Programming Assignment III

Shaun Pritchard & Isammel Lopez

Florida Atlantic University

CAP 5625

November 22, 2021

M.DeGiorgio

Programming Assignment III

The scope of this project was to build a working logistic (multinomial) ridge regression that could perform gradient descent and cross validation from scratch and with libraries (for comparison). For this assignment III we collaborated to finalize the assignment deliverables. This resulted in the completion of one method for creating the assignment III algorithms from scratch and one implementation for a working model based on python libraries. To complete both instances of the project, we implemented the following algorithm.

Assignment 1 - Algorithm 1 (vectorized):

Step 1. Choose learning rate α and fix tuning parameter λ

Step 2. Generate N×(p+1) augmented design matrix where column k has been centered and standardized such that feature k, k = 1,2,...,p, has mean zero and standard deviation one, and generate N×K indicator response matrix .

Step 3. Initialize the (p+1)×K-dimensional parameter matrix to all zeros, so that initially each class has the same probability.

Step 4. Create temporary (p+1)×K-dimensional parameter matrix

Step 5. Compute N×K unnormalized class probability matrix as where uik = exp($\beta 0k$ +$\sum xij\beta jk$)

Step 6. Compute N×K normalized class probability matrix

Step 7. For each j, j=0,1,...,p, and k, k =1,2,...,K, find the next value for parameter j for class k

Step 8. Update the parameter matrix as $\mathbf{B}$=$\mathbf{B}$temp

Step 9. Repeat Steps 5 to 8 for 10,000 iterations

Step 10. Set the last updated parameter matrix as $\mathbf{B}$

Deliverables:

This project required seven deliverables produced from the algorithms we created using the python programming language. The deliverables are as follows.

- **Deliverable 1:** Illustrate the effect of the tuning parameter on the inferred ridge regression coefficients by generating five plots
- **Deliverable 2:** Illustrate the effect of the tuning parameter on the cross-validation error by generating a plot with the y-axis as CV(5) error, and the x-axis the corresponding log-scaled tuning parameter value log10(λ) that generated the particular CV(5) error.
- **Deliverable 3:** Indicate the value of λ value that generated the smallest CV(5) error.
- **Deliverable 4:** Given the optimal λ, retrain your model on the entire dataset of N=183 observations to obtain an estimate of the $(p+1)\times K$ model parameter matrix as $\mathbf{B}$ and make predictions of the probability for each of the K =5 classes. Also, report all six values (probability for each of the K =5 classes and the most probable ancestry label) for all 111 test individuals.
- **Deliverable 5:** How do the class label probabilities differ for the Mexican and African American samples when compared to the class label probabilities for the unknown samples?
- **Deliverable 6:** Provide all the source code that you wrote from scratch to perform all analyses
- **Deliverable 7:** (extra credit): Implement the assignment using statistical or machine learning libraries in a language of your choice.

Instructions overview:

According to the deliverables listed, we have delivered two implementations of the assignment: one from scratch and one using libraries. Below you will find a summary of the assignment 3 instructions and deliverables.

From Scratch - demonstrates how a set of algorithms can be implemented from scratch in Python using only the numpy and pandas libraries.

With libraries - Demonstrates how assignment 3 is implemented using libraries like Sci-kit Learn.

Instructions:

For this assignment, Python and Google Colaboratory Jupyter notebooks were used. We created two separatTwo notebooks were created, one for the from scratch implementation and one using libraries. sound aFrom the notebook, the "Run All" command can be used to execute the algorithms for both instances.

All files are included in this repo and both implementations are accessible online for viewing and commenting at the following URLs:

From Scratch Google Colab implementation:

https://colab.research.google.com/drive/1w0O23DDN45NC96UMtNuBNqbyK_s3an8m?usp=sharing

With Libaries Google Colab Implementation:

https://colab.research.google.com/drive/1jsQiNbecYaXL0vqUE09P5EQeZx1EyEnx?usp=sharing

Note: For using standard python scripts on a local machine make sure the required imports are installed using pip, pip3, or anaconda. Also, rename the file-path if importing the dataset from another location other than the original implementation.

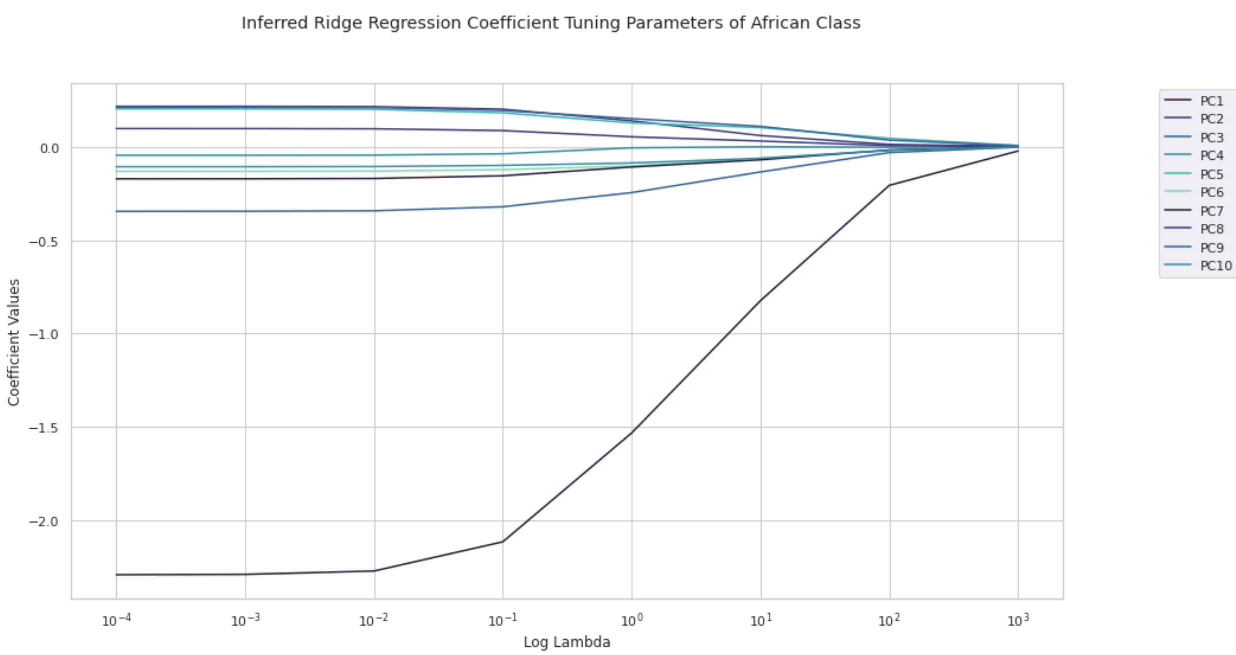
Files and directories:

