squares between the pikachu’s start position and jumped piece should be empty and also all the squares between the jumped piece and ending position should also be empty.

Raichu: First of all Raichu is created when a pichu or pikachu reached the opposite side of the board. Raichu’s possible moves are that it can move any number of squares which are forward, backward,left,right, and also diagonal to an empty square as long as all the squares in between are empty. A raichu can also jump over a pichu, pikachu or raichu as long as they are of opposite color and can land on any number of squares as long as the squres between the jumped square and the landing square are empty.

In this game the winner is the one who captures all of the pieces of other player. We have to write a code that plays raichu well.

**State Space**

The state space here is the set of all possible places pichu, pikachu and raichu will be on the board which includes black and white both.

**Initial State**

The initial state is the state of the board in which there are pichus and pikachus of white and black on both sides in equal numbers.

**Successor Function**

We written successor function for pichu, Pickachu and also Raichu.

Since pichu can only move diagonally we have written a successor function which returns all the possible moves. Similarly in the case of pikachu since it can only move forward, left or right we have written a successor function with those constraints and returned all the possible moves. In the case of Raichu, as it can basically move in all directions we wrote a successor function which also considers some constraints like if the square is empty or not and also since it can jump over pichu, pikachu and also raichu we considered those while building our successor function and then returned all the possible moves for raichu.

**Heuristic**

We have come up with two heauristics for this problem. Both return the counts for where as in the first we are giving the weights for the respective move based on the moves of the pices. second one is simple count. for which we were getting 1.77 as the value at the end

**Goal State**

The Goal state is a state in which a player captures all the pieces of the other player.

**Reference**

* [Ref-1](https://stackoverflow.com/questions/33644353/better-heuristic-function-for-a-game-ai-minimax)
* [Ref-2](https://github.com/njmarko/alpha-beta-pruning-minmax-checkers)

**Truth be Told**

**Problem Statement:**

Many practical problems involve classifying textual objects — documents, emails, sentences, tweets, etc. —into two specific categories —spam vs nonspam, important vs unimportant, acceptable vs inappropriate, etc. Naive Bayes classifiers are often used for such problems. They often use a bag-of-words model, which means that each object is represented as just an unordered “bag” of words, with no information about the grammatical structure or order of words in the document. Using this clasifier we need to classify reviews into faked or legitimate, for 20 hotels in Chicago from the given data sets

**Approach:**

Initially we cleaned the text in the given data set using string translate, regex functions and sets. to get only the text and remove the charecters and any other unuseful information. Then from nltk we used the corpus english words to get the stop words. We've also tried to give teh stop words manually which didn't improve the accuracy that much. Then we tried doing the lemmatization to get the base root word with out loosing any information even this didn't improve the accuracy that much. But at this point we've considered the stop words from nltk and went further.

Once the text is clear then we moved to calculating the posterior probabilities for two classes truthful and deceptive. In this process first we calculated the prior probabilies for the classes that is deceptive/total length and likewise for truthful. So step 1 is done. Now we move to step 2: where we calculated the likehood probabilities for the words given classes using the dictionaries. For example P(poor/ truthful) is calculated based on occurences of poor in truth divided by number of words in truthful here to avaoid the case of no word poor we did smoothing by adding a positive value of 1 in numerator and denomenator. and finally mutiplied this with priors of respectives class. In the latest try updated the formula by considering the prior probability in the denominator as well which helped in increasing the accuaracy of the model.

Lastly we compare this values of posterior of truth with posterior of false based on this we will be appending the label to test data.

