# Group ID - MSc in Data Analytics

Author: Zhe Luo

Lecturers : Muhammad Iqbal **David McQuaid** Taufique Ahmed Sam Weiss

Github : https://github.com/ZheLuo2023/CA1.git

e-mail: [2023234@student.cct.ie](mailto:2023234@student.cct.ie)

Student ID: 2023234

**Introduction:**

This paper focuses on the demographic analysis of the Irish population. The data used in this study is sourced from <https://data.cso.ie/table/PEA11>, which was collected by The Central Statistics Office in Ireland in relation to the population of Ireland. The dataset is titled "Population estimates from 1926."

One of the objectives of writing this paper is apply various statistical and machine learning analysis techniques that we have learned, combined with visualization methods. Python, with its numerous scientific computing libraries, is used for this purpose.

Taking into account the characteristics of the dataset, this paper selects a research direction to explore the population of Ireland based on year, age, and demographic trends. Machine learning algorithms are employed to attempt regression predictions, with the goal of finding the most accurate model.

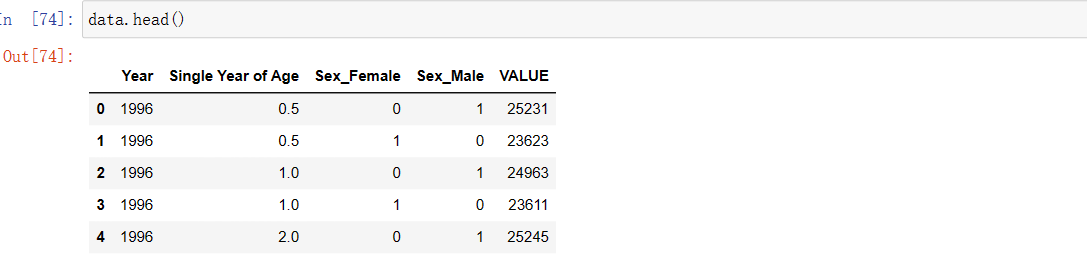
**Main Body :**

Due to different areas of emphasis, this paper will be divided into four distinct chapter, each dedicated to discussing machine learning, statistics, data preparation and visualization, and Python programming separately:

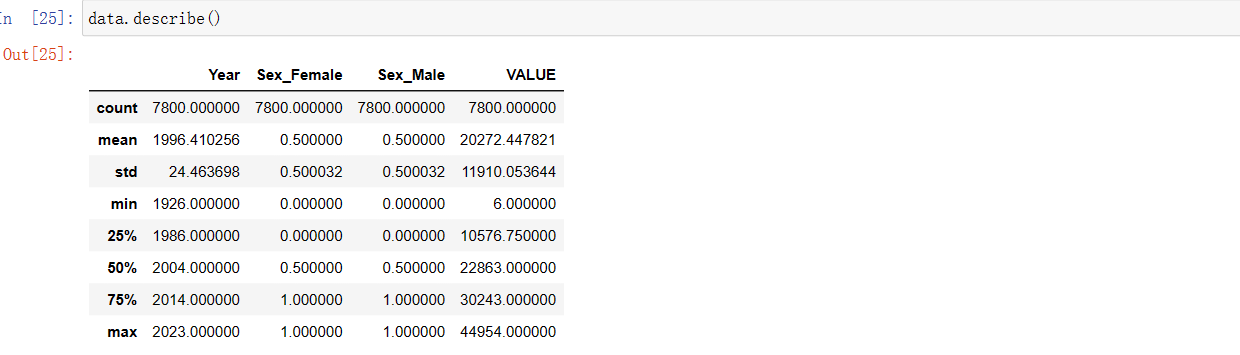
**Statistics**

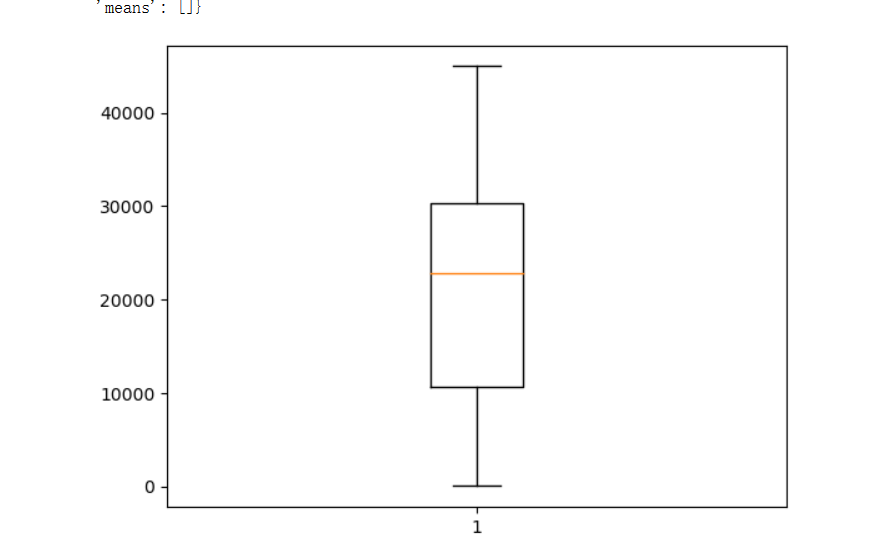
The dataset is titled "Population estimates from 1926," and it pertains to the population of Ireland. It covers a time span from 1926 to 2023 and consists of 11,817 rows and 6 columns. The variables as follows :"STATISTIC Label" ,"Year","Single Year of Age","Sex","UNIT","VALUE" (Population count)

The "VALUE" column, which represents the population count, is considered the target variable. The objective is to find the relationships between this target variable and the other variables. After data preprocessing, which includes removing unnecessary variables such as "STATISTIC Label" and "UNIT," and converting all data to integers or floats, the dataset is transformed into 7,800 rows and 5 columns.



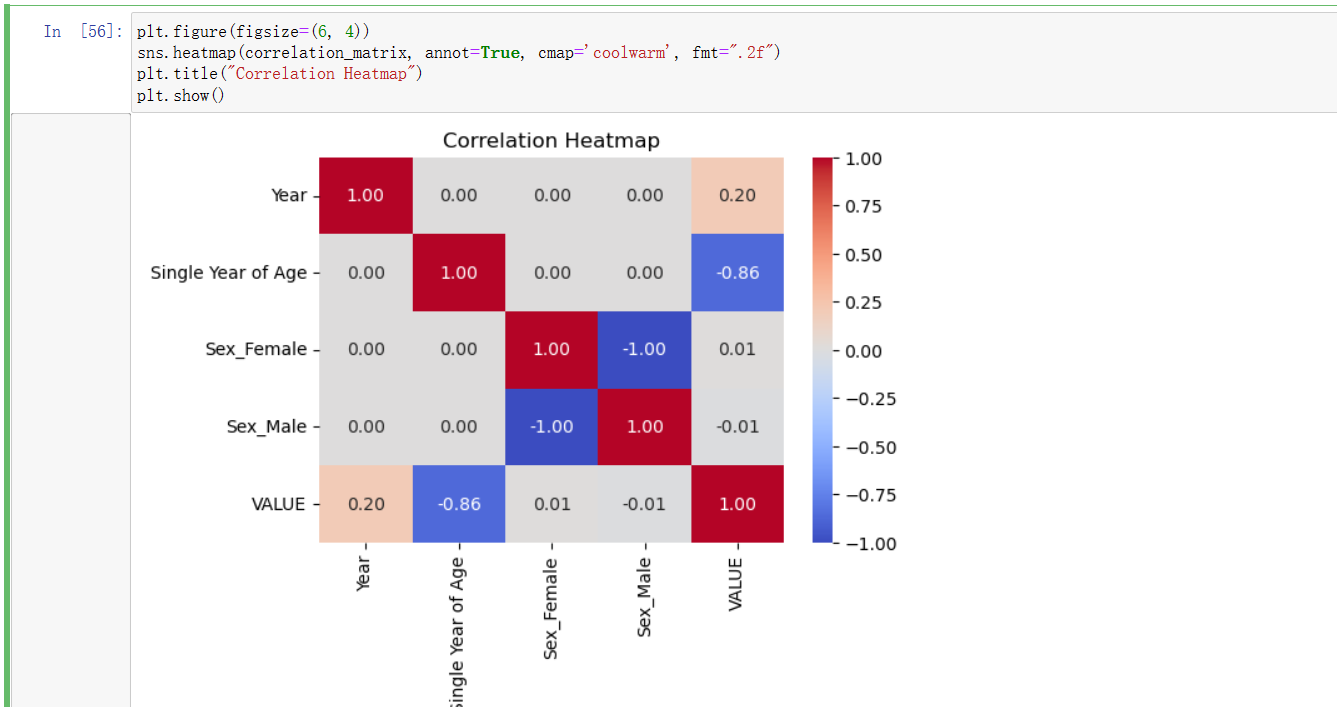
We would like to get a preliminary understanding of the data, and we want to know some basic information about the 'VALUE' column. So, we used **'describe()**' and '**boxplot()**





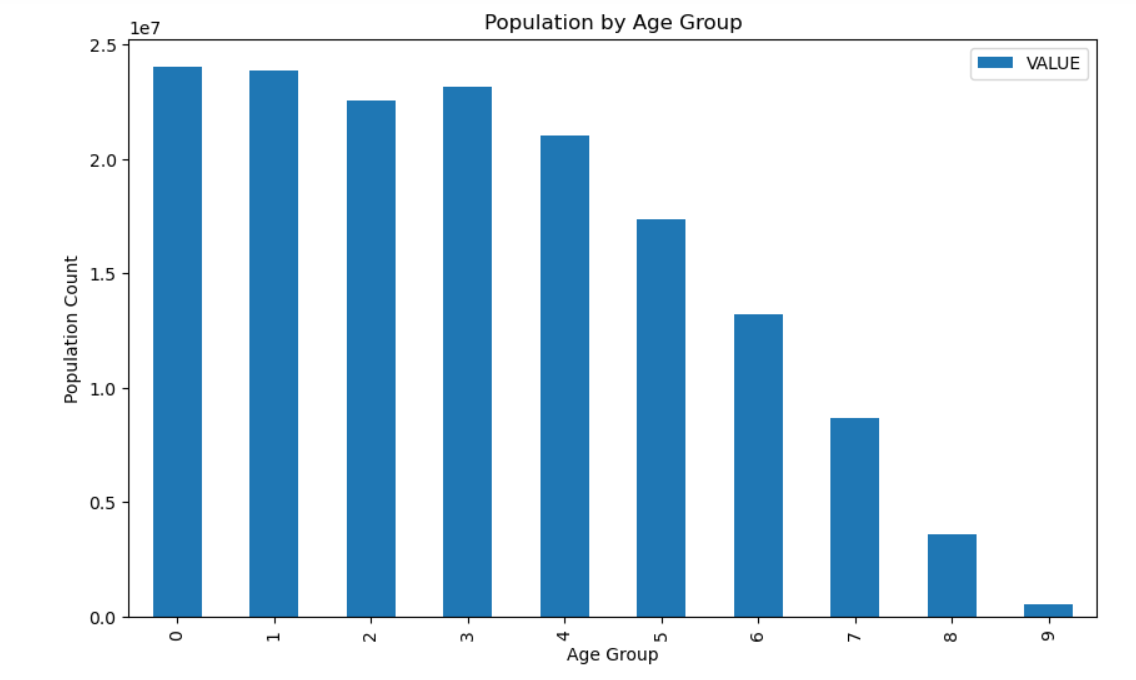
From the above statistical data, we can see that the median population, represented by the red line in the boxplot, is located at 22,863. The rectangular box in the box plot represents the middle 50% range of the data, which is the data between the first quartile, which is 10,576, and the

third quartile, which is 30,243

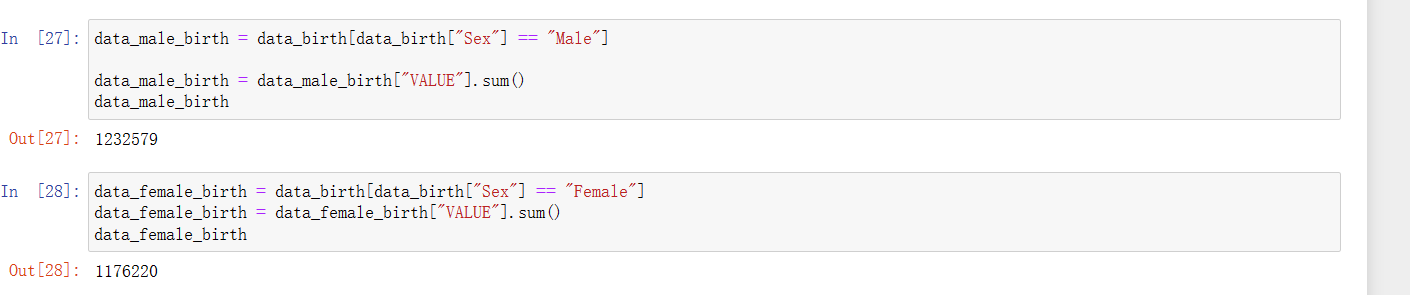


By interpreting the heatmap, we can observe that the correlation values between most variables and the population count are quite low. The only notable strong negative correlation exists between age and the population , with a coefficient of -0.86. To explore their relationship further, we created a bar chart illustrating the relationship between age and population count, dividing age into 10-year intervals, resulting in a total of 10 groups.

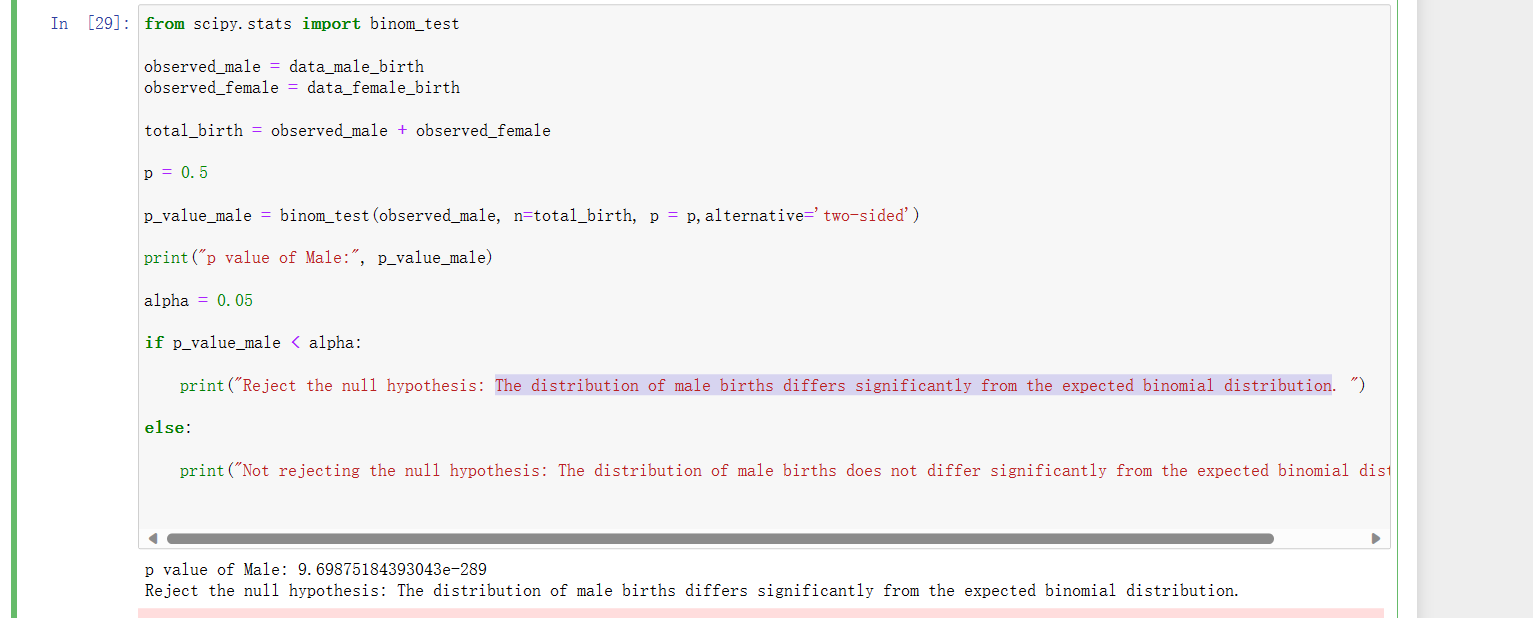




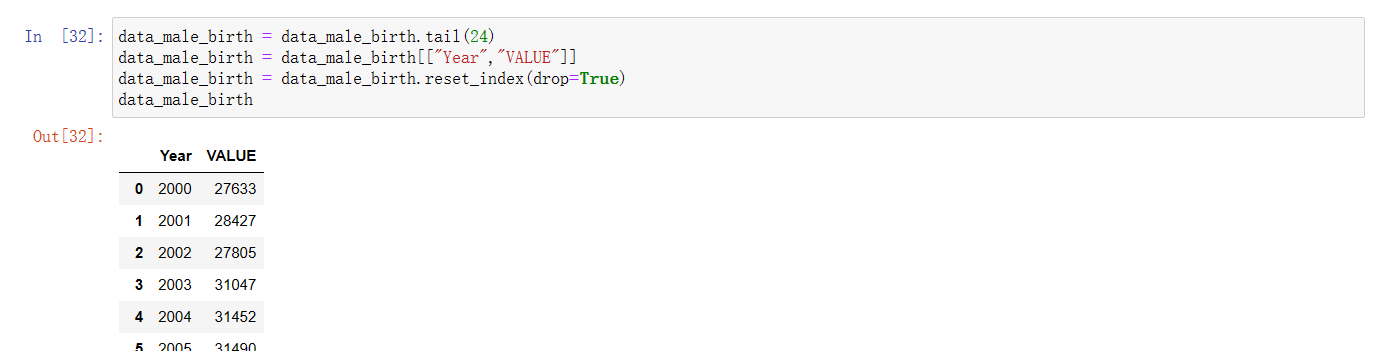
We can see that in the historical population of Ireland, there is a clear decreasing trend in population count with increasing age, which is logical as in a naturally evolving society, the elderly population is expected to be relatively smaller. Additionally, it is evident from the graph that this does not conform to a normal distribution. Next, we want to explore whether the probability of birth for males and females is equal, i.e., a binomial distribution with p = 0.5. Therefore, we will use hypothesis testing for verification. In the dataset, we consider individuals under the age of one as the population born in that year. By cumulative summation, we obtain the male and female births in these years:

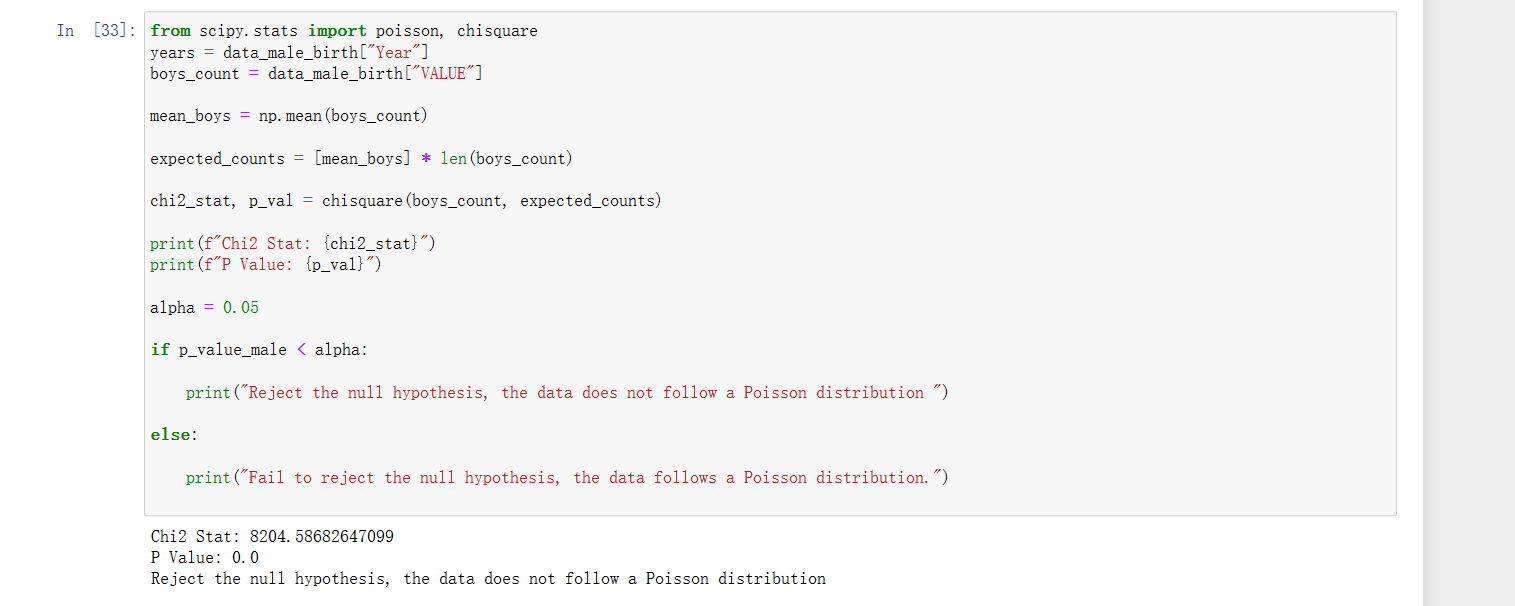


We will set the null hypothesis as follows: "The distribution of male births differs significantly from the expected binomial distribution." Below is the specific process of the hypothesis test:



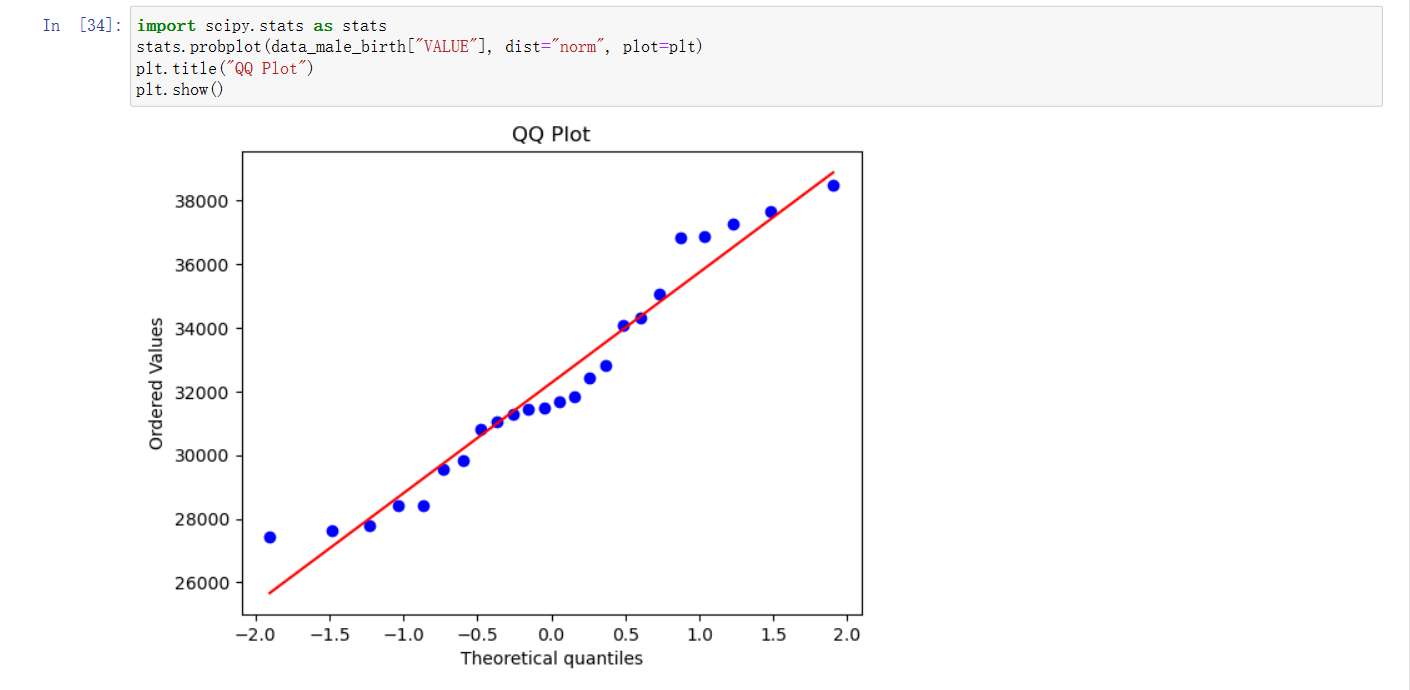
From the results, we can see that the birth rate of males does not conform to the binomial distribution with p = 0.5. Therefore, we reject the null hypothesis. The reason we chose the binomial distribution for validation is that we treated population births as a series of independent and identical experiments, each with two possible outcomes: boys or girls. Based on this dataset, we can conclude that the birth rates of males and females in Ireland's history are not the same. Now, with a keen interest in the number of male births, we are curious about what kind of distribution it follows. With this in mind, the first thing we want to verify is whether it conforms to a Poisson distribution. We have extracted data from 2000 to 2023. The reason for this choice is that during this period, there is no missing data in the dataset. We treat the occurrence of male births in the past 23 years as a limited event, which fits the description of a Poisson distribution.



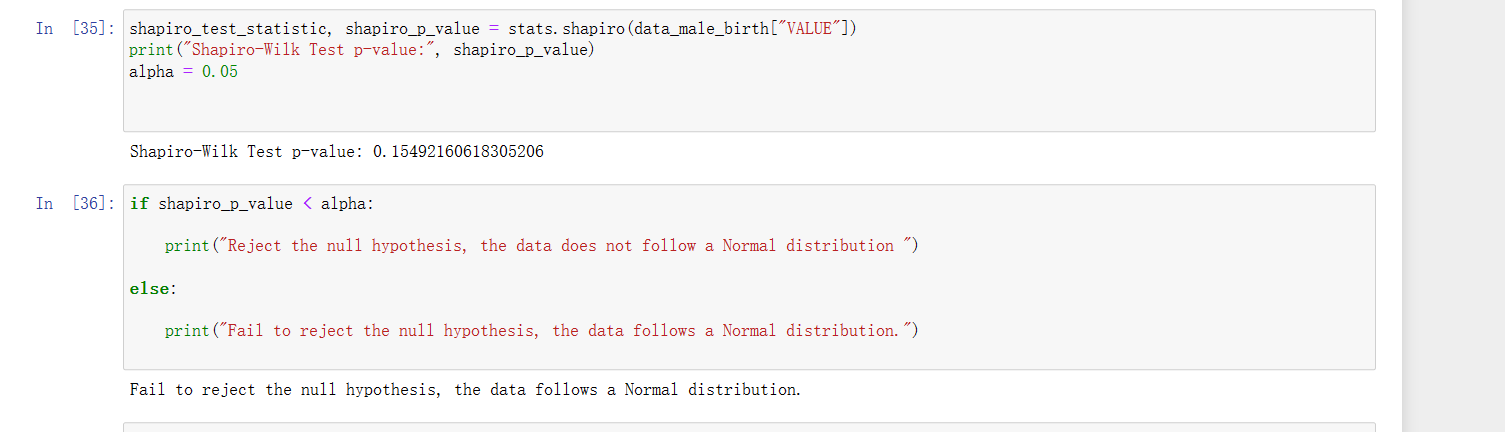


Therefore, from the analysis results, we can see that the P-value is almost 0, falling into the significant area. Thus, we reject the null hypothesis and conclude that the data does not follow a Poisson distribution. Now the question arises: what distribution does the population birth data for boys actually follow? Since it is a natural event, we would like to attempt to verify whether it conforms to a normal distribution.

To do this, we will start by creating a QQ plot to observe the distribution of the data.



From the graph, we can see that the data points are mostly clustered around the red reference line (representing the normal distribution). Therefore, we can tentatively conclude that the data follows a normal distribution. To confirm this conclusion, we used the common Shapiro statistical test for normality.



From the final results, we can see that the null hypothesis cannot be rejected, indicating that the data follows a normal distribution. With this, we conclude the statistical analysis section of this dataset.

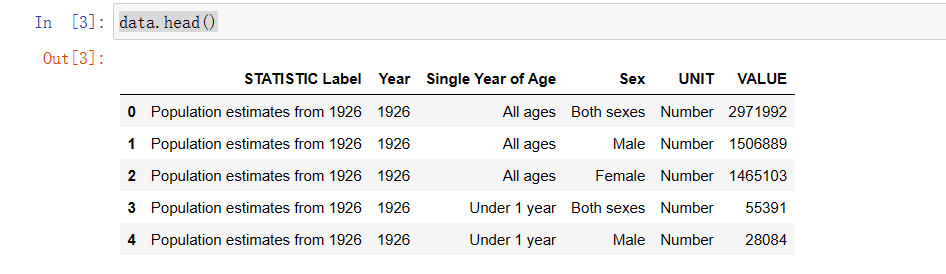
**Data Preparation and Visualization**

This chapter will be divided into three parts:

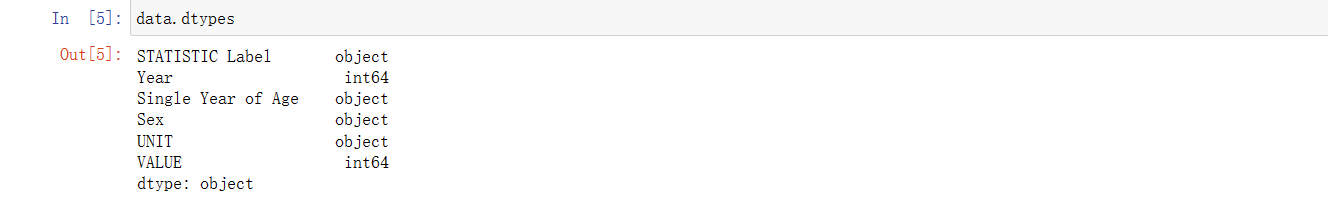
1. **Exploratory Data Analysis (EDA)**

Early exploration of data is crucial as it helps us quickly understand the dataset's structure, basic parameters, data deficiencies, and expands our research ideas. It also plays a crucial role in data cleaning, feature engineering, and the study of statistical or machine learning algorithms. First, we would like to explore the first five rows of the data.

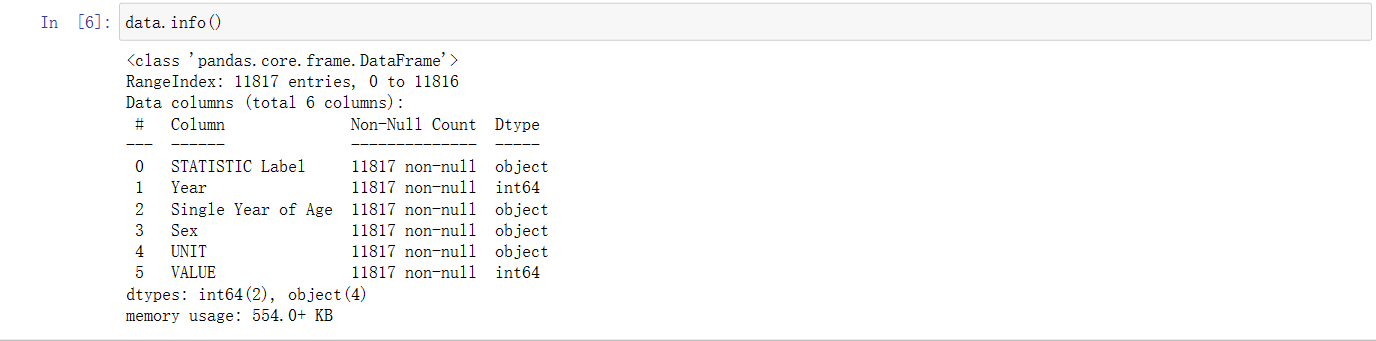
窗体顶端



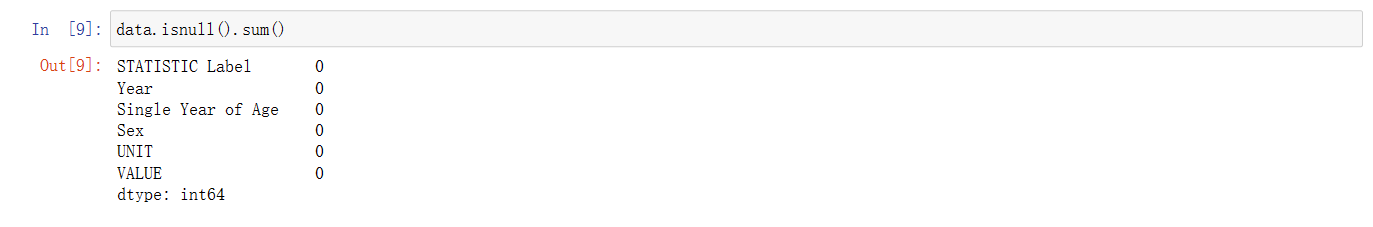
From this, we can get a rough idea of all the variables in the dataset and the approximate form of each variable's values.



By examining the **dtypes**, we can determine the original data types of each variable before any data processing. Some variables are labeled as "object" because they contain at least two or more data types, making it challenging for Python to determine the specific data type. This information will guide us in the upcoming data cleaning process.