- *Assignment 3 Instructions:* = instructions as per assignment for CAP 5625 Assignment 3
- *Dataset:* = contains the working training and test dataset for this project
- *From Scratch Implementation directory:* = PDF of code and results, ipynb file of code, and .py script file of code, and generated plots for deliverables specific to the code used with libraries only.
- *With Libaries Implementation directory :*= PDF of code and results, ipynb file of code, and .py script file of code, for deliverables specific to the code used with libraries only.
- *Instructions and comparison file:* = two files one in word and one in pdf format

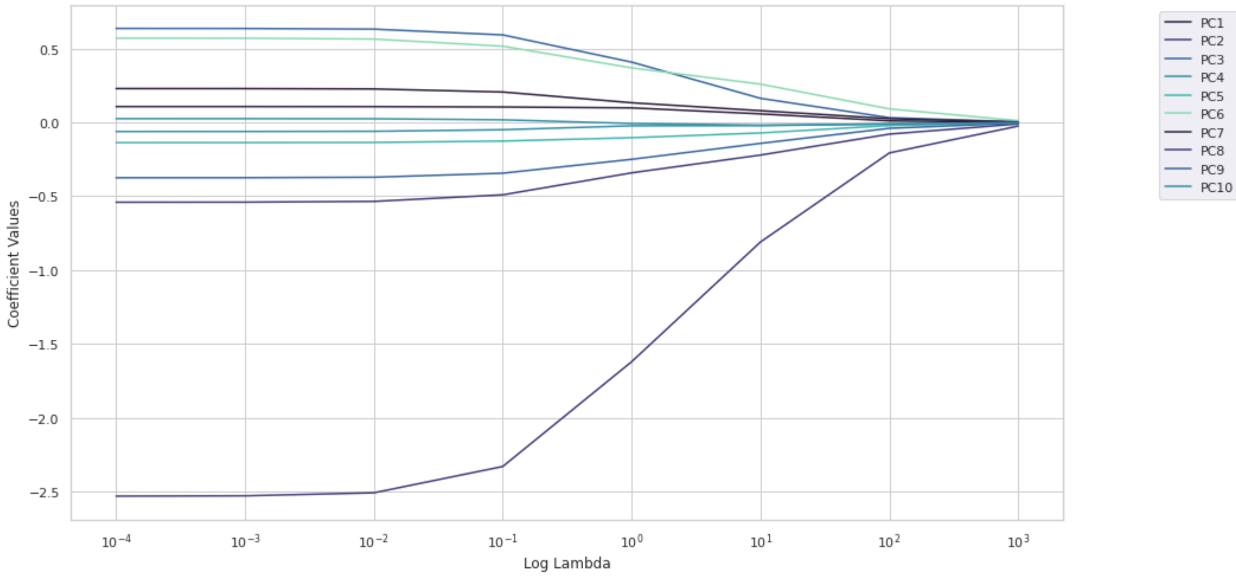
From scratch Logistic Multinomial Ridge Regression

We built a polarized logistic multinominal regression algorithm from scratch using Numpy and the Pandas libraries in Python. Using a batch gradient descent algorithm, we computed the initial algorithm. According to the output for the * tuning parameters, the results of these two algorithms were comparable:

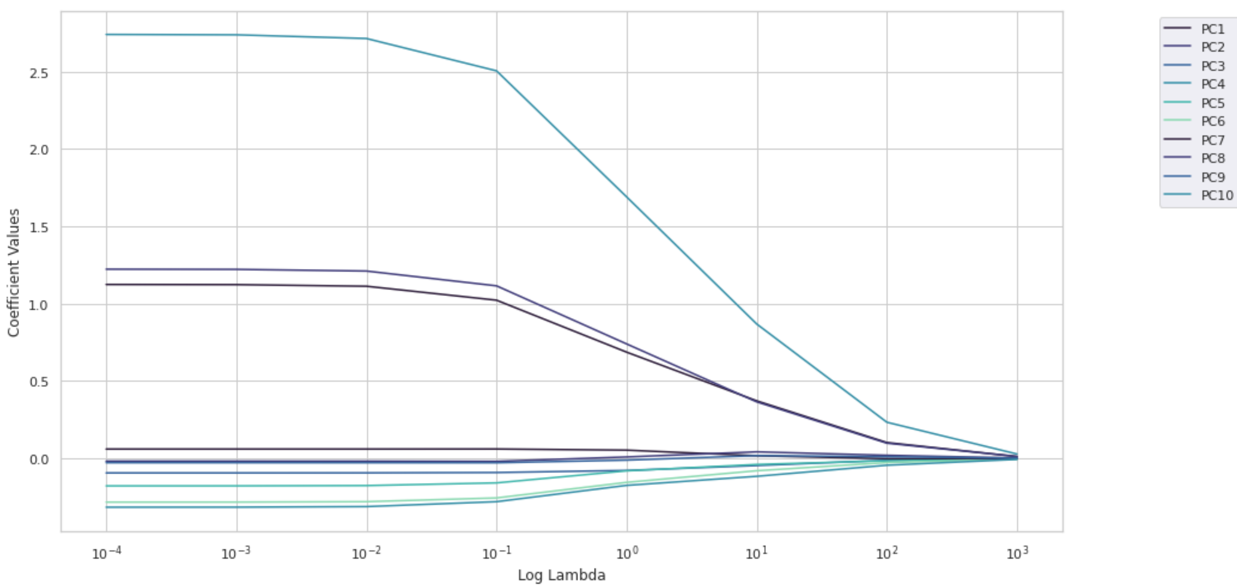
Deliverable 1 (5-plots):