Finally got at accuracy of 81.50%

**References**

* [REf-1](https://jaimin-ml2001.medium.com/stemming-lemmatization-stopwords-and-n-grams-in-nlp-96f8e8b6aa6f)
* [REf-2](https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/)
* [REf-3](https://medium.com/@rangavamsi5/na%C3%AFve-bayes-algorithm-implementation-from-scratch-in-python-7b2cc39268b9)
* [REf-4](https://leasetruk.medium.com/naive-bayes-classifier-with-examples-7b541f9ffedf)
* [REf-5](https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/)

**KNN**

**Approach**

I initially started with examining the data for better understanding of the train and test data that we are using then I moved to the fit where in we are assigning the the train of data and target. Then in predict I used the manhattan distance and euclidean distance to calculate the nearest k neighbors. so in predict i initially created a empty list for the predicted y. then initialised one varialble and two lists. SO here I am doing calculcating od distances and parlley appending that to the distance\_target list and then sorting based on the list 1 element and after doing that I am getting only the k nearest neighbors using the for loop after this i am getting the most frequent element and appending it to the ypred list. this will be the final list with predicted values.

So for this problem i initially used the np.sum and np.square and the rest of the np functions for calculating the eucleadian and manhattan distances so it was taking a 40 misnutes to give the output later i changed this to normal math functions which helped in reducing the time taken to run the program to 5 and then finally after changes again it came down to 2to 3 minutes.

**Final Result**

Obtained an accuracy of 100 for iris data set and 97 % for digits

**Multi Layer Perceptron**

**Appraoch**

So after a lot of reading and understanding I started with the activation functions for this program. I worte the activation functions for normal call and when the derivative is true. Then i moved to fir where in i started with initialising the weights of of the input layer and hidden layer. After that I came back the the fir and started coding the function for fit where in we will be considering the matrix created in initialise. Based on the activation function (hidden and output) we will be considering the respective functions and calculating the forward propogation once this is done then we will calculate the cross entropy loss and the gradient of this which is used in calculating the sigmoid function

I faced lot of issues while calculating the back propagation function.

Then in predict we just simply write the feed forward

**References**

* <https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/>
* <https://towardsdatascience.com/how-to-build-knn-from-scratch-in-python-5e22b8920bd2> (KNN- code ref)
* <https://blog.devgenius.io/implementing-k-nearest-neighbors-from-scratch-in-python-d5eaaf558d49> (KNN- code ref)
* <https://www.digitalocean.com/community/tutorials/k-nearest-neighbors-knn-in-python>
* <https://www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python/>
* <https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/>
* <https://www.digitalocean.com/community/tutorials/sigmoid-activation-function-python>
* <https://towardsdatascience.com/secret-sauce-behind-the-beauty-of-deep-learning-beginners-guide-to-activation-functions-a8e23a57d046#:~:text=Identity%20or%20Linear%20Activation%20Function,proportional%20to%20the%20input%20data>.
* <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>
* <https://www.kaggle.com/code/vitorgamalemos/multilayer-perceptron-from-scratch> (Code- ref\_mlp)
* <https://towardsdatascience.com/building-neural-network-from-scratch-9c88535bf8e9>
* <https://www.statology.org/one-hot-encoding-in-python/>
* <https://github.com/eriklindernoren/ML-From-Scratch/blob/a2806c6732eee8d27762edd6d864e0c179d8e9e8/mlfromscratch/supervised_learning/multilayer_perceptron.py#L27> (code- ref-mlp)
* <https://chat.openai.com/chat> (reference for mlp resources)
* <https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd>
* <https://neuralthreads.medium.com/backpropagation-made-super-easy-for-you-part-2-7b2a06f25f3c>
* <https://towardsdatascience.com/deriving-backpropagation-with-cross-entropy-loss-d24811edeaf9>

**Multi-variate Time Series Forecasting - Air Quality**

**Project Overview**

This project focuses on multi-variate time series forecasting applied to air quality prediction. Using a large-scale dataset from the UCI Machine Learning Repository, we investigated various forecasting techniques including VAR, SARIMAX, FB Prophet, LSTM (with a focus on stacked bidirectional architecture), Ensemble approaches, TCN, and GRU models. Each model was fine-tuned for optimal performance, and their accuracy was evaluated using the Root Mean Squared Error (RMSE) metric.

