**3.1 Methodology**

**3.1 Data Collection and Description**

Our research leveraged three comprehensive datasets, each providing unique insights into the factors influencing the selection of an optimal city based on income, family size, weather preferences, and population size.

1. **US Cost of Living Dataset**: This principal dataset, obtained from Kaggle, consists of extensive data from 3,171 counties. It includes 31,431 records encompassing 15 attributes such as housing, food, and healthcare costs, alongside median family income. This dataset's richness and diversity make it a cornerstone of our analysis. [Dataset Link](https://www.kaggle.com/datasets/asaniczka/us-cost-of-living-dataset-3171-counties).
2. **Climate at a Glance County Mapping Dataset**: Provided by NOAA, this dataset illuminates the climatic aspects of our study by presenting the average temperatures of the year 2022 for 3,108 U.S. counties. It comprises seven key features, including detailed temperature data and geographic identifiers. [Dataset Link](https://www.ncei.noaa.gov/access/monitoring/climate-at-a-glance/county/mapping).
3. **County Population by Characteristics (2020-2022)**: Sourced from the U.S. Census Bureau, this dataset offers a highly detailed view of demographic variations across U.S. counties, detailing population estimates segmented by gender and age for 2020-2022. This dataset is instrumental in understanding population dynamics in various locales. [Dataset Link](https://www.census.gov/data/tables/time-series/demo/popest/2020s-counties-detail.html).

**3.2 Methodological Approach**

**3.2.1 Data Preprocessing**

Our preprocessing phase was meticulously designed to ensure data integrity and relevance:

* **Cost of Living Dataset**: We meticulously split the 'family\_member\_count' into separate counts for adults and children. This is done in order to both extract additional data and to account for more non-traditional family structures.
* **Weather Dataset**: In the original dataset the state’s names we fully written, however in Cost\_of\_living dataset the state names were an abbreviation of the full name. Thus, we changed the state name in the weather dataset to match with states on the other dataset. and the column 'NAME' was renamed to 'COUNTY' for clarity and to match the other dataset. Then we Combined the Cost\_of\_living dataframe with data from weather dataframe. Then removed the Columns 'ID', 'Rank', 'Anomaly (1901-2000 base period)', '1901-2000 Mean', 'areaname', 'case\_id', as we only require Average Temperature for our specified years which is provided by the 'Value' Column.
* **Population Dataset**: Initialy the dataset had the data from 2020 until 2022 but we only needed 2022 data Thus, the focus was narrowed to the year 2022 to capture the most current demographic insights. We only take values for the 4th and latest years demographics as this is the most relavent for our research. State names were abbreviated for consistency. We droped extra columns from it since they were redundant. We only take the values of the columns 'POPEST\_MALE', 'POPEST\_FEM', 'MEDIAN\_AGE\_MALE', 'MEDIAN\_AGE\_FEM'. This is done because the original population dataset has too many features for each group that might lead to unnecessary increased complexity for our model. These chosen columns contain the total population and Median ages for both sexes, which manage to capture most of the information with significant reduced number of features. We Then merged both the original and the population dataset based on the common columns 'county' and 'state'. The features for our final data frame can be observed on table1.

|  |  |
| --- | --- |
| Features | Description |
| State |  |
| IsMetro |  |
| county |  |
| housing\_cost |  |
| food\_cost |  |
| transportation\_cost |  |
| healthcare\_cost |  |
| other\_necessities\_cost  childcare\_cost  taxes  total\_cost  median\_family\_income  adult\_count  child\_count  avg\_temp  popest\_male  popest\_fem  median\_age\_male  median\_age\_fem |  |

