Tailored for Diversity: Adapting Garment Patterns to Fit Diverse Body Shapes

Author:

Lily Bharati Sharma (ls989), Xinru Zheng(xz844), Xuran Zhang(xz854), Yanan Zhang (yz2999), Yuze Gu (yg348)

Project Advisor: Sreyoshi Das

Client: Fatma Baytar

1 Acknowledgement

We would like to express our deepest gratitude to Professor Fatma Baytar, an exceptional client, for entrusting us with the “Tailored for Diversity: Adapting Garment Patterns to Fit Diverse Body Shapes” project. Your collaboration and support throughout the project were invaluable, and we appreciate the opportunity to work with you.

We are also thankful to Professor Xiaolong Yang, Director of the MPS program, for his continuous encouragement and support during the project. His guidance and insights greatly contributed to the success of our work.

Special thanks to Professor Sreyoshi Das, our advisor, for her invaluable guidance and support at every step of the project. Her expertise and willingness to address our queries significantly enriched our understanding and helped us navigate through various challenges.

We are also grateful to Professor Sumantu Basu, our Guest Advisor, for his exceptional advice on machine learning techniques and approaches. His insights and clarity on the intricacies of the project were immensely valuable. Despite his busy schedule, he generously devoted time to assist us whenever needed.

Finally, we extend our gratitude to all those who contributed directly or indirectly to the completion of this project. Your support and encouragement have been instrumental in our journey.

2 Executive Summary

The diverse body shapes and sizes found among various ethnic groups form a robust basis for improving the inclusivity and precision fit of clothing. This project is centered on accurately analyzing anthropometric data to predict crotch curves and using machine learning models to predict ethnicity. The team seeks to enhance pants design, comfort, and fashion by studying body measurement patterns, especially the lower torso for different ethnicities. The project will provide important insights into how garment pattern shapes can be changed to provide better cloth-fitting for all ages and various races. The research adopts the CAESAR database, which is located at Dr. Baytar's lab and contains measurements from three NATO countries. Dr. Fatma Baytar is an Assistant Professor of Human Centered Design at Cornell University, and she is the sponsor of the research project. Moreover, a subset from women's lower torso shapes is created from the database for the team to study how lower torso shapes change and how this change can be predicted from body measurements. The team completes a preview of literature on similar topics, uses a spline model to fit the curve data and extract features from the regression, and adopts multiple machine learning methods to make predictions. The results have shown that adding the curve features into the regression model along with the anthropometric numeric variables improves the prediction about the ethnicity of the experimental units.

3 Table of Contents

4 Introduction

Studying changes in body measurements and posture for men and women across diverse ethnic backgrounds helps people gain a deeper understanding of human diversity. These insights are meaningful for medical purposes, clothing designs, and ideas about differences between various cultures. Specifically, the team's research concerns figuring out related knowledge to improve ways to design clothes that fit well for men and women of all ages. The purpose has profound meaning in human civilization because simply everyone has the right to wear clothes that fit them comfortably no matter their age and race, and through the research, we can help people find clothes that feel cozy and look fashionable on them.

Predicting the crotch curve has been an essential task in garment design due to multiple reasons. First of all, it ensures that clothes fit wearers well by improving their outlook and comfort. Since women's bodies have various shapes and sizes, predicting crotch curves precisely helps make garments reflect these differences. Besides, predicting crotch curves helps improve the quality and durability of clothes. If the crotch curve parts are designed poorly, chafing between legs and pants can lead to discomfort and even break the pants. By studying the crotch curve, the manufacturers can extend the longevity of their pants. Additionally, predicting crotch curves contributes to the inclusivity of the products. By considering the diverse body types of women with different races, pants can be more inclusive and accessible to a larger variety of people, and therefore the research includes the study of the crotch curve of women with different ethnicities. Finally, predicting crotch curves can also enhance the production process by reducing alterations and waste, thus improving the sustainability of the whole industry. With all these benefits, it is reasonable to conclude that predicting the crotch curve is a significant task for researchers to dive into.

However, measuring crotch curves directly is not a simple task. It requires the measurements of the shape and size of the area between legs, and since it is a 3D curve instead of a flat part, it is very difficult. Even if tailors use tools such as rulers and tapes, it is hard to get the precise measurements, because its 3D nature always causes measurement errors. Factors such as body shape can also interfere with the measurements, and even minor errors may lead to poor design and discomfort wearing experience.

In these ways, the team aims to use data of human characteristics such as weight, height, and BMI as independent variables to predict crotch curve for women with different races such as White, Asian, and African American and determine whether adding features of the curve improve prediction about ethnicity. Moreover, the team analyzes how women's lower torso shapes change with age using body measurements from the whole dataset, and we try to discover patterns in how lower torso shapes change over time. And this purpose concerns beyond observation. Instead, the team tries to develop predictive models that anticipate the changes based on body measurements. Through the results, understanding of women's physiology is going to be facilitated, which is going to be one step forward towards enhancing human civilization.

5 Literature review

We have firstly conducted a literature review process on the research being done by professionals in this field trying to build on what they have learned in their studies.

The first objective in these literatures is to solve the fitness problem associated with pants by considering the body shape of crotch curves. Separate experiments are conducted for men and women. For example, the article "predicting distance ease distributions on crotch curves of

customized female pants" mentions that the distance ease at the crotch is the determinant factor ensuring the pant fit during unrestricted movements. The second objective is to enhance the trouser pattern construction, and one of the techniques is the utilization of 3D body scanning. The third objective is to classify body shapes for garment design. Studies classify body shapes to tailor garments including pants for diverse body types.

Various statistical techniques are used for prediction in the literature. The first technique is regression analysis. Predictive models are developed to estimate distance ease distributions along crotch curves. There is linear regression, polynomial regression, and maybe ridge and lasso potentially. The second technique is to find factors influencing body shapes and pattern construction for better garment fit. The third technique is to categorize body shapes into clusters based on similarities for tailored garment design. For these two techniques, in the article called "an image-based shape analysis approach and its application to young women's waist hip leg position", they found factors that influence the waist-hip shape and leg shape characteristics, and they also put the waist-hip shapes and leg-shapes into a few clusters. A fourth technique is the PCA. It is a dimension-reduction method that is used for regularization. Here it is utilized to analyze variance in body shape data and streamline pattern generation.

Different data collection methods are employed in literature to collect size and shape data. The first one is anthropometric measurement which is a direct measurement technique employed to gather body dimensions, including crotch curve features. The second one is 3d body scanning which is an advanced technology capturing precise body measurements, facilitating accurate pattern drafting. In the article called "enhancing pattern construction by body scanning the importance of curves", it states that in using 3D body scans to map the body we must take into

consideration all the curved surfaces of the body and their measurements. The third one is image-based shape analysis which is basically parameter extraction based on body photos.

The next section is the exploration of somatic factors and pattern drafting techniques. The first technique is crotch curve width considerations because studies suggest that optimal crotch curve width is based on hip depth and wearer's age. The second technique is age variation: variations in crotch curve dimensions are observed across different age groups, highlighting age-specific design needs. The third one is the measurement and analysis technique. Accurate measurement and analysis of body parameters are crucial for effective pattern drafting.

6 Exploratory data analysis

6.1 Data Overview

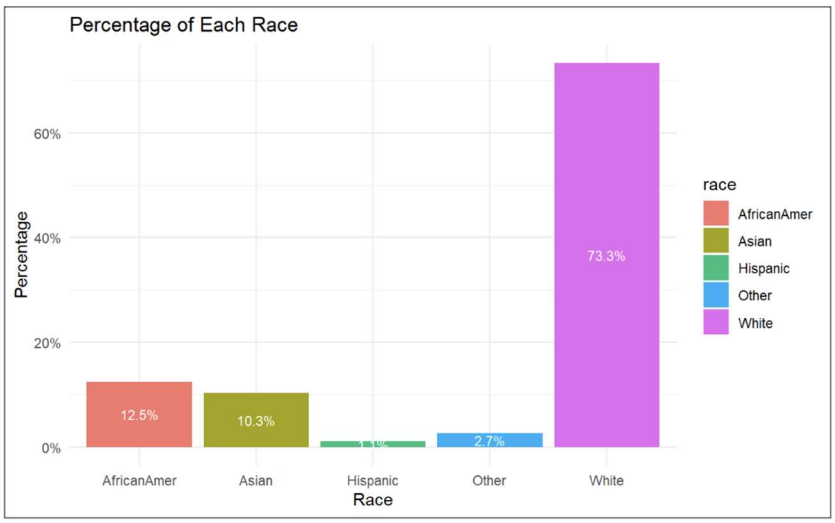
The used for the analysis was from the CAESAR database which is located at Dr. Fatma Baytar's lab. It contains measurements from three NATO countries: North America, Italy, and the Netherlands. Our objective was to analyze only North America data. In total, there are 12 independent variables: subject, BMI, height, max_hip, anterior_posterior_length, depth, crotch_curve_length_at_back_waist, front_crotch_(left_side), back_crotch_(right_side), comments, curve, race.

Table 1: List of variables in the original data client given to us

Variable Name	Description	Type
BMI	body weight relative to height	quantitative
Height [mm]		quantitative
Max.Hip [mm]	The maximum circumference of the body measured between waist and crotch, parallel to the floor	quantitative
Anterior-Posterior.Length [mm]	Distance between front and back at the waist level on the sagittal plane	quantitative
Depth [mm]	Distance between the waist level and the crotch point	quantitative
Crotch.curve.length.at.back.waist [mm]	Distance from front to back at the waist level on the sagittal plane	quantitative
Front.Crotch.(Left.side) [mm]	Distance from the front waist level to the crotch point on the sagittal plane	quantitative
Back.Crotch.(Right.Side) [mm]	Distance from the back waist level to the crotch point on the sagittal plane	quantitative
Race	Hispanic, Other, White, Asian, African American	categorical
Subject	database number	
Comments	notes	
Curve	crotch curve shape picture	
Data_points	crotch curve shape coordinates	quantitative

There are 551 subjects in total, and 405 subjects of them are white, which takes about 30% of the data from all races. The distribution of data by race can be seen in the following image.

Chart 1: Bar chart – Distribution of Population across Race



Total –551 subjects

White – 405

Approximately-30% of the data from all other race.

6.2 Data Pre-processing

6.2.1 Missing Value

Each variable's missing values were checked, as shown in Table 2 below. At most, 7 data points were found to be missing, and these data points are either funky shapes or bad crotch curves. The total percentage of missing data points accounts for approximately 1%. Since this percentage is negligible, it will not affect our analysis if we delete it. Therefore, we exclude these data points from the analysis. Detailed information about missing values can be found in the following chart.Having these missing values in the analysis would introduce potential biases or errors, particularly in statistical measures like means, variances, and correlations. It could also impact the generalizability and reliability of our findings. By excluding these data points, we ensure that our analysis is based on more complete and accurate information, thereby enhancing the validity of our results. Detailed information about missing values can be found in the following chart.

