12-2023

GENERAL POPULATION PROJECTION MODEL WITH CENSUS POPULATION DATA

Takenori Tsuruga

GENERAL POPULATION PROJECTION MODEL WITH CENSUS POPULATION DATA

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Takenori Tsuruga

December 2023

GENERAL POPULATION PROJECTION MODEL

WITH CENSUS POPULATION DATA

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Takenori Tsuruga

December 2023

Approved by:

Dr. Haiyan Qiao, Project Advisor, School of Computer Science and Engineering

Dr. Ronald Salloum, Project Committee Member

Dr. Yan Zhang, Project Committee Member

ABSTRACT

The US Census Bureau offers a wide range of data, and within this array, the American Community Survey 5-Year Estimate (ACS5) serves as a valuable resource for understanding the US population. This project embarks on an exploration of Machine Learning and the Software Development process with the goal of generating effective population projections from ACS5 data. The project aims to provide methods to make predictions for every city and town in the US, encompassing their total population and population divided into 5-year age groups. It's worth noting that while the generation of these projections is grounded in the generalized statistical likelihood computed by the machine learning models, there remains an expected margin of error for each prediction.

To effectively convey this margin of error alongside the series of predictions, the project leverages a technique known as conformal prediction, which delivers the error range in the form of conformalized quantile regression. The modeling process encompasses a variety of approaches, including both Statistical Machine Learning Models and Deep Learning Models. The ultimate results take the form of visualizations, comprising combined plots featuring selected statistics for specific cities and towns within the County of Riverside, serving as the test dataset. These final machine learning models successfully yield persuasive population growth projection curves.

ACKNOWLEDGEMENTS

This project was a challenging one, requiring slow and steady learning, experimentation, and perseverance. I am grateful to my project advisor, Dr. Haiyan Qiao, for patiently providing continuous support and guidance throughout the process. I would also like to thank my project committee members, Dr. Ronald Salloum and Dr. Yan Zhang, for kindly allocating their time to participate in the committee and offering valuable feedback.

Most importantly, I would like to express my sincere appreciation to my close friends and family members for their unflinching support and encouragement throughout this journey. Without their unwavering belief in me, this project would not have been possible.

DEDICATION

This work is dedicated to the people who work in the field of aging society, with the hope that the results of this project, a method to model and generate population growth projections to bring inspiration, and leads to aid in their effort in effective strategic planning for the aging society in the coming future.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

LIST OF EQUATIONS ix

CHAPTER ONE: MOTIVATION 1

Background 1

Significance of Effective Projection 2

Objective 2

CHAPTER TWO: PROJECT OVERVIEW 4

General Workflow 4

Variation in Trained Models 6

Conformalized Quantile Regression 8

CHAPTER THREE: DATA 9

Data Preparation 9

Data Collection 9

Data Analysis 10

Data Cleaning 13

Data Clustering / Dataset Splitting / Outlier Detection 14

Data Clustering 15

Outlier Detection / Data Splitting 16

t-SNE Plot 17

Data Engineering 19
Data Wrangling 19
Data Normalization 21
Data Generation 21
Data Augmentation 23
CHAPTER FOUR: MODEL 25
Statistical Machine Learning Model 25
Random Forest 26
Deep Learning Model 28
Training Methods 28
RWKV 39
CHAPTER FIVE: RESULTS 43
Visualization 43
Evaluation 48
Discussion 50
CHAPTER SIX: CONCLUSION 51
Summary 51
Challenges 52
Limitations 53
Future Work 54
APPENDIX A: SAMPLED STATISTICAL VISUALIZATIONS 55
REFERENCES 60

LIST OF TABLES

Table 1. Abstract of Subject Tables 10

Table 2. Data Collection Information 14

Table 3. Data Split Information 18

Table 4. Data Wrangling Information 20

Table 5. Random Forest Training Parameters 27

Table 6. RWKV Training Parameters 42

Table 7. Quantitative Evaluation of Models 49

LIST OF FIGURES

Figure 1. Project Workflow 4

Figure 2. World Population Projection by U.N. 6

Figure 3. US Age Group 2010-2021 11

Figure 4. US Growth Rate Distribution 12

Figure 5. Age Group Ratio Density Shift 2010-2021 12

Figure 6. Distance Measurements for Time Series 16

Figure 7. t-SNE Plot 10 Clusters 17

Figure 8. t-SNE Plot Dataset Split 18

Figure 9. Quantile Loss Plots 33

Figure 10. Training Monitoring with MLflow 39

Figure 11. RWKV Cell Architecture 41

Figure 12. Population Projections of Riverside County 43

Figure 13. Age Group Projection by Random Forest 44

Figure 14. Age Group Projection by RWKV 45

Figure 15. Total Population Quantile Projection 45

Figure 16. Group Ratio Quantile Projection 46

Figure 17. Age Group Ratio Density Distribution 47

LIST OF EQUATIONS

Equation 1. Huber Loss Equation 30
Equation 2. SSIM Equation 31
Equation 3. Quantile Loss Equation 32
Equation 4. Pseudo Huber Loss Equation 33
Equation 5. Pseudo Huber Like Quantile Loss 34
Equation 6. SSIM Components Equations 35
Equation 7. SSIM Luminance Replacement Equations 35
Equation 8. Model Training Loss 36

CHAPTER ONE

MOTIVATION

Background

This project found its inspiration in a segment of my work with the Office of Aging in the County of Riverside during my participation in the Data Science Path Fellowship at UCR in the summer of 2022. The core objective of the entire fellowship project revolved around the creation of a data visualization dashboard for the office, utilizing data not only from the office's internal service databases but also from the US Census Bureau. The choice to use Census data for visualization was made during the data collection phase of the project, and it was guided by the office's specific interests. These interests encompassed various aspects, including the total population, population divided into age groups, and data pertaining to seniors, such as poverty, disability, living arrangements, and more. As the project progressed, the importance of trend forecasting in the ratio between the working-age population and the senior-age population was underscored by the office as a means to prevent potential caregiver shortages. While trend forecasting was not initially part of the project's scope, the existing code for collecting Census data was already in place. With some time remaining before the end of the fellowship, a straightforward population growth model was trained to provide this trend forecasting. Both the model's architecture and the data used were intentionally kept simple to ensure that the challenge could be met within the given timeframe, and the results proved its effectiveness in

offering foresight into the aforementioned trends across various cities and towns in Riverside County. Conversely, due to the constraints of time, the scale of the model and the dataset were intentionally kept minimal. This left a substantial amount of room for further exploration and improvement as part of a broader data science project. Within the framework of this Master project, I thoroughly explored these potential enhancements for the population growth model, striving to maximize their development within the established limitations.

Significance of Effective Projection

The primary aim of developing this model is to establish a method for identifying potential signs of an impending caregiver shortage in the years to come within each city and town in Riverside County. This scenario is expected to emerge when the elderly population (65+) begins to surpass the working-age population (25-64). The insights generated by this model will empower the county to make informed decisions and strategically prepare for this inevitable situation. This strategic preparation encompasses various measures, including the allocation of funding for caregiver training, the construction of new caregiver facilities, and the strategic placement of such facilities based on the geographical distribution of the anticipated increase in the elderly population.

Objective

The primary objective of this project is to establish methods for a general population projection model using Census Population data. The goal is to provide

a valuable resource that can assist any interested party in their pursuit of similar general population growth projection modeling with machine learning. These methods are designed to be adaptable, even if the data differs from the Census Population data, or if the modeling task involves high-dimensional multivariate time series projections.

A supplementary objective is to enhance one's own knowledge and skills as a machine learning practitioner through the process of conducting research, performing experiments, and resolving challenges within the data science project cycles using Census Population data. It is anticipated that such projects will present numerous challenges and issues, and the experience of overcoming them will serve as a valuable asset for future endeavors.

The project originated within the context of Riverside County, thus the final results were visualized for the entire county, encompassing every city and town within it. However, it's important to note that the model itself is not restricted to this county; it is designed to be applicable to any city or town in the US with ACS5 data. Additionally, visualizations were created for selected cities and towns in Riverside County, showcasing the diversity in population data characteristics, and these visualizations have been included in the appendix for reference.

CHAPTER TWO

PROJECT OVERVIEW

General Workflow

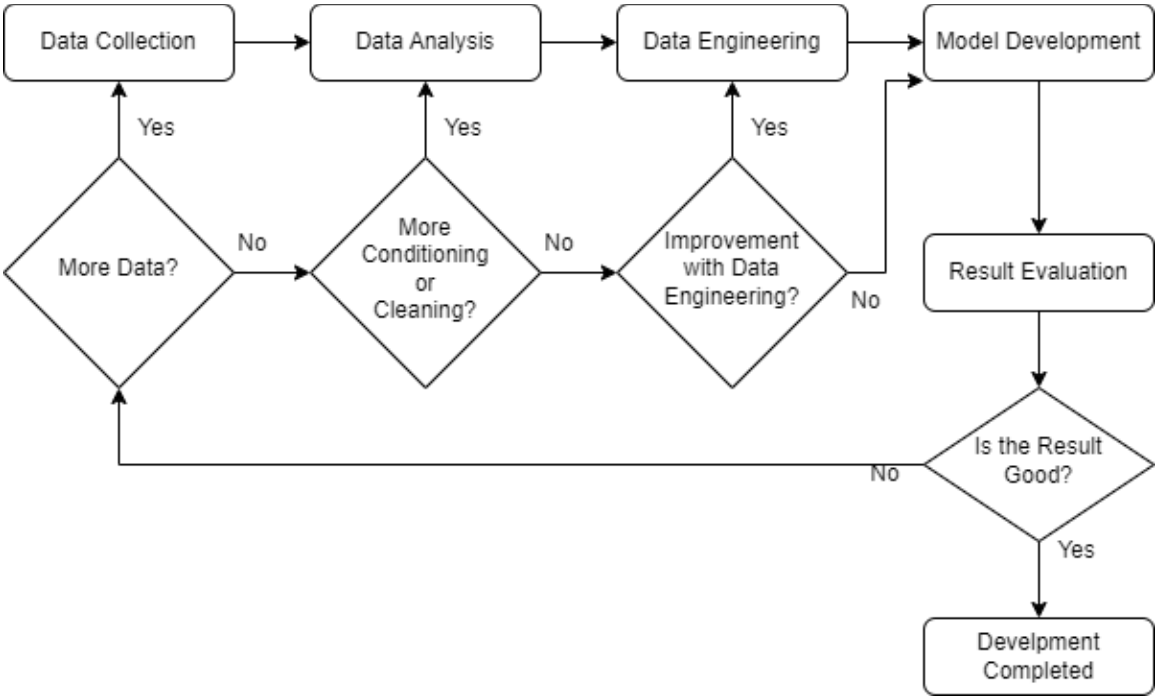


Figure 1. Project Workflow

The project workflow begins with data collection, which is done through the US Census Bureau API [1] and its wrapper library for R [2]. Next, data analysis is performed to determine the characteristics of the data. This includes data visualization, clustering, format alignment, cleaning, and normalization / standardization. After data analysis, data engineering is performed, which includes developing data generators and processing the data. Data generator development includes developing window-based, and time-series lagged data

generators for statistical machine learning and deep learning models, respectively. Data processing includes transforming the data according to model training needs.

Model development and training follows data engineering. The prepared data from the previous step is used to train various models to find the one that produces the most convincing results. With the trained model, recursive prediction is made on the data of cities/towns in Riverside County until year 2060 to produce population growth projection curves. The predicted data is then visualized to assess the resulting curves.

It is important to note that although model performance is measured with a validation set during training, a lower validation loss does not always mean that the model is capable of making a good projection curve. Therefore, the validation loss measurement is used as an indicator of model convergence, and the actual performance as a projection model is determined by the resulting projection curve.

For reference, figure 2 on next page is an example of population growth curve on world population by the UN [3]. Evaluation of the resulting curve from the train model is done based on the expectation of its continuity and smoothness like the example curve.