**Table of Contents**

1. [Introduction](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#introduction)
2. [Methods](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#methods)
   * [Data Preprocessing](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#data-preprocessing)
   * [Exploratory Data Analysis](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#exploratory-data-analysis)
   * [Modeling](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#modeling)
   * [Testing](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#testing)
   * [Evaluation](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#evaluation)
   * [Time Series Forecasting](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#time-series-forecasting)
3. [Experimental Setup](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#experimental-setup)
   * [VAR Model](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#var-model)
   * [SARIMAX](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#sarimax)
   * [FB Prophet](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#fb-prophet)
   * [LSTM Models](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#lstm-models)
   * [Ensemble Models](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#ensemble-models)
   * [Temporal Convolutional Networks (TCN)](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#temporal-convolutional-networks-tcn)
   * [Gated Recurrent Units (GRU)](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#gated-recurrent-units-gru)
4. [Results](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#results)
5. [Conclusion](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#conclusion)
   * [Key Findings](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#key-findings)
   * [Limitations](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#limitations)
   * [Future Scope](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#future-scope)
6. [References](https://github.com/preetham315/Data_expedition/blob/main/Multi_Variate_Timeseries_Forecasting/README.md#references)

**Introduction**

Air quality poses a significant risk to public health and the environment. Accurate forecasting of air quality indices (AQI) is crucial for effective environmental monitoring and pollution control. This project systematically evaluates various machine learning and deep learning models to predict air quality indices, providing insights into their predictive capabilities and real-world applications.

**Methods**

**Data Preprocessing**

* Cleaned the dataset by removing irrelevant columns and NaN values.
* Standardized numerical values and merged 'Date' and 'Time' columns.
* Imputed missing data and corrected outliers to ensure data integrity.

**Exploratory Data Analysis**

* Conducted visual analyses to compare pollution levels across different time periods.
* Analyzed relationships between temperature, humidity, and various pollutants.

**Modeling**

* Used statistical and machine learning models including VAR, SARIMAX, FB Prophet, LSTM, TCN, and GRU.
* Performed hyperparameter tuning for each model to optimize performance.
* Evaluated models using RMSE and plotted forecasting graphs.

**Testing**

* Validated models on test sets to assess their performance on new data.

**Evaluation**

* Used RMSE as the primary evaluation metric to compare model accuracy.

**Time Series Forecasting**

* Plotted forecasting graphs to visualize the predictive power of each model.

**Experimental Setup**

**VAR Model**

* Conducted stationarity tests and Granger causality tests.
* Selected the best lag order and evaluated the model using RMSE and MAE.

**SARIMAX**

* Handled seasonal effects and selected hyperparameters using ACF and PACF plots.
* Evaluated the model with RMSE and plotted 30-day forecasts.

**FB Prophet**

* Tuned hyperparameters and extended the model for multivariate analysis.
* Evaluated the model using RMSE and plotted forecasting graphs.

**LSTM Models**

* Experimented with various LSTM configurations including Vanilla, Stacked, Bidirectional, and CNN-LSTM.
* Selected the best model through hyperparameter tuning and evaluated it using MAE and forecasting graphs.

**Ensemble Models**

* Used Random Forest, Gradient Boosting, AdaBoost, Histogram GB, and XGBoost.
* Tuned hyperparameters using Randomized Search CV and evaluated models using RMSE and other metrics.

**Temporal Convolutional Networks (TCN)**

* Performed hyperparameter tuning and evaluated the model using RMSE.
* Plotted forecasting graphs to visualize predictions.

**Gated Recurrent Units (GRU)**

* Tuned hyperparameters using grid search and evaluated the model using RMSE.
* Plotted forecasting graphs to visualize predictions.

**Results**

* The Stacked Bidirectional LSTM model achieved the lowest RMSE of 0.424, indicating superior prediction accuracy.
* The SARIMAX model performed well with an RMSE of 1.4, effectively handling seasonal patterns.
* GRU and TCN models showed competitive performance with RMSEs of 7.43 and 10.47, respectively.
* Ensemble methods, particularly XGBoost, required more tuning for optimal performance.