Table 2: Missing Values

	Missing
Anterior_posterior_Length	3
Depth	3
Crotch_curve_length_at_back_waist	7
Front_Crotch_Left_side	7
Back_Crotch_Right_Side	7
Depth	3
Crotch_curve_length_at_back_waist	7
Front_Crotch_Left_side	7
Back_Crotch_Right_Side	7

6.2.2 Outlier Detection

Outliers are identified prior to the main analysis by calculating the first quartile (Q1) and the third quartile (Q3) for each quantitative variable. The lower and upper bounds are then determined using the formula: lower bound = Q1 – 1.5 * IQR and upper bound = Q3 + 1.5 * IQR, where the Interquartile Range (IQR) is the difference between Q3 and Q1. Values that fall outside these calculated bounds are classified as outliers, indicating potential anomalies or extreme values in the data.

In relation to other races, the White group has the highest number of outliers across the majority of measurements, particularly in BMI, Max_Hip, and Anterior_posterior_Length. This suggests that individuals in the White group may exhibit more extreme values in these measurements compared to other racial groups. The presence of outliers in these variables could skew statistical

analyses and affect the interpretation of results. Therefore, it's essential to account for these outliers appropriately, such as through robust statistical methods or, if necessary, exclusion from certain analyses to ensure the validity and reliability of the findings.

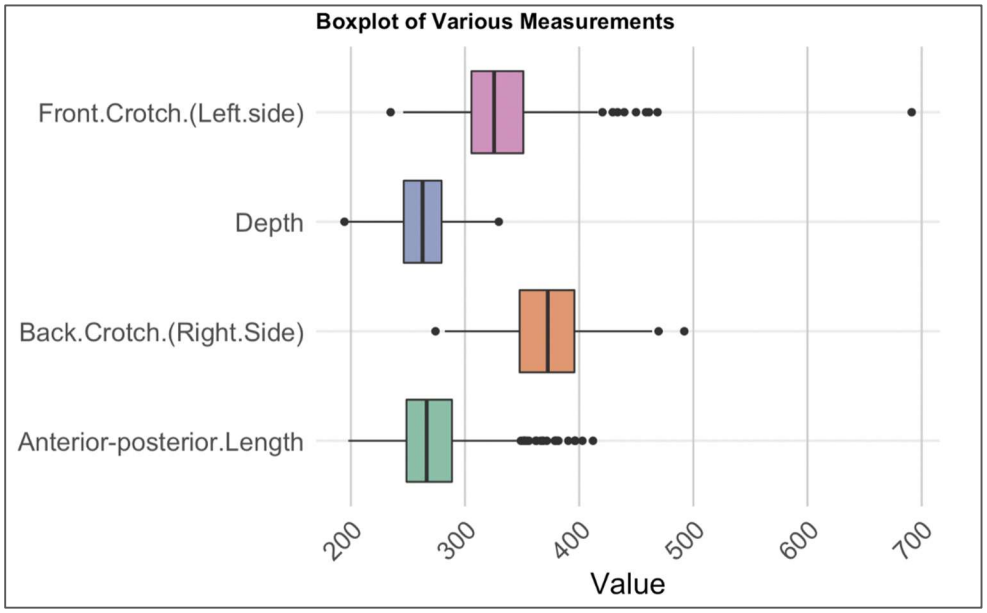
Table 3: Outliers

Race	AfricanAmer	Asian	Hispanic	Other	White
BMI	2	1	1	0	20
Height	3	0	0	2	5
Max_Hip	1	1	0	0	13
Anterior_posterior_Length	2	2	1	0	17
Depth	0	5	1	0	6
Crotch_curve_length_at_back_waist	1	2	1	0	5
Front_Crotch__Left_side	1	1	1	0	6
Back_Crotch__Right_Side	1	3	0	0	2

The boxplot offers a visual summary of the distributions of four distinct body measurements pertinent to the lower torso: Front Crotch (Left side), Depth, Back Crotch (Right Side), and Anterior-posterior Length. It illustrates the median, interquartile range, and outliers for each measurement, providing insight into the variability and central tendencies within the data set.

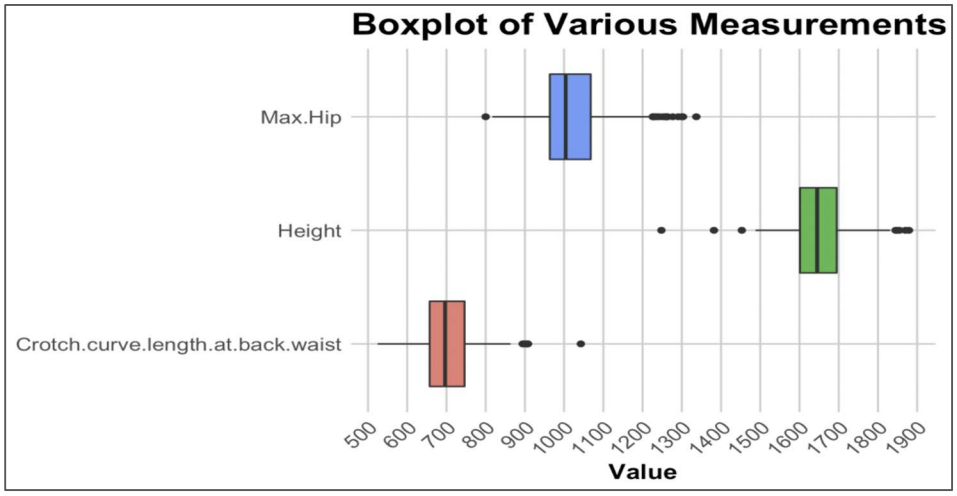
The 'Front Crotch' and 'Back Crotch' measurements, likely related to the front and back rise in garment fitting, display a moderate spread of values, indicating a diverse range of lower torso shapes. Notably, the 'Depth' measurement shows a narrower interquartile range, suggesting less variability in this particular dimension across the sampled population. The 'Anterior-posterior Length' reveals a wider range of values, including several outliers, which may be indicative of significant variations in body posture or type. These outliers could be of particular interest when considering ergonomic and apparel design to accommodate a wide array of body shapes.

Chart 2: Outlier detection -Box Plot – variables with similar scale



Examining the boxplots above, the 'front crotch of the left side' shows a few outliers that fall outside of the general spread of the rest of the data. 'Depth' and 'back crotch of the right side' both have outliers situated at the upper and lower whiskers area. 'Anterior-posterior length' shows relatively small variations with the box being very compact, and outliers closely situated around the whiskers.

Chart 3: Outlier detection -Box Plot – variables with similar scale



The boxplot under examination showcases the distribution of three key body measurements: Max Hip, Height, and Crotch Curve Length at Back Waist. Height varies most among these three variables with a wide IQR and the whiskers extending further, suggesting a greater spread of data. Outliers are present on both lower and upper ends, which indicates some individuals are significantly shorter or taller than the average. The plot for Max Hip demonstrates a relatively compact interquartile range with a consistent spread of values, which suggests a moderate variation in hip size within the population. The measurement for crotch curve length at back waist has the narrowest IQR, indicating fewer variations in the measurement. Note that BMI is excluded in boxplots since it has a different measurement scale than the rest of the variables.

To compare variables across various races, ANOVA analysis was conducted to discover whether there are differences between the means of the ethnic groups. ANOVA (Analysis of Variance) is a statistical technique that tests whether the means of multiple groups are statistically different. It

is a very useful way of recognizing whether to separate different racial groups in the following analysis.

In the analysis, we first separate the whole data into three ethnic groups: White, African American, and Others according to the data size of each of these groups. The following chart includes all the ANOVA results. The F-statistic shows if the means between populations are significantly different. Larger F-statistic values show more significant differences. And each of these values has a corresponding P-value, where values smaller than 0.05 show significant differences at the significance level of 0.05. ANOVA tests was conducted on all the quantitative variables, where we conclude that there are significant differences between means of the tested variables across different ethnic groups, and thus White, African American, and Other needs to be separated in the analysis.

Table 4: Analysis of Variance (ANOVA)

Variable name	F-statistic	P-value
BMI	15.34	3.29e-07
Height	29.99	4.38e-13
Depth	6.929	0.00107
Front.Crotch.(Left.side)	6.559	0.00153
Back.Crotch.(Right.Side)	9.463	9.13e-05
Max.Hip	23.12	2.29e-10
Crotch.curve.length.at.back.waist	9.502	8.79e-05
Anterior-posterior.Length	18.94	1.12e-08

6.2.3 Correlation Analysis

Correlation analysis was conducted to explore relationships between different variables in our dataset. By calculating correlation coefficients, we can determine the strength and direction of

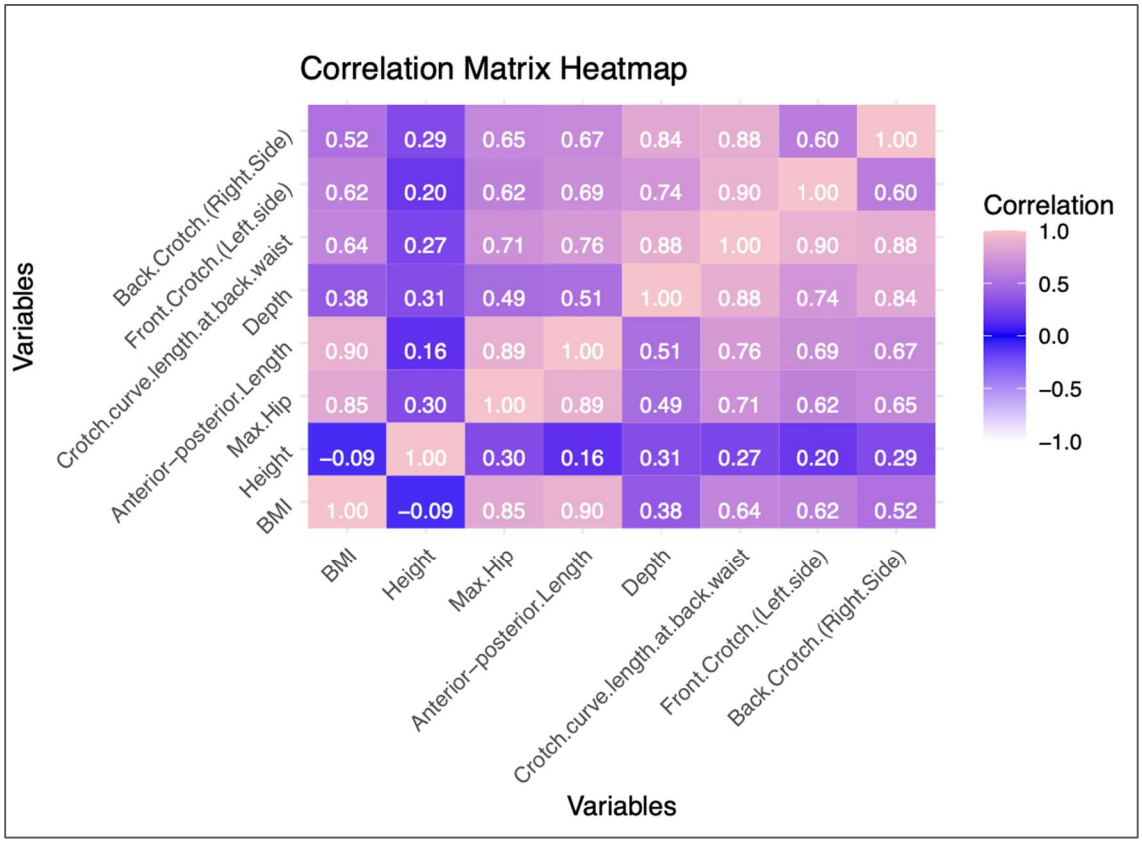
associations between pairs of variables. This analysis helps us understand how changes in one variable relate to changes in another, allowing us to identify potential patterns, dependencies, or trends in the data.