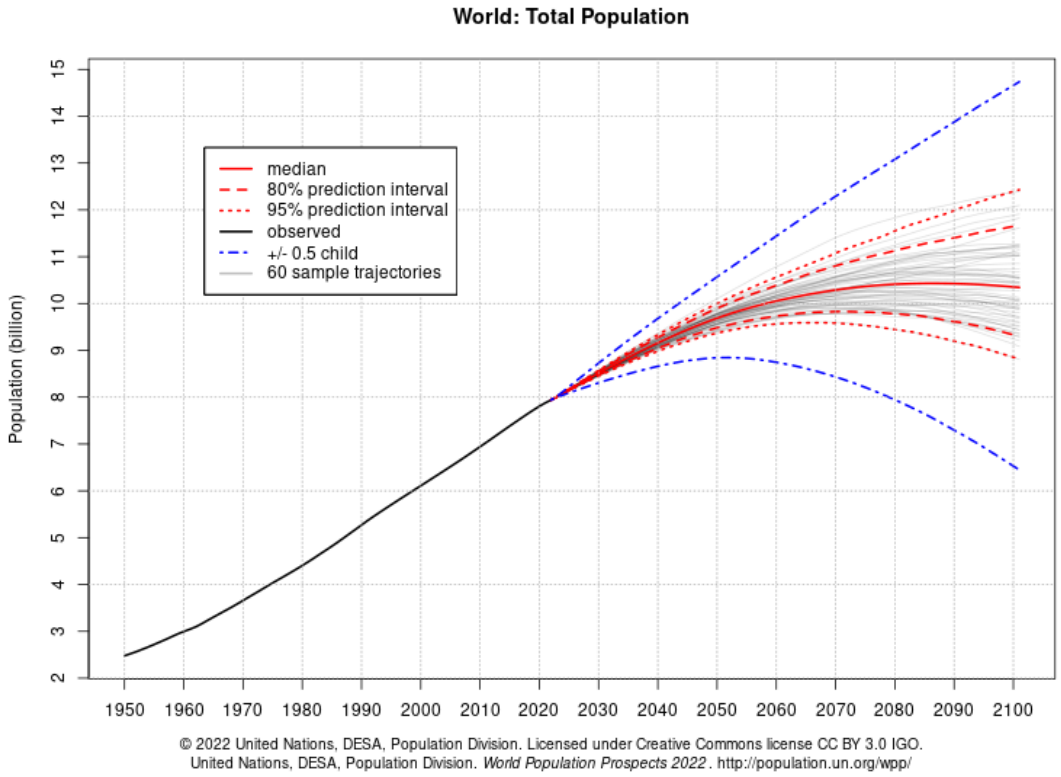


Figure 2. World Population Projection by U.N.

Variation in Trained Models

This project encompasses a diverse range of models, categorized into two main groups: statistical machine learning models and deep learning models. Statistical machine learning models rely on their underlying algorithms to learn from data and construct a model. On the other hand, deep learning models leverage artificial neural networks inspired by the structure of biological neurons. These networks consist of multiple interconnected layers of neurons that process information in a manner similar to our brains, and they, too, learn from data to establish a model.

While many types of machine learning models from both statistical and deep learning models were trained during the project experiments, best performing ones from each statistical and deep learning model were picked for this report for concise and clear representation of the result. Models chosen for the statistical machine learning model are the Random Forest model [4], and for the deep learning model is the modern Recurrent Neural Network (RNN), called Receptance Weighted Key Value (RWKV) model [5].

Data processing requirements differ between statistical machine learning models and deep learning models due to the different preprocessing needs required to train a well-fitting model. Statistical machine learning models require more data preprocessing than deep learning models. In other words, deep learning models are more capable than statistical machine learning models and therefore require less data preprocessing. However, deep learning models require a large amount of data to train effectively, and they may not be able to properly fit a model if the data is sparse in information content. While statistical models require training a model for each feature, deep learning models are trained for all the features at once as a multivariate regression model per quantile. This leads to a large number of model training requirements for statistical machine learning models (as many as the number of features), compared to only three for deep learning models (upper, middle, and lower quantile). Considering the above facts, statistical machine learning models are

trained with a selected set of minimal features, while deep learning models are trained with all features.

Conformalized Quantile Regression

Quantile regression is a technique for providing a confidence range for regression model predictions. The provided range indicates the range of error at a specified confidence level. Conformal prediction, on the other hand, is a technique for quantifying model prediction uncertainty based on past learning [6]. This conformalization technique can be applied to the quantile range by quantifying the model prediction uncertainty with a separately prepared set of data called a calibration set. The adjusted quartile prediction thus becomes conformalized. In this project, conformalized quantile prediction is used to provide an error range for each prediction. The model prediction quantiles are conformalized at an appropriate confidence level depending on each model's performance to maintain the meaningfulness of the displayed quantile range. Higher variable dimensionality of the data leads to higher sensitivity to quantile change in prediction. Therefore, the quantile range is adjusted to be narrower for models trained with higher-dimensional data to produce reasonable quantile projection.

CHAPTER THREE

DATA

Data Preparation

Data Collection

Prior to the commencement of the data collection process, an assessment of data availability and data requirements was conducted to guide data collection decisions. Data availability was evaluated by exploring data tables on the Census Data website. The Census API provides a wide variety of data in numerous reports [1]. Data requirements were determined by referencing the methodology employed in the Census population projection from 2017 [7]. According to the methodology, population data, migration data, and fertility data are the most critical components for population projections. Each data category is accessible from the American Community Survey in the form of a subject table. While there are two variations of subject tables, 1-year estimate and 5-year estimate, the 5-year estimate was selected for the project due to its anticipated higher data accuracy. Subject tables were chosen for each data category accordingly: ACS5 Subject Table S0101 for population data, ACS5 Subject Table S0701 for migration data, and ACS5 Subject Table S1301 for fertility data.

Following the selection of data, the Census API was utilized to acquire the required data. The specific implementation of the data collection code was developed in R using the “censusapi” wrapper package [2]. The specific targets of the data collection were the three subject tables for every city/place and every

year in which the ACS5 data is available in consistent data resolution (currently ranging from 2010 to 2021). While the execution of the data collection code involved a substantial number of query requests, the code was meticulously tested unit by unit to ensure its functionality. After the testing phase, the code was executed to retrieve the data from the Census API. The results of the data collection were stored in CSV format for use in the subsequent project step. The table below summarizes the information contained in each subject table from which the data was collected.

Table 1. Abstract of Subject Tables

Subject Table	Table Name	Row Binn Abstract	Column Binn Abstract
S0101	Age and Sex	Age	Sex
S0701	Geographic Mobility	Age, sex, race and ethnicity, nativity, marital status, education, income, poverty	Moved from same county, different county same state, different state, abroad
S1301	Fertility	Woman age 15 - 50 Age, race and ethnicity, nativity, education, poverty, labor force status, public assistance income	Woman with birth in past 12 months, Unmarried ratio of woman with birth in past 12 months

Data Analysis

Data analysis is a crucial step in the project, as it serves as the foundation upon which all models are built. The process commences with data visualization, which involves plotting a sample of the data to gain insights into its

characteristics. A rapid visualization of the sampled data was conducted using the R package "ggplot2" [8]. Observations of a few sampled cities and towns across sampled features revealed that most numeric data from 2021 exhibited unusual fluctuations compared to other years. Given that the COVID-19 pandemic began in 2020, it is plausible that either the collected data lacked its usual accuracy or that the data was accurate but the population was indeed disrupted due to the pandemic. Regardless of the underlying cause of the data's unusual movement, it constitutes a time series data anomaly. As the goal of model fitting is to develop a generalized model, including the data anomaly would introduce unwanted noise into the model. Consequently, the decision was made to exclude data from 2021 from the training dataset. Conversely, for the prediction phase, the data from 2021 was included in the prediction dataset, treating the data as factual. This approach introduces a challenging element into the modeling task, as the model is required to make predictions with a sequence of data that contains an anomaly in its last time step (2021 data).

By totaling the collected data, Figure 3 on the right shows the population status from 2010 to 2021 for the whole US. Each colored stacked bar indicates a different age group in the population. The "Youth" age group in yellow includes ages 0 to 24, the "Working"

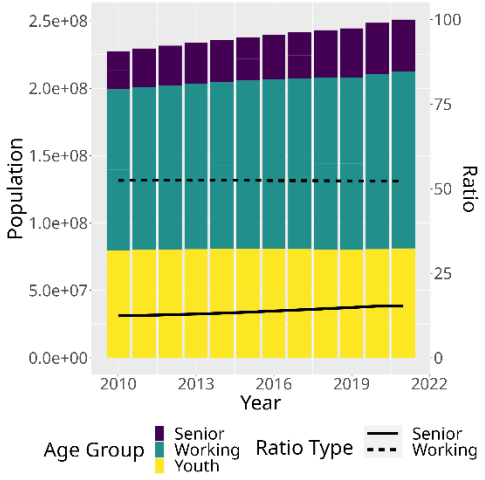


Figure 3. US Age Group Flow 2010-2021

age group in green includes ages 25 to 64, and the "Senior" age group in purple includes ages above 65. There are also two line plots on top of the bars, which indicate the percentage ratio of the "Working" age group population against the total (dotted line), and the "Senior" age group population against the total (solid line). From the plot, stagnation in the growth of the youth population, and a slight increase in both the Working age group and the Senior age group can be observed, but not significant enough to be an obvious flow of population.

Figure 4 on the right presents a density plot of the growth rate (in percentage) distribution among all US cities/towns between 2010 and 2021. The plot reveals that the majority of cities/towns experienced growth rates within a very narrow range. Notably, the positive growth rate side is more prevalent than the negative growth rate side, which aligns with the observed slight increase in population in the previous plot.

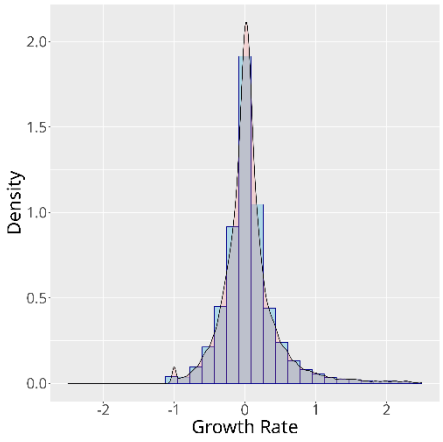


Figure 4. US Growth Rate Distribution

Figure 5 on the right illustrates the distribution of the population ratio between the Working age group and the Senior age group, comparing the years 2010 and 2021. Regions with a senior age population exceeding the

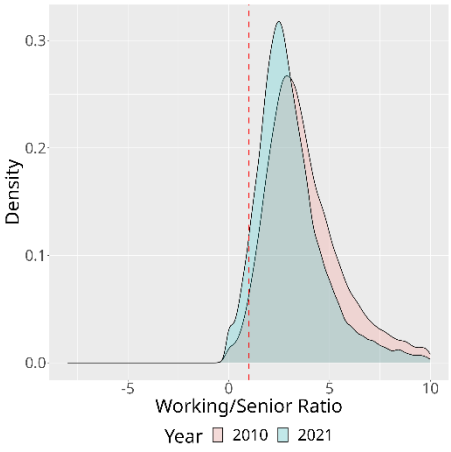


Figure 5. Age Group Ratio Density Shift 2010-2021

working age population are indicated by the area to the left of the red dotted vertical line, representing a ratio less than 1. While these regions constitute a small minority in both 2010 and 2021, a noticeable increase in their density can be observed over time. Upon comparing the overall density distributions, it is evident that the distribution is gradually shifting towards the left, indicating a nationwide trend towards a higher proportion of senior population relative to the working age population.

Based on the preceding analysis, a successful projection of the population growth curve is anticipated to exhibit a smooth and continuous trajectory with a modest increase in population and a gradual shift in the ratio between the working-age group and the senior-age group. For certain regions, it is plausible to anticipate a transition in age group dominance, with the senior-age group surpassing the working-age group.

Data Cleaning

The raw data obtained from the Census API contained values below -111111111, indicating that the corresponding cell values were not applicable due to the format of each subject table. These values, along with any "NA" (Not Available) entries, were replaced with 0. Subsequently, the mean of each feature value was calculated, and features with a mean value of 0 were eliminated from the dataset. This step aimed to remove meaningless features that would introduce noise into subsequent analyses. Additionally, certain locations with incomplete time-series data, spanning less than the entire 12-year

period, were identified as either newly incorporated or obsolete and were consequently excluded from the dataset due to their missing values.

Table 2. Data Collection Information

Category	Information
Collected Year Range	2010 - 2021
Number of Place	32375
Number of Place w/ Full Range	29180
Number of Place w/o Full Range	3195
Number of Attributes Raw	482
Number of Attributes After Cleanup	470

Data Clustering / Dataset Splitting / Outlier Detection

A crucial step in data preparation involves partitioning the available data into training and validation sets. This process typically entails dividing the data into training and validation sets in a 4:1 ratio. While data from cities/towns within Riverside County is designated for testing purposes, data from all other US cities/towns must be further divided into training and validation sets. To ensure the effectiveness of this split, similar types of samples should ideally be distributed across both the training and validation sets. To achieve this optimal distribution, data clustering techniques were employed to group data points with similar patterns.

Data Clustering

Numerous clustering techniques exist, and for this project, the time-series nature of the data necessitated careful consideration to ensure proper clustering application. The R package "dwtclust" [9] was employed for this purpose, offering a variety of clustering methods tailored to time-series data. Specifically, the "k-means" clustering method with "Soft Dynamic Time Warping" was selected for this project.