**Conclusion**

**Key Findings**

* The Stacked Bidirectional LSTM model was the most effective for air quality forecasting.
* Hyperparameter tuning significantly improved model accuracy.
* The graphical representation of forecasts provided valuable insights into model performance.

**Limitations**

* Model performance is highly dependent on data quality and quantity.
* Computational intensity may limit the application of complex models.

**Future Scope**

* Further refine SARIMAX for better seasonal accuracy.
* Improve Prophet model by adding more datasets and adjusting hyperparameters.
* Develop adaptive GRU architectures for handling dynamic data.

**References**

1. Vito, Saverio. (2016). Air quality. UCI Machine Learning Repository. <https://doi.org/10.24432/C5060Z>
2. Beijing Multi-Site Air-Quality Data. <https://archive.ics.uci.edu/dataset/501/beijing+multi+site+air+quality+data>
3. Time series analysis and forecasting of air quality index. <https://doi.org/10.1063/5.0045753>
4. U. A. Bhatti et al., "Time Series Analysis and Forecasting of Air Pollution Particulate Matter (PM2.5): An SARIMA and Factor Analysis Approach," in IEEE Access, vol. 9, pp. 41019-41031, 2021, doi: 10.1109/ACCESS.2021.3060744. <https://ieeexplore.ieee.org/abstract/document/9359734>
5. Brian S. Freeman, Graham Taylor, Bahram Gharabaghi Jesse Thé (2018) Forecasting air quality time series using deep learning, Journal of the Air Waste Management Association, 68:8, 866-886, DOI: 10.1080/10962247.2018.1459956. <https://www.tandfonline.com/doi/full/10.1080/10962247.2018.1459956>

**Part-1 - Parts of Speech Tagging**

**Simplified Bayes Net**

**Formula**

* P(S/W)= (P(W/S)\*P(S))/P(W)

**Approach**

For this approach we are expected to find the probability of a part of speech given word. So using bayes theorem we calculate the posterior i.e P(W/S) and prior P(S). P(W) can be neglected as it will be very small when compared to the list of total words. So for calculating the posterior we have constructed a frequencies dictionary which has a word and the in the keys we will be having the dict of all the possible parts of speech. So we are going to divide this by totol frequency of that particular pos .Then this output is multiplied to the P(S) which is the ratio of occurence of a particular pos to tolat length of words in the given data.

So here we've encountered problems where in the word is not present in the test data so to handle this we have given some basic rules of grammer for the most occuring parts of speech and by default we are considering the rest to be be may be noun with a probability of 0.2.

Some of the important dicts and variables to be noted are:

* frequencies: gives us the freq of word beign that particular pos
* parts of speech : gives us the freq of ocurance of a parts of speech
* words: gives us the total number of words given in the training data set
* Obtained accuaracy is 49.28%

**Viterbi- HMM**

**Formula**

* P(S/W)= max(Transition\_probability)\* P(Si/W)--> Emission probabity

**Appraoch**

For this approach we have started initialyy by creating a viterbi table which is a dict. Then here the states are pasrts of speech and the observed values are words. So firstly, we initialised the viterbi table with all the initial values for pos. Then moving on we calculated the transition probabilities of respective states and used the max of all these values to mulitply with the emission probability. Finally storing the probability and the previous state. After doing this we iterated over the v\_table to get the best possible sequence to generated for the observed sequence.

We only considered the previous state as in HMM the crux is that we only consider just the prior state for the current state. Used most of the code that i did for in class assignment

Some important variables and dicts to be noted are:

* V\_talbe: stores the probability and the previous state
* Max\_prob : gives the final max probaility
* opt: final ouput sequence obtained that has highest probability
* Obtained accuaracy is 70.30%

**Markov Chain Monte Carlo : Gibbs Sampling**

**Approach**

In this approach firstly we tried calculating hte transition probabilities from states n to n+2 as every state that is current is dependent on the previous two states and the emission from the observed to hidden states is also dependent on the current state and the previous state. WE tried creating a dictionary that has the transition proababilities for the n, n+2 pos but after that we struggled to do the sampling and obtain the stationary distribution from where we can ignore the previous burn in values and consider the balanced probabilities for which we have high efficiency. Attempts were made to solve for the balanced proabability but the time to run ofr the least amout of smaples is also exceeding the 10 min interval mentioned and these were the final tries made to calcualte the MCMC.