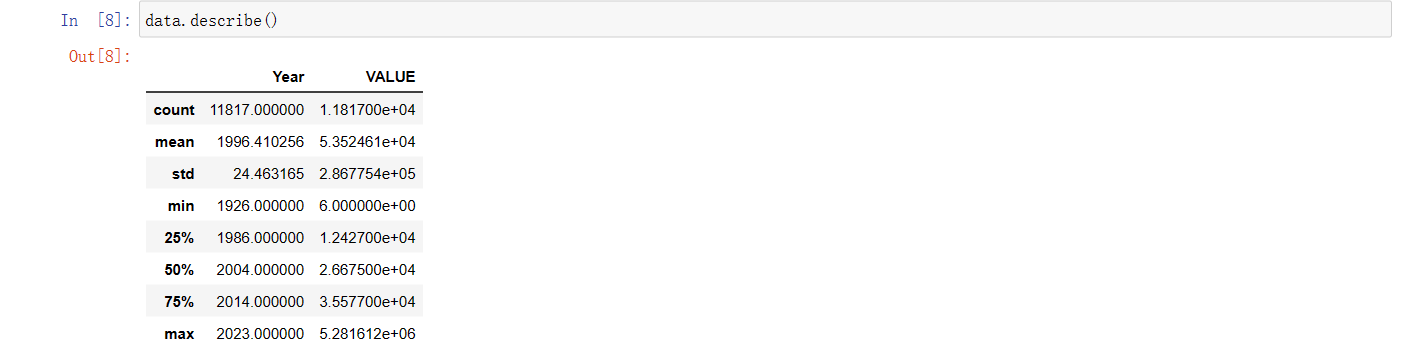
Usually, we can use the **info()**  method to understand which variables in the data have missing values, providing guidance for the subsequent data cleaning. For the dataset used in this session, the number of entries for all variables is 11,817, so we can reasonably conclude that there are no missing values in the dataset.



We have also confirmed this idea by using the  **isnull()** method.



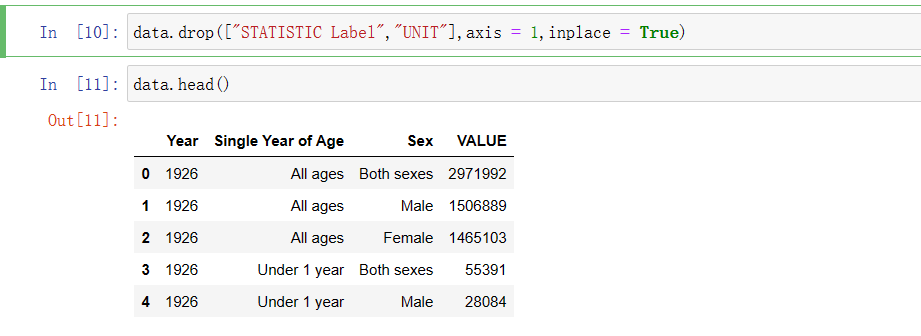
With the **shape** method, we can instantly determine the number of rows and columns in the data.



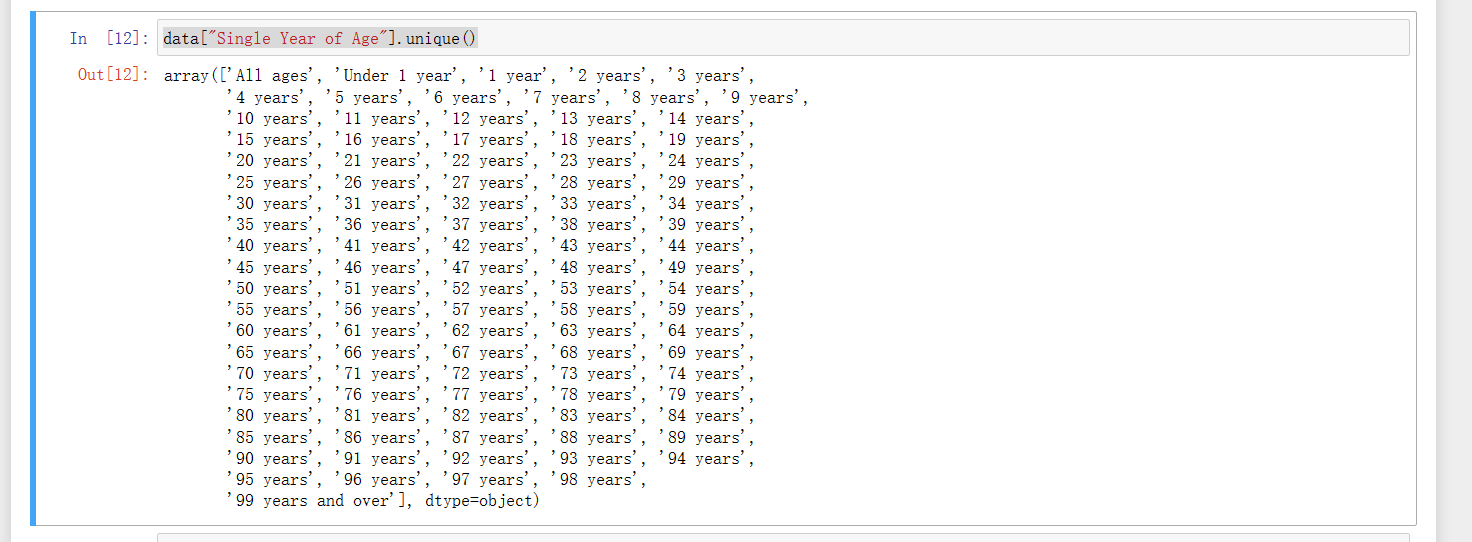
Through **describe()**, we can gain insight into some basic statistical information about the dataset, such as extreme values, medians, and by calculating Q3-Q1, we can understand the range within which 50% of the data falls. Exploratory Data Analysis (EDA) also includes some early data cleaning, as well as the presentation of **boxplots, heatmaps, and bar plots**. These will be detailed in the data processing and visualization sections.

1. **Data Preprocessing and Feature Selection**

This section is primarily aimed at preparing for subsequent statistical analysis and machine learning algorithms.

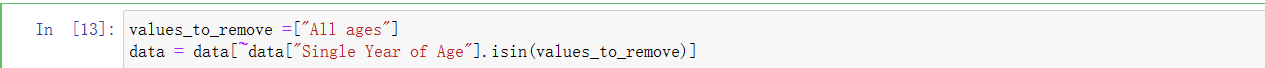


First, we removed the unnecessary columns "STATISTIC Label" and "UNIT."



Then, we started processing the first important feature, "Single Year of Age." By using **unique()**, we could understand all the possible values contained in this variable.

窗体顶端

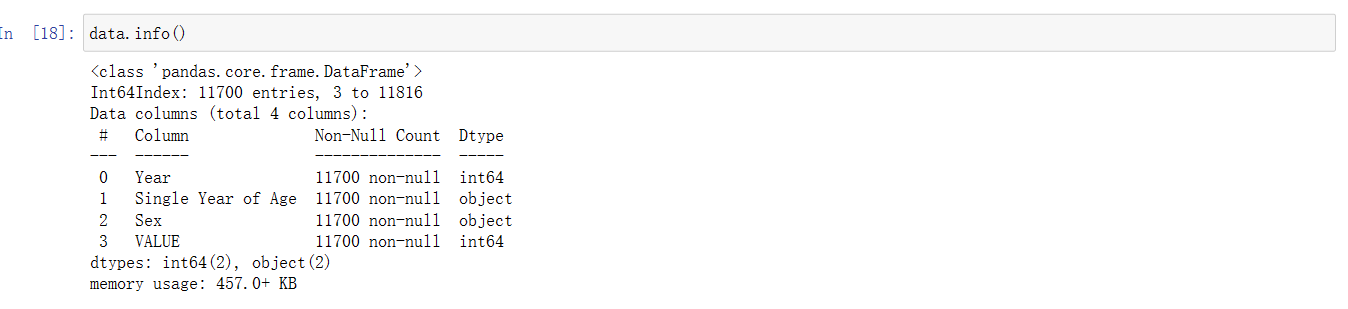


First, we remove the unnecessary value of "All ages."

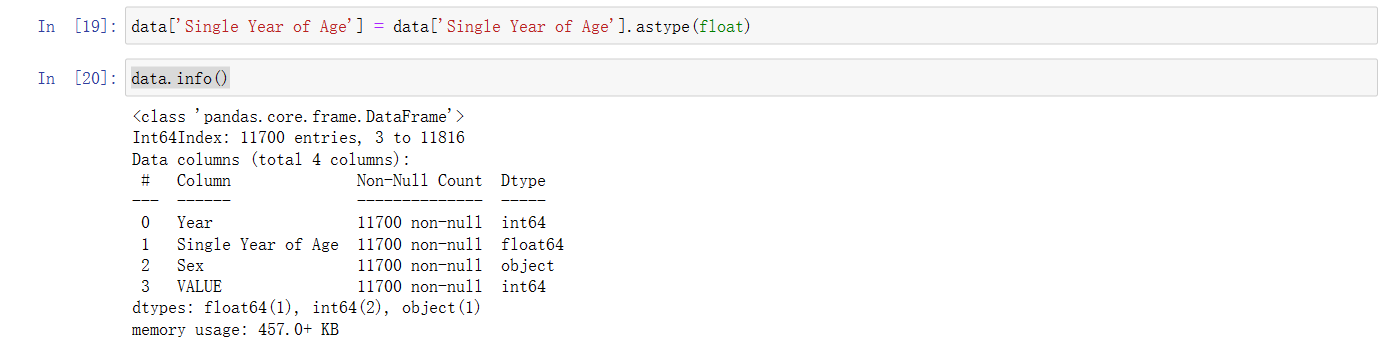


Then, we constructed a Class AgeDataProcessor to replace all values less than one year with 0.5 and converted the remaining values from strings to float format.

窗体顶端

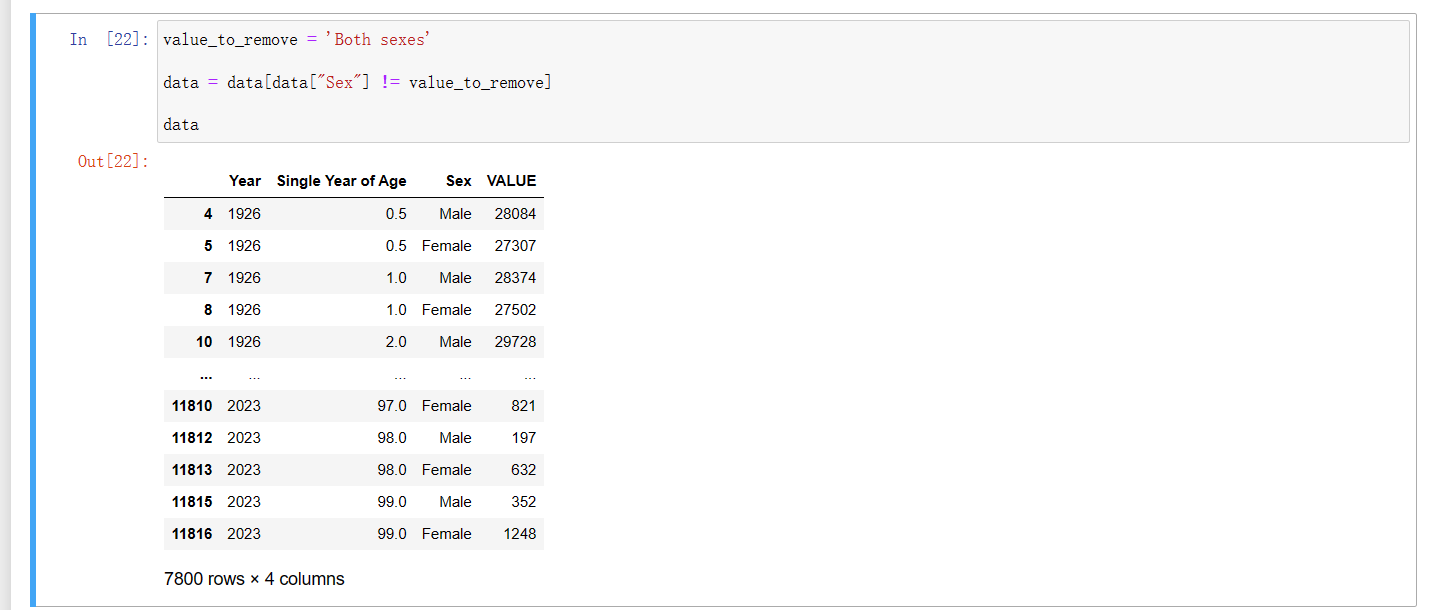


However, we found that the data type of this column is still "object." ,because during the creation of the array, some elements were identified as Python objects rather than numerical types. This might be due to special characters, spaces, or other unusual characters.

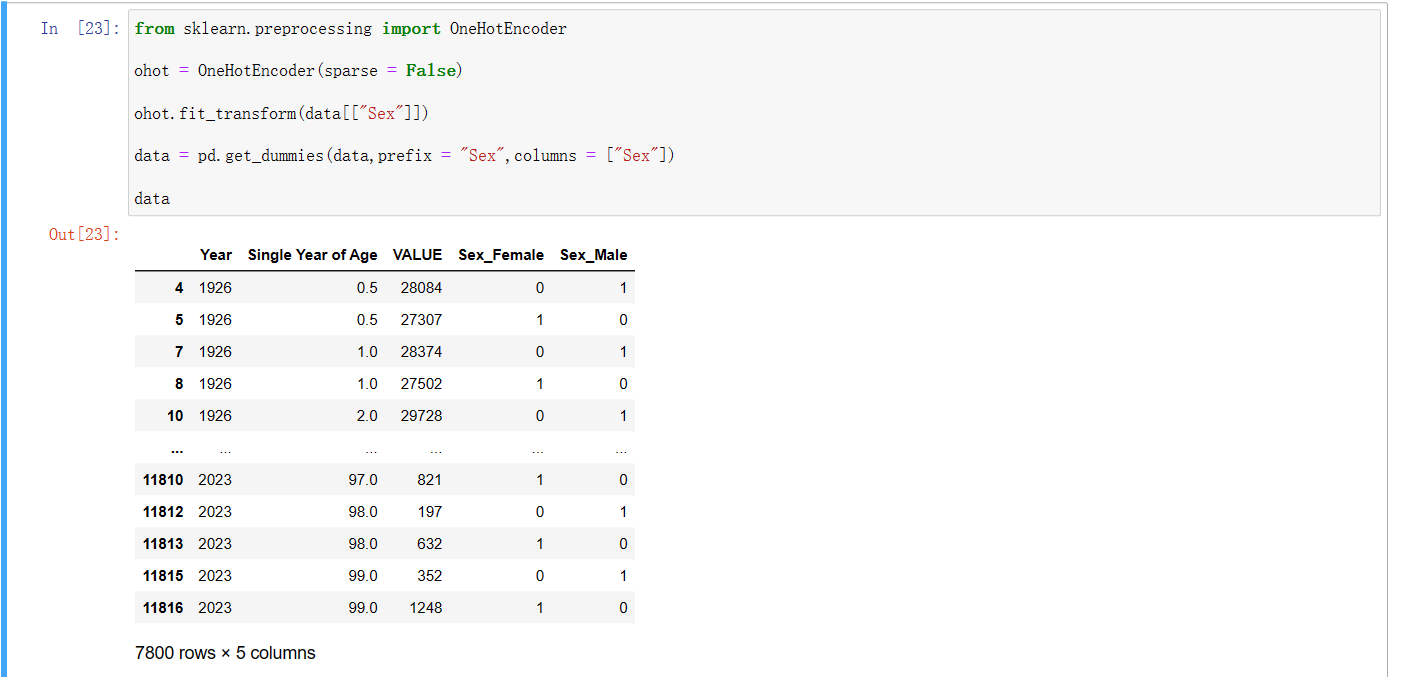


So, we used the method **astype(float)** to forcefully convert the data into floating-point numbers. From the results, we can see that the data type has finally become the floating-point type we need.

Next, we will process another important variable, "Sex." First, we want to remove all rows containing "Both sexes" in the entire column, as it is not needed in our study.



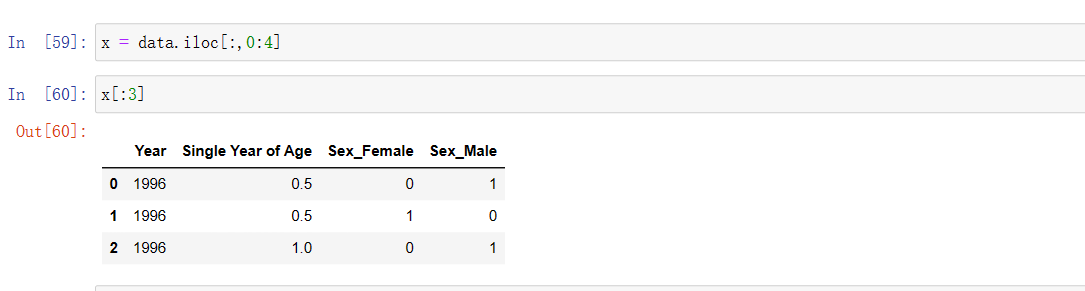
We achieved this goal using only slicing.