The k-means clustering algorithm is an iterative data partitioning method that aims to group similar data points into k predefined clusters. The algorithm initializes k centroids randomly, which serve as the initial cluster representatives. Subsequently, it calculates the distance between each data point and all centroids. Data points are then assigned to the cluster with the closest centroid. This process is repeated iteratively, recalculating centroids as the means of the data points within each cluster and re-assigning data points accordingly. After numerous iterations, the centroids converge to a stable state, minimizing the overall distance between data points and their respective centroids. The data points are then considered to be clustered into k distinct groups.

As shown in figure 6 [10] on the next page, Euclidean distance and dynamic time warping (DTW) are two commonly used distance calculation methods for time series data. Euclidean distance measures the point-to-point distance between two time series, while DTW takes into account the fact that time series patterns may not be perfectly aligned and finds the optimal alignment

for minimum distance calculation. Additionally, the soft dynamic time warping (SDTW) method allows for a certain amount of flexibility in the alignment, making it more robust to noise in the time series data at the cost of extra computation.

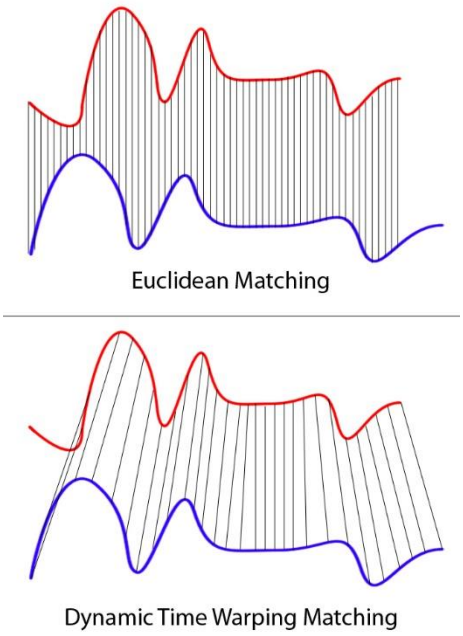


Figure 6. Distance Measurements for Time Series

Employing the methodologies and tools, data exhibiting similar time-series patterns were effectively grouped into 1000 clusters. This clustering step facilitated a more efficient data split in the subsequent stage.

Outlier Detection / Data Splitting

The clustering process yielded a small number of clusters with only one data point. These clusters were deemed outliers and excluded from the training and validation datasets due to the uniqueness of their data, which was unlikely to contribute to the development of a generalized model. For the remaining clusters, data points were divided into training and validation sets in a 4:1 ratio whenever possible. In cases where a cluster contained only two data points, each point was assigned to either the training or validation set. All clustering and splitting information was recorded in a dedicated column of the dataset to facilitate efficient querying of training and validation data.

t-SNE Plot

Dimensional reduction techniques are commonly employed to visualize the distribution of data samples in two or three dimensions. Two frequently used methods are principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE). While PCA reduces dimensionality by analyzing the linear correlation between features, t-SNE operates in high-dimensional space and projects data down to two or three dimensions, effectively capturing non-linear correlations between features. To visualize the outcome of the data clustering and dataset splitting, a t-SNE plot was employed due to its ability to identify distribution patterns in training data. The R package "Rtsne" [11] was utilized in conjunction with the soft DTW distance matrix generated during the clustering process. This distance matrix contains information about the distances between each sample, as measured by soft DWT, and has dimensions of (number of samples x number of samples).

Figure 7 employs a two-dimensional representation, with data points color-coded according to their cluster membership. While visualizing 1000 distinct clusters using different colors would result in an overly complex and difficult-to-interpret representation, the clustering was

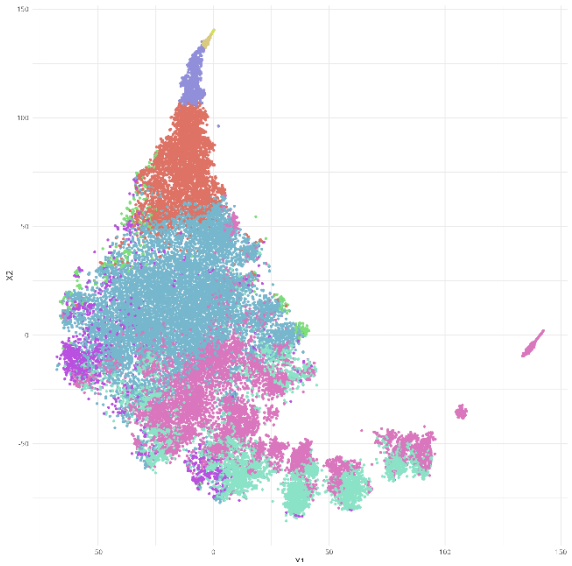


Figure 7. t-SNE Plot 10 Clusters

simplified by utilizing only 10 clusters specifically prepared for this visualization purpose.

Figure 8 utilizes the same t-SNE plot but color-codes the data points based on their split information (training set in dark blue and validation set in yellow dots). This visualization reveals that the data points of the validation set are distributed evenly across the entire dataset, confirming the effectiveness of the data splitting process.

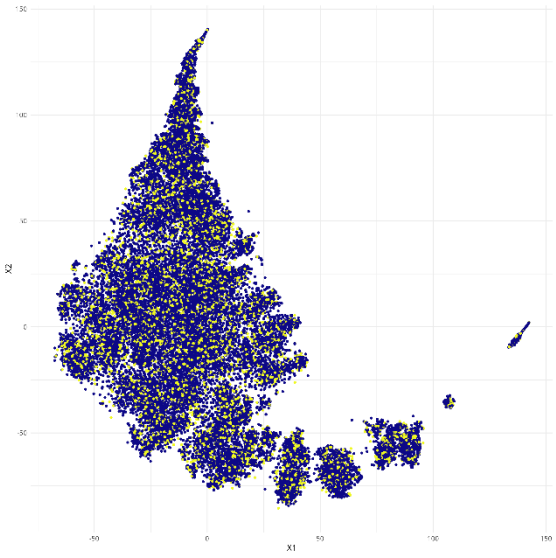


Figure 8. t-SNE Plot Dataset Split

Table 3. Data Split Information

Category	Information
Number of Place in Training Set	23696
Timeseries Sample in Training Set	71088
Number of Place in Validation Set	5472
Timeseries Sample in Validation Set	16416
Number of Outlier Place	12

Data Engineering

While an appropriate data split is essential for constructing well-generalized models and effectively validating their performance, it is equally crucial to ensure that the model is trained on well-prepared data. Training a model on poorly prepared data hinders its ability to learn the underlying patterns effectively, ultimately leading to suboptimal predictions. Therefore, proper data engineering is a cornerstone of any data science workflow. In this project, data engineering encompasses data wrangling, data normalization, data generation, and data augmentation.

Data Wrangling

Prior to applying major data transformations, it is essential to ensure data alignment. The data obtained from the Census predominantly utilizes population count units as its primary measurement. However, certain columns employ percentage number units relative to the total population, average age, or average income. To achieve unit alignment, the percentage number units were converted to population count units by multiplying the percentage value by the corresponding total population figure. Data with units of average age or income was subsequently removed, as population distribution information across various age and income ranges is already provided by other variables. Additionally, instances of redundant information in different units within the table for fertility were identified, and all associated variables were consequently removed from the dataset.

There were some variables added to the dataset which were derived through simple arithmetic operations. The geological mobility table lacked a column for the total number of in-migrations; therefore, an additional column was added by summing the number of in-migrations from each origin column. Additionally, the data included information about all increases in population directly within each category but did not provide corresponding values for decreases. While the decrease in population is typically calculated as the sum of out-migrations and end-of-life counts, it can also be estimated using the total population count, total in-migration count, and total birth count. Since the decrease in population is a crucial element for understanding population growth trends, this value was calculated for each sample and each year and added as an additional feature. As the calculation involves data from two consecutive years, the first time slice (data from 2010) lacked the additional feature and was consequently removed from the dataset. The raw calculation results are inherently negative values, and the sign of the variable was intentionally preserved to align the over/underflow of quantile calculation with other variables.

Table 4. Data Wrangling Information

Category	Information
Number of Converted Attributes	234
Number of Removed Attributes	66
Number of Added Attributes	50
Number of Attributes After Wrangling	454

Data Normalization

Following data wrangling procedures, including unit alignment and the calculation of additional features, the data underwent normalization using two distinct approaches: one tailored for statistical machine learning models and the other for neural network models. For statistical machine learning models, data was scaled solely by the standard deviation of all values within the dataset. This approach preserves information regarding relative differences between features and samples, proving to be the most effective strategy for decision tree-based machine learning models. In contrast, data normalization for neural network models employed a standardization process. This process involves calculating the mean and standard deviation of each feature, then subtracting the mean from each value and dividing it by the standard deviation. Consequently, each feature exhibits a mean of 0 and a standard deviation of 1.

Data Generation

This stage encompasses the implementation of two primary functions: a data generator and a data augmentation flow. When dealing with large datasets, it is essential to efficiently supply models with the specific data segments and formats they require. This is achieved through the implementation of a data generator. By encapsulating this functionality within a function, it becomes convenient to generate data for both model training and prediction purposes. As statistical machine learning models and neural network models necessitate

distinct data formats, separate data generator functions are implemented for each type of model.

For statistical machine learning models, each sample is expected to be a one-dimensional array as input. While time series data typically requires a two-dimensional format, with time slices as the first dimension and features as the second dimension, this format is only applicable for chronologically continuous data. As the population data from the census is not chronologically continuous, this format is not suitable. To address this, a data generator was created with the following process steps: first, lagged data columns for each feature were created for the number of lookback time slice count (6 in this case). Second, one slice of lagged data from the year ahead was created for each feature as the prediction target value (y). Finally, only the rows that correspond to the last year of time series data were selected (in this case, 2017, 2018, and 2019).

For neural network models, time series data must be formatted as a three-dimensional array. The first dimension represents the number of batched samples, the second dimension represents the number of time series slices, and the third dimension represents the number of features. Additionally, the data generator for neural network model training must be implemented to prepare the data as efficiently as possible. To fulfill these requirements, the data generator for the neural network model was implemented in two stages: creation of pre-formatted cached data and slice selection and packaging of requested data from the cached data. The initial stage generates every slice of time series data

sample in a two-dimensional array first (with time steps as the first dimension and features as the second dimension) and then caches them in a three-dimensional array (with the number of samples as the first dimension, time steps as the second dimension, and features as the third dimension). The cached data is saved in the form of a NumPy array file for quick loading in the next stage. Slice selection in the next stage of the process is handled differently depending on the data generation need. For model training, slice selection is performed randomly without replacement, while for prediction, slice selection is performed sequentially. The data generator function for the neural network model is implemented to accommodate both model training and prediction scenarios.

Data Augmentation

Despite the limited time series length of the data, data augmentation can be beneficial for enhancing model performance, particularly for statistical machine learning models. Data augmentation aims to introduce additional information about the data flow through the time steps, enabling models to better capture underlying patterns. One technique for data augmentation is moving average, which involves calculating the average value of data within a specific time window. A simple moving average is calculated by taking the plain average of the time window, while an exponential moving average places greater weight on recent time steps. In this project, exponential moving average was employed for data augmentation. The exponential moving average was calculated using a

time window of three years, and the resulting values were incorporated into the dataset as additional features.

Polynomial feature crafting is an additional technique for augmenting data when the model's learning capability is restricted to forming linear correlations between features. It involves generating polynomial features from the original features, excluding lagged data columns and exponential moving average columns. Adding polynomial features for model training significantly increases the number of features, requiring substantially more computational resources to fit a model if all added polynomial features are utilized. To mitigate these computational resource challenges, polynomial features were evaluated for each target variable, and the number of polynomial features incorporated into each model training was deterministically reduced. This polynomial feature crafting was employed solely for linear quantile regression model training and not for the Random Forest model presented here due to its inherent ability to model nonlinearity.

CHAPTER FOUR

MODEL

Statistical Machine Learning Model

Considering the availability of software for quantile regression capabilities, Quantile Linear Regression [12], Gradient Boosted Regression Tree [13], and Random Forest [4] models were evaluated for the statistical machine learning models. As previously mentioned, the Random Forest model proved to be the most effective for the given data and objective.

For the Linear Regression model, including all variations of penalized models (lasso, ridge, elastic net, adaptive lasso, scad, mcp), polynomial feature crafting was applied to the second degree. However, the results suggested that a further degree of polynomial feature crafting was necessary, which was not achievable due to computational environment limitations and memory size constraints.