The heatmap in Chart 4 shows the correlations among various variables, revealing the nature of their relationships. The color scale on the right highlights the strength of these correlations, ranging from -1 (deep blue, indicating a strong negative correlation) to +1 (deep pink, indicating a strong positive correlation).

For example, the correlation between "Crotch.curve.length.at.back.waist" and "Front.Crotch.(Left.side)" is approximately 0.9, showing that changes in one of these crotch length measurements often coincide with changes in the other. This insight could be valuable in ensuring better clothing fit and design. Another strong positive correlation is between “BMI” and “anterior-posterior length”, meaning that as BMI rises, the anterior-posterior length usually increases as well. This association can be useful for health assessments, helping understand body composition better. Similarly, the correlation between “anterior-posterior length” and “Max-Hip” is 0.89, indicating that individuals with greater anterior-posterior length also tend to have a larger hip circumference. This means as the body depth (from front to back) grows, hip size generally increases proportionally.

Conversely, the negative correlation of -0.09 between “BMI” and “height” shows a slight inverse relationship between the two variables. Although not particularly strong, it suggests that taller individuals might have a lower BMI, especially if their weight does not increase proportionally to their height, as BMI is calculated by dividing weight by the square of height.

Chart 4: Correlation Analysis-Heat Map



7 Crotch Curve Fitting

One of the crucial steps before delving into the real analysis is the crotch curve fitting process. The crotch curve for a subject is depicted in Chart 5 below. These images for each subject were provided in the master data in the form of images, and their respective x and y coordinates were provided in separate data.

The initial step involved merging this data with the master data, which contained all the variables for analysis along with the crotch curve image. As a second approach, we explored different

polynomial regression models to estimate this curve, and the most effective approach was using a Spline regression model.

7.1 Spline Regression Model

The key concept in the curve fitting process is the idea of spline regression. A spline regression model is a fundamental tool in statistical modeling and data analysis, widely adopted to model the relationship of data by combining piecewise polynomial regressions. Regression splines, as an example of a basis function approach, can capture the trend of a complex curve effectively. Splines are very useful in constructing smooth models over data in a particular range where the relationship between variables is too complex to be modeled by a linear relationship or a polynomial regression. The most significant idea in spline modeling is to separate the entire data range into a few segments. A polynomial regression with a small degree is fitted for each segment, and all polynomials join at knots that are pre-determined in the entire range of data. The placement of knots is usually not a very important aspect to the analysis, but the number of knots is crucial. For instance, an appropriate way of choosing K is to use cross-validation. The number of knots can influence how well the spline model fit the data set, specifically, the rule is that placing K knots will lead to K + 1 polynomials. Generally, more knots lead to the piecewise polynomial being more flexible. The spline regression ensures continuity at each of the knots, making the entire curve smooth.

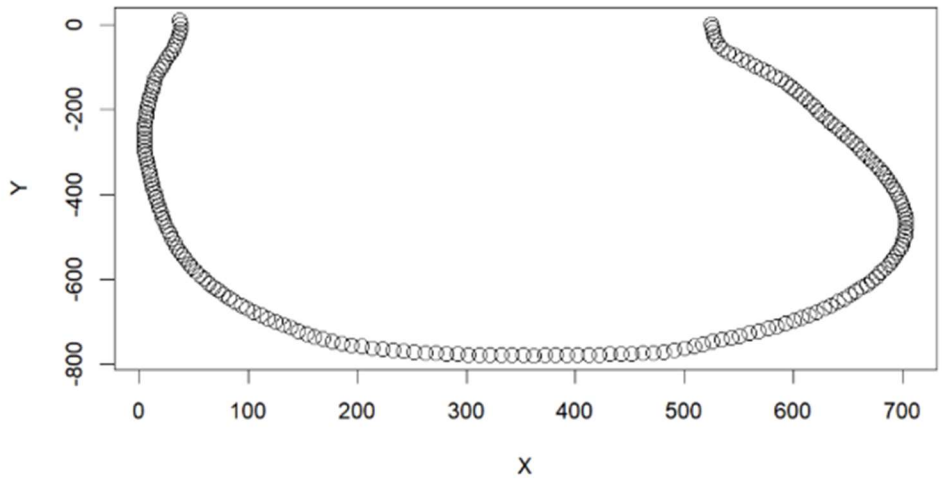
There are different types of spline models. A linear spline is a piecewise linear polynomial continuous at each knot, and a cubic spline is a piecewise cubic polynomial at each knot, where the derivatives are continuous up to the degree of two. A natural spline is a spline with constraint that at additional boundaries the function must be linear.

Spline models are commonly adopted in data science because they create smooth trends even in very complex datasets. In the process described below, the team adopts the concept of spline fit the curve and obtains the spline model coefficients.

7.2 Estimating Crotch Curve

To begin the process, the coordinates provided in the data of the crotch curve of every valid experimental unit. By using these coordinates, every crotch curve can be drawn. The goal is to adopt statistical methods to fit the curve and extract features of the curve in order to get necessary data to proceed with further analysis.

Chart 5: Crotch curve in the Data



The above chart is an example of the crotch curve of one data point, specifically the crotch curve of subject number 321.

The first step of curve fitting is to separate the lower part of the graph. The team begins by identifying the leftmost and rightmost points of the curve. Once they are determined, we select a specific point where the y-values are relatively low, and this point is a significant reference for

cutting the lower section of the curve. After obtaining this point, we can draw a horizontal line that intersects with it to segment the lower portion of the curve. The following Chart 6 and 7 illustrates how the whole process described above is being done on the crotch curve of the 321st subject. The right graph is the lower portion that is cut from the original curve.

Chart 6: Crotch curve split in half

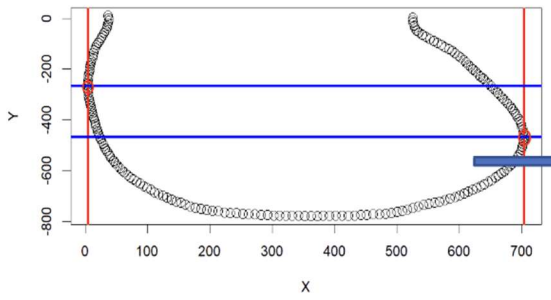
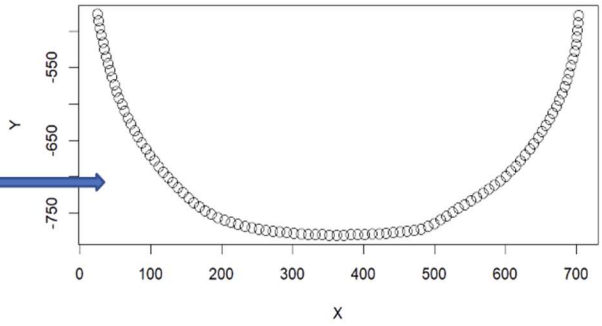
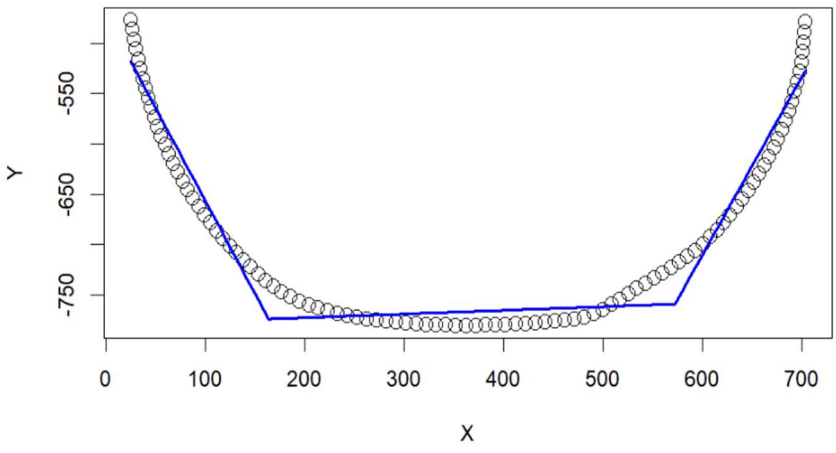


Chart 7: Crotch Curve Lower portion



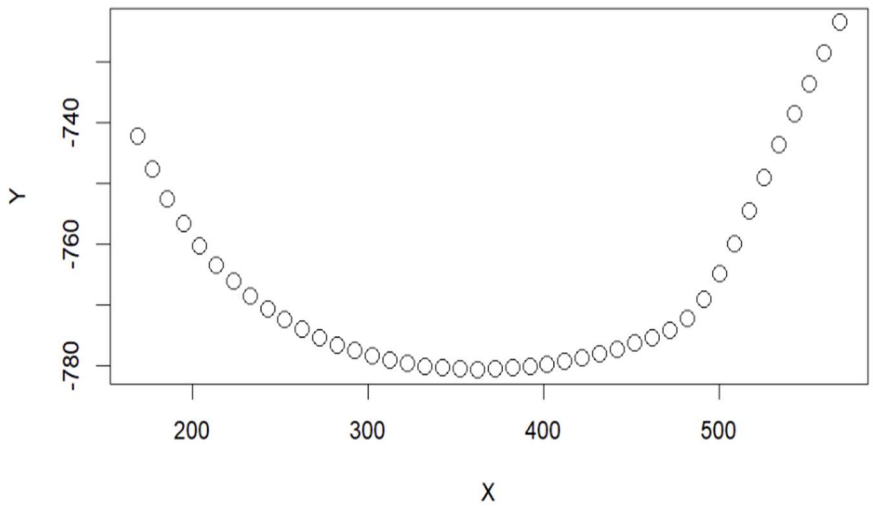
The second step is to separate the lower portion curve into three pieces, which is completed by getting two knots as shown in the below Chart 8. In order to find the two knots precisely, a package in R called "Segmented" was used. This package can help identify the two optimal knots and find them automatically.

Chart 8: Crotch curve -Knots



After obtaining the three parts, the middle part of the curve was separated from the lower curve to estimate it separately due to the fact that it is going to be fitted in a simpler way, as shown in the graph below.