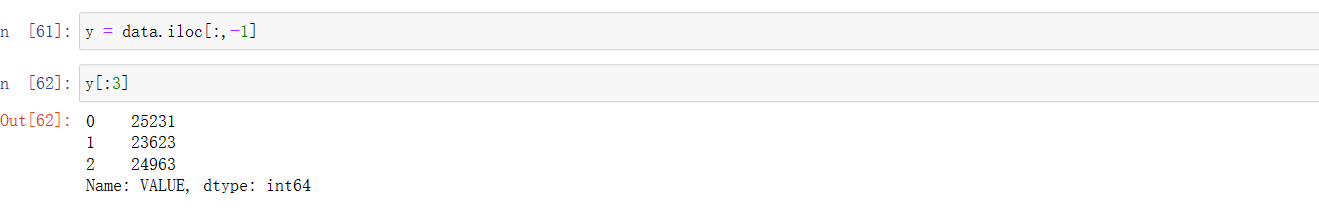


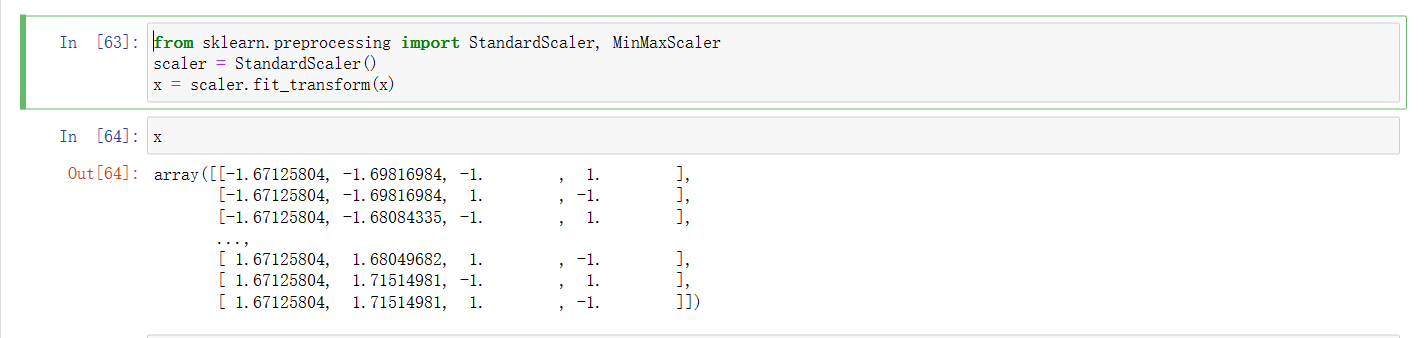
Then, we used the encoding method to convert categorical variables into data types. There are two methods: Ordinal Encoding and One-Hot Encoding. Here, the second method is logically more correct because gender does not have a natural order.

So far, the data cleaning, feature engineering, and feature encoding parts have been completed. However, before proceeding with machine learning, we still need to perform data standardization. In the statistical part, we have already verified that the majority of the data follows a normal distribution, so we choose standardization.

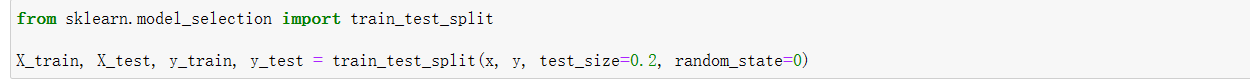
First, we will separate the data into a training set and a test set.







We can see that after standerdization, the training set eliminates the impact of different units.

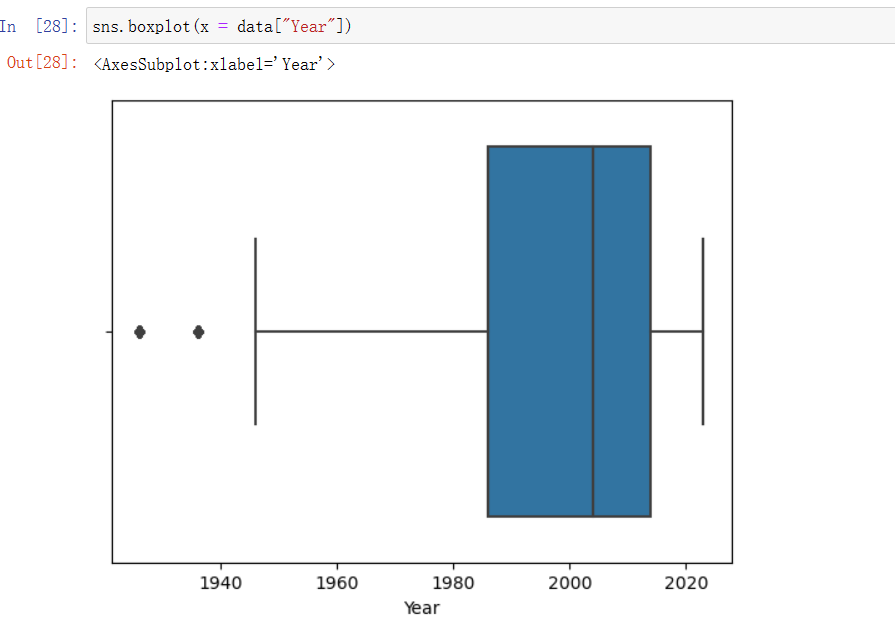


Finally, we split the data into training and test sets with a ratio of 4:1. With this, all data processing tasks have been completed.

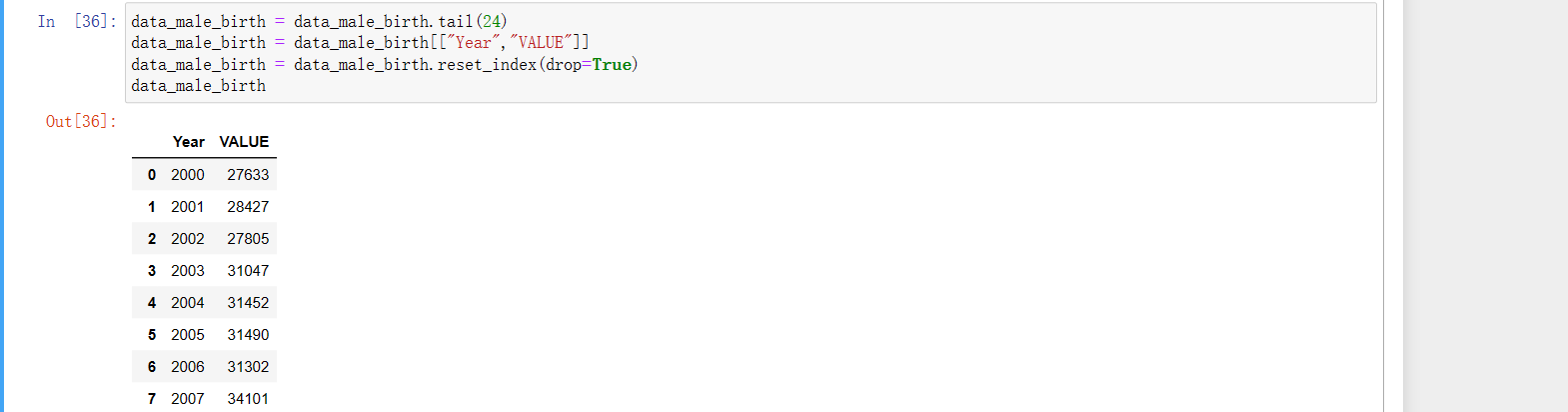
1. **Data Visualisation (Regarding requirements 3 and 4)**

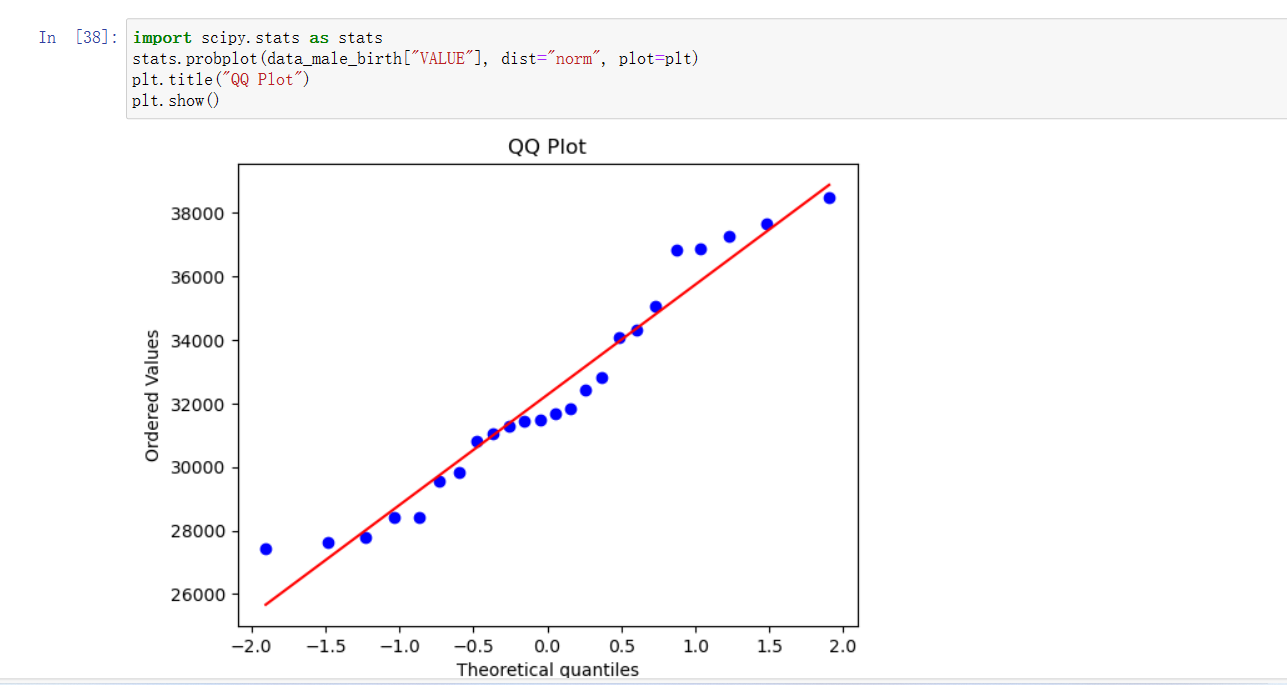
In the process of Exploratory Data Analysis (EDA), we initially used the boxplot from the **Seaborn** library to explore the data.

窗体顶端



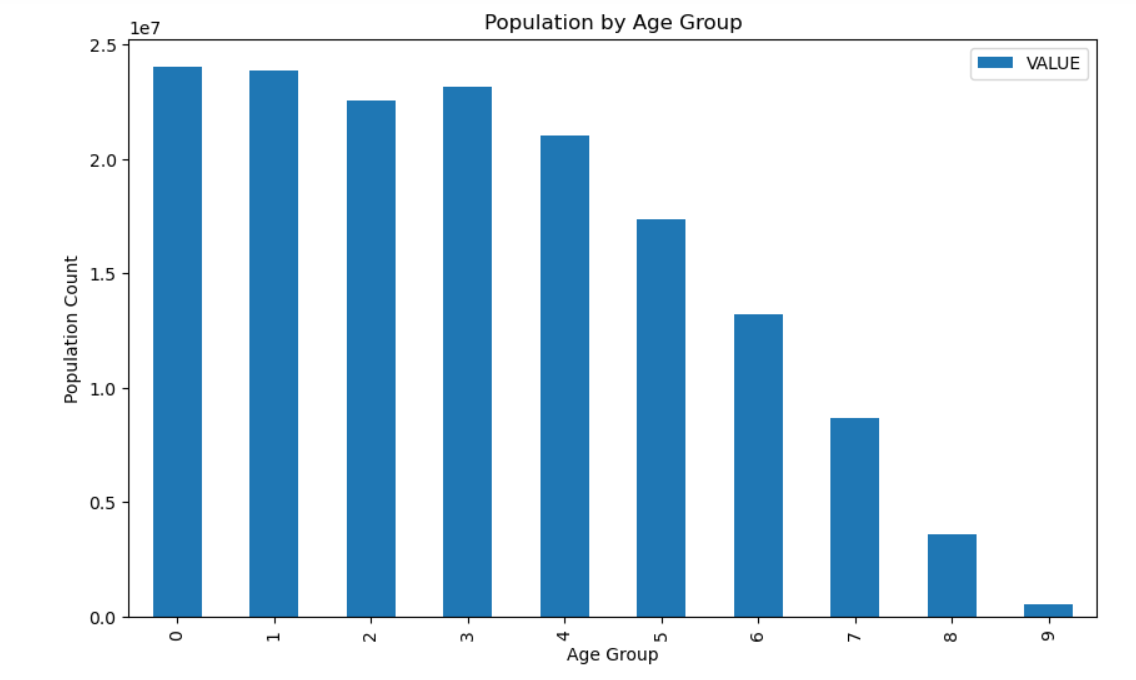
From the graph, we can see that the median of the data falls around the year 2000, and data before 1940 is identified as outliers. This indicates that the "Year" feature had a significant number of missing years in the early period, which could have a detrimental impact on regression analysis. Therefore, machine learning algorithms were trained using data after the continuous year 2000



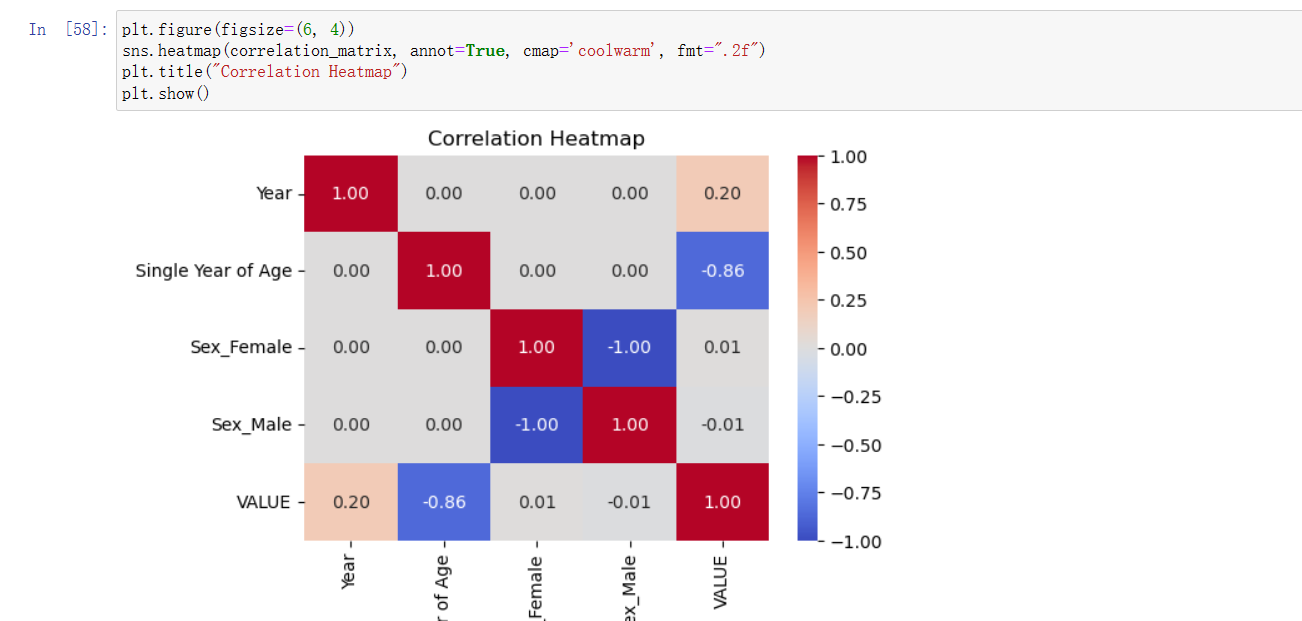


To assess the distribution of annual births, we used a **QQ plot**. In the plot, the red line represents the fit for a normal distribution, and the blue dots represent the data points. We can observe that most data points cluster around the red line, indicating that there is a strong likelihood that the data follows a normal distribution. This forms the basis for the subsequent normal distribution test.





During my exploration of the connection between birth population and age, I initiated some preparatory steps. Firstly, I categorized ages into 10-year intervals employing the "**cut"** method from the Pandas library. In the course of data manipulation, I harnessed the "**groupby**" method, which offers a convenient means to aggregate and sum birth populations within the specified age groups. Ultimately, I represented the data visually using a **bar chart**. This visualization clearly reveals that the birth population does not conform to a normal distribution, as it exhibits variations across different age groups. These observations furnish valuable insights for subsequent phases of our statistical analysis.



Prior to commencing machine learning training, I employed a heatmap to examine the interrelationships among various variables. By computing the correlation matrix, I obtained the values depicted in the heatmap, with darker hues denoting more pronounced correlations. The heatmap analysis unveiled a robust negative correlation between age and population, corroborating our earlier observations from the bar chart. While the correlations between age and other variables are not notably strong, I opted to retain all variables in the analysis, despite the potential suboptimal outcomes in regression exploration. This decision was influenced by the relatively limited number of variables available for consideration.