Gradient Boosted Regression Tree models are generally strong performers for various tasks, but they have a performance limitation in the presence of outliers in the target variable due to their gradient-boosted training nature. While the model performed well on inlier samples like small towns to small cities, it was incapable of handling samples of medium to large cities in the data, which are statistical outliers. To compensate for this weakness, per-sample normalization and data transformations (log, power transformation) were attempted, enabling the model to make similar predictions between inlier and

outlier samples. However, the projections became linear rather than curved, leading to the conclusion that the Gradient Boosted Regression Tree model was not suitable for this case.

Due to the substantial computational resources required for training statistical machine learning models with a full set of features, the number of features was reduced to a minimum of 23 features that are most relevant for the population projection task. These minimal features include age-binned population features, female population between the ages of 15 and 50, number of births, total number of incoming migrations, and total number of decreases in population.

Random Forest

Decision tree models are statistical machine learning algorithms that learn the underlying patterns in data by constructing a tree-like structure of decisions. Each decision in the tree splits the data into smaller subsets, leading to a hierarchical representation of the data. Decision tree models are relatively easy to understand and interpret, making them popular for both classification and regression tasks.

Random forest is an ensemble machine learning method that combines a large number of decision trees to make predictions. Each tree in the forest is trained on a randomly subsampled subset of the training data, and the randomness helps prevent the model from overfitting. Random forest is known to

be a robust and effective machine learning algorithm, making it widely used for various machine learning tasks.

Due to their ability to model non-linear relationships between features, decision tree-based models, including random forest, only required exponential moving average as data augmentation. This is a significant advantage of decision tree-based models compared to other statistical machine learning models.

The random forest model fitted the data well without significant challenges and produced reasonable prediction results using a minimal set of features. As mentioned earlier in the conformalized quantile regression section, the quartile range needed to be adjusted to generate reasonable quantile projections. Different quartile ranges were evaluated to determine the appropriate range, and 40th and 60th percentiles were found to be suitable in this case.

Model Training Parameter Configurations. Here is the final training configuration of model training parameters in a table below.

Table 5. Random Forest Training Parameters

Parameter	Config
Library	quantregForest
Training Options	Software Defaults
Data Normalization	Scale by Std Whole Dataset
Predictor Variable	34 (5 ema, 6 lag)
Target Variable	1
Target Variable Lookback	6
Number of Model	23, 1 per Target Variable

Deep Learning Model

Neural network models exhibit superior capabilities compared to statistical machine learning models, thereby reducing the reliance on extensive data preprocessing. Nevertheless, meticulous data preparation remains crucial for effective model training. In this case, the prior data normalization process adequately prepared the data. Additionally, a diverse range of deep learning model architectures were explored to identify the model that generated the most optimal outcome – a smooth and continuous population growth curve.

Training Methods

The training method employed for all neural network models encompasses several critical aspects, including data pipeline optimization, selection of an appropriate loss function tailored to population projection, optimizer selection, custom training loop implementation, custom layer implementation methodology, and energy-efficient training with mixed precision.

Training Data Loading. While the aforementioned data generator for neural network training fulfills the required specifications, its integration into the training loop's data pipeline remains necessary. The neural network training for this project was implemented in Python using TensorFlow, and a data loader provided by the framework was utilized to effectively incorporate the self-coded generator function into the training loop. The TensorFlow data loader accepts a generator function as input and prefetches data from the generator function as TensorFlow tensors, storing them in memory. This prefetching ensures that the

data is already in a format compatible with the model when it is needed, eliminating any computational overhead associated with data flow from memory to the model. Additionally, the data loader supports prefetching data directly into GPU memory, eliminating the data transfer overhead between CPU and GPU. Optimizing the data pipeline is crucial for neural network training, as inefficiencies can easily become a bottleneck in the training process. Given the lengthy training times involved in neural network training, neglecting data pipeline optimization can significantly prolong training duration, resulting in wasted time and energy.

Loss Function. In previous training sessions from Fellowship project, the simple Huber loss function provided by TensorFlow was employed, which proved adequate for model conversion when the multivariate dimensionality was relatively low (26). However, for the model training in this project, the data dimensionality is significantly higher (454), rendering the simple Huber loss ineffective for model conversion. To address this issue, a custom Huber loss function was implemented using TensorFlow based on the original Huber loss formula. The original Huber loss function provided by TensorFlow calculates the Huber loss term for each feature and then reduces these values to a single mean value. In contrast, the custom-implemented Huber loss omits the final step of mean reduction, preserving the loss term for each feature. This modification enables the model learning process to effectively handle high dimensionality, leading to faster model convergence. The Huber loss function equation is provided on the next page from Wikipedia [14].

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \le \delta, \\ \delta \cdot \left(|y - f(x)| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

(1)

While the custom Huber loss function resulted in substantial improvements in model training performance, it proved insufficient for this specific application. The data structure underlying the data exhibits inherent structural alignments, such as the total population count being expected to closely approximate the sum of age-binned population counts. When models trained solely with the Huber loss are employed for projection, these anticipated structural alignments begin to disintegrate after several prediction steps, leading to feature misalignment after numerous projection steps.

To address the data structural alignment issue, an additional custom loss function was implemented based on SSIM (Structural Similarity Index Measure). SSIM is a loss function commonly used for image processing tasks, effectively measuring structural similarity between two images. While the data in this case is not an image, the population data shares a similar characteristic of high dimensionality, resembling a one-dimensional array rather than a two-dimensional array. Hence, the SSIM loss function proved to be capable of monitoring structural alignment within the data. Equation 2 on the next page presents the SSIM function equation from Wikipedia [15].

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

(2)

While TensorFlow provides an SSIM calculation function for image data, the dimensionality of projection model outputs differs from image data, necessitating manual implementation of the function. The custom SSIM loss function was implemented in TensorFlow based on the original SSIM formula presented above. The SSIM loss term was added to the Huber loss term as a small additional term during training. Consequently, the model achieved convergence with structural alignment awareness, successfully maintaining structural alignment in projections after numerous steps.

The development of the aforementioned loss functions was sufficient to fit the model for the most likely projection, the 50th percentile quantile. However, the project's objective is to generate quantile projections, requiring a loss function capable of handling different quantiles, not just the middle 50th quantile. A commonly used loss function for quantile regression neural network model training is the Pinball loss function. It calculates the absolute error for each overflow and underflow error, then applies a quantile loss term to each of the errors. For example, when measuring the 70th quantile, the overflow error is applied with a 0.7 quantile term, and the underflow error is applied with a 0.3 quantile term. Equation 3 on the next page presents the pinball (quantile) loss equation from an online article [16].

$$\ell(y, \hat{y}) = \begin{cases} \alpha \cdot (y - \hat{y}) & , \hat{y} \leq y \\ (1 - \alpha) \cdot (\hat{y} - y) & , \hat{y} > y \end{cases}$$

(3)

A robust loss function is crucial for model training to minimize the influence of outliers compared to inliers. While the data for medium to large cities statistically represent outliers in the dataset, the robustness of the loss function is essential for the model fit to generalize effectively to the population data. The Pinball loss function mentioned above is inherently a mean absolute error, which is a type of robust loss function. However, the essence of Huber loss measurement is known to be more effective for robustness, so a combination of Pinball loss and Huber loss was implemented.

The custom loss described above provides enhanced robustness to model training and was found to be sufficiently effective for model fitting for quantile regression. Even though the loss function was already capable of handling the data for quantile projection, there was still room for improvement. Taking into account the concept of function continuity, the loss function exhibits discontinuities in its gradient and Hessian. The neural network training process utilizes the gradient and Hessian of the loss function to update the model weights, and both smoothness and continuity of the curves are critical for smooth and effective model fitting. To address the discontinuity of the loss function, a pseudo-Huber loss term was employed to replace the Huber loss term. The pseudo-Huber loss term is designed to be smooth and continuous while

maintaining similar loss term values as the Huber loss term. Equation 4 presents the equation of pseudo-Huber loss from Wikipedia [14].

$$L_\delta(a) = \delta^2 \left(\sqrt{1 + (a/\delta)^2} - 1 \right).$$

(4)

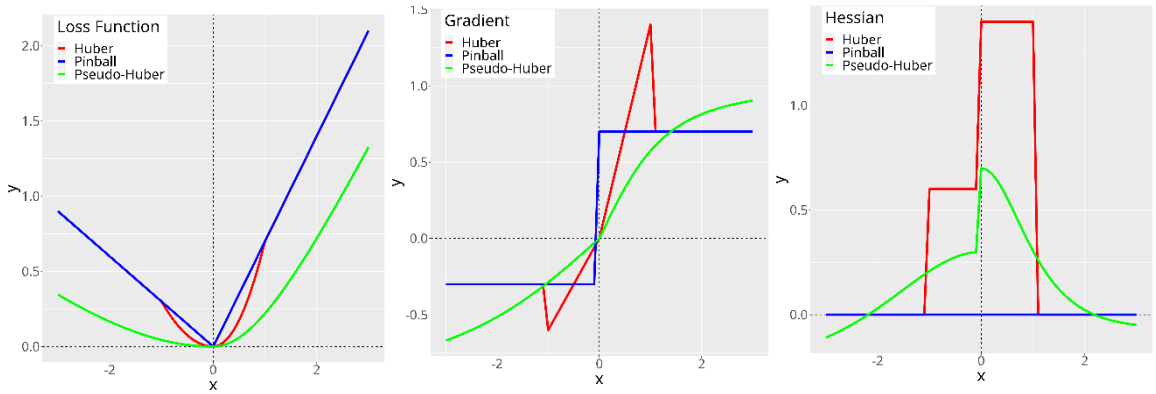


Figure 9. Quantile Loss Plots

Figure 9 illustrates the difference in the function, gradient, and Hessian curves for the 70th quantile. From left to right, the figure depicts the function plot, gradient plot, and Hessian plot. The curves are represented in the following colors: pinball (blue), Huber (red), and Pseudo Huber (green). While the Pseudo Huber function exhibits a lack of continuity at 0 in the Hessian, it demonstrates notable improvements in continuity and smoothness compared to the pinball and Huber functions. Equation 5, presented on the following page, outlines the formula for the Pseudo Huber-like quantile loss employed in this project.

$$Pseudo\ Huber\ Like\ Quantile\ Loss = phq(y, \hat{y})$$

$$= \begin{cases} \delta^2 \left(\sqrt{1 + \left(\frac{(y_i - \hat{y}_i)\alpha}{\delta} \right)^2} - 1 \right), & \hat{y}_i < y_i \\ \delta^2 \left(\sqrt{1 + \left(\frac{(y_i - \hat{y}_i)(1-\alpha)}{\delta} \right)^2} - 1 \right), & \hat{y}_i \geq y_i \end{cases}$$

(5)

The quantile measurement portion of the loss function was successfully adjusted using the method. However, the data structural similarity component of the loss function still required adaptation for quantile regression. The straightforward implementation of the SSIM loss term for a one-dimensional array, designed for a 50th percentile model, was deemed insufficient. The primary issue with directly applying the loss term to quantile regression lies in the inherent difference in mean distribution between quantile predictions and the original data. Measuring similarity between the original data and quantile predictions using the unmodified SSIM loss term would be ineffective, or even misleading, as the SSIM calculation inherently guides the mean towards the same level. To address this issue and accurately measure the mean similarity for quantile predictions within the SSIM loss term, a modification was made. According to research on the Multiscale Structural Similarity Index Measure (MS-SSIM), the original SSIM term can be decomposed into three components: luminance, contrast, and structure. Equation 6 from Wikipedia [15], presented on the following page, outlines the equations for each component and the overall

SSIM calculation. From left to right, the components are luminance, contrast, and structure, followed by the combined SSIM formula at the bottom.