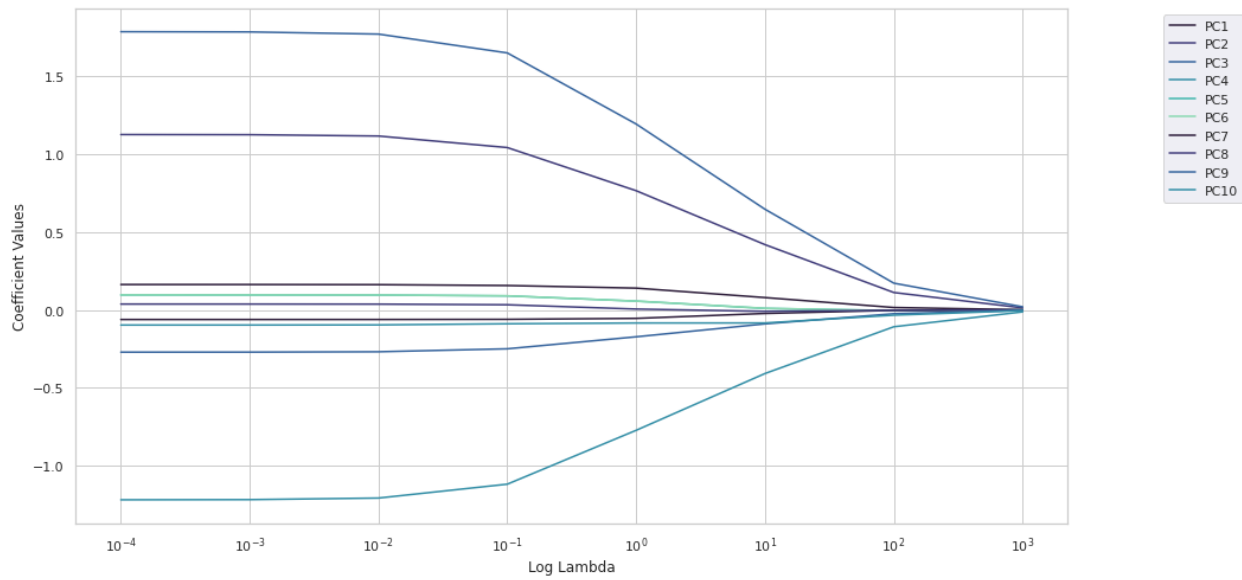
Inferred Ridge Regression Coefficient Tuning Parameters of European Class



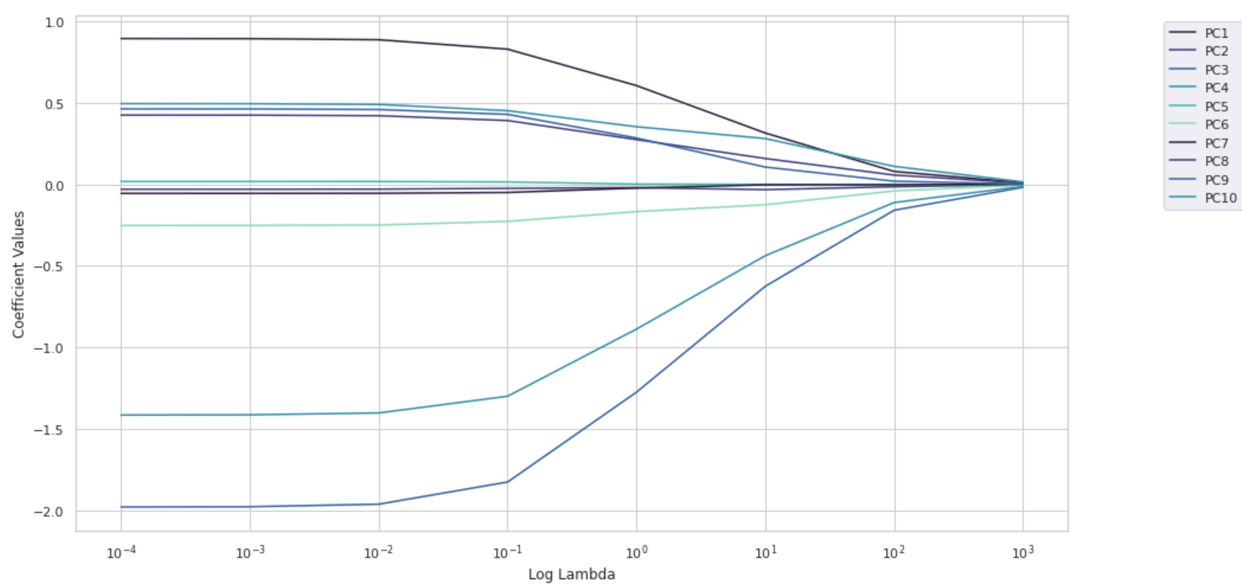
Inferred Ridge Regression Coefficient Tuning Parameters of EastAsian Class



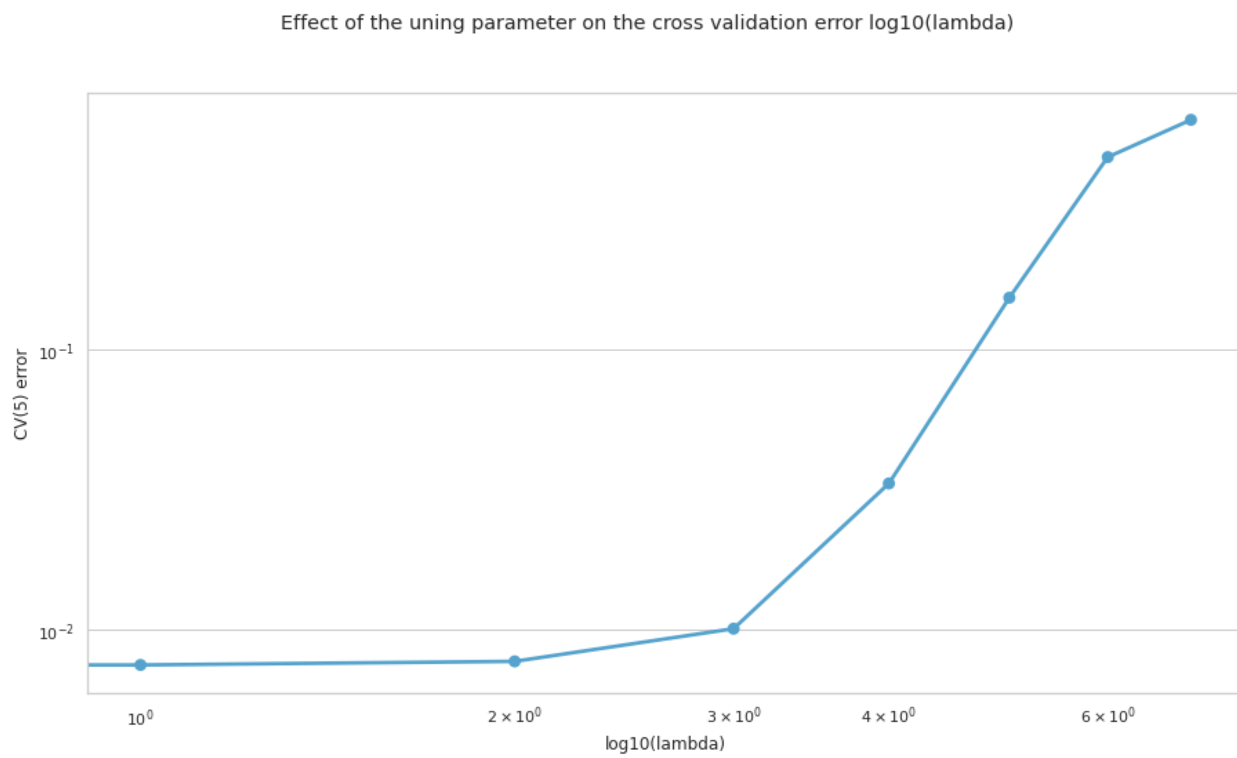
Inferred Ridge Regression Coefficient Tuning Parameters of Oceanian Class