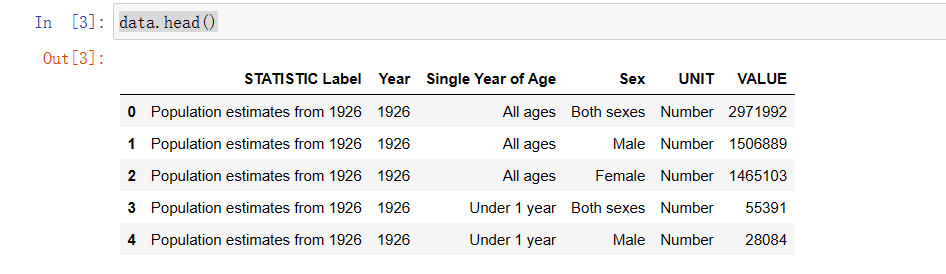
**Machine learning for Data Analytics**

This chapter will be divided into three parts for discussion:

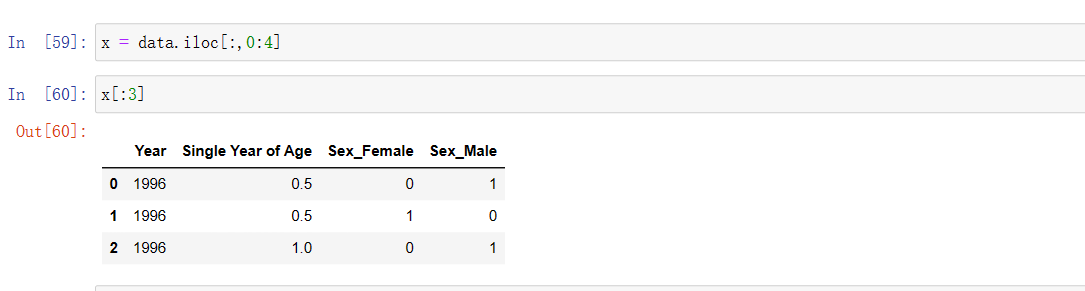
**1 Selecting the Right Framework and Machine Learning Technique for Data Science Projects**

In the real world, we often use the CRISP-DM framework, short for Cross-Industry Standard Process for Data Mining, to guide our work in solving various business and research problems. CRISP-DM provides a series of steps and activities to guide the entire process of data mining projects, including stages such as problem understanding, data understanding, data preparation, modeling, evaluation, and deployment. This study is also conducted based on the aforementioned framework. First, the data set is selected, followed by Exploratory Data Analysis (EDA). Next, data preprocessing is performed, and statistical hypothesis testing or machine learning models are built. Finally, the results are evaluated. Since this is not a real project, there is no deployment phase involved.

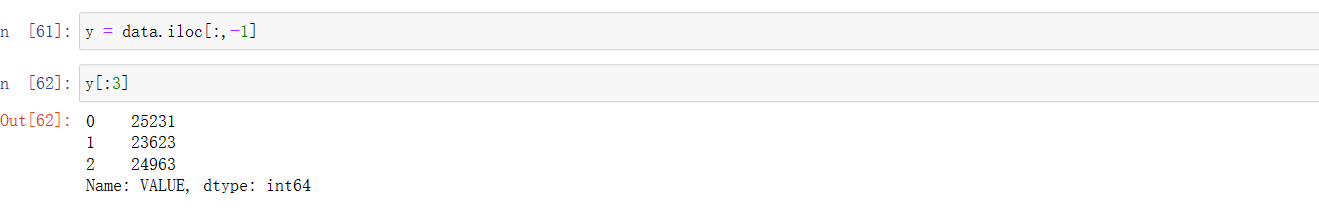
For the dataset "Population estimates from 1926" that we have selected.



We decided to use supervised learning for regression analysis right from the beginning because, based on the variables present in the dataset, it was easy to envision conducting research by finding relationships between population count and other variables. Since we have population count information, we could set it as the target variable. I did attempt to use this dataset for classification as well, but due to the limited number of variables and the lack of clear correlations between them, conducting research from a classification or clustering perspective would have been quite challenging, and the results would likely not have been satisfactory. Therefore, we ultimately chose supervised learning for regression analysis.



This is the traing set



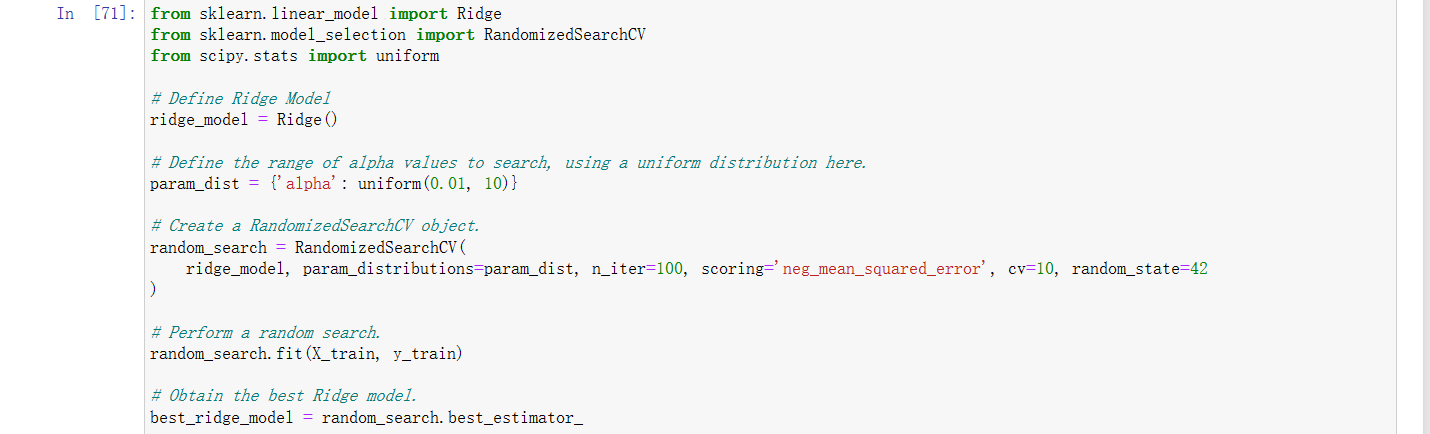
This is the test set

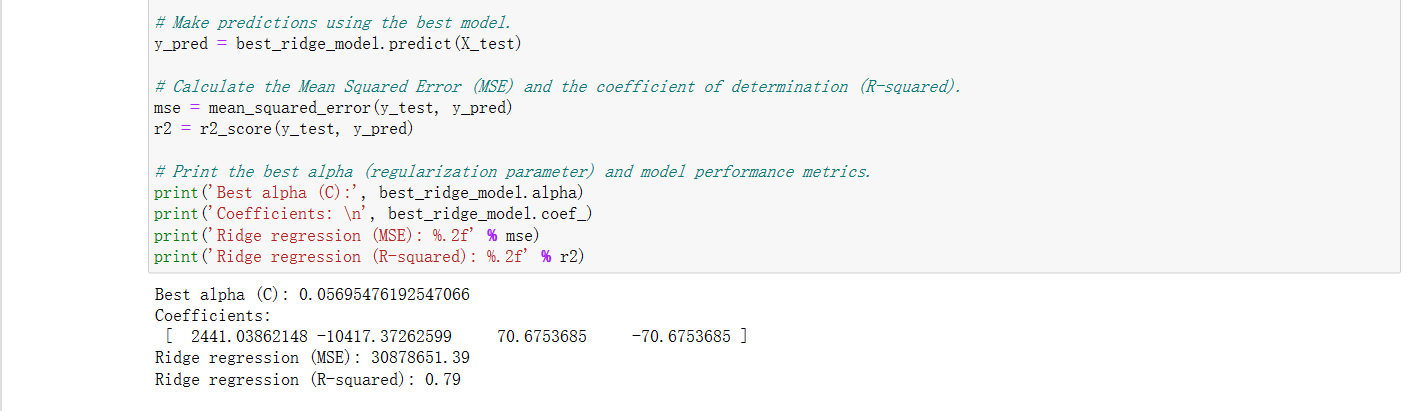
We will use multiple machine learning models to attempt regression analysis on the population count using "Year," "Single Year of Age," and "Sex" as variables.

**2 Optimizing Machine Learning Models through Hyperparameter Tuning**

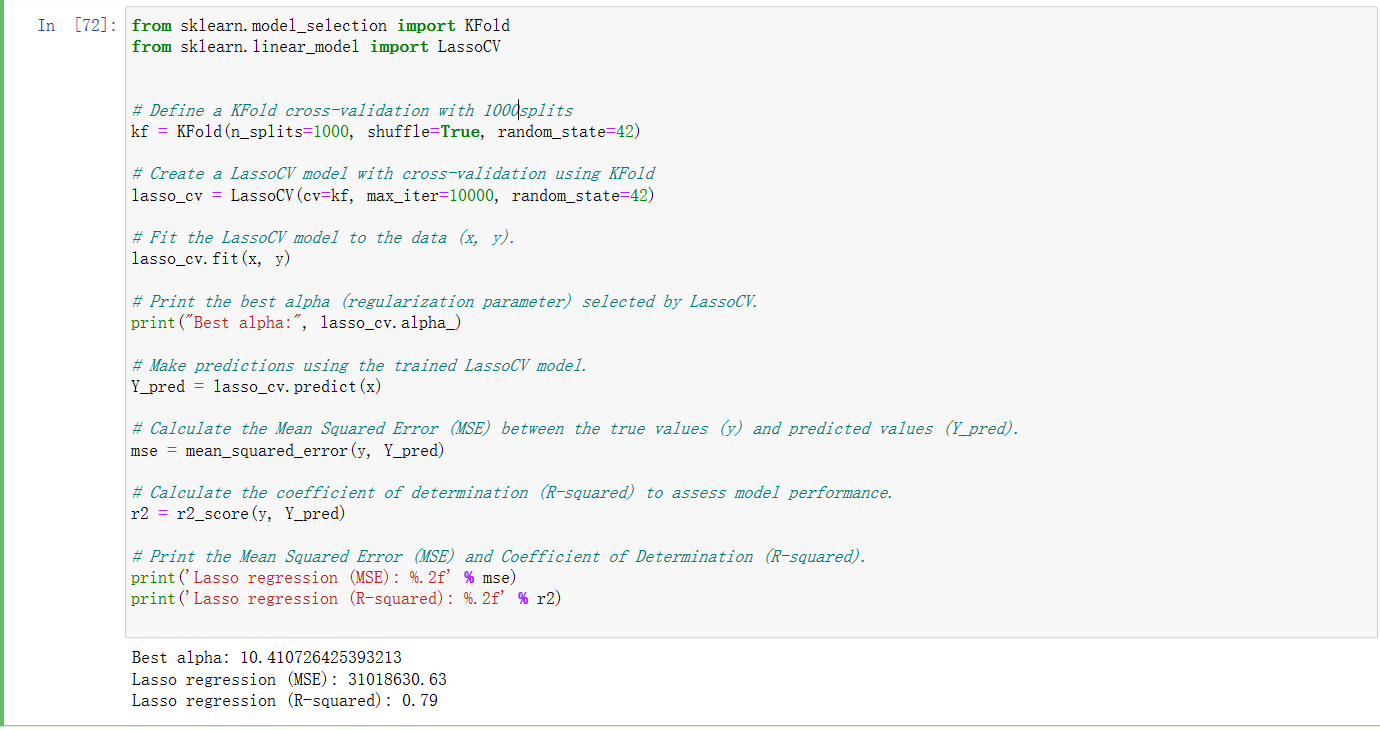
In this section, we will perform regression analysis using the prepared data with four different machine learning models and explore

hyperparameters using various methods.

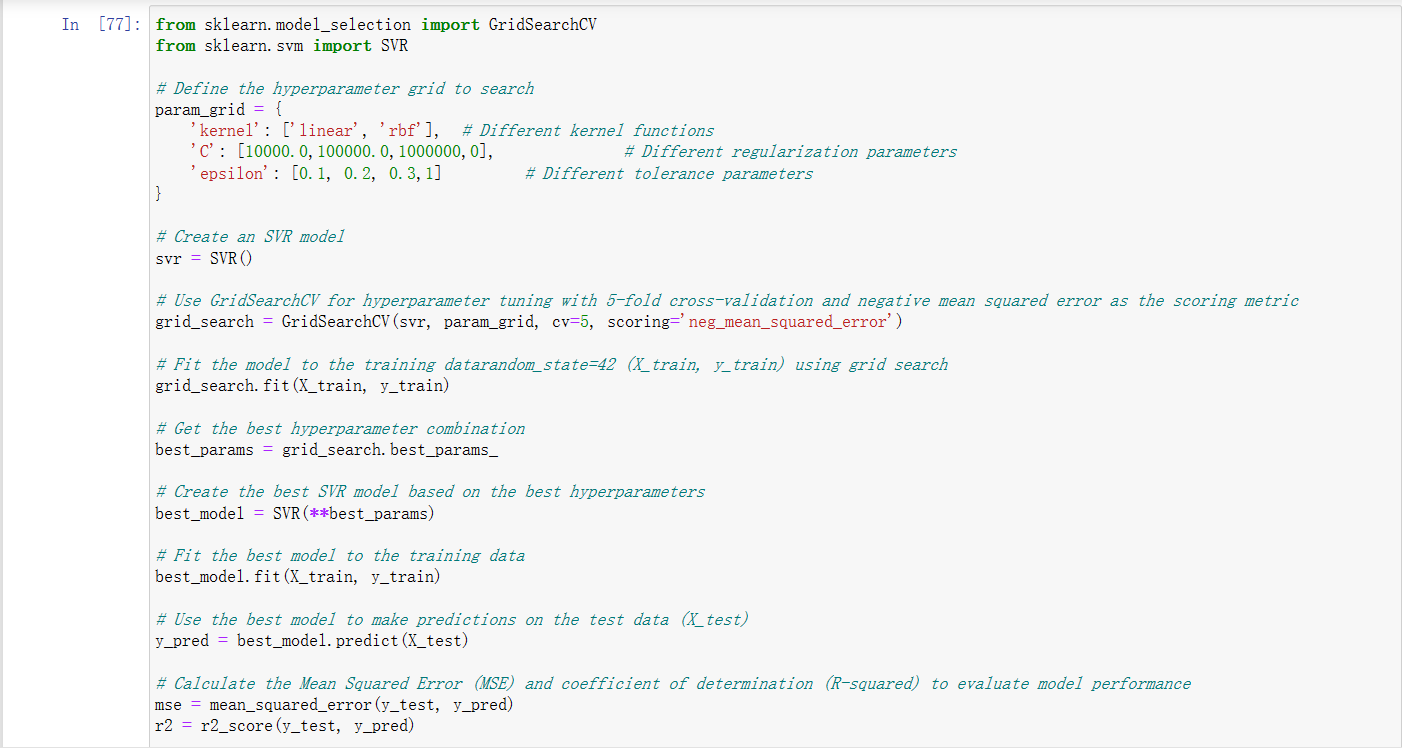


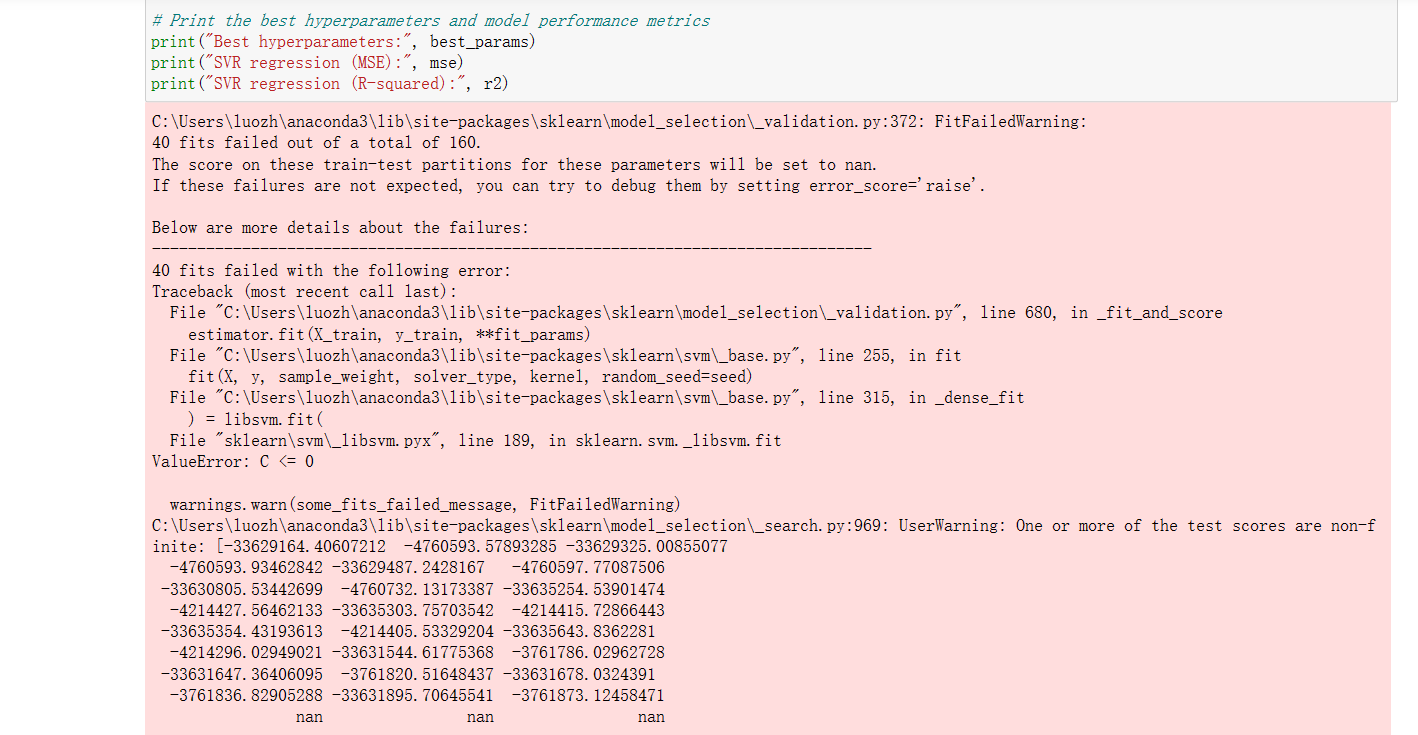


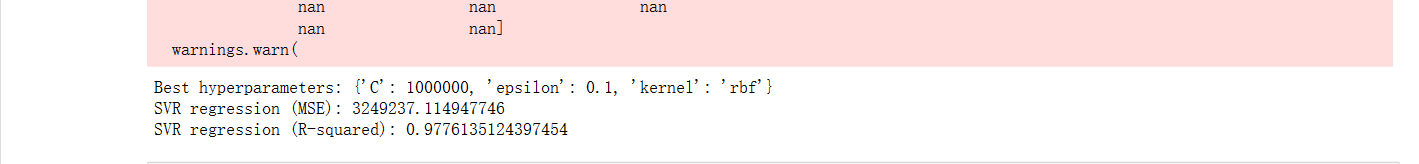
We first performed regression analysis using the Ridge model. For hyperparameter tuning, we utilized the RandomizedSearchCV method, which randomly selects a C value from a predefined range of 0.01 to 10 following a uniform distribution. We fitted the model to find the optimal C value and employed cross-validation to enhance the model's generalization and stability. Finally, we used the test set for predictions and evaluated the model's performance using Mean Squared Error (MSE) and R-squared (R^2) as two metrics. The obtained results were MSE of 30,878,651 and an R^2 of 0.79. The best parameter C value found was 0.057.