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \qquad c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \qquad s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

$$\mathrm{SSIM}(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \quad (6)$$

In the context of quantile regression with a one-dimensional array, luminance corresponds to the comparison of means. Consequently, it was necessary to replace the mean comparison term with a suitable metric for quantile prediction. While the quantile measurement component of the loss function developed above can provide the error measurement from the expected quantile, all measurements are performed in absolute value terms. By modifying the quantile measurement to retain the sign of the error and incorporate the reduced mean, it can be employed as a substitute for the luminance (mean) term in the SSIM loss function.

$$Signed\ Pseudo\ Huber\ Like\ Quantile\ Loss = sphq(y,\hat{y})$$

$$= \begin{cases} \delta^2\left(\sqrt{1 + \left(\frac{(y_i - \hat{y}_i)\alpha}{\delta}\right)^2} - 1\right), & \hat{y}_i < y_i \\ -\delta^2\left(\sqrt{1 + \left(\frac{(y_i - \hat{y}_i)(1-\alpha)}{\delta}\right)^2} - 1\right), & \hat{y}_i \geq y_i \end{cases}$$

$$Replacement\ Term\ for\ Luminance = m(y,\hat{y}) = \left| \frac{1}{n}\sum_{i=1}^{n} sphq(y_i,\hat{y}_i) \right| \quad (7)$$

By integrating the previously mentioned components, the final loss function for quantile regression was formulated. It comprises a combination of the Pinball loss term, the Pseudo Huber loss term, and the SSIM loss term with a modified luminance (mean) measurement term.

$$Loss = phq(y, \hat{y}) + m(y, \hat{y}) \cdot c(y, \hat{y}) \cdot s(y, \hat{y}) \quad (8)$$

Optimizer Selection. The Adam optimizer [17], the current standard for model training, was employed in the previous project. It leverages a momentum mechanism to accelerate model training and performs well in generalizing model conversion while minimizing the impact of noise or outliers during training. On the other hand, the AdamW optimizer [18] is a variant of Adam that incorporates a corrected weight decay mechanism, providing robust regularization to counter overfitting through weight decay at the expense of some additional computation. Initially, both Adam and AdamW were utilized for model training in this project for prototyping and experimentation, and they yielded satisfactory results. While model prototyping necessitates a substantial number of experiments, any improvement in training speed is highly beneficial. Subsequently, I was introduced to the Lion optimizer [19] through a Keras software update, which claims to offer faster training speeds than Adam or AdamW. Unlike Adam and AdamW, the Lion optimizer does not employ a momentum mechanism and features a significantly simpler algorithm, yet it can achieve faster training

speeds. After some experimentation with the Lion optimizer, it was determined that it outperformed AdamW in terms of training speed in this particular instance, and it was consequently adopted for the remainder of the model training process. It is important to note that the Lion optimizer requires a three to ten times smaller learning rate compared to AdamW due to its tendency to generate larger gradients.

Custom Training Loop. TensorFlow offers a high-level API for model training, which simplifies the process considerably. Additionally, TensorFlow provides the flexibility to create custom training loops, which are particularly valuable for optimizing model training. Custom training loops grant greater control over the training process. In this project, as mentioned earlier, several custom loss terms were combined to form the final loss function for model training. By implementing appropriate code, it was possible to log all intermediate losses for monitoring purposes, which proved to be extremely beneficial for the training process. Furthermore, a training technique known as gradient clipping was employed within the training loop to stabilize the training process, which was found to be essential in this case.

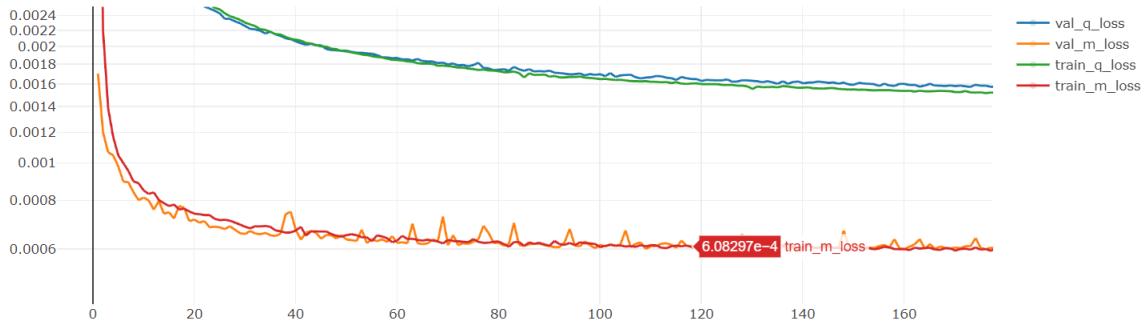
Custom Layer Implementation. Repurposing the same neural network pattern throughout a model or during rapid prototyping can be streamlined by encapsulating the pattern within a custom layer. Keras, a user-friendly wrapper layer around TensorFlow, provides a framework for creating serializable, and thus restorable, custom layers. Leveraging Keras serializable custom layers

proved highly beneficial for organizing the model code and was extensively utilized in the model training code.

Mixed Precision Training. Mixed precision training is a technique that utilizes lower precision floating-point numbers for model training computation while maintaining the model weights in higher precision floating-point numbers. For many types of tasks, employing float16 (half precision) has been found to be sufficient for model training computation and can provide faster training speeds and significantly reduced energy consumption. TensorFlow provides an API for mixed precision training, which was leveraged for rapid prototype model training. While the task at hand is a regression task, where every bit of precision is crucial, utilizing float16 mixed precision training proved sufficient for experimenting with the model prediction to explore curve convergence trends. However, to produce a smoother curve for better projection, the final fit was performed with float32 precision.

Training Data Collection / Monitoring. As the number of experiments conducted increases, managing training data becomes increasingly challenging. This highlights the importance of effectively tracking training data to optimize model training. While various tools are available for this purpose, MLflow [20] is a widely used and powerful option. MLflow provides comprehensive tracking capabilities for training data and offers insightful visualizations. In this project, MLflow was employed to monitor model training, proving to be an invaluable tool.

Figure 10 below illustrates an example of a monitoring plot depicting the training of multiple loss components.



Metric	Latest	Min	Max
val_m_loss	6.003e-4 (step=2000)	5.951e-4 (step=303)	0.002 (step=1)
val_q_loss	0.001 (step=2000)	0.001 (step=612)	0.024 (step=1)
train_q_loss	0.001 (step=2000)	0.001 (step=682)	0.149 (step=1)
train_m_loss	5.884e-4 (step=2000)	5.797e-4 (step=1242)	0.015 (step=1)

Figure 10. Training Monitoring with MLflow

RWKV

RWKV (Receptance Weighted Key Value) [5] is a type of RNN (Recurrent neural network) model emerged from the NLP (Natural language processing) field, developed to include the potential of the transformer model, effective use of attention mechanism, into Recurrent Neural Network.

RNN [21] are a type of artificial neural network specifically designed to process sequential data. They consist of interconnected neurons organized into

recurrent cells, where the output of each cell is fed back to its input. This structure enables RNNs to process sequential data by maintaining a memory of previous inputs. In time series data processing, RNNs excel at capturing not only the sequential patterns within the data but also the temporal dynamics, a unique capability that is crucial for constructing accurate population growth curves.

Transformer [22], on the other hand, are a type of neural network model that emerged from NLP research and have demonstrated exceptional capabilities in handling long sequences and high-dimensional data. Unlike traditional neural network architectures, transformers perform computations in parallel, leading to more efficient processing. Their versatility has extended their applications beyond NLP to fields such as computer vision, where they have consistently outperformed conventional neural network architectures. When applied to time series forecasting, transformers effectively capture data patterns but struggle to capture the temporal dynamics. Therefore, they are often combined with RNNs in ensemble models to compensate for this limitation. However, their parallel computation approach involves brute-force calculations, resulting in significant computational demands in terms of both processing time and memory usage.

RWKV effectively combines the strengths of both RNNs and transformers. It utilizes the transformer's attention mechanism within RNN cell structures while maintaining recurrent computation, resulting in computationally efficient processing and the ability to capture the temporal dynamics of data. By employing a modified attention mechanism within the RNN cell structure, RWKV

addresses the computational efficiency limitations of the transformer model. The RWKV cell comprises two main components: a time mixing block and a channel mixing block. The time mixing block receives input data along with recurrently passed data from the previous time mixing block. These two data streams are combined using an attention mechanism that involves multiplication of key and value vectors. The channel mixing block receives the output of the time mixing block and recurrently passed data from the previous channel mixing block. These two data streams are combined using a feed-forward network.

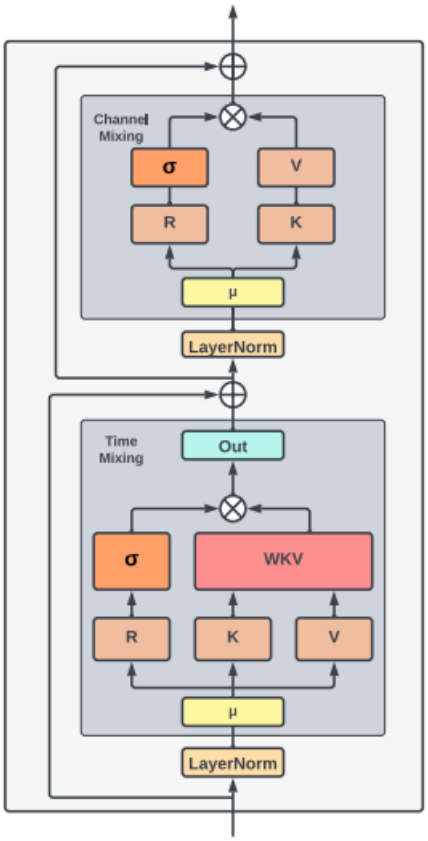


Figure 11. RWKV Cell Architecture

Structurally, RWKV cells resemble transformer blocks. However, the computations are performed in a recurrent manner, making them computationally efficient and capable of capturing the temporal dynamics of data.

Leveraging the information provided in the published paper, a straightforward RWKV RNN cell was implemented in TensorFlow, adhering to the custom RNN custom cell implementation specifications. This allows for seamless integration of the RWKV cell into TensorFlow RNN models. Consistent with the placement strategy employed in Transformer blocks, dropout layers were

introduced following each mixing block, applied to the input of the residual addition operation. The model architecture comprises three primary components: a linear transformation feed-forward network, an RWKV layer, and an output dense layer. The trained model achieved remarkably low validation loss and demonstrated its effectiveness in generating smooth population growth curves.

Model Training Parameter Configurations. Here is the final training configuration of model training parameters in a table below.

Table 6. RWKV Training Parameters

Parameter	Config
Feed Forward Layer	3 Layers
Feed Forward Units	512, 768, 1024 Units
RWKV Width	1024 Units
RWKV Attention Width	1024 Units
RWKV Feed Forward Width	4096 Units
RWKV Feed Forward Activation	tanh
RWKV Dropout Rate	0.8
Optimizer	Lion
Weight Decay	0.0
Learning Rate	1e-6
Learning Rate Scheduler	Exponential decay 0.99
Batch Size	64
Epochs	1500
Delta for Pseudo Huber	1.0
Gradient Clipping	Global Norm at 1.0
Data Normalization	Mean at 0, Std at 1, Per Variable
Predictor Variable	454
Target Variable	454
Time Series Lookback	6
Number of Model	3, 1 per Quantile

CHAPTER FIVE

RESULTS

Visualization

After each model training experiment, the models were evaluated by making predictions repeatedly to form a projection curve from 2022 to 2060. This visualization generating process is different for each model depending on its data conditioning and the library used for model training. The time it takes for the whole process differs between statistical models and neural network models. Since statistical models have a model per feature, they naturally take more time for the prediction process. R scripting was utilized for the task for all the models, including the neural network model. The following are the visualizations of raw projection data for the whole Riverside County from each model: Random Forest on the left, RWKV on the right.

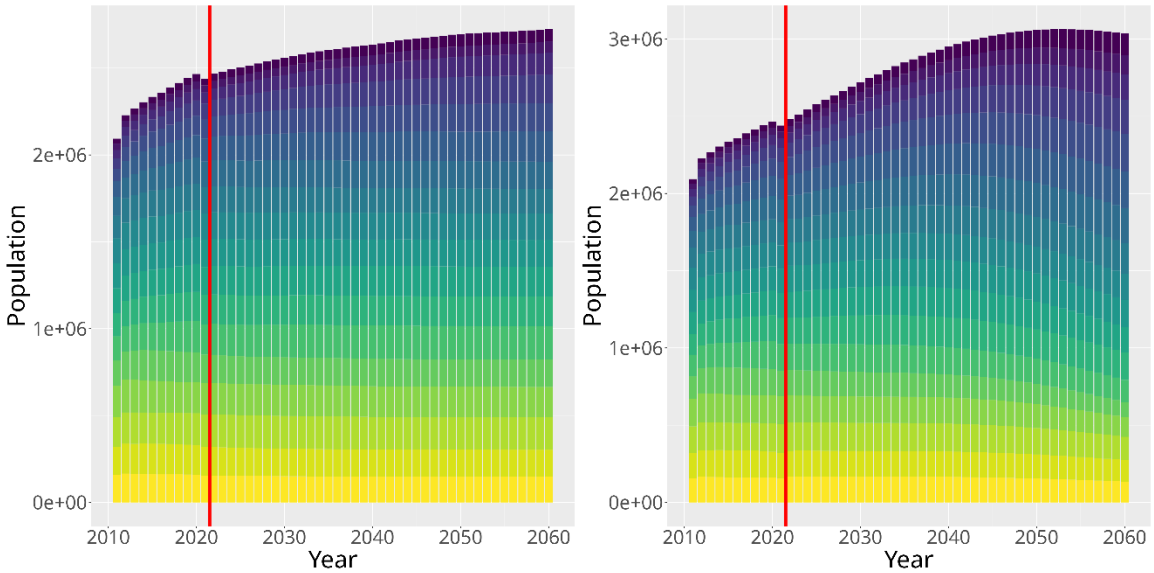


Figure 12. Population Projections of Riverside County

The red vertical line indicates the start of the projection by model prediction. A similar visualization was utilized for the evaluation of model fitting and performance to check on the smoothness and continuity expected for projection curves. Projections generated by each model show different but aesthetically pleasing curves as population growth projections. For more detailed statistics along with the use of quantile prediction, four more plots are generated for better visualization of results for the whole county of Riverside as a whole.