**Reading Text**

**Problem Statement:**

There are many techniques for image processing which can be used to get the required characters from the image that is given. But here we try to find the characters in the image using Simple Bayes Natwork and HMMs. Here the images are noisy and our main challenge is figureout the letters from the image with as much as noise can be removed

**Approach:**

In our first approach, we have used Simple Bayes Network to find the words formt he noisy image. At first we train our algorithm such that it can easily find the given text from given images. Our training set consists of Lower case characters(26), Upper case characters(26), numbers(10) and special symbols (,.-!?"' ). Once our image is converted, it will be consisiting of '\*' and ' '. The whole letter/symbol is in these two symbols based on which we derive the character. Initially we create a dictionary for the test data with indicies of all the characters in it. Then for each character we make a count of \* and ' ' respectively. Once the dictionary is created, our next task was to extract symbols and numbers individually form the constructed dictionary.

For the ease of our code, we categorize the characters into we have 3 dictionaries(original, symbols and numbers). If we are able to find a character as a number or a symbol it will be added to the respective dictionary and removed from the original dictionary. There are still some character which are misread, so to get the right values we calculate the symbols and numbers based on threshold values

**Result:**

Given: SUPREME COURT OF THF UNITED STATES Simple: SUPREME COURT OF THF UN.TED STATES HMM: Sample simple result

**References (referred the following pages to better understand the code for viterbi algorithm and rest of the parts)**

* <https://stackoverflow.com/questions/32103458/python-viterbi-algorithm>
* <http://www.adeveloperdiary.com/data-science/machine-learning/implement-viterbi-> algorithm-in-hidden-markov-model-using-python-and-r/
* <https://www.pythonpool.com/viterbi-algorithm-python/> (code-reference)
* <https://stackoverflow.com/questions/9729968/python-implementation-of-viterbi-> algorithm
* <https://medium.com/analytics-vidhya/part-of-speech-tagging-what-when-why-and-how-9d250e634df6>
* <https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31>
* <https://medium.com/codex/a-probabilistic-approach-to-pos-tagging-hmm-a557f963e159>
* <https://etn-sas.eu/2020/09/23/part-of-speech-tagging-using-hidden-markov-models/>
* <https://www.tweag.io/blog/2019-10-25-mcmc-intro1/>
* <https://www.tweag.io/blog/2020-01-09-mcmc-intro2/>

**Problem 1 Pichu-Route**

State space is the map1.txt and map2.txt. the initial state is the location of p at the start of the given map. Goal state is the destination that p is trying to reach with shortest number of steps. The Cost function for the map1.txt is 16.

Added the function **direction** to calculate the path of traversal to reach the destination of the given map. In the current function the value of next step is stored for every move. We are storing the values of Y in move 1 and X in move 0. In Y -1 for L and +1 for R. In X -1 for U and + for D. The main function search is calling the direction fuction to get the final output. If there is no path we return -1.

Rest of the functions are given already.

**References**

* Class presentation and Lectures
* Geeks for Geeks
* few Other websites in the direction of BFS, DFS and some blind search algorithms

**Problem 2 Arrange Pichu**

State space is the given map1.txt and map2.txt. The initial state is the given map with one pichu and the goal state is to palce the pichu's such that they dont attack each other. Cost function for this is high as the number of steps required are more.

Updated the successor function with a traverse list to traverse on the matrix such that we cover all the directions left,right,up, down and diagonally. based on the K we move up or down in row, column and diagonally. based on the position of the pichu we change that flag variable and generate the new housemap and keep on appending it to the final one. If the number of pichus to be inserted is more than suffiecient then we return the false.