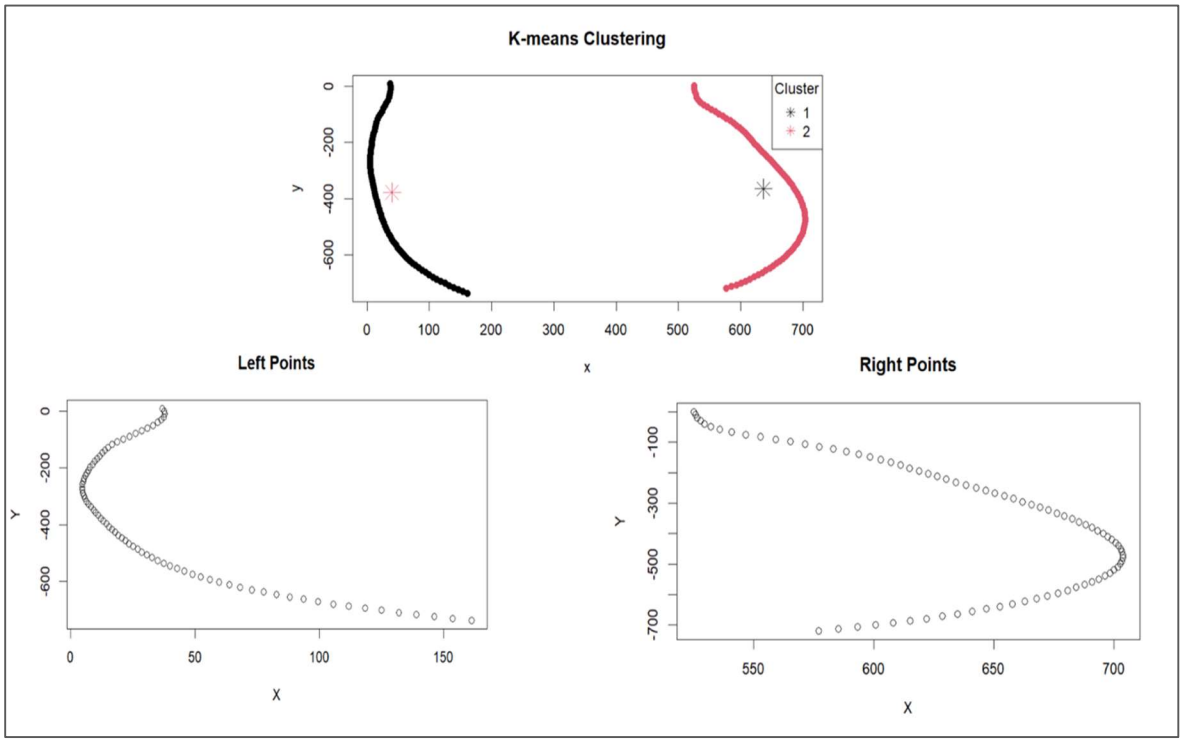
Chart 8: Crotch curve -Middle Portion



The next step is to separate the left and data points by using the method of K-means. K-means is a clustering machine learning and data mining method. It can partition a dataset into a certain number of clusters based on similarity measures. The method assigns data points to the nearest cluster based on the mean of the data points in each cluster, and this process is repeated until the means stop to change in a large scale. K-means is a largely favorable method due to its simplicity in the field of machine learning. Chart 9 below illustrates the left points and right points separated from the original curve. The team fit each part of the data with an appropriate regression model. The left part of the data is fitted with a third-degree polynomial, the right part of the data is fitted with a third-degree polynomial model, and the middle part of the data is fitted with a second-

degree polynomial model. In this way, all parts of the curve are fitted by appropriate regression methods.

Chart 9: Crotch curve -Left and Right curve



Finally, the last step is going to be plotting the fitted graph as shown in the graphs below. And at the current stage, all features of the curve can be extracted to become data that can be adopted in the following analysis. The following chart is to give a general overview of the data.

Chart 9: Crotch curve -Estimated vs Original Curve

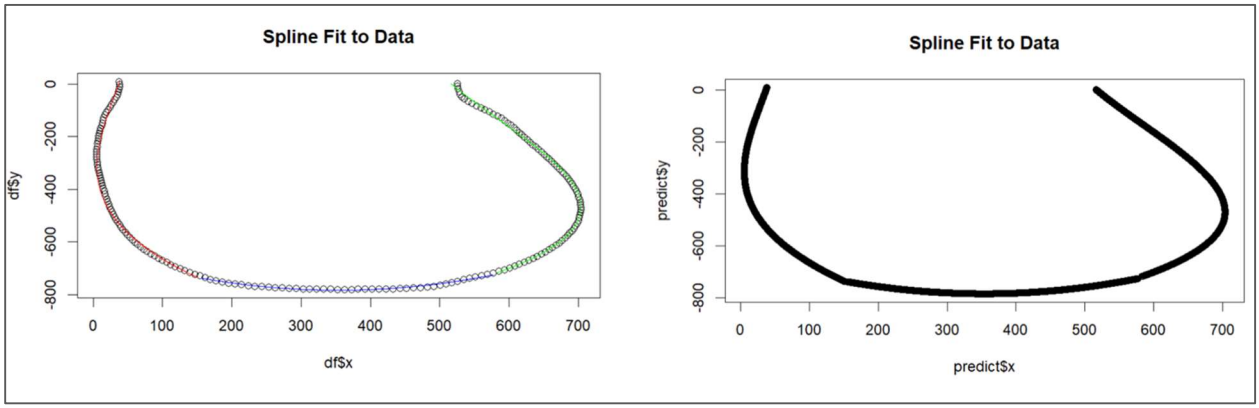


Table 5: Sample Master data

Left_I	Left_D1	Left_D2	Left_D3	Mid_I	Mid_D1	Mid_D2	Right_I	Right_D1	Right_D2	Right_D3	BMI	Height	Max.Hip	Anterior-pos	Depth	Crotch.curve	Front.Crotch.	Back.Crotch.(	race
734.428491	211.229306	76.8208939	-104.35068	-818.84	-88.670626	15.5875642	274.744735	-324.15438	-279.75474	-198.75285	25.4979605	1578	1067	293.1572	299.1631	788.713	384.521	404.192	White
170.718655	-174.89663	-196.19529	-123.15041	-680.27009	-40.719594	35.1892756	526.348238	206.229576	-17.857851	-28.657576	18.8218763	1583	911	214.3938	242.7403	618.077	296.48	321.597	White
726.041456	201.762483	25.5530464	-99.090271	-760.51557	-57.67701	9.70910241	222.963551	-270.83639	-279.81643	-221.29622	28.8944332	1619	1132	300.321	275.41	756.254	353.11	403.144	Other
208.782429	-206.60561	-185.5413	-195.12729	-679.29138	-61.220511	29.2428514	642.303539	183.093159	45.7408065	-113.03551	21.4	1601	934	254.29	245.012	662.811	311.565	351.246	AfricanAmer
694.735887	131.478702	55.981626	-62.296118	-784.79245	-95.738715	55.3175304	256.053544	-236.90149	-309.29568	-175.56081	20.8961904	1740	942	266.8546	285.9145	728.419	354.73	373.689	White
154.835993	-158.283	-226.5723	-186.55999	-685.69981	-70.527874	23.8493902	574.038617	193.489466	96.4878253	22.3293281	23.1534762	1611	988	260.2131	228.0892	609.998	279.52	330.478	White
666.543375	139.407149	-31.524659	-142.18692	-775.66105	-22.669873	106.024984	270.877128	-281.04888	-303.35276	-272.47691	22.1523644	1647	975	258.659	274.183	698.787	339.931	358.856	Asian

7.3 MANOVA - Racial Variations in Crotch Curve Characteristics

After the graph fitting process, we get spline coefficients, which are Left_I, Left_D1, Left_D2, Left_D3, Mid_I, Mid_D1, Mid_D2, Right_I, Right_D1, Right_D2, Right_D3. MANOVA analysis (Multivariate Analysis of Variance) examines whether spline coefficients, which reflect crotch curves, differ significantly among races. The null hypothesis suggests no significant differences in the means of the body measurements across different racial groups. On the other hand, the alternative hypothesis posits that these differences exist.

In terms of overall spline coefficients together, the p-value for the racial grouping is less than 2.2e-16, indicating extremely significant differences among the racial groups. This suggests that the overall shape of the crotch curves varies substantially between races.

For the left-side coefficient, the p-values between African American and Asian groups, as well as between Asian and White groups, are 0.007066 and 0.001366, respectively. These values are well below the standard significance threshold of 0.05, indicating significant differences in crotch curve measurements on the left side between these racial groups. For the middle coefficient, the p-values among African American and Asian, African American and White, and Asian and White groups are 2.97E-06, 3.74E-05, and 0.001365, respectively. These extremely low p-values highlight statistically significant differences in crotch curve measurements in the middle region between the different races. For the right-side coefficient, the p-values between African American and Asian groups, African American and White groups, and Asian and White groups are 0.00127, 8.28E-07, and 0.04478, respectively. These significant p-values indicate considerable differences in the crotch curve measurements on the right side among these racial groups.

In sum, the analysis demonstrates that spline coefficients for crotch curves significantly differ between racial groups, underscoring the need for tailoring clothing designs to accommodate these differences.

Table 6: MANOVA- Racial Variations in Crotch Curve Characteristics

Data	Race	P-value
Left Coefficient	AfricanAmer vs Asian	0.007066
	Asian vs White	0.001366
Middle Coefficient	AfricanAmer vs Asian	2.97E-06
	AficanAmer vs White	3.74E-05
	Asian vs White	0.001365
Right Coefficient	AfricanAmer vs Asian	0.00127
	AfricanAmer vs White	8.28E-07
	Asian vs White	0.04478
Spline Coefficient	Race	<2.2e-16

8 Machine Learning Model to Predict Race

8.1 Logistic regression

Logistic regression is a statistical method usually used to model a binary outcome. The regression model predicts the log-odds or the probability of this binary outcome based on one or multiple independent variables. One of the common extensions of logistic regression is multinomial logistic regression which is used when the response variable contains more than two outcomes. Multinomial logistic regression model predicts the probabilities of multiple outcomes of the response variable based on a few independent variables. The research objective is to determine whether curve features along with characteristics information improves the prediction for race. Since the race variable has more than two categories, multinomial logistic regression is used to simplify the analysis.

To test this objective, three sets of data was used to compare the model performance:

i. Spline model coefficients which are the curve features extracted from curve fitting process

ii. Numeric variables which are the body characteristics in the original data set, and

iii. Variables from both categories (Spline model coefficients and Numeric variables)