Figure 13 on right shows the age group population projection by Random Forest. The plot implies steady flat growth for youth and working age groups, and slight growth for senior group. The same can be confirmed from line plots which show age group ratio of working and senior age groups. The ratio stays stationary for working age group, and senior age group ratio keeps on growing at constant pace.

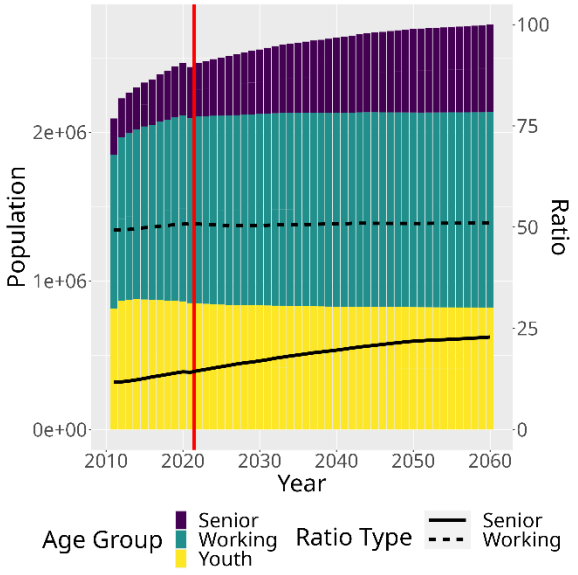


Figure 13. Age Group Projection by Random Forest

Figure 14 on the next page shows the same statistical plot of age groups from a projection generated by the RWKV model. Compared to the one from the Random Forest model, it shows a better capturing of population shifting flow among age groups, creating a smooth curvature on the projection. The slight

decaying of the youth age group population and a slightly higher rate of senior age group population growth are significant differences from the Random Forest model, which are noteworthy. For the age group ratio, both models show a similar trend for the working age group ratio. On the other hand, the senior group ratio growth is projected to be higher in this plot from the RWKV model.

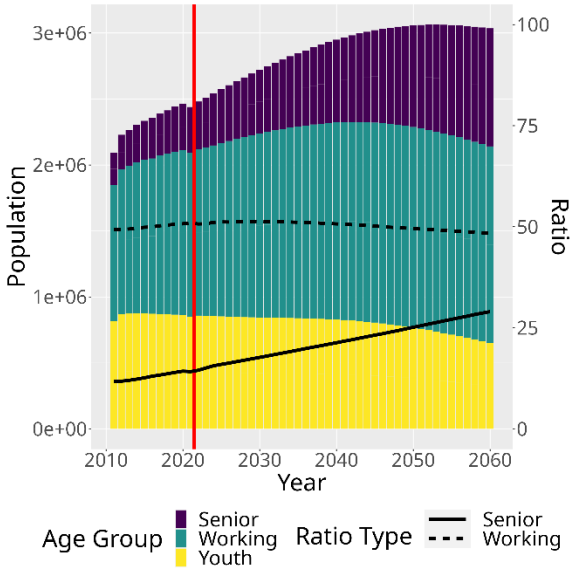


Figure 14. Age Group Projection by RWKV

Figure 15 presents a comparison of total population projections between the models along with quantile ranges. This quantile projection represents the entire Riverside County total population. As mentioned in earlier sections, the sensitivity of quantile regression increases significantly with the number of multivariate dimensions.

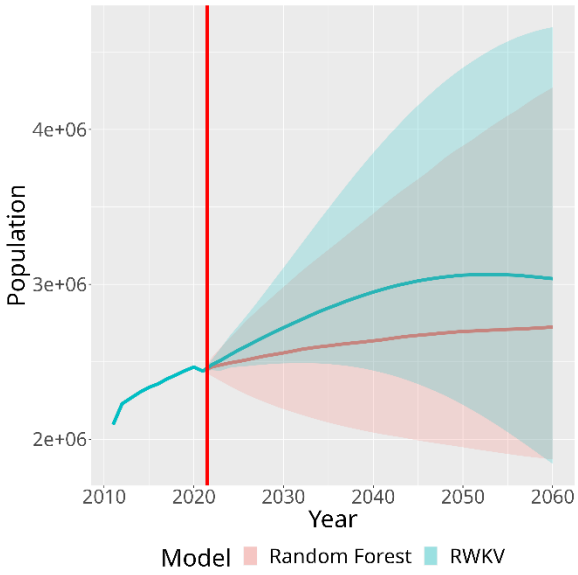


Figure 15. Total Population Quantile Projection

Accordingly, the 60th and 40th percentiles were chosen as the upper and lower

quantiles for Random Forest, while the 51st and 49th percentiles were chosen for RWKV. Given that Random Forest has 23 variables to recursively predict compared to RWKV's 454 variables, the difference in required quantile percentiles for reasonable projection generation is understandable. While both models exhibit a large area of overlap in quantile coverage, the general projection angles of the mid-quantiles (solid lines) differ significantly. The RWKV model projects a higher rate of total population growth, with a growth peak around 2050-2055. In contrast, the Random Forest model indicates a slower, more constant population growth throughout the projected time range.t likelihood of population projection course.

Figure 16 presents another critical aspect of the projection analysis: the ratio between the working-age and senior-age populations. The red lines represent Random Forest projections, while the green lines represent RWKV projections. For each model, the long-dashed line indicates the upper quantile, and the dot-dash line indicates the lower quantile. Unlike the previous quantile projection, the quantile range is intentionally not filled with a ribbon plot to differentiate between the

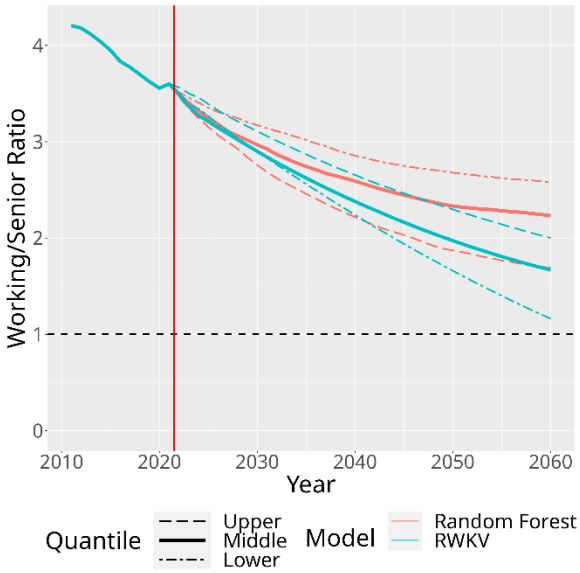


Figure 16. Group Ratio Quantile Projection

models. This quantile projection of the age group ratio reveals that each quantile projects a different scenario, rather than simply representing the upper and lower bounds. Since the projection of total population is the result of the combined upward or downward trends of all the variables, the ratio flow between quantiles can exhibit non-fixed trends. The trends of the upper and lower quantiles differ between Random Forest and RWKV, indicating a fundamental difference in the underlying projection behavior. Additionally, RWKV projects the ratio to be lower in the long run compared to Random Forest. It is important to note that the projection numbers for the entire Riverside County are cumulative results of all the projections made for every city and town within the county.

Finally, Figure 17 presents the age group ratio density distribution among cities and towns in Riverside County, comparing the current distribution in 2021 (red), the Random Forest mid-quantile projection for 2060 (green), and the RWKV mid-quantile projection for 2060 (blue). A red dashed vertical line is drawn at a ratio of 1, indicating the threshold where the senior-age group population outnumbers the working-age group. Both models project a trend of the entire density distribution shifting to the left, implying a

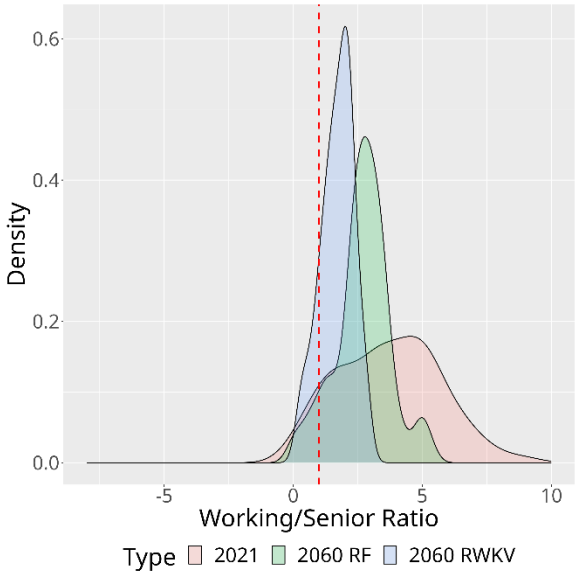


Figure 17. Age Group Ratio Density Distribution

decrease in the ratio expected at most locations. The distribution below the ratio of 1 (left side of the red dashed line) shows a similar pattern between 2021 and the Random Forest projection for 2060, indicating almost no change in the number of places with senior-age group dominance. On the other hand, the RWKV projection shows a notable increase in the number of such places. The projected shifts in age group ratio density distribution from both models are significant, highlighting the importance of preparing society for these changes.

Statistical visualizations of projections for selected locations were also generated and are included in the appendix for review. The selection of locations was considered to ensure a diverse representation of characteristics, including current total population and age group distribution. The selected locations are as follows: Riverside, Blythe, Hemet, Indio, Jurupa Valley, Lake Elsinore, Menifee, and Palm Springs.

Evaluation

In addition to qualitative evaluation through observation of the resulting projection curves, a quantitative evaluation of each model's performance was conducted using both mean absolute error (MAE) and mean absolute percentage error (MAPE) for all age-binned population variables. The evaluation was based on a test dataset comprising population data for Riverside County from 2018 to 2020. It is important to note that the data for 2021 was excluded from the evaluation due to the unpredictable time series anomaly observed in that year, likely attributed to the global pandemic and not fully captured by the utilized data.

While MAE can produce large spikes in measurements for predictions on places with small populations due to the division by small numbers, the calculated MAPE values were rounded by eliminating values exceeding the 95th percentile of the entire prediction error for each model prediction. Table 3 on the next page presents the quantitative evaluation of each model. Each row indicates an age-bin population variable, the 'Mean' row represents the mean of all age groups, and the 'Total' row shows the results for total population prediction.

Table 7. Quantitative Evaluation of Models

Age Bin	MAE		MAPE	
	RF	RWKV	RF	RWKV
0-4	117.76	180.28	14.47	22.14
5-9	146.41	164.6	13.54	18.66
10-14	136.74	192.55	11.04	17.86
15-19	129.74	154.76	11.71	18.56
20-24	115.06	169.26	13.04	22.2
25-29	145.44	237.02	12.07	23.68
30-34	132.7	176.6	14.61	23.36
35-39	124.84	202.67	13.21	21.67
40-44	103.15	170.8	10.47	20.8
45-49	103.1	147.1	11.01	18.68
50-54	100.21	138.03	11.69	17.75
55-59	106.15	165.43	11.16	19.39
60-64	107.76	134.79	10.71	19.04
65-69	83.71	118.9	10.61	17.63
70-74	70.77	90.79	12.44	19.1
75-79	61.37	82.56	13.34	20.15
80-84	47.95	57.58	15.11	28.48
85+	48.03	66.94	16.03	39.26
Mean	104.49	147.26	12.57	21.58
Total (0+)	433.61	593.44	4.61	5.2

Based on these quantitative measures shown in Table 7, Random Forest consistently outperforms RWKV, indicating that smoothness in the continuity of the projected curve does not necessarily translate into better performance in terms of the quantitative metrics used for evaluation in this study.

Discussion

Qualitative evaluation through statistical visualization suggests that RWKV generates smoother projection curves with a better sense of sequential continuity compared to Random Forest. However, quantitative evaluation indicates that Random Forest outperforms RWKV by a significant margin, raising the question of which model is actually better or more accurate. This question becomes challenging to answer when considering the inherent nature of machine learning projection models, which make predictions recursively based on statistical patterns observed in the training data. Instead of judging the correctness of model predictions, assessing the validity of the model fit to the data might be a more appropriate approach in this context. Upon closer examination of the projections generated by both models and their respective error statistics, it becomes evident that while they differ in their predictions, both models can be considered valid fits to the data. Given this inherent variability in the validity of model fits for projection models, it seems crucial to develop multiple valid fits in different ways to identify commonalities among them.