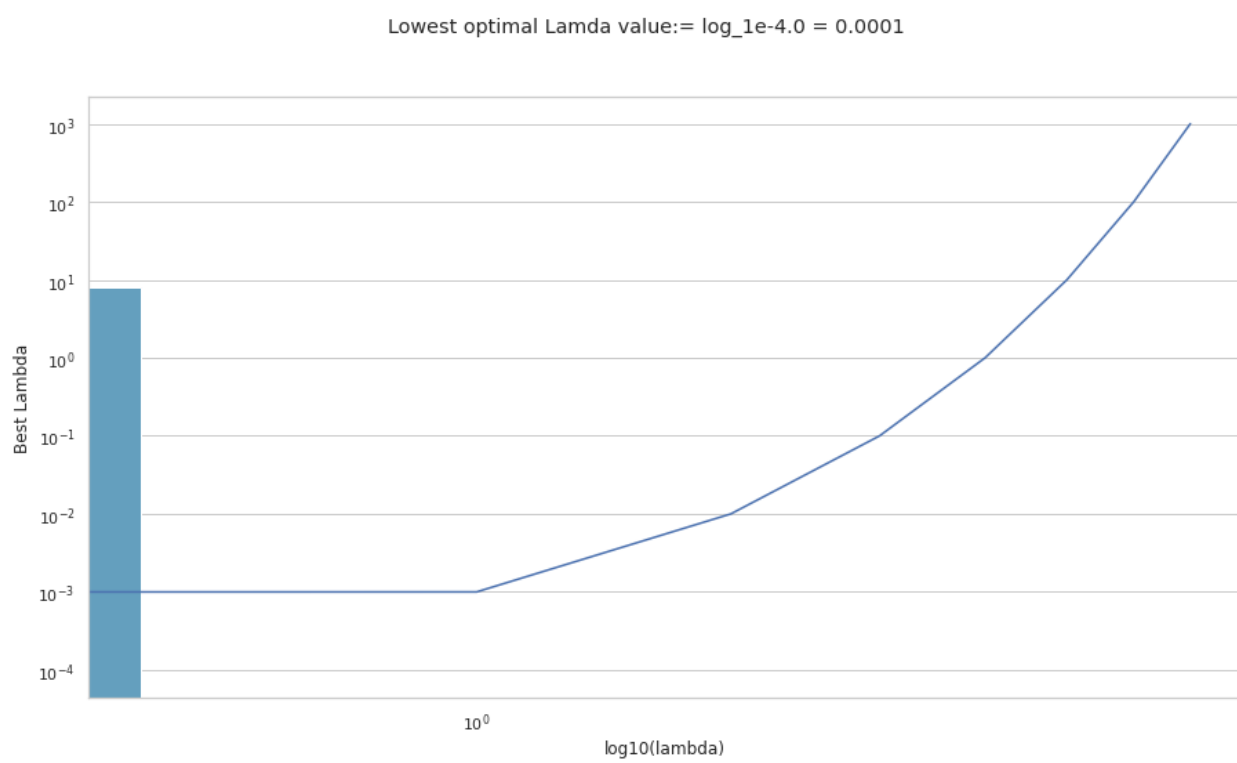
Inferred Ridge Regression Coefficient Tuning Parameters of NativeAmerican Class