Table 7: Logistic Regression Accuracy and AUC

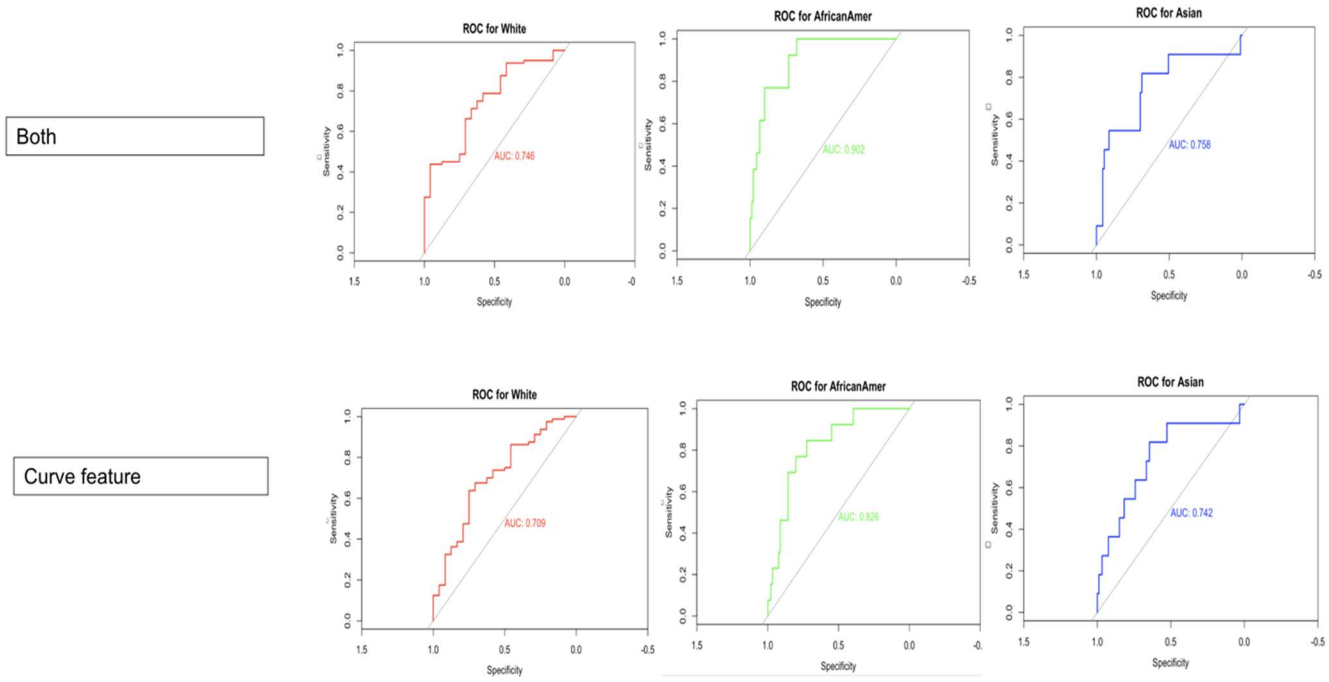
Data	Stepwise	Variables	Number of Variables	Accuracy	AUC for AfricanAmer	AUC for Asian	AUC for White
Both	Before	Left_I, Left_D1, Left_D2, Left_D3, Mid_I, Mid_D1, Mid_D2, Right_I, Right_D1, Right_D2, Right_D3, BMI, Max Hip, Height, Anterior-posterior Length, Depth, Crotch curve length at back waist, Front Crotch(Left side), Back Crotch (Right Side)	19	0.7788462	90.2	75.8	74.6
	After	Left_I, Left_D1, Left_D3, Mid_I, Right_I, Right_D1, Right_D2, BMI, Height, Max Hip, Depth, Crotch curve length at back waist, Front Crotch(Left side)	13	0.7788462	89.2	77.9	76.9
Spline Model Coefficients	Before	Left_I, Left_D1, Left_D2, Left_D3, Mid_I, Mid_D1, Mid_D2, Right_I, Right_D1, Right_D2, Right_D3	11	0.7692308	82.6	74.2	70.9
	After	Left_I, Left_D1, Left_D2, Left_D3, Mid_I, Right_I, Right_D1, Right_D2	8	0.7692308	80.1	74.5	68.1
Numeric Variables	Before	BMI, Max Hip, Height, Anterior-posterior Length, Depth, Crotch curve length at back waist, Front Crotch(Left side), Back Crotch (Right Side)	8	0.7596	71.7	84.9	73
	After	Max Hip, Height, Depth, Crotch curve length at back waist, Front Crotch(Left side)	5	0.7884615	71	83.4	73.4

As shown in the above Table 7, each of the three sets has a relatively large number of variables. The problem with a large number of variables is that the regression is potentially so complex that the model has an overfitting problem. Overfit models tend to have larger variability than a good model that balances well between bias and variance. One of the best ways to solve this problem is to perform model selection on the set of variables. So, stepwise regression- a process called bidirectional selection, which is a combination of forward and backward selections was performed.

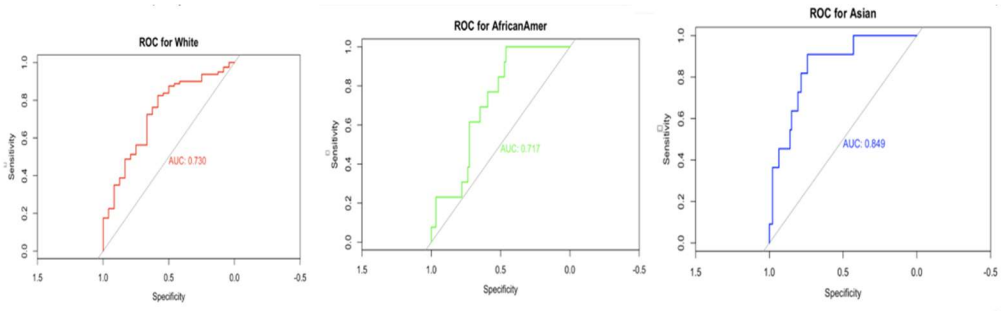
It adds variables as in forward selection and checks to remove any unnecessary variable at the same time. Among all the models for each number of variables, the model with the smallest AIC leaves out the final best logistic model.

Then, multinomial logistic regression was used to each of the three sets of variables and ROC curve to test the performance of them. ROC (Receiver Operating Characteristic) curve provides a visual way to test the performance of logistic regression. It plots the true positive rate (sensitivity) versus the false positive rate (1 – specificity). The threshold is that the more top-left the curve is, the larger the area under the curve, the better the classifier is. ROC curve is a very useful statistical method used to test the performance of different classifiers. In this analysis, after fitting logistic regressions, ROC curves is obtained, and their AUC values are evaluated to determine the actual performance. The graphs below illustrate the ROC results for regressing the three sets of variables on the three races: White, African American, and Asian.

Chart 10: ROC Curve- Logistic Regression Model Performance



Numeric variables- Body characteristics



As noted in previous sections, more than 70% of data comes from the White category. However, the AUC values of White are relatively low compared to the two other races, indicating that logistic regression is not predicting as accurately as expected for White. There are two potential causes for the problem: imbalance data and variation within the White race.

The first problem to check is the imbalance data problem. To test the problem, 100 experimental units were randomly selected from White, and repeated the regression analysis with all other variables remaining constant. The results are shown in the chart below, which has shown that the AUC values have only increased slightly, indicating that the imbalance data problem is not likely to be the problem that causes the poor performance for White.

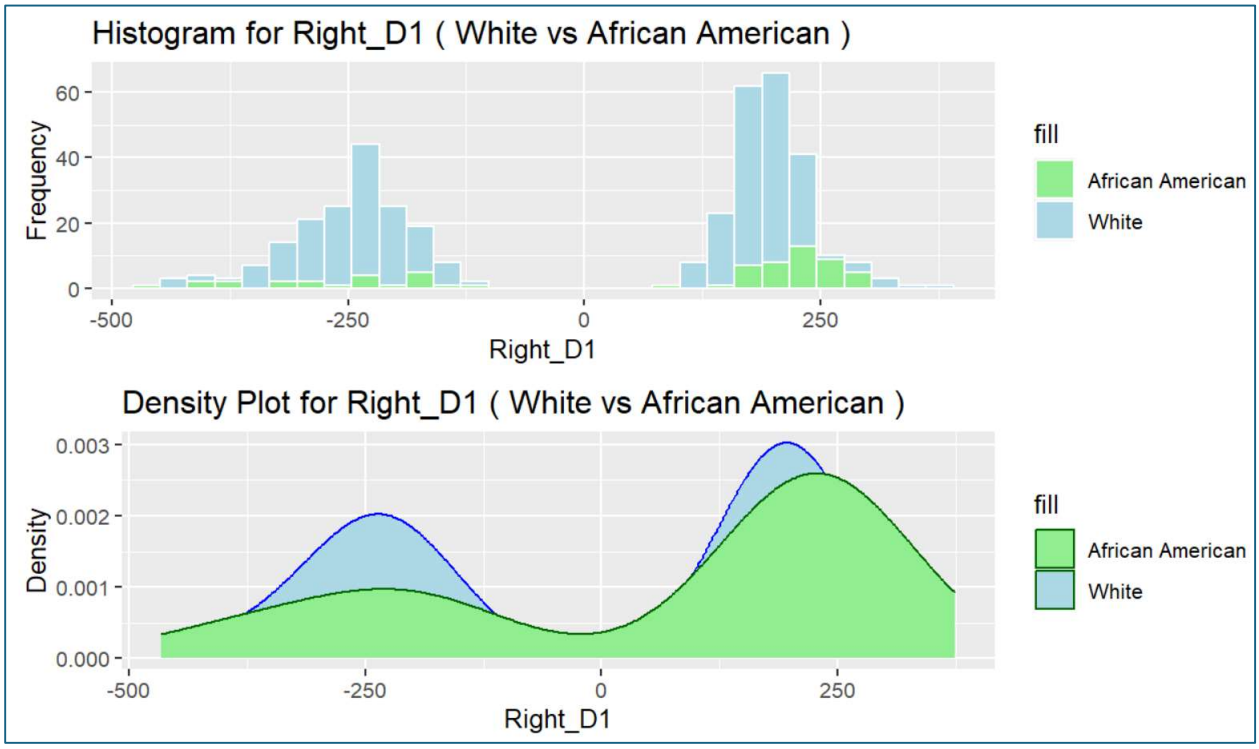
Table 8: Logistic Regression & Stepwise Regression - Accuracy and AUC

Both	Before	0.7788462	74.6	90.2	75.8
	After	0.815534	75.7	85.8	78.4
Spline	Before	0.7692308	70.9	82.6	74.2
	After	0.7692308	70.9	82.6	74.2
Numeric	Before	0.7596	73	71.7	83.4
	After	0.7572816	70.4	69.4	83.5

The next problem to check is the variation within White, which may also be causing poor performance. To investigate further the distribution of data with white group was studied . The

analysis is shown in the histogram (Chart 11) and the density plot of a feature extracted from the curve.

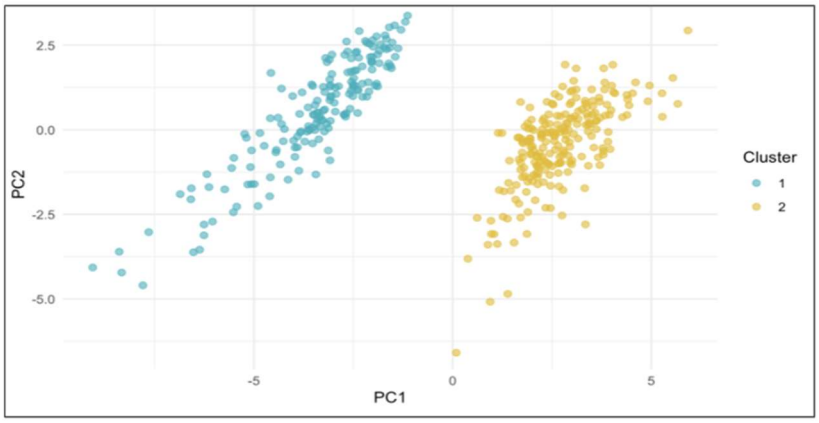
Chart 11: Spline coefficient -Variation within Data of White Race



Right_D1 is a feature extracted from the curve and was used as input in the machine learning model. Upon examining its distribution, we observed significant variation within the White race. This indicates that even though White race data constituted the majority, some of our machine learning models failed to capture its pattern and did not predict as expected for this race.

To solve the problem, K-means was adopted to perform cluster analysis to cluster the data points to groups where they belong to according to the nearest mean value. The cluster analysis is illustrated in the graph below. The image shows how the group of data points as a whole is being separated into two clusters, where the data points are being distributed through the similarity in their group means.