Next, we conducted regression analysis using the Lasso model. This is a somewhat unique model, and the LassoCV function automatically determines the range of alpha values and performs cross-validation within that range to find the best alpha value. We don't need to specify the alpha range, but we only need to set the number of cross-validation folds to 1000. In the end, we found the best alpha value to be 10.41, and the model's MSE and R-squared values were 31,018,630.6 and 0.79, respectively.

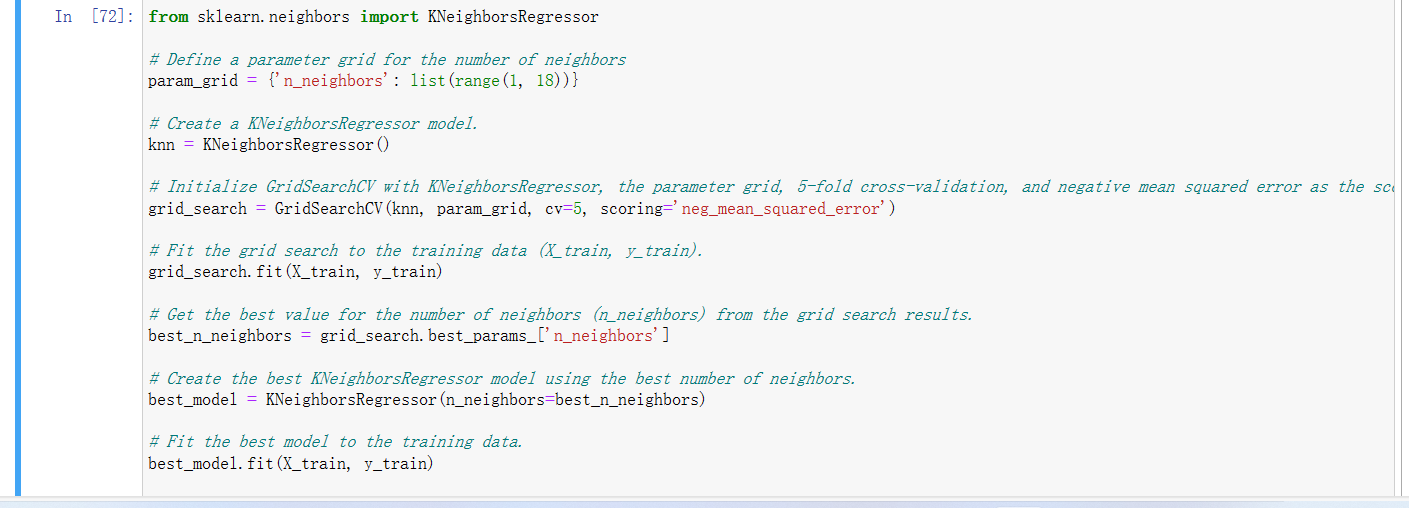


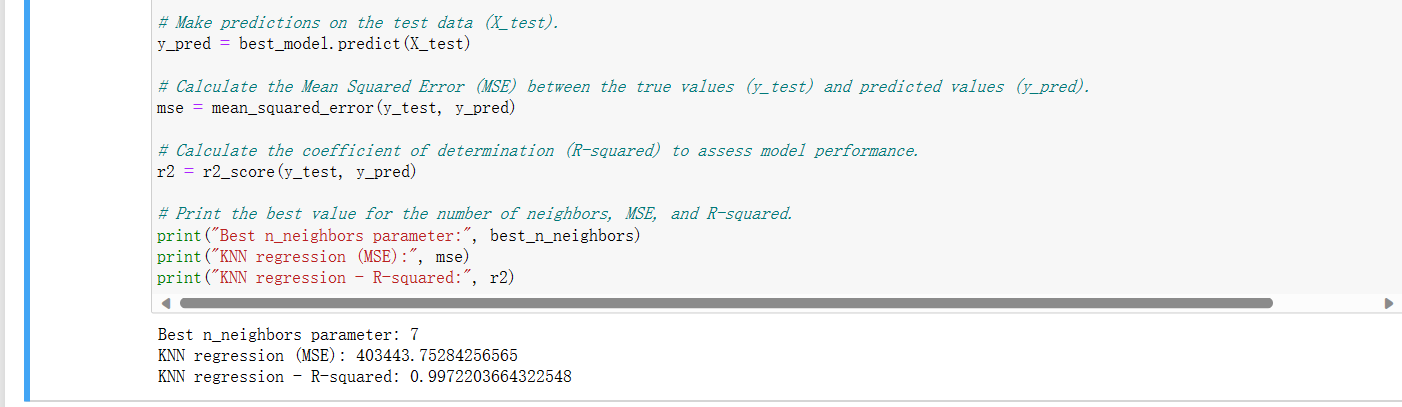




Then, we used Support Vector Machine (SVM) for regression analysis. This model has three hyperparameters: "kernel" for the kernel function, "C" for the regularization parameter, and "epsilon" for the epsilon parameter. Therefore, we conducted a grid search. During the search, we noticed that the results for "kernel" and "epsilon" tended to stabilize, while the value of C consistently approached the maximum boundary. We increased the value of C, which meant that the model's classification margin became smaller, but the model's performance continued to improve. We eventually raised C to 1,000,000. The computation demands almost exceeded my computer's capabilities, but the R-squared value remained stable at around 0.975. It became challenging to further enhance performance, and the MSE value was 3,249,237, which is already a good result.

Although increasing C further might yield better results, it would also carry the risk of overfitting the model.



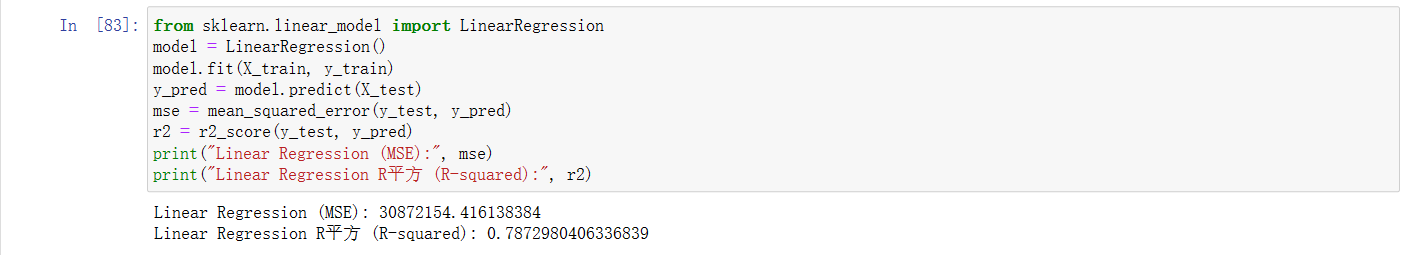


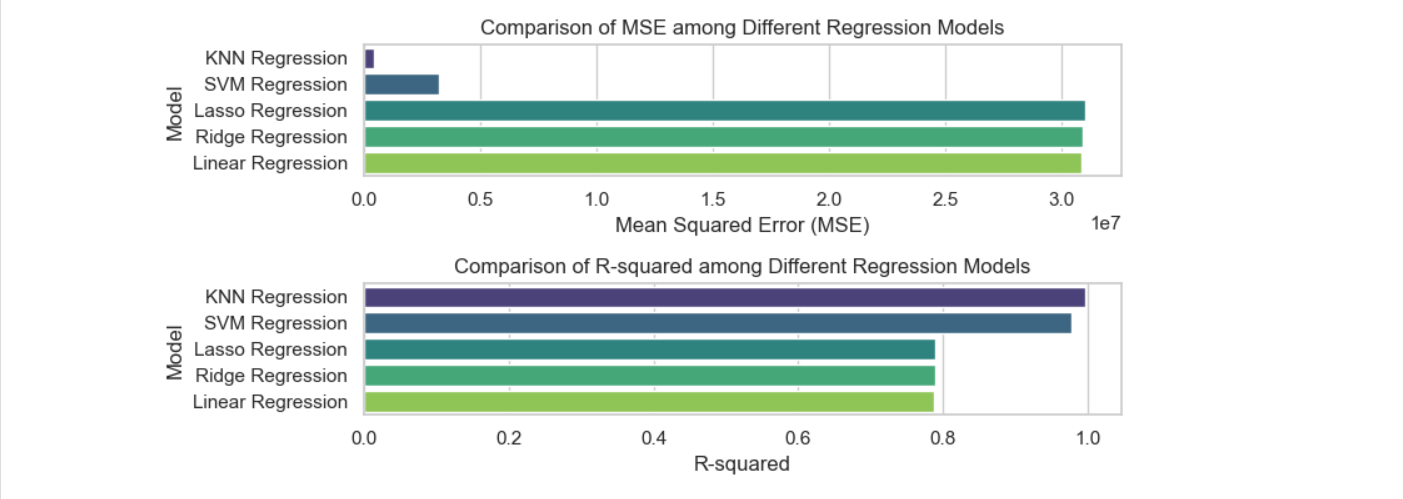
In the end, we used K-Nearest Neighbors (KNN) for regression analysis. Since KNN has only one hyperparameter, this example was well-suited for using GridSearchCV to find the best hyperparameter. We set the parameter "k" in the range from 1 to 17. The final results were excellent, with an MSE of 403,443 and an R-squared (R^2) value of 0.997. In this dataset's context, the model fits extremely well. The best parameter K found by the model was 7.3

**3 Comparison of Machine Learning Models（Regarding requirements 3 and 4）**

We also conducted analysis using a linear regression model:

Because the linear regression model is relatively simple, the regression results were not very ideal, with MSE and R-squared (R^2) values of 30,872,154 and 0.787, respectively. I created a bar chart to visualize the results of the five models for easier comparison:





From the chart, it's evident that KNN and SVM perform well, with lower MSE and higher R^2 values. Particularly, KNN nearly perfectly captures the complex relationships between variables. However, SVM's excellent performance is based on a large input regularization parameter, which comes with the risk of overfitting.

In contrast, Lasso, Ridge, and Linear Regression show similar and less favorable results, indicating that they did not capture the data's information well and may not be particularly suitable for this dataset.

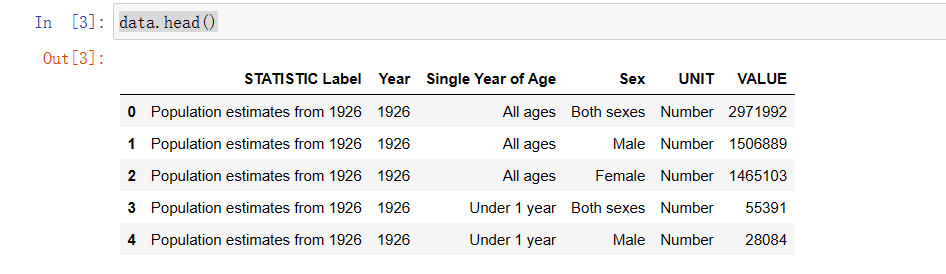
In summary, I have utilized nearly all possible machine learning models for regression analysis, employing supervised learning methods. In the end, I found that the K-Nearest Neighbors (KNN) model performed the best, along with its associated hyperparameters. I believe that this model will also perform well in practical population analysis tasks.

**Python Programming**

This chapter is primarily divided into two parts to introduce:

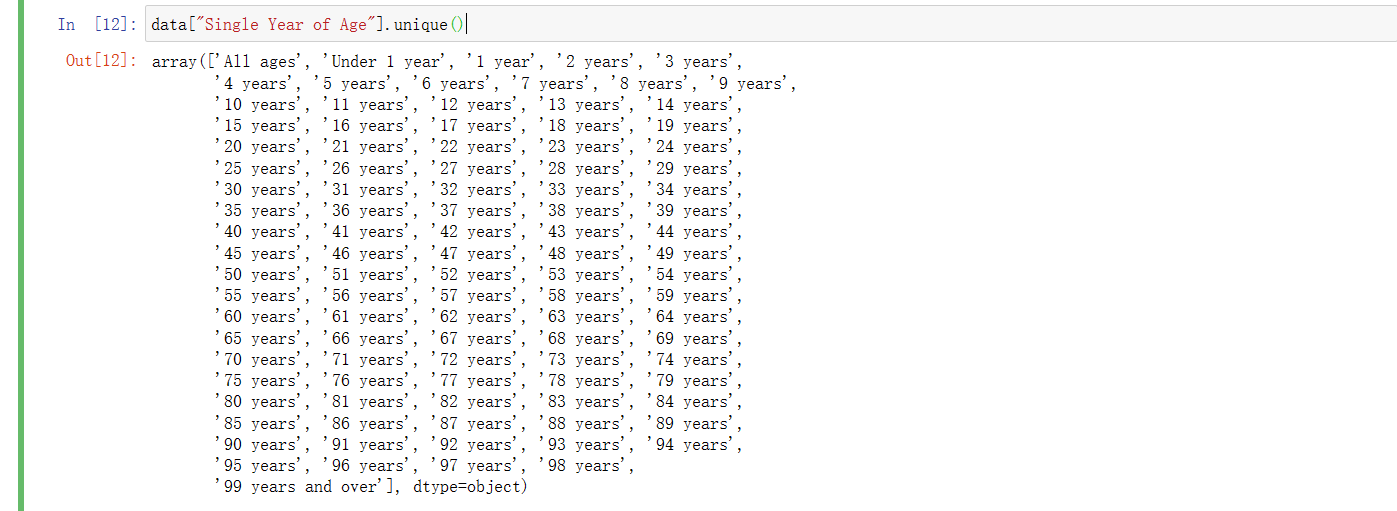
**1 Explain the reasons for choosing some code and provide explanations**

Throughout the project, we have made extensive use of built-in Python methods. These methods played a crucial role in the initial data exploration and cleaning phases.