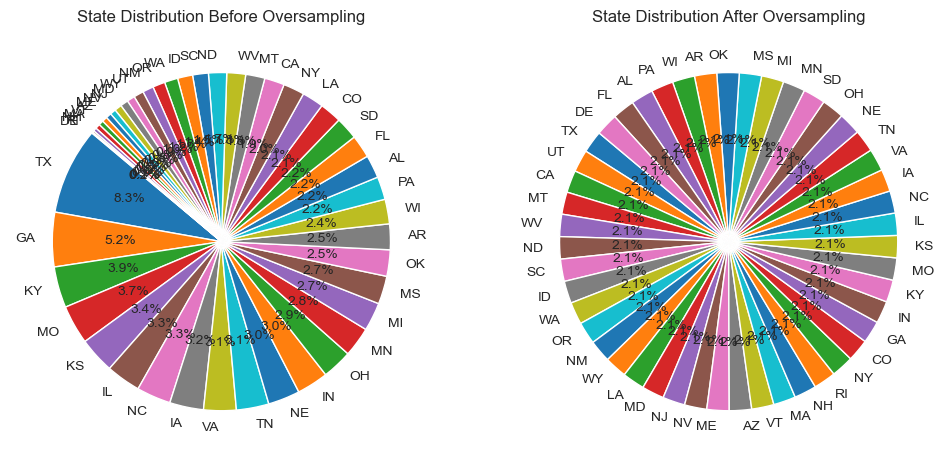
Based on figure[1] we realized that this dataset isn’t balanced. To overcome data imbalance, we employed oversampling techniques, ensuring a representative distribution across states.

Figure 1- State Distribution before and after Oversampling

We now finalize our data preprocessing with the following steps:

1. Check for any null values that might exist in the dataset.
2. Use Label Encoding to encode all categorical data to numeric data to be more freely usable to ML and Deep Learning Models.
3. Change Types of all columns in the dataset to be numeric in case some columns do not have the correct type despite all of them being numerical data.
4. We shuffle the dataset so that no model can find any patterns from the arrangement of the data

We separate out all labels from the original dataframe and put in into a new dataframe. Then wo do the following transformations to the new Dataset:

1. We convert all columns to float64. This is done so that all columns in the dataframe share a type to make future transformations easier.
2. We scale the data using a Standard Scaler. This is done because most of the columns contains data that are not on similar scales, which might lead to errors for several machine learning algorithms.
3. We then shift the scaled values to only contain positive values. This is done by subtracting the lowest negative value from all datapoints. This way, the lowest value in the dataframe becomes 0, thereby placing all other datapoints on the positive scales. This is done because some algorithms like Naive Bayes work best on positive values.

We then seperate out the labels into their own dataframe. Note that since we are only predicting states, we only use the states columns for all predictions. We then split the x and y dataframes into train test sets to be used by the models. These sets are then turned into tensors for use in our custom neural network.

After integrating and aligning our datasets, we undertook a series of critical final preprocessing steps to ensure the data's readiness for machine learning and deep learning models:

1. **Null Value Examination:**
   * Our first task was to rigorously check the dataset for any null or missing values. Identifying and addressing these gaps is crucial as they can significantly impact the performance and accuracy of our models.
   * There found no null value on our data frame.
2. **Categorical Data Encoding:**
   * We employed Label Encoding to transform all categorical data into numerical formats. This conversion is essential since most machine learning algorithms operate more efficiently with numerical inputs.
   * This step involved converting text labels into a series of unique integers, making the data more interpretable for our algorithms.
3. **Data Type Standardization:**
   * We reviewed and standardized the data types of all columns. Despite the numerical nature of our data, inconsistencies in data types can arise during data integration.
   * Ensuring uniform data types across the dataset is vital for maintaining computational efficiency and avoiding type-mismatch errors during model training.
4. **Dataset Shuffling:**
   * To prevent any potential biases arising from the original ordering of the data, we shuffled the dataset. This step is pivotal in ensuring that the models do not inadvertently learn patterns from the sequence of data rather than the actual features.
5. **Transformation of the Labelled Data:**
   * We extracted the labels (target variables) from our dataset and stored them in a separate dataframe. Since our primary goal is to predict optimal states, the labels consisted of state identifiers.
   * For the remaining data (features), we performed several transformations to optimize them for model input:
     + **Type Conversion:** All feature columns were converted to **float64**. This uniformity in data type eases further data manipulations and transformations.
     + **Data Scaling:** We utilized a Standard Scaler to normalize the feature data. This scaling is critical because it brings all features to a similar scale. This is done because most of the columns contains data that are not on similar scales, which might lead to errors for several machine learning algorithms.
     + **Positive Value Shift:** To adapt our data for algorithms that perform better with positive values (like Naive Bayes), we shifted all scaled data values to be positive. This was achieved by subtracting the minimum (negative) value across the dataset from all data points, effectively resetting the lowest value to zero.
6. **Data Splitting and Tensor Conversion:**
   * We split the feature and label dataframes into training and testing sets. This separation is fundamental to evaluating our models effectively, ensuring they are tested on unseen data.
   * Finally, we converted these datasets into tensors, the required format for feeding data into our custom neural network built with PyTorch. Tensors facilitate efficient computations and are a standard format for neural network inputs.