Chart 12: Cluster Analysis - White data



We randomly selected 100 samples from one cluster to match other race data count and then ran the machine learning algorithm. As shown in the chart below, the White model's predictions are performing significantly better after the cluster analysis, indicated that the variation in White data is likely to be the major cause to the initial poor performance.

Table 9: Cluster Analysis - Accuracy and AUC for Both

Cluster Analysis	Accuracy	AUC for White	AUC for African Amer	AUC for Asian
Before	0.7788462	74.6	90.2	75.8
After	0.8409091	89.8	84.4	96.1

8.2 Other Machine Leaning Models

Besides logistic regression, four other types of Machine Learning models were developed . The first method adopted is random forest, which is a robust machine learning method used for regression. It is one of the tree-based methods that are very easy to interpret. Each tree is built from a different sample of the data set, and the best class is chosen as the final result among all the

trees. The method then averages the predictions from all the trees to form the final regression model. Randomness is ensured for this process because each tree is picked from a random subset of data.

In a normal tree-based method, a very strong predictor in the data set is going to influence all the trees at the very top level, then all of them will be correlated and thus give poor prediction results. Random forest solves this problem by making each individual tree base on a unique subset of predictors. In this way, the very strong predictor will not influence the final regression result in a very large scale. This advantage of the random forest method will make the model robust against overfitting.

The AUC results are shown in the below chart. All the values for both the numeric variables and spline coefficients are relatively high, which validates Random Boosting as an appropriate machine learning method to analyze our data set. This shows that including crotch curve features in the model improves model prediction. Additionally, the model captures the variation within white data well and still performs as expected. This mostly is because of the underlying structure of the Random Forest Model itself.

Table 10: Random Forest - Accuracy and AUC

Data	Accuracy	AUC for AfriAmeri	AUC for Asian	AUC for White
Both	81.95	73.9	73.1	74.5
Numeric	71.43	57.3	52.6	53.5
Spline Coeff	75.47	53.0	56.2	54.3

The second method we use is Support Vector Machine. It is a direct method mainly for the purpose of classification. The core of the method is to find a hyperplane that separates classes in the vector space. When the classes can be separated clearly, SVM is a better approach than logistic regression, otherwise, they are similar approaches.

The biggest advantage of SVM is that it can deal with high-dimensional data by using kernel functions to convert the data to a higher dimension where a hyperplane can separate the classes. The method is particularly good at arranging data points nearest to the decision boundary. It is commonly applied in analyzing complex datasets.

The AUC results are shown in Table 11 below. As shown by the numbers, SVM is not performing as well as other models for the given data set because the AUC values for both numeric variables and spline coefficients are very low, showing that the predictions are not as expected.

Table 11: Support Vector Machine - Accuracy and AUC

Data	Accuracy	AUC for AfriAmeri	AUC for Asian	AUC for White
Both	71.43	60.9	52.6	54.4
Numeric	72.64	53.0	53.1	52.7
Spline Coeff	69.52	49.5	50.0	49.3

The third method is gradient boosting, which is a machine learning method that is effective in classification and regression. The method mainly builds on decision trees to enhance them to form a strong model for prediction. Gradient boosting involves the addition of predictors one by one,

and each model in the next level corrects and improves the previous one. The internal algorithm of gradient boosting is to optimize the loss function, which enhances the accuracy of the model in every next step.

The tuning parameters of boosting are significant in the learning process. The number of trees is important because it may cause overfitting problems. The shrinkage parameter affects the learning rate. The number of splits, which is also the interaction depth, manages the complexity of the model. All these parameters are essential to the algorithm because they contribute to the balance between bias and variance of the final fitted model.

The AUC results are shown in the chart below, which shows that gradient boosting is not performing well for the data set because the values for both numeric variables and spline coefficients are not higher than the values for numeric variables and spline coefficients individually, showing that including crotch curve features in the model has not improved model prediction effectively.

Table 12: Gradient Boosting - Accuracy and AUC

Data	Accuracy	AUC for AfriAmeri	AUC for Asian	AUC for White
Both	73.1	80.1	77.0	69.8
Numeric	72.1	70.2	71.9	64.8
Spline Coeff	76.9	85.7	77.6	71.4

The final method is extreme gradient boosting which is one type of gradient boosting. XGBoost follows the big concept of gradient boosting with the difference that XGBoost adopts advanced

regularization, which can potentially lead to better outcomes than the regular gradient boosting. Just like gradient boosting, XGBoost builds models one by one to enhance the accuracy in each step by minimizing the loss functions. The biggest advantage of XGBoost is that it can provide accurate results at a relatively fast rate.

The numeric results of XGBoost are shown in the chart below. The values for both numeric variables and spline coefficients are markedly higher than their corresponding numbers individually, which indicates that XGBoost is an effective machine learning method in showing that including crotch curve features in the model has improved model prediction effectively and productively.

Table 12: Xtreme Gradient Boosting - Accuracy and AUC

Data	Accuracy	AUC for AfriAmeri	AUC for Asian	AUC for White
Both	80.95	77.0	74.0	76.6
Numeric	76.19	72.8	62.9	70.1
Spline Coeff	77.36	63.2	58.8	64.3

9 Conclusion

The features derived from spline regression and high-degree polynomial curve fitting along with the numeric variables have proven to significantly enhance the accuracy of the ethnicity prediction model. Spline regression that can create smooth models by separating piecewise polynomials successfully extracts features from the curve data provided by the coordinates in the original data

set, which are incorporated through various machine learning methods along with the numeric variables to enhance the prediction accuracy.

Despite the presence of significant variation within the White ethnic group, models like random forest and XGBoost have shown proficiency and effectiveness in extracting and illustrating the patterns from the data. These machine learning methods use algorithms to combine the outcomes of decision trees or improve the outcomes of the trees, providing strength and stability of the final prediction.

The effectiveness of these models is evident compared with the simpler models that only rely on individual characteristics as predictors. The holistic approach integrates efficient statistical modeling with cutting–edge data science knowledge and demonstrates the superior capacity for precise prediction about ethnicity, which will provide profound insights and impacts on improving clothing designs and understanding similarities and differences of humans.

Additionally, it is limited what the team can do on this huge topic in this period. One promising future study on this topic is to use advanced machine learning techniques to incorporate age , genetic data into the existing models.

10 Bibliography

Gu, Bingfei, et al. “Predicting distance ease distributions on crotch curves of customized female pants”. International Journal of Clothing Science and Technology, vol. 36, page 47 – 59.

Hu, Yupeng, et al. “An image-based shape analysis approach and its application to young women’s waist-hip-leg position”. Ergonomics. DOI: 10.1080/00140139.2023.2184366.

Alrushaydan, Tarfah, et al. “Enhancing pattern construction by body scanning: the importance of curves”. 11th International Conference and Exhibition on 3D Body Scanning and Processing Technologies, No. 56.

Ning, Yang. “STSCI 5740 Data Mining and Machine Learning”. Lecture notes. Cornell University

11 Project R-code

11.1 Initial Data manipulation

---

title: "Enhancing Clothing Design through Anthropometric Data Analysis"

author: "Lily Bharati Sharma (ls989), Xinru Zheng(xz844), Xuran Zhang(xz854), Yanan Zhang (yz2999), Yuze Gu (yg348) "

date: "2024-05-12"

output: html_document

11.1.1 ## README

Tailored for Diversity: Adapting Garment Patterns to Fit Diverse Body Shapes

This project leverages anthropometric data from the CAESAR database to enhance the design and fit of pants across diverse ethnic groups. By accurately analyzing body measurement patterns, particularly the lower torso, and utilizing machine learning techniques, our team aims to predict crotch curves and identify ethnicity-based differences. The research, sponsored by Dr. Fatma Baytar from Cornell University, uses a spline model for curve fitting and combines these features with traditional anthropometric variables to improve predictions of ethnicity and inform adjustments in garment patterns for better inclusivity and comfort.

11.1.2 ## Prerequisites

Before you can run the examples provided, ensure you have the following R packages installed:

```{r install_packages, eval=FALSE}

install.packages("ggplot2")

install.packages("dplyr")

install.packages("readxl")

install.packages("tidyr")

install.packages("reshape2")

install.packages("splines")

install.packages("segmented")

install.packages("MASS")

install.packages("nnet")

install.packages("caret")

install.packages("pROC")

install.packages("cluster")

install.packages("gridExtra")

install.packages("xgboost")

install.packages("lightgbm")

install.packages("adabag")

install.packages("randomForest")

install.packages("tidyverse")

install.packages("openxlsx")

library(openxlsx)

library(readxl)

library(dplyr)
```

```
library(ggplot2)

library(tidyr)

library(reshape2)

library(MASS)

library(segmented)

library(splines)

library(nnet)

library(caret)

library(pROC)

library(cluster)

library(xgboost)

library(lightgbm)

library(adabag)

library(randomForest)

library(gridExtra)

library(tidyverse)
```

##Usage

To use this project's functionalities, you first need to install and load the necessary R package, then utilize the provided functions to analyze anthropometric data and predict crotch curves and ethnicity.

##Parameters

data: A data frame where each row represents an individual and columns represent their body measurements and race.

output_results: A combination of both data and the spline model coefficients.

##Authors

Lily Bharati Sharma (ls989)

Xinru Zheng(xz844)

Xuran Zhang(xz854)

Yanan Zhang (yz2999)

Yuze Gu (yg348)

##Acknowledgments

You might see some "Warning" while running the code, do not stop generating the code. You are on the right track.

```{r}

#Raw data merge file

#Asian data replace with appropriate Path

# Replace the path appropriately

zip_file_path_Asian <- "F:/Graduate/Project/AsianWaistXYpoints.zip"  # refer data AsianWaistXYpoints.zip

csv_file_path_Asian <- "F:/Graduate/Project/AsianWaistXYpoints.csv"

xlsx_file_path_Asian <-"F:/Graduate/Project/AsianWaist.xlsx"

merged_output_Asian  <- "F:/Graduate/Project/merged_AsianWaist.xlsx"# path to output

#Hispanic data replace with appropriate Path
```

```
zip_file_path_hispanic <-
"C:/Users/lilyb/OneDrive/Desktop/Project_capstone/HispanicWaistXYpoints.zip"

csv_file_path_hispanic <-
"C:/Users/lilyb/OneDrive/Desktop/Project_capstone/HispanictXYpoints.csv"

xlsx_file_path_hispanic <-
"C:/Users/lilyb/OneDrive/Desktop/Project_capstone/Hispanic.xlsx"

merged_output_hispanic <- "C:/Users/lilyb/OneDrive/Desktop/Project_capstone/" # path
to output


#White data replace with appropriate Path


zip_file_path_White <- "F:/Graduate/Project/WhiteWaistXYpoints.zip"

csv_file_path_White <- "F:/Graduate/Project/WhiteWaistXYpoints.csv"

xlsx_file_path_White <-"F:/Graduate/Project/WhiteWaist.xlsx"

merged_output_White <- "/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI" # path
to output


#AfricanAmer data replace with appropriate Path

zip_file_path_AfricanAmerican <-
"/Users/gracezhang/Desktop/AfricanAmerWaistXYpoints.zip"

csv_file_path_AfricanAmerican <-
"/Users/gracezhang/Desktop/AfricanAmerWaistXYpoints.csv"

xlsx_file_path_AfricanAmerican <- "/Users/gracezhang/Desktop/AfricanAmerWaist.xlsx"

merged_output_AfricanAmerican  <-
"/Users/gracezhang/Desktop/merged_AfricanAmerWaist.xlsx

5999/Dataset/OtherWaistXYpoints/merged_WhiteWaist.xlsx" # path to output
```

```
#Other race data replace with appropriate Path

zip_file_path_other <- "/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI
5999/Dataset/OtherWaistXYpoints.zip"

csv_file_path_other <- "/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI
5999/Dataset/OtherWaistXYpoints.csv"

xlsx_file_path_other <- "/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI
5999/Dataset/OtherWaistXYpoints.csv"

merged_output_other <- "/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI
5999/Dataset/OtherWaistXYpoints.csv"

# Input path of the data
# Specify the file path
file_path <- "/Users/gracezhang/Desktop/STSCI5999/data.xlsx"  # Replace with the actual
file path
# Define the output directory
output_dir <- "/Users/gracezhang/Desktop/STSCI5999/Data"

# Define the output Excel file path
output_excel <- "/Users/gracezhang/Desktop/STSCI5999/Data/output_results.xlsx"