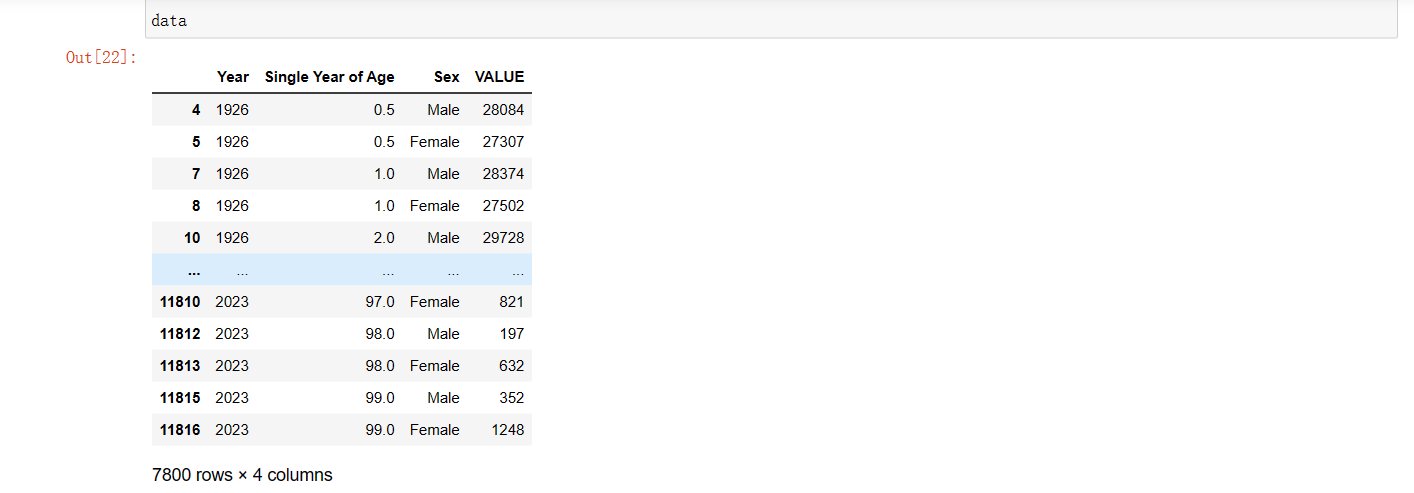


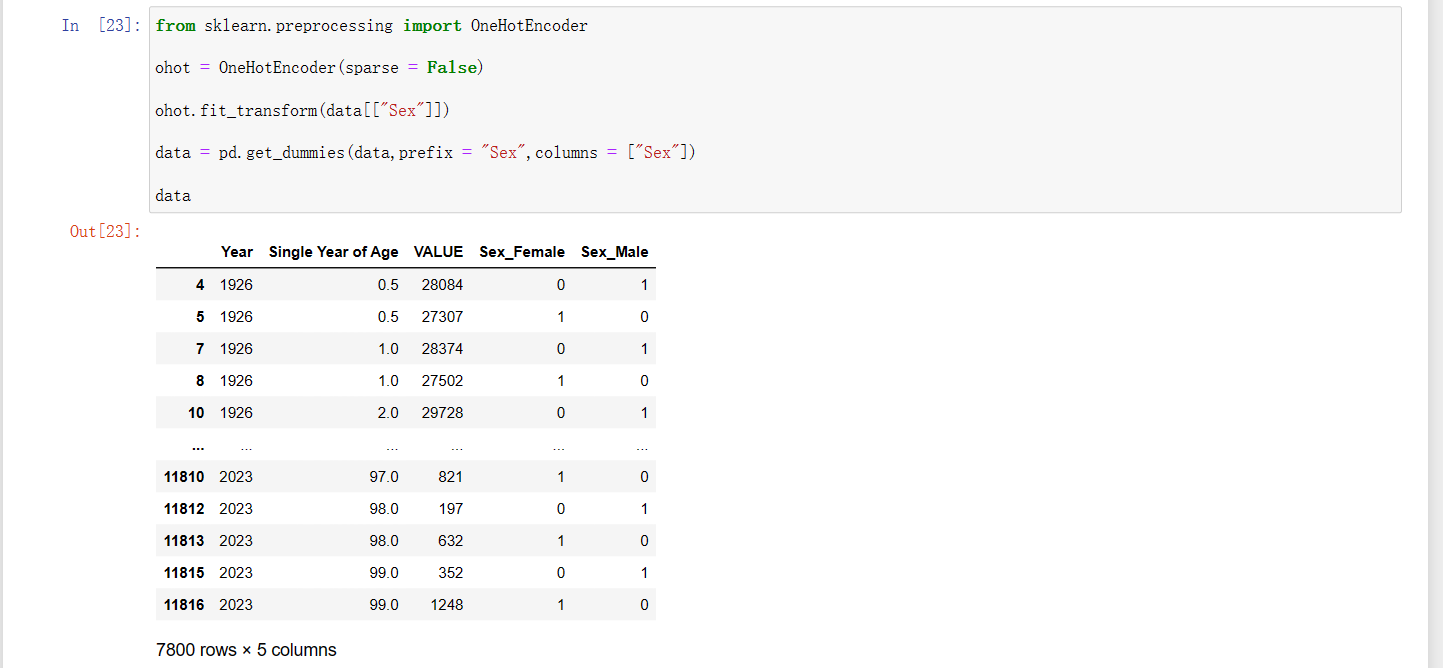
For the dataset "Population estimates from 1926" that we have selected:

The **head()** method allows for a straightforward understanding of the dataset's variables and the values of its features.



The **unique()** method is valuable for quickly understanding all possible values within a variable, and it plays a crucial role in later data cleaning tasks. These methods have significant importance. I have also used a wide range of Python libraries, such as Pandas, which offers various data structures and data analysis tools.

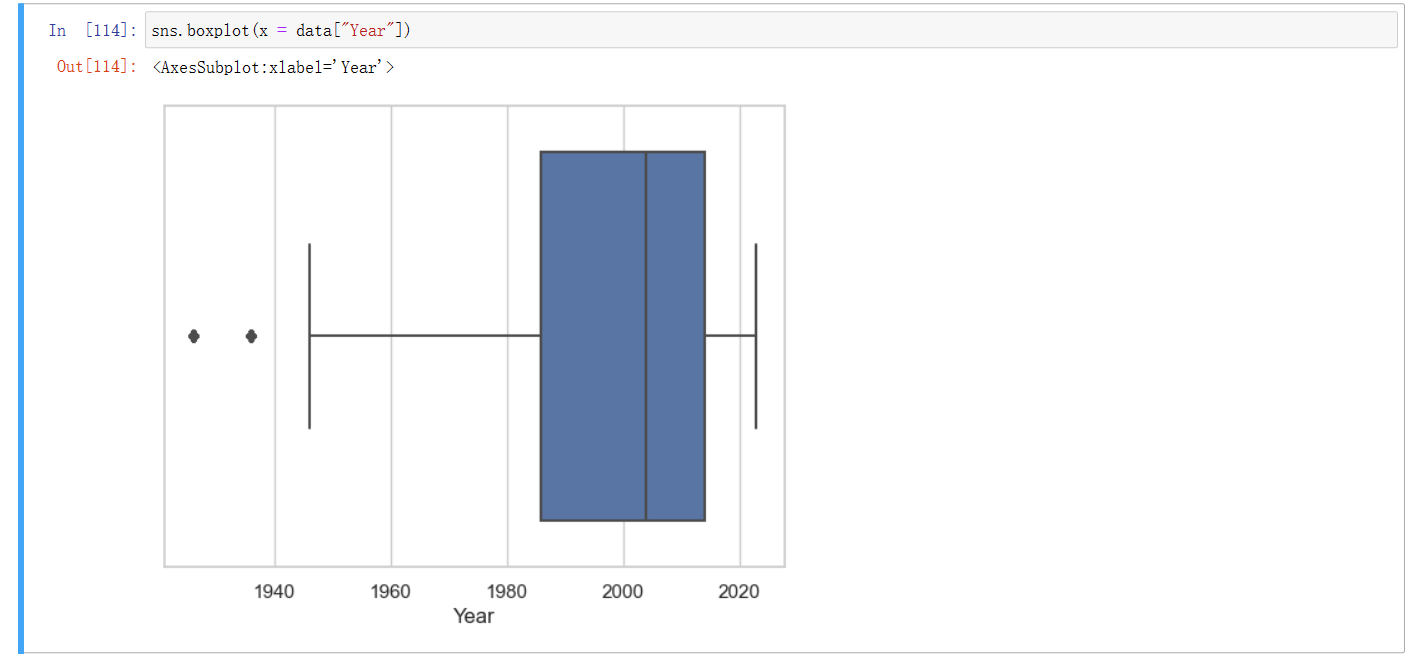


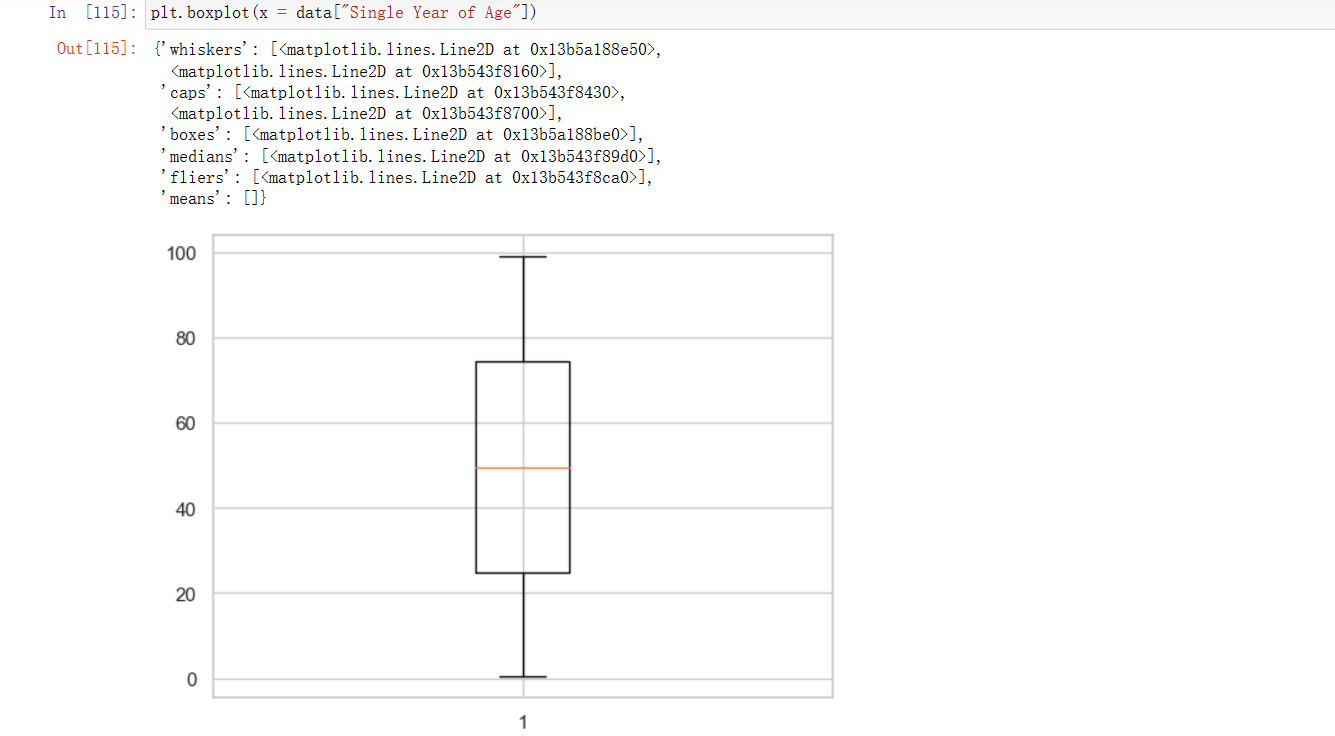


For instance, here we used the **pd.get\_dummies()** method, which conveniently transforms each category within a feature into a separate column.



Here, **pd.cut()** has played a crucial role by helping me group ages in a specified way, preparing the data for the subsequent visualization. Additionally, the use of Python's built-in method **groupby().sum()** allowed me to quickly group and sum population counts for different age groups. For visualization, I just used two libraries: **Seaborn** and **Matplotlib**.

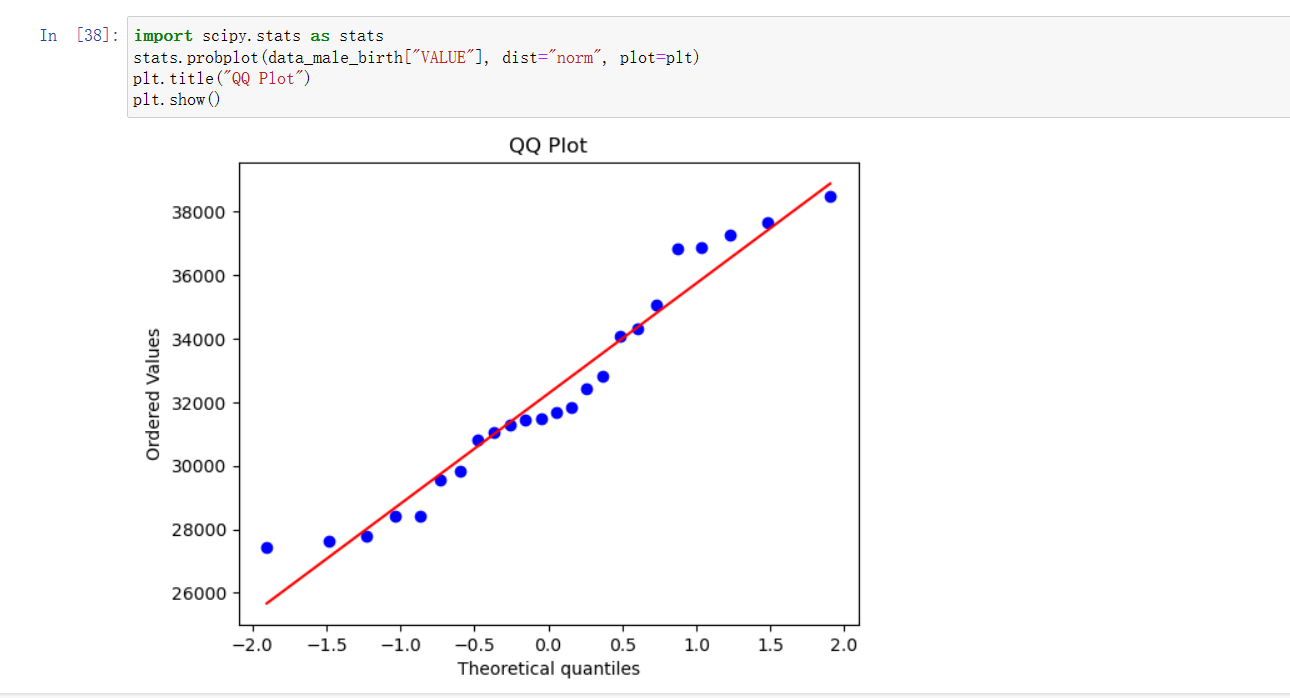




By comparing these two boxplots, we can observe that **Seaborn**'s statistical plot appears more aesthetically pleasing, as it typically comes with more attractive default styles.

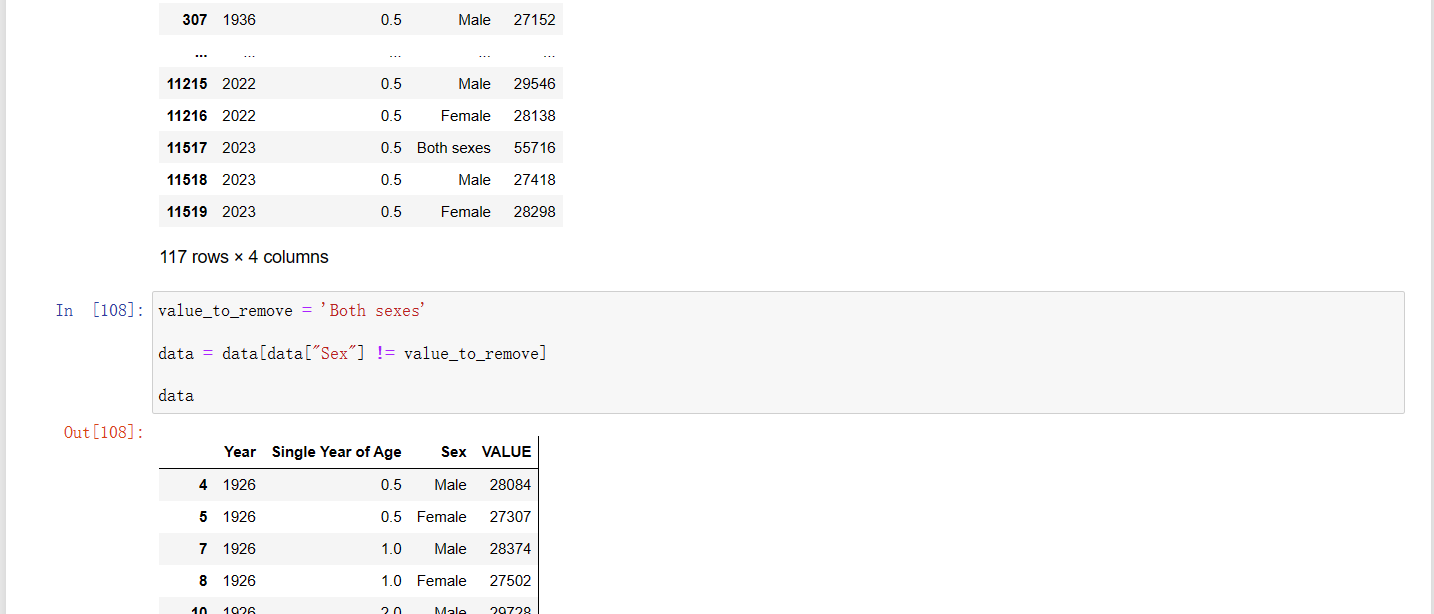
Sometimes, combining **Matplotlib** and **Seaborn** can result in exceptionally attractive plots, as **Seaborn** enhances **Matplotlib**'s capabilities by providing advanced interfaces and built-in themes. Overall, **Matplotlib** offers greater customization and flexibility, while **Seaborn** is often able to quickly create more visually appealing statistical plots. Additionally, we mentioned using the **stats** module from **SciPy** for statistical analysis, indicating a well-rounded approach to data analysis.