**3.2.2 Model Development and Training**

We adopted a multi-model approach, utilizing Python and its robust libraries:

1. **Logistic Regression**: Logistic Regression, particularly Multinomial Logistic Regression, is a statistical model used for predicting the probabilities of the different possible outcomes of a categorically distributed dependent variable. In our case, it predicts the best-suited state based on various features. We employed the **LogisticRegression** class from **sklearn.linear\_model**, specifying **multi\_class='multinomial'** to handle multiple classes (states). Logistic Regression is particularly effective for binary or multiclass classification problems where relationships between independent variables are linear.

**Advantages**: It's a simple, fast, and interpretable model, making it suitable for baseline comparisons.

1. **Support Vector Machine (SVM)**: SVMs are powerful classifiers that work by finding the best hyperplane to separate data points of different classes. In our project, we used both linear SVMs and explored One-vs-One (OvO) and One-vs-All (OvA) strategies for multiclass classification. We used the SVC class from sklearn.svm, experimenting with linear kernels and both OvO and OvA strategies. The linear kernel was chosen for its efficiency in high-dimensional spaces.

**Advantages:** SVMs are effective in high-dimensional spaces and relatively memory efficient. However, they require careful tuning of the hyperparameters and can be sensitive to the choice of kernel.

1. **Naïve Bayes Classifier**: Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. We used the MultinomialNB class from sklearn.naive\_bayes, which is well-suited for discrete feature models like ours. It's particularly effective when the dimensionality of the input is high, as in text classification problems.

**Advantages:** Naive Bayes is known for its simplicity, efficiency, and effectiveness, especially in large datasets. It performs well in multi-class prediction and when the assumption of independence holds.

1. **Custom Neural Network**: Neural Networks are a set of algorithms, modeled loosely after the human brain, designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling, or clustering raw input. For our custom neural network, we opted for a moderately complex network, with 5 hidden layers between the input and output layers. We used ELU activation function for these hidden layers to avoid the dying Relu problem, which occurs when the Relu activation function outputs only 0 for all inputs. We used Cross Entropy loss for multi classification. We also used Adam optimizer as it tends to be the most well rounded optimizer for neural networks. We train our model with a learning rate of 1X10^-4 for 100 epochs. This ensures that the model will be able to slowly learn the patterns hidden in the data without a high risk of overfitting on the training data.

**Advantages:** Neural networks are highly flexible and capable of modeling complex non-linear relationships. They are particularly effective for large datasets and can approximate almost any function given enough data and compute power.

Each model was rigorously trained and evaluated based on its ability to accurately predict the best-suited state based on income, family size, preferred weather, and city population. The choice of models encompassed both traditional statistical approaches and advanced machine learning techniques to provide a comprehensive analysis.

**3.3 Evaluation Metrics and Model Assessment**

In-depth analysis of model performance was conducted using accuracy, precision, recall, and F1-score. These metrics provided a holistic view of each model's strengths and weaknesses, guiding us in fine-tuning our models and selecting the most effective algorithm for predicting the best city based on our criteria.