```
```
```

11.1.3 Data merge (White)

R Markdown

White Race original data merge with xy coordinates

```{r}

unzip(zip_file_path_White)

file_list <- list.files(pattern = "*.txt")

df <- data.frame(File_Name = character(), Content = character(), stringsAsFactors = FALSE)

for (file_name in file_list) {
 content <- readLines(file_name, warn = FALSE)
 df <- rbind(df, data.frame(Subject = sub("^csr(.*)a\\.txt$", "\\1",file_name), Data_Points =
paste(content, collapse = "\n"), stringsAsFactors = FALSE))
}
print(df)

```

```{r}
#save in csv white data after merge
write.csv(df, file = csv_file_path_White, row.names = FALSE)
```

```

```{r}

csv_data <- read.csv(csv_file_path_White) #Read white csv file

wb <- loadWorkbook(xlsx_file_path_White)  #load white xlsx file

xlsx_data <- read.xlsx(wb)

merged_data <- merge(xlsx_data, csv_data, by.x = "Subject", by.y = "Subject", all.x = TRUE)
merged_data <- mutate(merged_data, race = 'White')

writeData(wb, sheet = "Sheet1", x = merged_data, startCol = 1, startRow = 1, colNames = TRUE)

saveWorkbook(wb, xlsx_file_path_White)

```

11.1.4 Data merge (Other race)

#Other race data merge
```

```{r}

unzip(zip_file_path_other)

file_list <- list.files(pattern = "*.txt")

df <- data.frame(File_Name = character(), Content = character(), stringsAsFactors = FALSE)

for (file_name in file_list) {
 content <- readLines(file_name, warn = FALSE)
 df <- rbind(df, data.frame(Subject = sub("^csr(.*)a\\.txt$", "\\1",file_name), Data_Points =
paste(content, collapse = "\n"), stringsAsFactors = FALSE))
}

print(df)

write.csv(df, file = csv_file_path_other, row.names = FALSE)

csv_data <- read.csv(csv_file_path_other)

wb <- loadWorkbook(xlsx_file_path_other)
```

```
xlsx_data <- read.xlsx(wb)

merged_data <- merge(xlsx_data, csv_data, by.x = "Subject", by.y = "Subject", all.x = TRUE)
merged_data <- mutate(merged_data, race = 'Other')

writeData(wb, sheet = "Sheet1", x = merged_data, startCol = 1, startRow = 1, colNames = TRUE)

saveWorkbook(wb,merged_output_other )
```

11.1.5 Data merge (AfricanAmerican)

#Data merge code (AfricanAmerican)

```{r}

unzip(zip_file_path_AfricanAmerican)

file_list <- list.files(pattern = "*.txt")

df <- data.frame(File_Name = character(), Content = character(), stringsAsFactors = FALSE)

for (file_name in file_list) {
 content <- readLines(file_name, warn = FALSE)
 df <- rbind(df, data.frame(Subject = sub("^csr(.*)a\\.txt$", "\\1",file_name), Data_Points =
```

```
paste(content, collapse = "\n"), stringsAsFactors = FALSE))
}

print(df)


write.csv(df, file = csv_file_path_AfricanAmerican, row.names = FALSE)


csv_data <- read.csv(csv_file_path_AfricanAmerican)

wb <- loadWorkbook(xlsx_file_path_AfricanAmerican)

xlsx_data <- read.xlsx(wb)

merged_data <- merge(xlsx_data, csv_data, by.x = "Subject", by.y = "Subject", all.x = TRUE)
merged_data <- mutate(merged_data, race = 'AfricanAmer')

writeData(wb, sheet = "Sheet1", x = merged_data, startCol = 1, startRow = 1, colNames =
TRUE)

saveWorkbook(wb, merged_output_AfricanAmerican  )
```
```
```

11.1.6 Data merge (Asian)

Data merge Code (Asian)

```{r}

unzip(zip_file_path)

file_list <- list.files(pattern = "*.txt")

df <- data.frame(File_Name = character(), Content = character(), stringsAsFactors = FALSE)

for (file_name in file_list) {
  content <- readLines(file_name, warn = FALSE)
  df <- rbind(df, data.frame(Subject = sub("^csr(.*)a\\.txt$", "\\1",file_name), Data_Points = paste(content, collapse = "\n"), stringsAsFactors = FALSE))
}

print(df)

```

```{r}
# creating CSV file of the data
```

```
write.csv(df, file = csv_file_path, row.names = FALSE)

```

```{r}

csv_data <- read.csv(csv_file_path_Asian)

wb <- loadWorkbook(xlsx_file_path_Asian)

xlsx_data <- read.xlsx(wb)

merged_data <- merge(xlsx_data, csv_data, by.x = "Subject", by.y = "Subject", all.x = TRUE)

merged_data <- mutate(merged_data, race = 'Asian')

writeData(wb, sheet = "Sheet1", x = merged_data, startCol = 1, startRow = 1, colNames = TRUE)

saveWorkbook(wb, merged_output_Asian)
```

11.1.7 Data merge (Hispanic)

```
#       Data merge code (Hispanic)

## Data-coordinates of each subject appended for Hispanic race
```

```{r}

unzip(zip_file_path_hispanic)

file_list <- list.files(pattern = "*.txt")

df <- data.frame(File_Name = character(), Content = character(), stringsAsFactors = FALSE)

for (file_name in file_list) {
 content <- readLines(file_name, warn = FALSE)
 df <- rbind(df, data.frame(Subject = sub("^csr(.*)a\\.txt$", "\\1",file_name), Data_Points =
paste(content, collapse = "\n"), stringsAsFactors = FALSE))
}

print(df)

write.csv(df, file = csv_file_path_hispanic, row.names = FALSE)

csv_data <- read.csv(csv_file_path_hispanicn)

wb <- loadWorkbook(xlsx_file_path_hispanic)
```

```
xlsx_data <- read.xlsx(wb)

merged_data <- merge(xlsx_data, csv_data, by.x = "Subject", by.y = "Subject", all.x = TRUE)

merged_data <- mutate(merged_data, race = 'AfricanAmer')

writeData(wb, sheet = "Sheet1", x = merged_data, startCol = 1, startRow = 1, colNames = TRUE)

saveWorkbook(wb, merged_output_hispanic  )
```

11.1.8 Data merge (All race)

```
#         Data Merge all Race



# combining all the Race data along with respecting x-y coordinate of Curve
```{r}
library(readxl)
library(dplyr)

file_paths <- c(merged_output_Asian,
merged_output_Other,merged_output_AfricanAmerican, merged_output_White,
merged_output_hispanic) # List of file paths
dfs <- lapply(file_paths, read_excel) # Read each Excel file into a list of data frames
```
```

```{r}
library(readxl)
# Read the Excel file into a data frame
df <- read_excel(file_path)

# View the first few rows of the data frame
head(df)

# List names of variables
variable_names <- names(df)

# Print variable names
print(variable_names)

```

11.2 Data Overview code

Summary

```{r}
#summary(df)

library(dplyr)
library(ggplot2)
```

```
# Function to calculate summary statistics for a variable
calc_summary <- function(x) {
  c(Mean = mean(x, na.rm = TRUE),
    Median = median(x, na.rm = TRUE),
    Min = min(x, na.rm = TRUE),
    Max = max(x, na.rm = TRUE))
}

# Apply the function to each numeric variable in the dataset
numeric_vars <- Filter(is.numeric, df)
summary_table <- summarise_if(numeric_vars, is.numeric, calc_summary)
summary_table
# Load the tibble package
library(tibble)

# Convert tibble to data frame
summary_table <- as.data.frame(summary_table)

# Add row names as a column
summary_table <- rownames_to_column(summary_table, var = "Variable")

# Print the summary table
print(summary_table)
```

11.2.1 Summary of data, Histogram, Stacked Bar graph distribution

```
#Histogram of each variable
```

```
#hist(data$`Max-Hip`)

# Set the width of the histogram bars
bar_width <- 0.5

# Iterate through each column in the dataframe
for (col in names(df)) {
  # Check if the column is numeric
  if (is.numeric(df[[col]])) {
    # Generate a histogram for numeric columns
    hist(df[[col]], main = paste("Histogram of", col),
         xlab = col, ylab = "Frequency",
         col = "skyblue", width = bar_width)
  } else {
    # For non-numeric columns, you can skip or handle them differently
    cat("Skipping non-numeric column:", col, "\n")
  }
}
```

#Percentage of each race

```{r}
```

```
# Calculate the percentage of each race
race_percentage <- df %>%
  group_by(race) %>%
  summarise(count = n()) %>%
  mutate(percentage = count / sum(count))

# Plot the graph
ggplot(race_percentage, aes(x = race, y = percentage, fill = race)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = scales::percent(percentage)),
            position = position_stack(vjust = 0.5),
            size = 3, color = "white") +  # Add text labels on bars
  labs(title = "Percentage of Each Race",
       x = "Race",
       y = "Percentage") +
  scale_y_continuous(labels = scales::percent_format()) +
  theme_minimal()
```

11.2.2 Box plot by race code

#BMI variable distribution by race

```{r}
library(dplyr)
```

```
library(ggplot2)

# Assuming 'data' is your dataset containing BMI and race variables

# Round down BMI to zero decimal digits
df$rounded_BMI <- floor(df$BMI)
data_clean <- na.omit(df) # Remove rows with NA values
frequency_table <- table(df$race, df$rounded_BMI)



# View the frequency table
print(frequency_table)
# Group by race and BMI, calculate count and percentage
data_summary <- df %>%
  group_by(rounded_BMI, race) %>%
  summarise(count = n()) %>%
  group_by(rounded_BMI) %>%
  mutate(percentage = count / sum(count) * 100)

# Create 100% stacked bar plot
ggplot(data_summary, aes(x = as.factor(rounded_BMI), y = percentage, fill = race)) +
  geom_bar(stat = "identity", position = "fill") +
  labs(title = "100% Stacked Bar Graph of BMI by Race",
       x = "BMI",
       y = "Percentage",
       fill = "Race") +
```

```
  scale_y_continuous(labels = scales::percent_format()) +
  theme_minimal()
```