CHAPTER SIX

CONCLUSION

Summary

This project aimed to develop methods for training machine learning models using population data from the Census American Community Survey to generate population growth projections. Effective projections are expected to aid in city and town planning efforts to address the challenges of an aging society. Two types of machine learning models were trained: a statistical machine learning model and a deep learning model. The Random Forest statistical machine learning model was trained using a minimal set of 23 features along with data augmentation techniques, exponential moving average. This resulted in a model capable of making numerically accurate predictions with smooth population growth curves. The RWKV deep learning model, trained using all 454 features, produced very smooth population growth projection curves with a strong sense of time series continuity. Both models were trained using quantile regression techniques to generate reasonable quantile ranges: 40th to 60th percentiles for Random Forest and 49th to 51st percentiles for RWKV. The resulting quantile-based projections were used to generate statistical visualizations to better reveal the trends of each population growth projection. After undergoing both qualitative and quantitative evaluation processes, both models were found to generate projections with distinct characteristics, yet both were considered valid models fit for the task of population projection.

Challenges

There have been countless number of challenges encountered throughout this project, which can be separated into 2 different aspects: Continuous learning and experimentation, and creative thinking.

Continuous learning and experimentation. Addressing the scale and complexity of this project required a continuous learning process that involved acquiring a vast body of knowledge from diverse sources. The learning materials encompassed a wide range of topics, including machine learning theory like attention mechanisms, data engineering techniques like data transformation and augmentation, software engineering techniques like custom layer implementation and efficient data pipeline implementation, relevant research papers on robust loss functions and RWKV, and software tools like MLflow and TensorFlow. After each learning phase, it was essential to apply this newly acquired knowledge to evaluate its applicability to the project. This iterative process continued throughout the project's development, culminating in its successful completion. Effective learning often involves periods of stagnation or even setbacks, making maintaining motivation a significant challenge. To overcome these obstacles, cultivating extreme patience was essential to foster resilience and ensure the continuity of the learning process.

Creative Thinking. When confronting challenges with no clear solutions, creative thinking becomes essential to devise innovative approaches. Such a situation arose during the development of the multivariate quantile loss function

employed for RWKV model training. Two aspects of this development demanded creative thinking: formulating a method to combine the essence of quantile measurement with a robust loss function and devising a replacement for the luminance/mean measurement in the SSIM loss term. While experienced data scientists or mathematicians might find these tasks straightforward, I encountered significant challenges during the development process. To achieve breakthroughs in such situations, it's crucial to think outside the box.

Limitations

This project faced two primary resource limitations: time and computational environment. As with any project, time constraints restrict the scope of research exploration. Due to the iterative nature of machine learning experimentation, each iteration consumes considerable time, limiting the number of iterations possible within the project's timeframe. To effectively manage this time constraint, careful scheduling of experiments with adequate time estimations was found to be crucial. Additionally, minimizing the number of experiments and focusing on their quality proved to be essential. Another limitation arose from the computational environment. The project was conducted using a personal computer with a single GPU, which restricted the rapid prototyping process. Memory size limitations affected the ability to systematically explore high-degree polynomial feature engineering, hindering the exploration of statistical machine learning model training. Furthermore, limitations in CPU computational power

impacted the ability to explore high-dimensional data with statistical machine learning models.

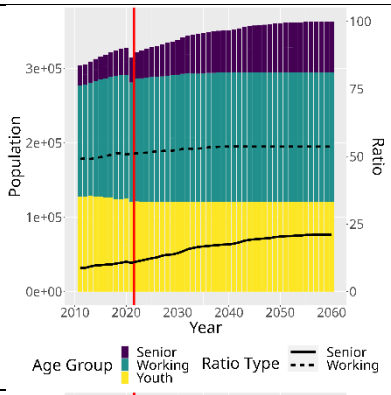
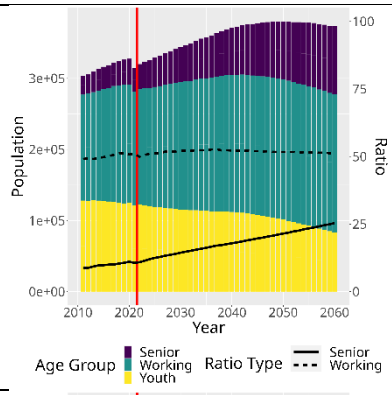
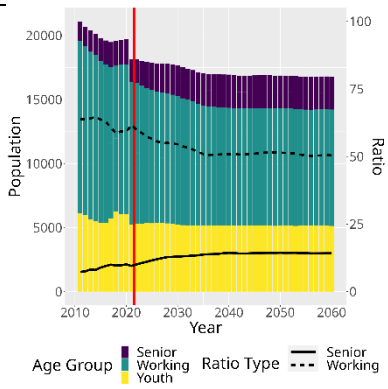
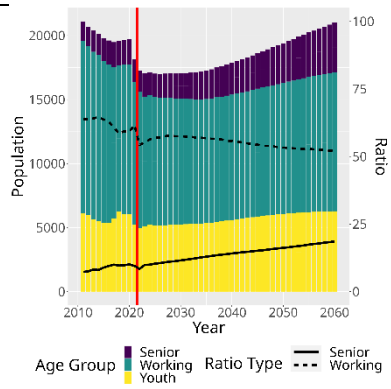
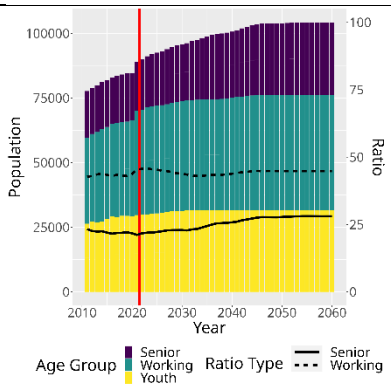
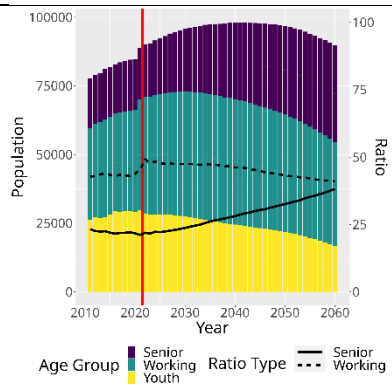
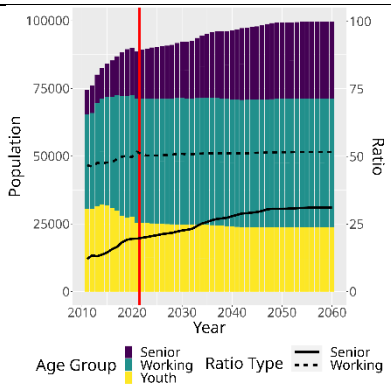
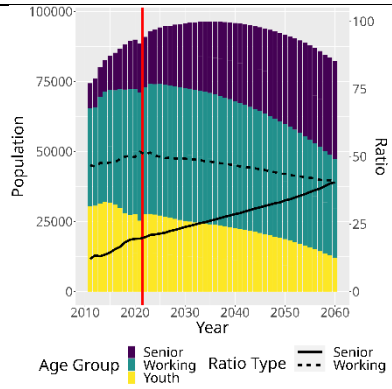
Future Work

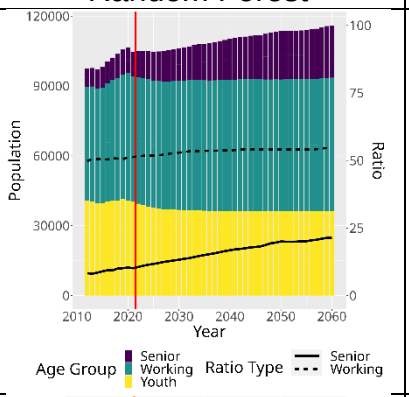
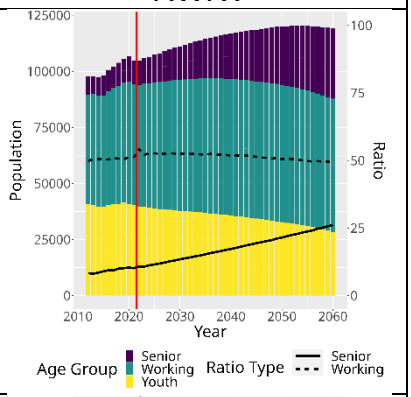
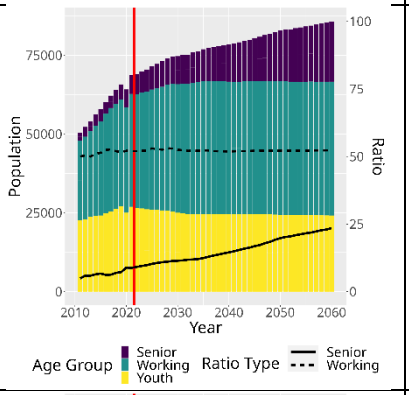
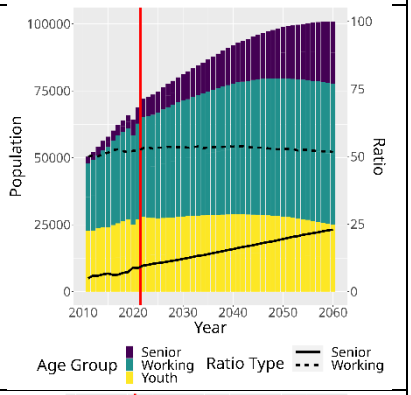
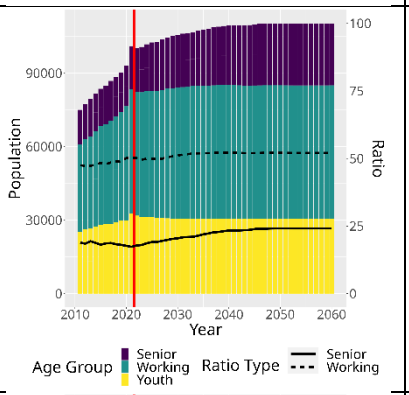
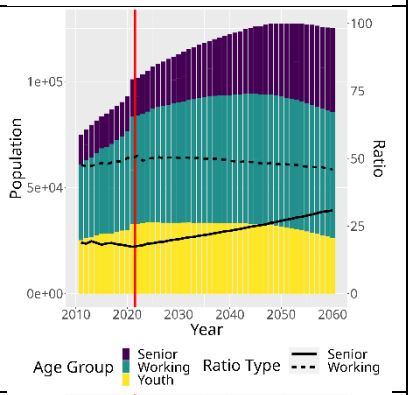
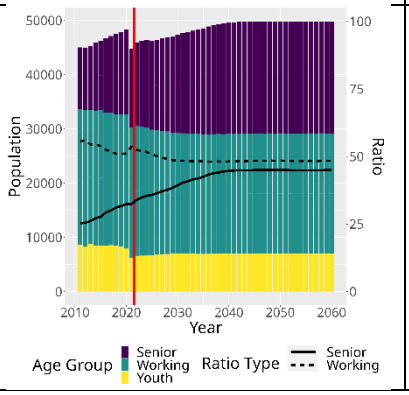
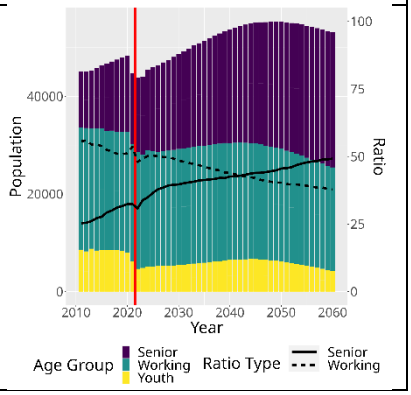
Scaling appears to be the natural next step for this project, with two primary aspects to consider: data scaling and model scaling. While the project utilized high-dimensional data, it was extracted from a limited selection of subject tables from ACS5. Additional subject tables are available, potentially providing more data dimensions. Although the impact of increased data dimensionality on the general population growth projection modeling task is uncertain, it is worth exploring. On the model side, while the RWKV model was trained with a single RWKV layer, scaling by stacking RWKV layers would be a worthwhile experiment. Moreover, during the experiments, it was observed that changing the feed-forward activation function in the RWKV layer altered the characteristics of the model's predictions. Therefore, training multiple models with varying activation functions to form a mixture of experts model [23] would be another experimentally appealing model scaling approach.