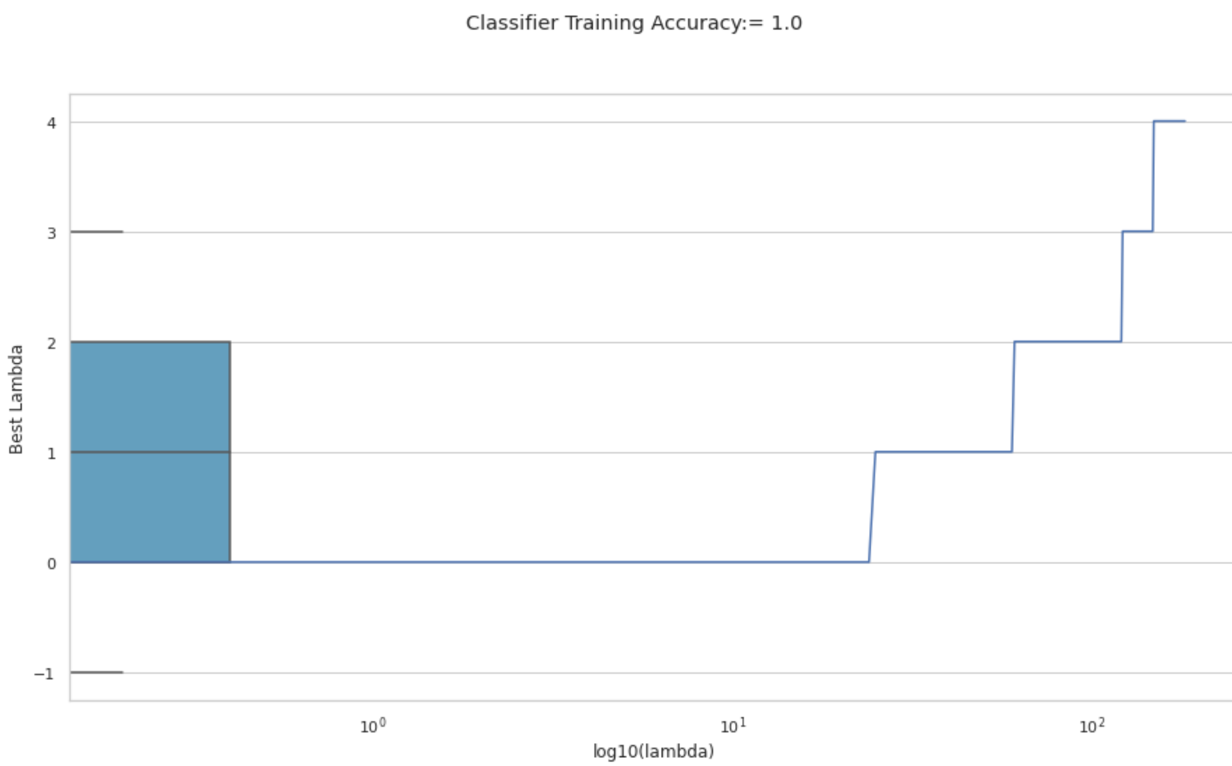
Deliverable 2:



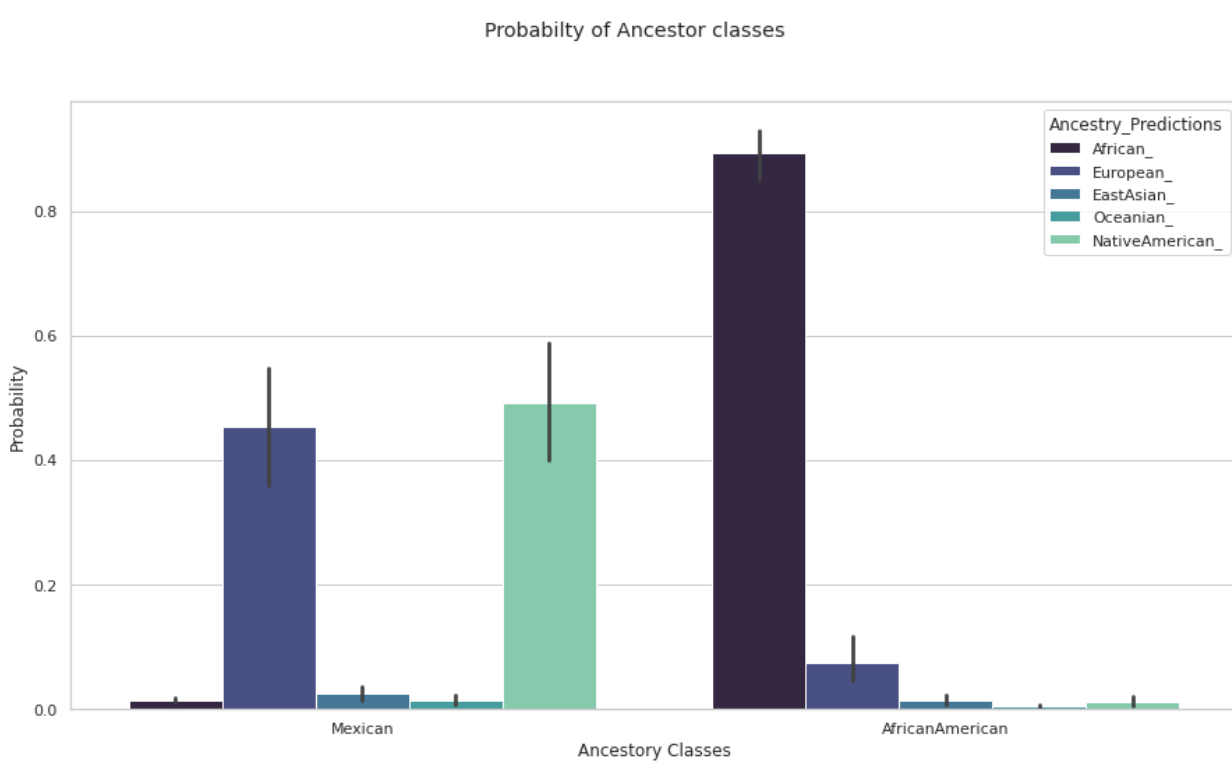
Deliverable 3.1:



Deliverable 3.2:



Deliverable 4.1:



Deliverable 5:

Deliverable 5

How do the class label probabilities differ for the Mexican and African American samples when compared to the class label probabilities for the unknown samples?

> In comparison to the class label probabilities for the unknown samples, those with unknown ancestry show a probability close to or equal to one while the other classes show a probability close to zero or less than one. African American samples showed similar results. The model assigned high probabilities to the African ancestry class for each of these samples. However, both Native American and European ancestry contribute high probabilities to the Mexican population on average with Native American slightly higher than European.