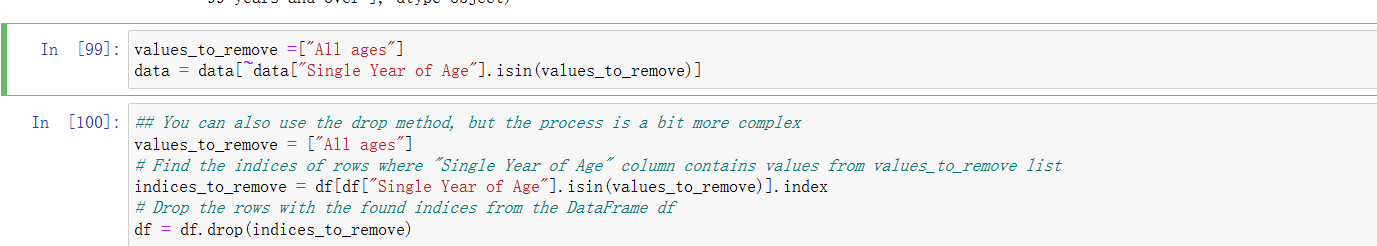


For instance, I mentioned the use of the **stats.probplot()** function in conjunction with **Matplotlib** to generate **QQ plots**. This illustrates my utilization of multiple tools for conducting comprehensive data analysis. Furthermore, my extensive reliance on the **scikit-learn (sklearn)** library is noteworthy. It is undeniably one of the most widely employed libraries for machine learning endeavors, providing a comprehensive repertoire of machine learning models and tools. Within **sklearn**, I have access to a diverse array of machine learning models to tackle various tasks.



I used a total of 5 machine learning algorithms, and their overall analysis processes are quite similar. Apart from some additional requirements that necessitated adding code, such as using grid search to find parameters, almost all machine learning models can be completed with just three lines of code. These steps include model creation, importing the training dataset, and making predictions. Because Scikit-Learn is highly integrated, it is very convenient to use.



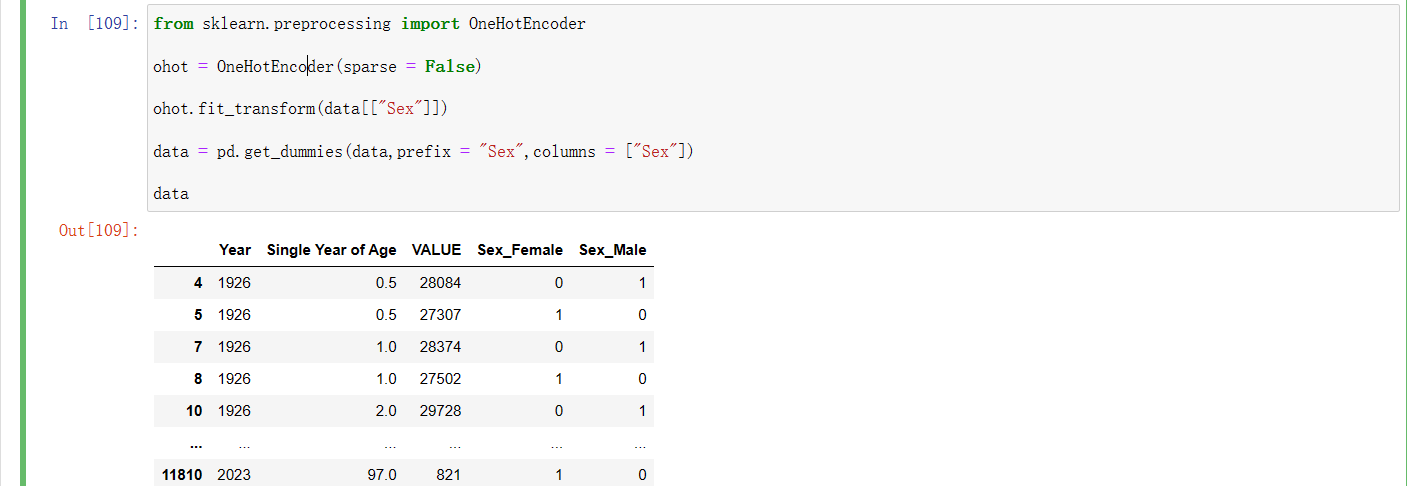


Finally, I'd like to discuss a topic. When it comes to removing rows containing a specific value in a column, I found three different methods. The first method involves using boolean indexing, directly checking if the values in a specific column are equal to the given value, and then selecting the rows where this condition is not met. This method can handle only a single value. On the other hand, the second and third methods involve creating a list of values to be removed and then using the "**isin**" method to check if the values in the specific column are in that list. However, they use different approaches for removal. The second method uses the negation operator "**~**" and boolean indexing, while the third method uses the "**drop()**" method. Personally, I lean towards using the second method because it is more versatile and concise in its functionality.

**2 Programming Paradigm Integration in Project Development**

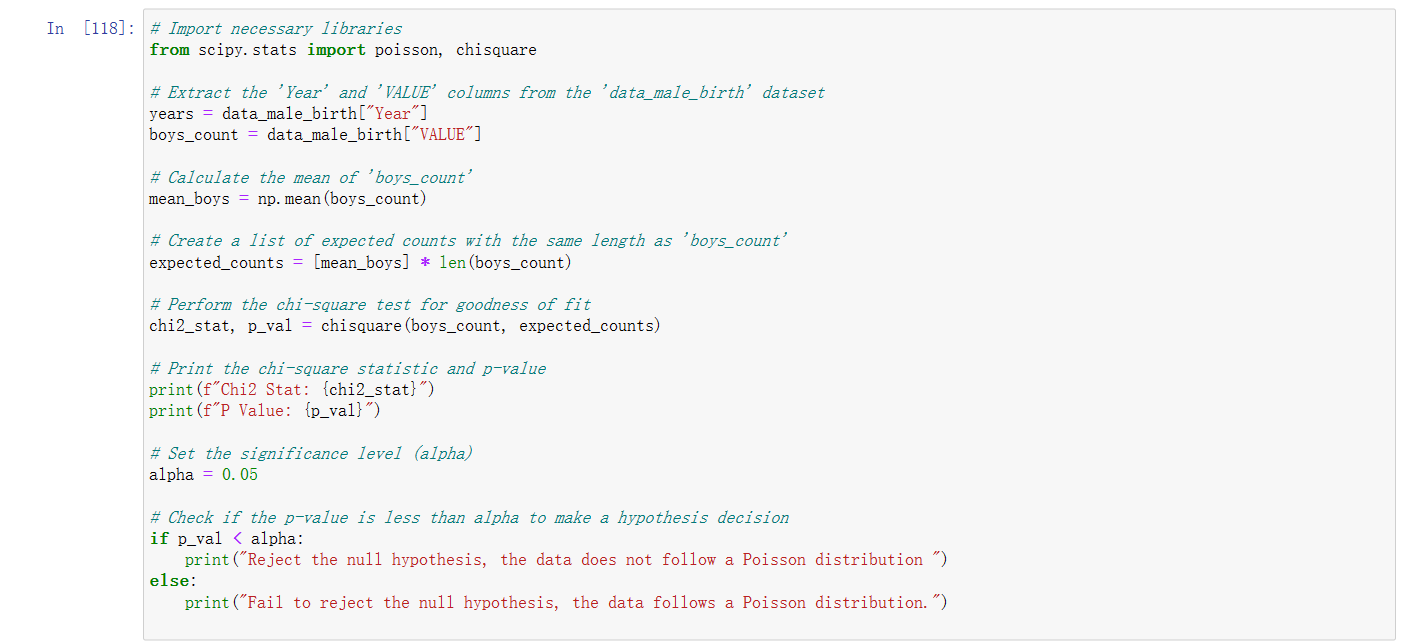
In terms of programming styles, there are primarily five main programming paradigms: Imperative Programming, Procedural Programming, Functional Programming, Declarative Programming, and Object-Oriented Programming.

Let's delve into the discussion of the programming style for the current project. In the early stages of this project, a significant amount of Imperative Programming was used. For example:

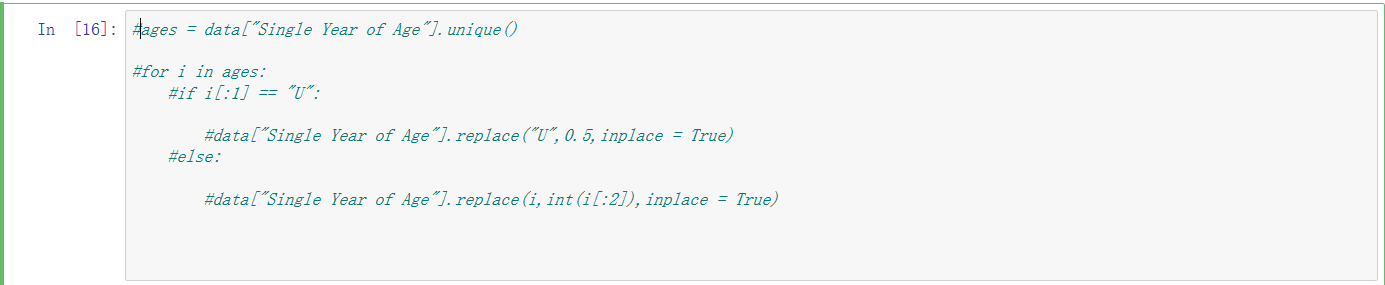




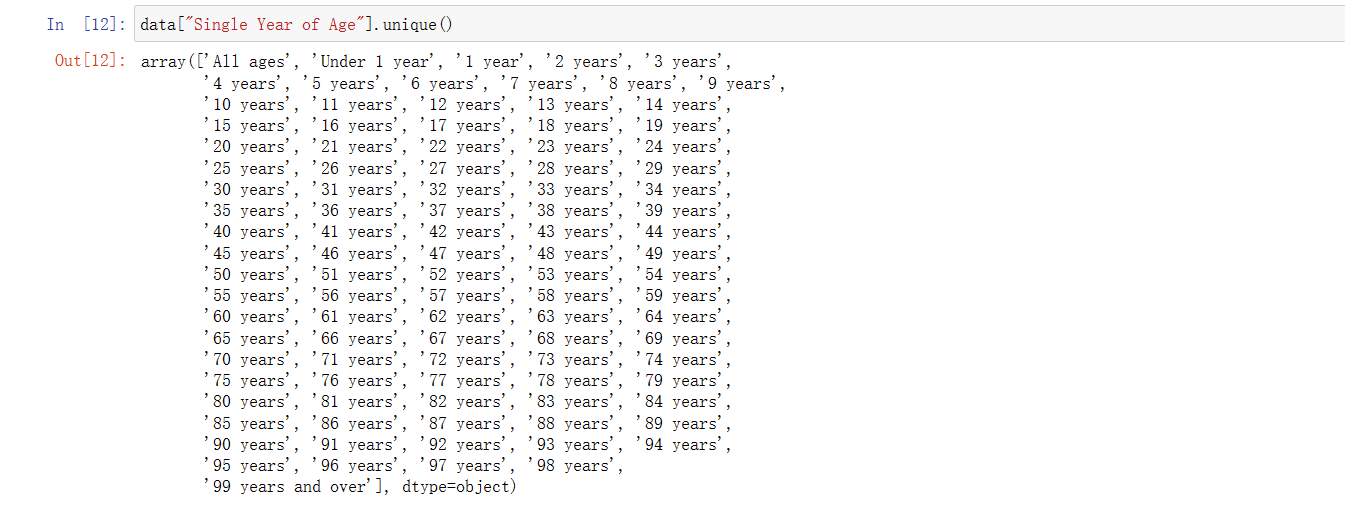
Both of these examples make use of Imperative Programming. Its advantages include strong intuitiveness, readability, and ease of debugging. However, it also has some drawbacks, such as making code appear more complex and having a relatively loose structure, which can result in poor maintainability.

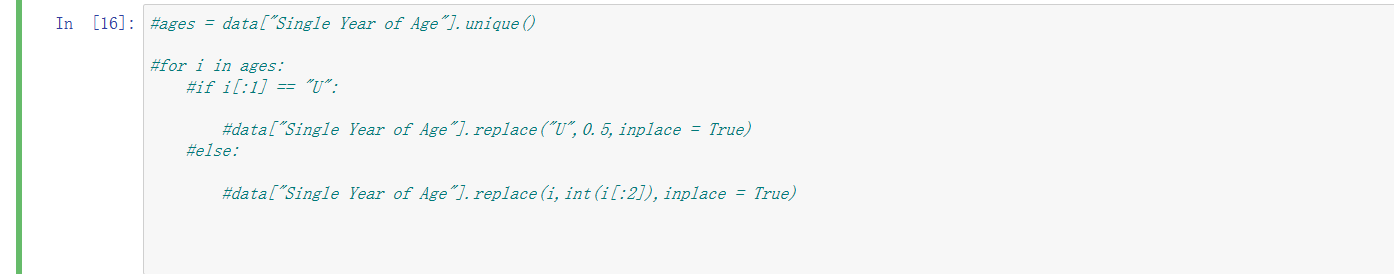


This piece of code falls under the category of Procedural Programming. It exhibits several evident characteristics, such as defining multiple functions like "**chisquare**" and "**print**" that perform specific tasks, promoting code modularity. There is also explicit control flow using **"if"** statements. In the code, "years" and "boys\_count" are shared and used in multiple sections. The sequential execution starts from importing modules and proceeds through each statement. Procedural Programming shares similarities with Imperative Programming in terms of good intuitiveness and readability. However, as a project grows, code can become more complex and challenging to maintain.

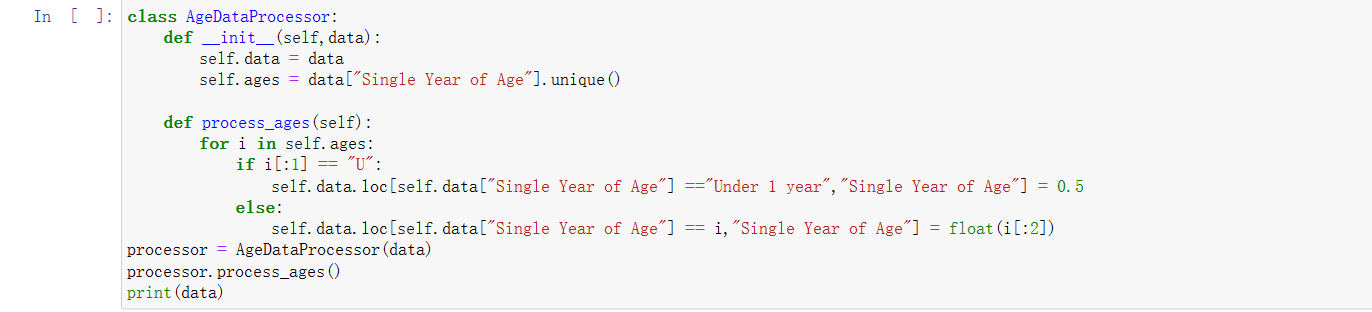


Finally, I'd like to discuss this piece of code.





In the beginning, when I wanted to convert all values in the "Single year of age" variable to floating-point numbers, I initially used a **for loop** and the replace method. While the code was functional, its structure was relatively loose. I was looking for a better coding approach to make the code more modular. So, I decided to try Object-Oriented Programming (OOP).



I defined an AgeDataProcessor class and then created a process\_ages method for handling age data. In this method, I processed each age value by iterating through the "ages" list. This was an attempt at Object-Oriented Programming (OOP). After this modification, the code became significantly more modular and maintainable, although it might introduce some additional computational overhead.

**Conclusion:**

In conclusion, this study has provided valuable insights into the population estimates of Ireland spanning from 1926 to 2023. Through rigorous statistical analysis, data preprocessing, and machine learning techniques, we have uncovered significant patterns and relationships within the dataset.

Our findings indicate a clear negative correlation between age and population count, aligning with the expected demographic trends in a evolving society. Moreover, the examination of birth rates has revealed that the distribution of male births does not conform to a binomial distribution with p=0.5, suggesting disparities between male and female birth rates in Ireland's history.

Machine learning models have played a pivotal role in understanding and predicting population counts. Among the models explored, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) emerged as the most promising, with KNN exhibiting exceptional accuracy. Furthermore, our use of Python programming, combining various libraries and programming paradigms, has showcased the flexibility and versatility of the language in data analysis and machine learning projects. In essence, this study contributes to a comprehensive understanding of population dynamics in Ireland and underscores the power of data-driven insights in addressing complex societal trends. It serves as a testament to the importance of interdisciplinary approaches in data science and offers a foundation for further research and policy implications.

**References:**

Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media

Müller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.

Harrington, P. (2012). Machine Learning in Action. Manning Publications.