**References**

* Class Presentations and Lectures
* Blind Search algorithms
* <https://github.com/pjhanwar/N-Queens-with-Obstacles/blob/master/Nqueens.py>

**Top Spotify Songs Analysis**

**Introduction**

This repository contains all the resources and findings of an exploratory data analysis project on the "Top Spotify Songs" dataset, which encapsulates trending songs across over 73 countries. The project aimed to uncover patterns and gain insights into global music preferences using advanced data analytics tools and techniques.

**Dataset**

The dataset includes 127,306 entries, representing individual songs, complete with various musical attributes like danceability, energy, popularity, etc. The data was sourced from Kaggle and is updated daily to reflect the latest trends.

**Methodology**

The project follows a systematic approach starting from data procurement to analysis and visualization:

* Plan: Formulating the objective to delve into the world of global music trends.
* Obtain: Acquiring the dataset from Kaggle that reflects global music trends.
* Assure: Setting up the infrastructure on GCP by creating a storage bucket named bucket\_spotify.
* Transform: Accessing and preprocessing the data using Google Colab, cleaning and structuring for effective analysis.
* Store: Storing the preprocessed data on GCP for accessibility and further processing.
* Publish: Sharing the cleaned data and scripts in this repository for community use and further exploration.

**Preprocessing Steps**

The preprocessing includes:

* Handling missing values, particularly in the 'country' column.
* Normalizing the 'mode' attribute using z-score calculations.
* Conducting correlation analysis to explore relationships between different attributes.
* Performing a detailed analysis of the 'popularity' metric using the Interquartile Range (IQR).
* Generating box plots and histograms for all numerical attributes.
* The transformation process was meticulously documented to ensure that the data was refined and ready for deeper analysis.

**BigQuery Integration and Visualization**

The cleaned dataset was uploaded to BigQuery, where various SQL queries were executed to generate insights. Corresponding visualizations were created using Looker Studio and are presented in this repository.

**Contribution**

This project is a testament to the power of combining academic knowledge with practical application. It showcases the use of various cloud computing and big data analytics technologies and the importance of data governance in the field of data science.

Feel free to explore the dataset, replicate the analysis, or build upon it for your research or projects.

**Installation and Usage**

Please refer to the individual scripts and notebooks for specific instructions on setting up your environment and running the code.

**Contributions**

Suggestions and improvements are welcome. Please fork the repository and submit a pull request for any enhancements.

**Contact**

For any queries, please reach out to [savinn@iu.edu](mailto:savinn@iu.edu).

**Acknowledgments**

Gratitude to Kaggle for providing the dataset and to all the contributors who have made this project possible.

**Wellness Tracker**

**Project Summary**

Our project entails creating a database application for fitness tracking that will help users monitor and improve their fitness and health. Users of the website will be able to set and track fitness goals as well as gain insights into their fitness advancement.

**Project Description**

**Objectives**

Our objective is to develop a database application for fitness tracking that will enable users to quickly perform several operations on their fitness data. Users will be able to add new fitness metrics, monitor their development over time, edit already-entered data, and remove records as necessary. By utilizing the Fitbit dataset, we hope to give users accurate and thorough information about their fitness routines, empowering them to make wise decisions about their health and wellbeing. The application will also offer tips for enhancing fitness levels based on the users' data and goals.

**Usefulness**

Our fitness tracking database will be helpful for anyone interested in monitoring their fitness progress and enhancing their general health and wellness. Users of the database will be able to track their daily step total, calories burned, walking distance, and other metrics, enabling them to set and meet fitness objectives. Our database will give users precise and in-depth information about their fitness routines by utilizing data from Fitbit fitness trackers, empowering them to decide on their health and wellness routines.

Users will benefit from our database's interactive interface because it will allow them to easily view and analyze their data. Users will be able to track and visualize their fitness progress over time, giving them actionable insights and advice to improve their wellness routine.

**Dataset**

This dataset was created by participants in a distributed survey via Amazon Mechanical Turk between December 3, 2016, and December 5, 2016. Fitbit fitness trackers collect a variety of fitness metrics, including step count, calories burned, total distance walked, and many more. Thirty eligible Fitbit users agreed to the submission of their personal tracker data for these metrics and many more. This data set's primary goal is to develop a better understanding of Fitbit users' exercise routines.