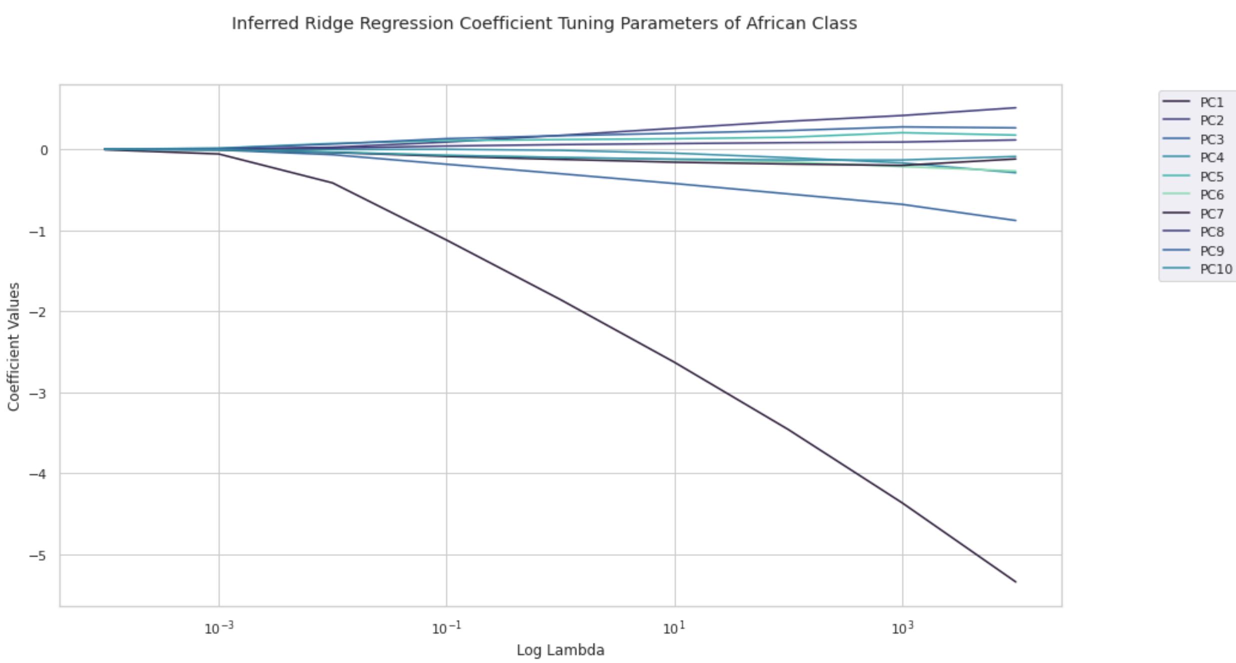
Deliverable 6:

All source code and project files incuded

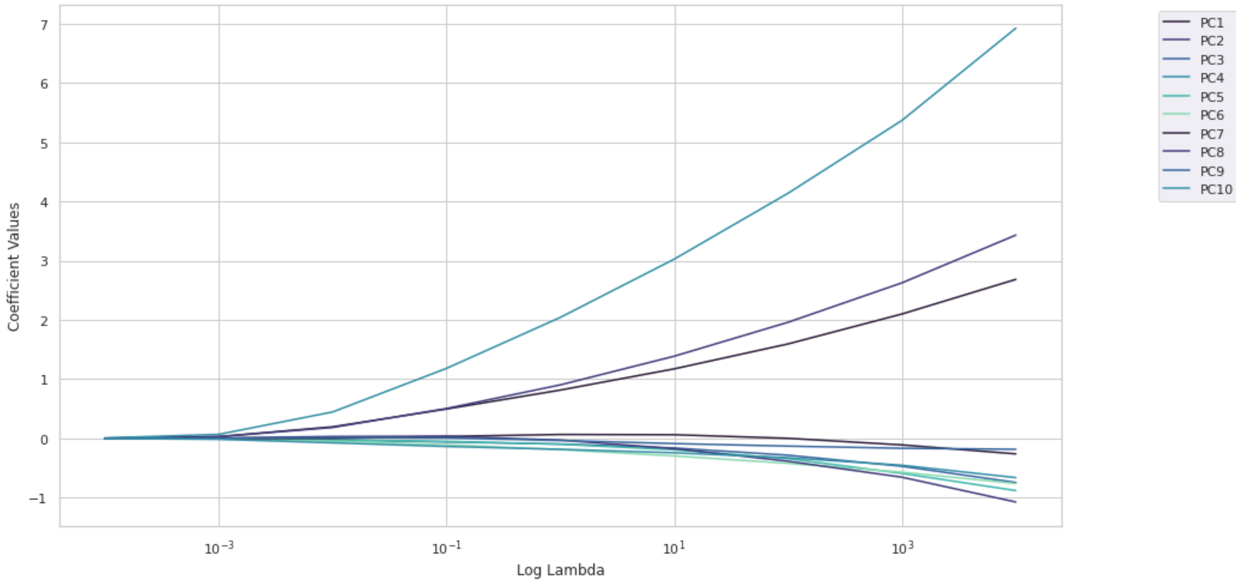
Deliverable 7:

From Libaries implementation output:

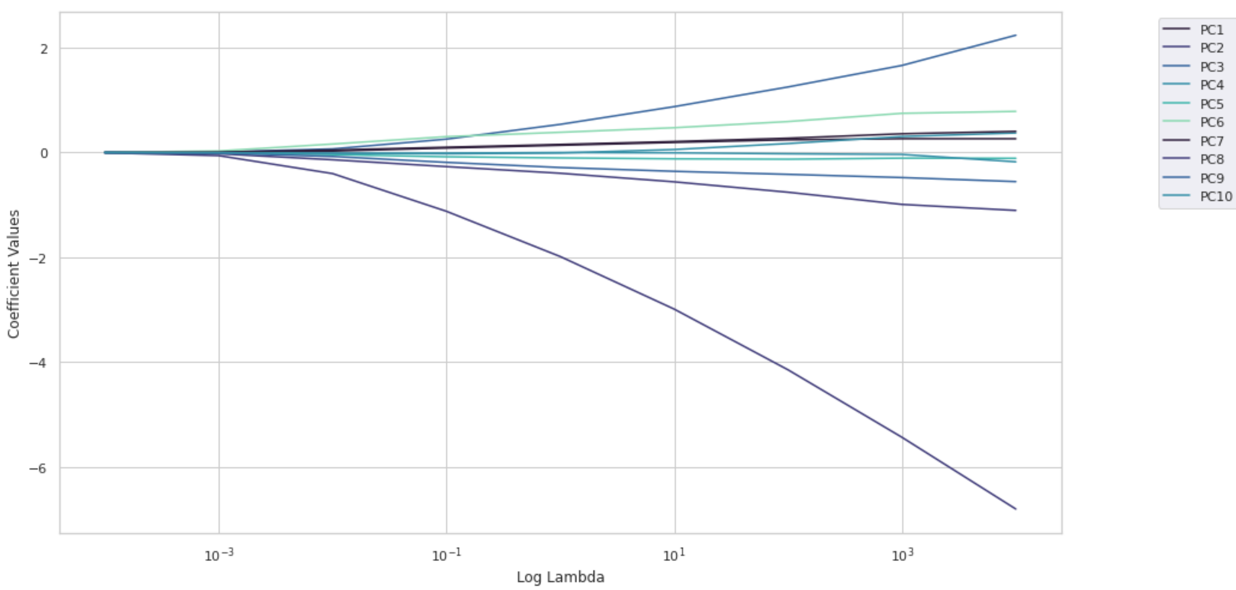
- **Deliverable 7-1:**



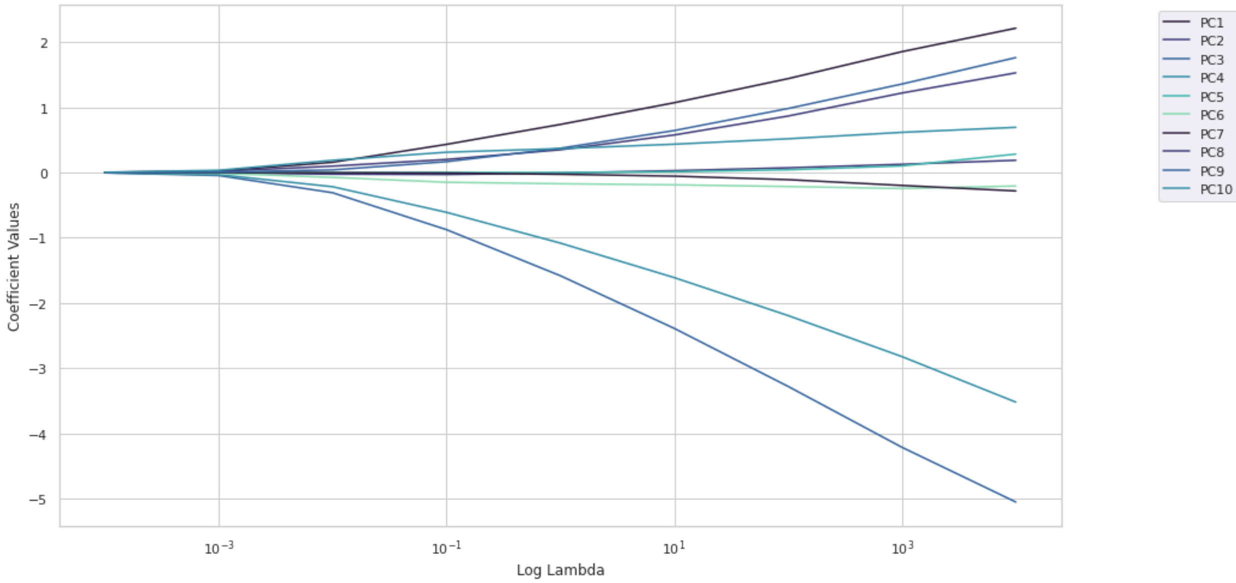
Inferred Ridge Regression Coefficient Tuning Parameters of European Class



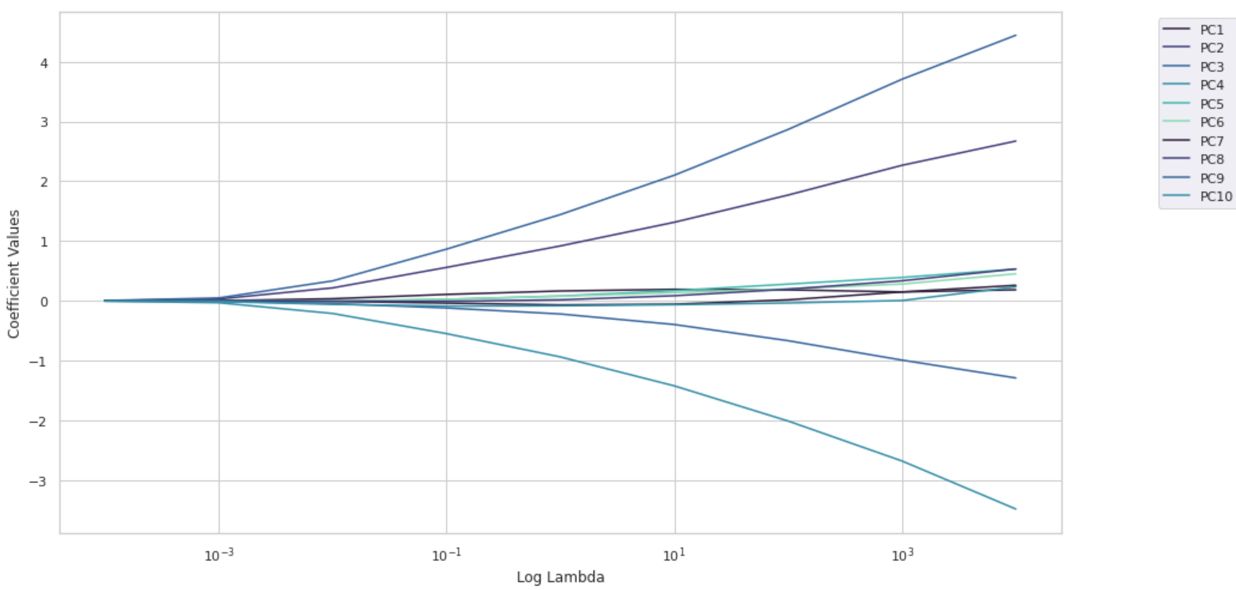
Inferred Ridge Regression Coefficient Tuning Parameters of EastAsian Class