APPENDIX A

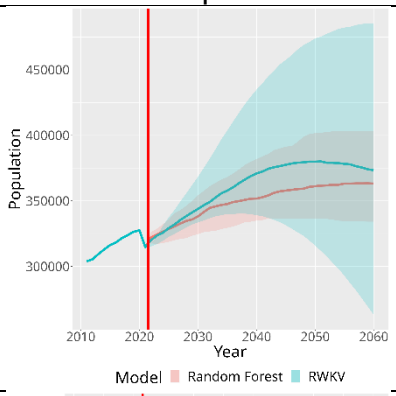
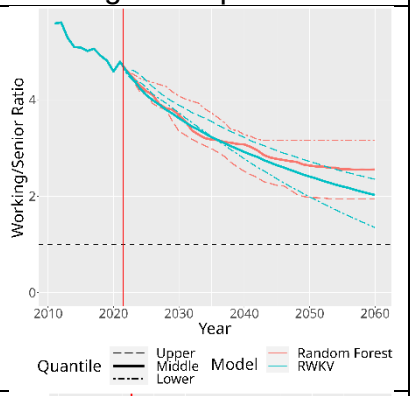
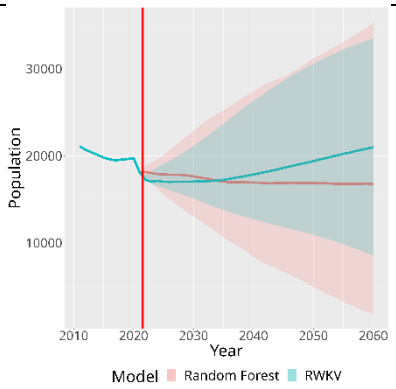
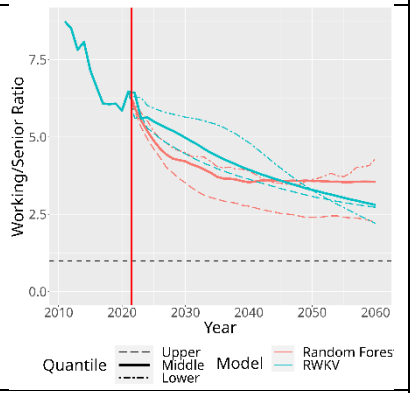
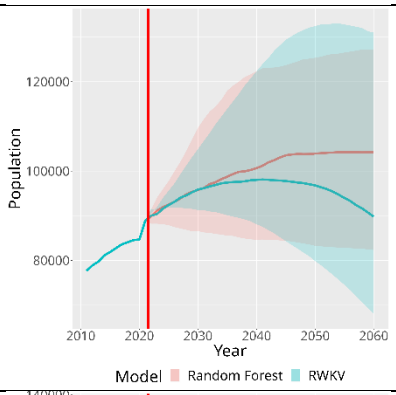
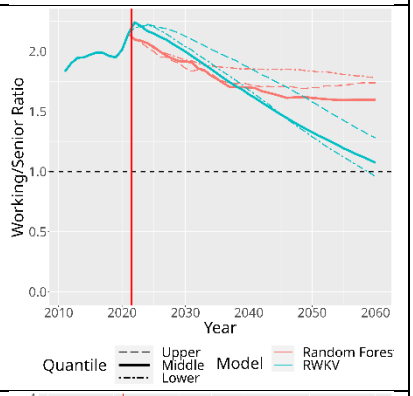
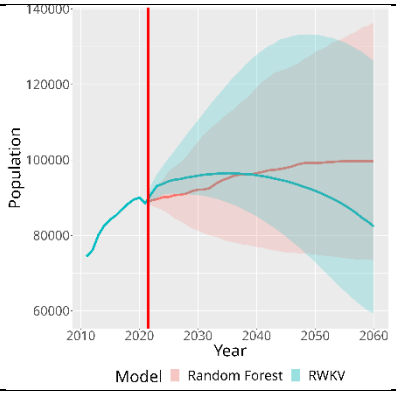
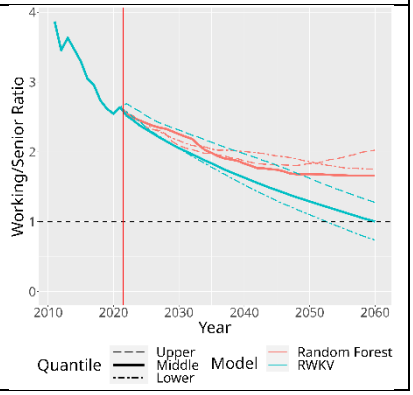
SAMPLED STATISTICAL VISUALIZATIONS

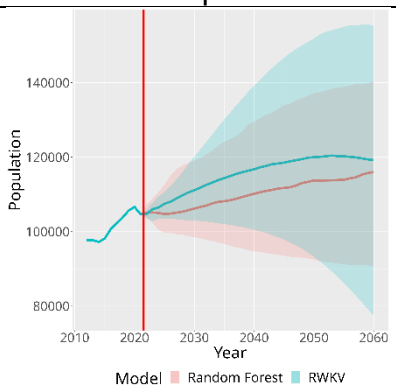
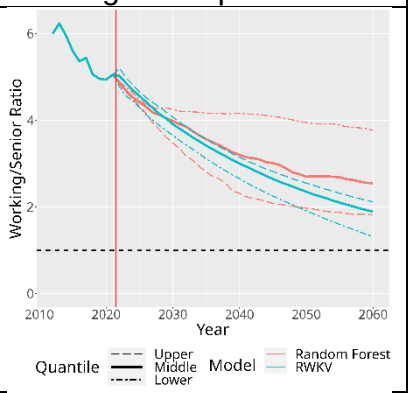
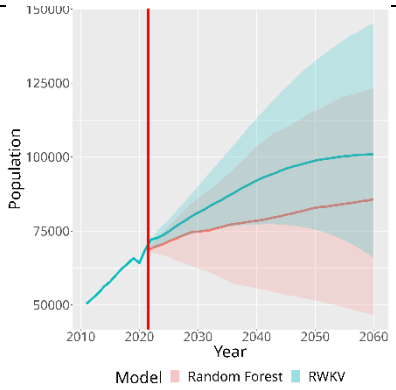
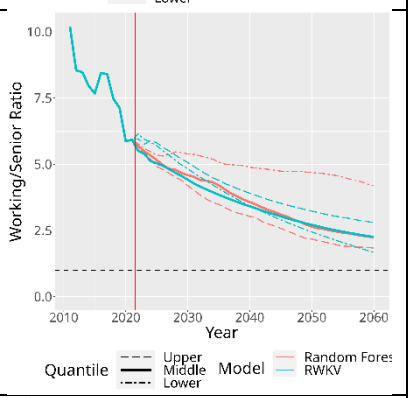
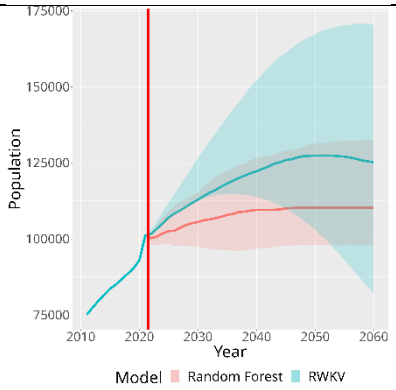
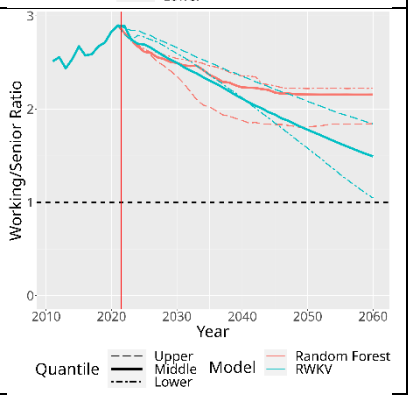
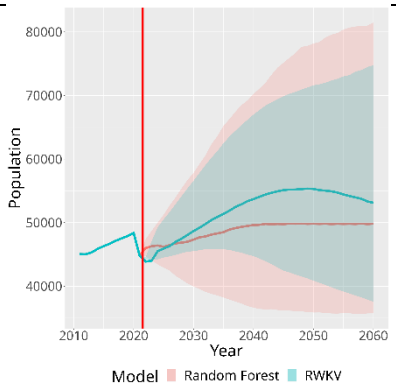
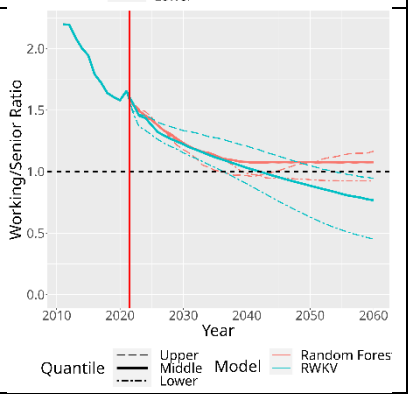
Age Group Projection in Sampled City from Country of Riverside

Place	Random Forest	RWKV
Riverside City		
Blythe		
Hemet		
Indio		

Place	Random Forest	RWKV
Jurupa Valley		
Lake Elsinore		
Menifee		
Palm Springs		

Total Population Age Group Ratio Quantile Projection RF=Red RWKV=Green

Place	Total Population	Age Group Ratio
Riverside City		
Blythe		
Hemet		
Indio		

Place	Total Population	Age Group Ratio
Jurupa Valley		
Lake Elsinore		
Menifee		
Palm Springs		

REFERENCES

[1] U. C. Bureau, "Available APIs," [Online]. Available: https://www.census.gov/data/developers/data-sets.html.

[2] H. Rech, "censusapi: Retrieve Data from the Census APIs," [Online]. Available: https://cran.r-project.org/web/packages/censusapi/index.html.

[3] D. P. D. United Nations, "World Population Prospects 2022," [Online]. Available: https://population.un.org/wpp/Graphs/Probabilistic/POP/TOT/900.

[4] N. Meinshausen, "quantregForest: Quantile Regression Forests," [Online]. Available: https://cran.r-project.org/web/packages/quantregForest/index.html.

[5] e. a. Bo Peng, "RWKV: Reinventing RNNs for the Transformer Era," [Online]. Available: https://arxiv.org/abs/2305.13048.

[6] S. B. Anastasios N. Angelopoulos, "A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification," [Online]. Available: https://arxiv.org/abs/2107.07511.

[7] U. C. Bureau, "2017 National Population Projections Tables: Main Series," [Online]. Available: https://www.census.gov/data/tables/2017/demo/popproj/2017-summary-tables.html.

[8] W. C. L. H. T. L. P. K. T. C. W. K. W. H. Y. D. D. Hadley Wickham, "ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics," [Online]. Available: https://cran.r-project.org/web/packages/ggplot2/index.html.

[9] A. Sarda-Espinosa, "dtwclust: Time Series Clustering Along with Optimizations for the Dynamic Time Warping Distance," [Online]. Available: https://cran.r-project.org/web/packages/dtwclust/index.html.

[10] W. Commons, "File:Euclidean vs DTW.jpg," [Online]. Available: https://commons.wikimedia.org/wiki/File:Euclidean_vs_DTW.jpg.

[11] J. Krijthe, "Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation," [Online]. Available: https://cran.r-project.org/web/packages/Rtsne/index.html.

[12] X. P. K. M. T. W.-X. Z. Xuming He, "conquer: Convolution-Type Smoothed Quantile Regression," [Online]. Available: https://cran.r-project.org/web/packages/conquer/index.html.

[13] C. G. Tianqi Chen, "XGBoost: A Scalable Tree Boosting System," [Online]. Available: https://arxiv.org/abs/1603.02754.

[14] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Huber_loss.

[15] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Structural_similarity.

[16] V. Efimov, "Quantile Loss & Quantile Regression," [Online]. Available: https://towardsdatascience.com/quantile-loss-and-quantile-regression-b0689c13f54d.

[17] J. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," [Online]. Available: https://arxiv.org/abs/1412.6980.

[18] F. H. Ilya Loshchilov, "Decoupled Weight Decay Regularization," [Online]. Available: https://arxiv.org/abs/1711.05101.

[19] C. L. D. H. E. R. K. W. Y. L. H. P. X. D. T. L. C.-J. H. Y. L. Q. V. L. Xiangning Chen, "Symbolic Discovery of Optimization Algorithms," [Online]. Available: https://arxiv.org/abs/2302.06675.

[20] M. Project, "MLflow Documentation," [Online]. Available: https://mlflow.org/docs/latest/index.html.

[21] R. M. Schmidt, "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview," [Online]. Available: https://arxiv.org/abs/1912.05911.

[22] N. S. N. P. J. U. L. J. A. N. G. L. K. I. P. Ashish Vaswani, "Attention Is All You Need," [Online]. Available: https://arxiv.org/abs/1706.03762.

[23] A. M. K. M. A. D. Q. L. G. H. J. D. Noam Shazeer, "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," [Online]. Available: https://arxiv.org/abs/1701.06538.