Data set link: [Fitbit Fitness Tracker Data](https://www.kaggle.com/datasets/arashnic/fitbit)

We will be using dailyActivity\_merged.csv from this Kaggle Fitbit Fitness Tracker dataset.

**Conceptual Diagram/Schema**

The database is designed to capture and store a user's daily health and fitness data such as activity, sleep, and weight. The main table, dailyActivity, is split into several tables to understand the data better:

* daily\_calories: Tracks a user's daily calorie intake.
* steps: Records the total number of steps taken by a user on a given day.
* minutes: Records a user's active minutes on a given day, broken down by type (very active, fairly active, lightly active, sedentary).
* distances: Records the distance traveled by a user on a given day, broken down by type (total, tracker, logged activities, very active, moderately active, lightly active, sedentary).
* sleeping: Stores a user's sleep data such as total sleep records, total minutes asleep, and total time spent in bed.
* weight\_log: Contains data related to weight tracking, including columns for ID, Date, WeightKg, WeightPounds, Fat, BMI, and IsManualReport.

All entities have a primary key made up of the combination of Id and ActivityDate to establish relationships between them.

**Database**

**Constraints**

Several constraints exist in the database to ensure data integrity. The primary keys in the tables ensure uniqueness. Foreign key constraints reference the daily\_calories table in the steps, minutes, and distances tables to ensure consistency. All columns in the tables are set to NOT NULL.

**Views, Functions, and Procedures**

We have planned to use views and functions for tasks such as finding average steps, average distance, average sleep taken, and average calories for each person to gain more insights from the visualizations. More views and functions will be created based on the application requirements.

**Code**

The SQL file attached contains all of the codes for creating the database and building queries.

**Web App Architecture**

**Data Storage**

Currently, our data is stored stand-alone, and later we will migrate to the cloud.

**Backend Languages**

To build the backend, we will be using R and SQL, as we are developing our entire application in RShiny.

**Database Access**

We are using a Postgres database, and we are connecting that to R using the following packages:

install.packages("RPostgreSQL")

install.packages("RPostgres")

library(DBI)

db <- "postgres"

db\_host <- "localhost"

db\_port <- "5432"

db\_user <- "<your\_user>"

db\_pass <- "<your\_password>"

conn <- dbConnect(

RPostgres::Postgres(),

dbname = db,

host = db\_host,

port = db\_port,

user = db\_user,

password = db\_pass

)

dbGetQuery(conn, "SELECT \* FROM dailyCalories\_merged LIMIT 5")

Only the fitness specialist (Admin of our application) has the ability to delete records; no other users have this capability. Users can only perform create, read, and update operations on their own data.

**Front-end Layout**

For the front end, we will be using R Shiny.

**Deployment**

Our application is deployed on the shiny server.

**Interactivity**

Users can log in, view insights on their fitness goals, update and add new activity details, and register for new accounts.

**Web App Layout**

**Initial Layout**

When a user first visits our website, they will see the Login page with fields for their email address and password and a navigation menu.

**Menu Panel**

The menu panel is located in the upper right corner and includes navigation buttons such as Home, Login, Your Fitness Insights, Update Details, and Logout.

**Pages/Tabs**

We have five pages: Home, Login, Your Fitness Insights, Update Details, and Register. We will be using tabs to switch between them.

**Color Schema**

For now, we are using the default schema provided by RShiny.

**Page Content**

* **Home**: Provides basic information about the application.
* **Login**: Contains a form for email ID and password.
* **Register**: Fields for full name, email ID, password, and confirm password.
* **Your Insights**: Displays various insights on user data through visualizations.
* **Update Details**: Displays user data and allows CRUD operations.
* **Logout**: Logs the user out of the session.

**References**

* [Fitbit Fitness Tracker Data](https://www.kaggle.com/datasets/arashnic/fitbit)