Inferred Ridge Regression Coefficient Tuning Parameters of Oceanian Class

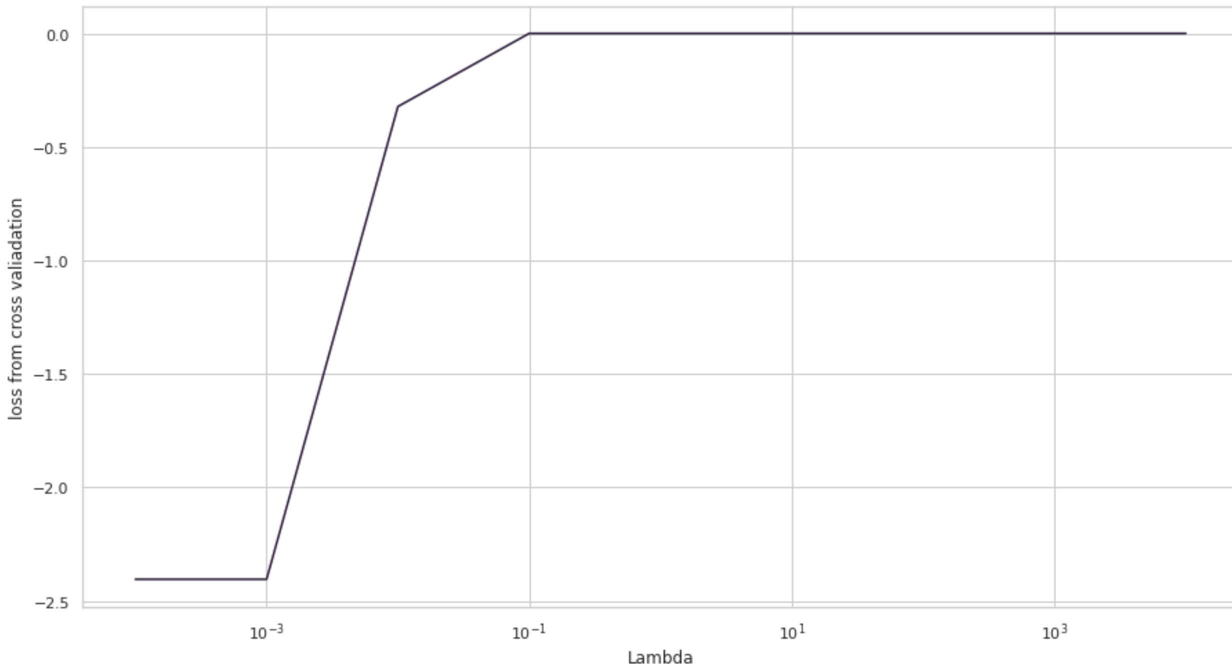


Inferred Ridge Regression Coefficient Tuning Parameters of NativeAmerican Class



- **Deliverable 7-2:**

Effect of the uning parameter on the cross validation error log10(lambda)



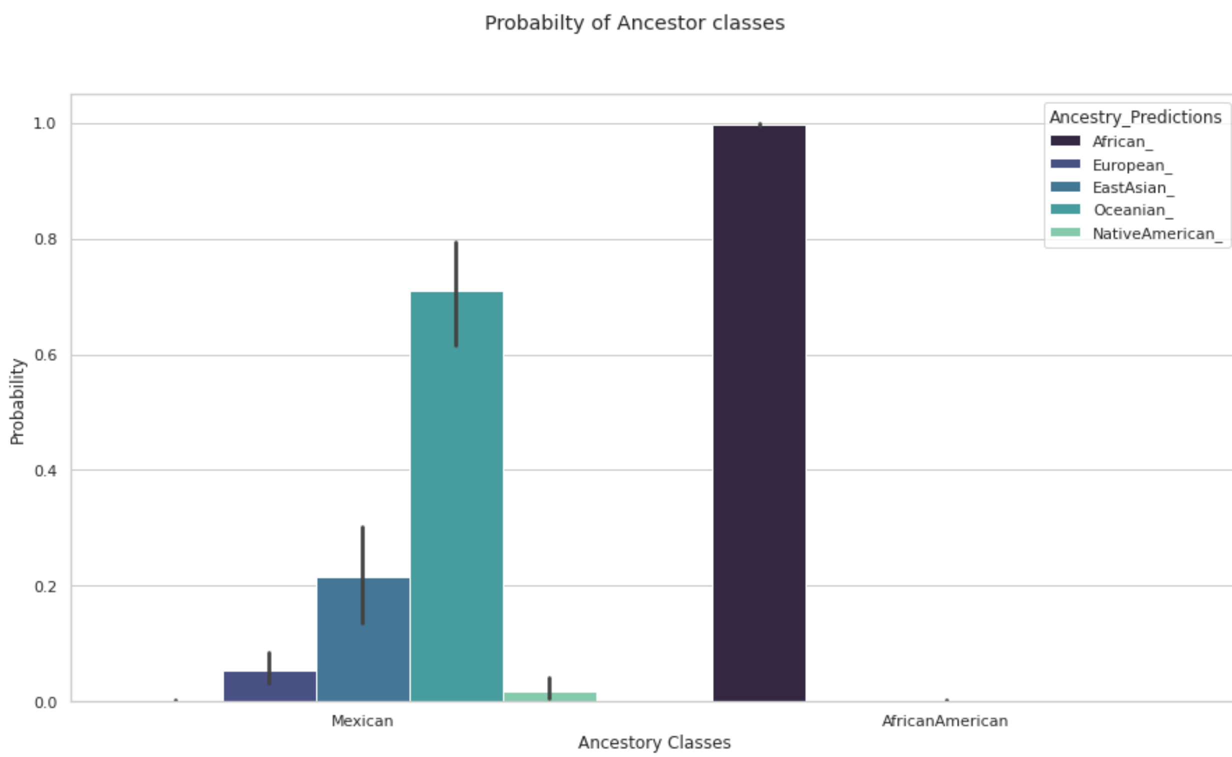
- **Deliverable 7-3-1:**

```
Best:  {'C': 0.1}
Best CV mean squared error: 0.000
```

- **Deliverable 7-3-2:**

	learning_rates_λ	mean_test_score
3	0.1	0.0

- **Deliverable 7-4:**



- **Deliverable 7-5:**

Conclusion:

Our implementation with the sci-kit learn libraries in Python came very close to the same precision and convergences as our algorithm from scratch. We see that the estimates both compare to the estimates obtained from logistic ridge regression from scratch optimal lambda (optimal λ = 0.0001 1e-4) and with libraries (optimal λ = 0.1). The convergence based on the effects of the tuning parameter and cross validation error converge at relatively the same frequency for both from scratch and with libraries.

In the data, there is greater prediction and performance when calculating the probabilities with all ancestry prediction closer to zero for both Mexican and Africanas having more significant probabilities. In addition, we measure the normal distribution for the performance value.

After calculating probability, the standard deviation of the implementation with libraries was approximately 3.713757e-01, while the standard deviation of the implementation from scratch was

0.340312 with a mean of 2.0 and cumulative z-score of 49.9% from the mean which shows a more precise and significant measurement of probability as compared to our implementation from scratch.