#Height by Race

```{r}
# Plot the box plot with y-axis breaks at intervals of 50
ggplot(df, aes(x = race, y = Height, fill = race)) +
  geom_boxplot() +
  labs(title = "Height by Race",
       x = "Race",
       y = "Height")  +
  theme_minimal() +
  scale_y_continuous(breaks = seq(0, max(df$Height), by = 50)) # Set breaks at intervals of 50
```

#Depth by race

```{r}
library(ggplot2)

# Remove missing values from the 'Anterior_posterior_Length' column
```

```
clean_df <- na.omit(df$Depth )

# Plot the box plot with y-axis breaks at intervals of 50

ggplot(df, aes(x = race, y = Depth, fill = race)) +

 geom_boxplot() +

 labs(title = "Depth by Race",

     x = "Race",

     y = "Depth")  +

 theme_minimal() +

 scale_y_continuous(breaks = seq(0, max(clean_df), by = 30)) # Set breaks at intervals of
50


```

11.2.3 Box plot by Race code-Over all view

```
#         Box plot by Race code-Over all view


```{r}

library(readxl)

library(dplyr)

library(ggplot2)


# Step 1: Read the Excel file

# data <- read_excel("data.xlsx")

data <- df

summary(data)
```
```

```
 # You may need to adjust the column names to exactly match those in your dataset
data_long <- data %>%
  pivot_longer(cols = c("Depth",
                        "Anterior-posterior.Length"
                        , "Front.Crotch.(Left.side)",
                        "Back.Crotch.(Right.Side)"),
               names_to = "Measurement",
               values_to = "Value")

data_stats <- data_long %>%
  group_by(Measurement) %>%
  summarise(Q1 = quantile(Value, 0.25, na.rm = TRUE),
            Q3 = quantile(Value, 0.75, na.rm = TRUE),
            IQR = Q3 - Q1,
            Lower_Bound = Q1 - 1.5 * IQR,
            Upper_Bound = Q3 + 1.5 * IQR)

print(data_stats)

ggplot(data_long, aes(x = Measurement, y = Value, fill = Measurement)) +
  geom_boxplot() +
  theme_minimal(base_size = 14) +  # Increase the base font size for all text
  scale_fill_brewer(palette = "Set2") +  # Use a color palette that is clear and distinct
  labs(title = "Boxplot of Various Measurements", y = "Value", x = "") +
  theme(
    plot.title = element_text(face = "bold", size = 12),         # Make the title bold and larger
```

```
  axis.text.x = element_text(angle = 45, hjust = 1, size = 14),  # Rotate X axis labels for better readability
  axis.text.y = element_text(size = 14),                 # Increase Y axis text size
  axis.title = element_text(size = 16),                  # Increase axis titles size
  legend.position = "none",                              # Remove legend if not needed
  panel.grid.major.x = element_line(color = "grey80", size = 0.5),  # Add major grid lines for x
  panel.grid.minor.x = element_blank(),                  # Remove minor grid lines for x
  panel.spacing = unit(10, "lines"),                     # Adjust spacing between panels
  plot.margin = unit(c(0.15, 0.15, 0.15, 0.15), "cm")                # Adjust plot margins
 )+
 coord_flip()
```

11.2.4 Outlier Detection

#Outlier

```{r}
library(tidyr)
library(ggplot2)

data_long <- data %>%
 pivot_longer(cols = c("Height", "Max.Hip", "Crotch.curve.length.at.back.waist"),
         names_to = "Measurement",
         values_to = "Value")

data_stats <- data_long %>%
```

```
 group_by(Measurement) %>%
 summarise(Q1 = quantile(Value, 0.25, na.rm = TRUE),
        Q3 = quantile(Value, 0.75, na.rm = TRUE),
        IQR = Q3 - Q1,
        Lower_Bound = Q1 - 1.5 * IQR,
        Upper_Bound = Q3 + 1.5 * IQR)

print(data_stats)

# Assuming data_long has been created from your dataset as previously described
ggplot(data_long, aes(x = Measurement, y = Value, fill = Measurement)) +
 geom_boxplot() +
 theme_minimal() +
 labs(title = "Boxplot of Various Measurements Across Individuals", y = "Value", x = "") +
 theme(plot.title = element_text(face = "bold", size = 20),
     axis.title.x = element_text(face = "bold", size = 14),
     axis.title.y = element_text(face = "bold", size = 14),
     axis.text.x = element_text(size = 12, angle = 45, hjust = 1),
     axis.text.y = element_text(size = 12),
     legend.position = "none",
     panel.grid.major = element_line(color = "grey80", size = 0.5),
     panel.grid.minor = element_blank()) +
 coord_flip() +
 scale_y_continuous(breaks = function(x) pretty(x, n = 20))  # Adjust n for more or less
granularity
```

```

### 11.2.5 Correlation analysis -all the variables

# Correlation analysis -all the variables

```{r}

# excel_data <- read_excel("/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI 5999/data.xlsx")

excel_data <- df

excel_data = select_if(excel_data, is.numeric)

# Remove rows with any missing values

clean_data <- na.omit(excel_data)

# Step 3: Perform correlation analysis

correlation_matrix <- cor(clean_data[-1])

# View correlation matrix

print(correlation_matrix)

melted_corr <- melt(correlation_matrix)

ggplot(melted_corr, aes(Var1, Var2, fill = value)) +

  geom_tile() +

  geom_text(aes(label = sprintf("%.2f", value)), vjust = 1, color = "white", size = 3) +

  scale_fill_gradient2(low = "white", high = "pink", mid = "blue", midpoint = 0, limit = c(-1, 1), space = "Lab", name="Correlation") +

  theme_minimal() +

  theme(axis.text.x = element_text(angle = 45, hjust = 1),
```
```

```
      axis.text.y = element_text(angle = 45, hjust = 1)) +
  labs(title = "Correlation Matrix Heatmap", x = "Variables", y = "Variables")
```

11.3 Crotch Curve Fitting code for all the subjects -Spline regression

Crotch Curve Fitting code for all the subjects -Spline regression

```{r}
library(readxl)
library(segmented)
library(dplyr)
library(splines)

# Read the excel file
excel_data <- df

# Define the output directory
#output_dir <- "C:/Users/lilyb/Box/Capstone_Proj_Internal_Team/Data"

# Define the output Excel file path
#output_excel <- "C:/Users/lilyb/Box/Capstone_Proj_Internal_Team/Data/output_results.xlsx"

# Print rows with empty Data_Points column before deletion
cat("Rows with empty Data_Points column before deletion:\n")
```

```
print(excel_data[is.na(excel_data$Data_Points), c("Subject", "Data_Points")])

# Filter out rows with empty Data_Points
excel_data <- excel_data %>% filter(!is.na(Data_Points))

# Print rows with empty Data_Points column after deletion
cat("\nRows with empty Data_Points column after deletion:\n")
print(excel_data[is.na(excel_data$Data_Points), c("Subject", "Data_Points")])




# Define a function to process each row
process_row <- function(row, row_number) {
 data <- as.character(row$Data_Points)

 lines <- strsplit(data, "\n", fixed = TRUE)[[1]]

 x_values <- numeric()
 y_values <- numeric()

 for (line in lines) {
  values <- strsplit(line, ",")[[1]]
  x_values <- c(x_values, as.numeric(values[1]))
  y_values <- c(y_values, as.numeric(values[2]))
 }
```

```
 df <- data.frame(x = x_values, y = y_values)


 #df$y <- df$y*-1
 if (any(df$y < 0)) {
 df$y <- df$y
} else {
 df$y <- df$y * -1
}


 max_index <- which.max(df$x)
 min_index <- which.min(df$x)
 max_point <- df[max_index, ]
 min_point <- df[min_index, ]


 df2 <- df[df$y < min(df$y[max_index], df$y[min_index]), ]


 lm.mod <- lm(y ~ x, data = df2)



start1 = ((min(df2$x)+max(df2$x))/2 + min(df2$x))/2
start2 = ((min(df2$x)+max(df2$x))/2 + max(df2$x))/2


#Segmented regression
model.segmented2 <- segmented(lm.mod, seg.Z = ~x, psi = c(start1,start2), control =
seg.control(quant = TRUE, display = FALSE))
```

```
#Extract first & second breakpoints value estimated by segmented regression
psi1 <- model.segmented2$psi[1, 2]
psi2 <- model.segmented2$psi[2, 2]

# extract data points which is middle portion i.e between first cutoff psi1 and psi2

middle <- df2[df2$x > psi1 & df2$x < psi2, ]

# Find all the data points which is not middle datapoints

points_only_in_df <- anti_join(df, middle, by = c("x", "y"))

#Using k-mean sepearte the left and right part of the curve
k <- 2
set.seed(123)
kmeans_result <- kmeans(points_only_in_df, centers = k)
cluster_assignments <- kmeans_result$cluster
Left <- points_only_in_df[cluster_assignments == 1, ]
Right <- points_only_in_df[cluster_assignments == 2, ]

#Fitting a spline model of degree 4 to the data. Using Basis Spline Transformation (bs) of x (predictor variable with degree of 4) to capture or fit non-linear relationship

fit_middle <- lm(y ~ bs(x, degree = 2), data = middle)
```

```
fit_left <- lm(x ~ bs(y, degree = 3), data = Left)
fit_right <- lm(x ~ bs(y, degree = 3), data = Right)

coef_middle <- as.data.frame(t(coef(fit_middle)))
colnames(coef_middle) <- c('Mid_I', 'Mid_D1', 'Mid_D2')
coef_left <- as.data.frame(t(coef(fit_left)))
colnames(coef_left) <- c('Left_I', 'Left_D1', 'Left_D2', 'Left_D3')
coef_right <- as.data.frame(t(coef(fit_right)))
colnames(coef_right) <- c('Right_I','Right_D1', 'Right_D2', 'Right_D3')

df_coef <- data.frame(coef_left, coef_middle, coef_right)

# For plot
  # Get the row number
#row_number <- as.numeric(rownames(row))

# Extract the subject from the test_data dataframe
subject <- as.character(row$Subject)

# Plot the original data
plot(df$x, df$y, main = paste("Spline Fit to Data - Row:", row_number, ", Subject:",
subject))

# Add the spline curve
y.range_left <- range(Left$y)
y.seq_left <- seq(from = y.range_left[2], to = y.range_left[1], by = -0.1)
```

```
  predict_left <- data.frame(x = predict(fit_left, data.frame(y = y.seq_left)), y = y.seq_left)


  y.range_right <- range(Right$y)

  y.seq_right <- seq(from = y.range_right[1], to = y.range_right[2], by = 0.1)

  predict_right <- data.frame(x = predict(fit_right, data.frame(y = y.seq_right)), y = y.seq_right)


  x.seq_middle <- seq(from = tail(predict_left$x, 1), to = head(predict_right$x, 1))

  predict_middle <- data.frame(x = x.seq_middle, y = predict(fit_middle, data.frame(x = x.seq_middle)))


  lines(x.seq_middle, predict(fit_middle, data.frame(x = x.seq_middle)), col = 'blue')

  lines(predict(fit_left, data.frame(y = y.seq_left)), y.seq_left, col = 'red')

  lines(predict(fit_right, data.frame(y = y.seq_right)), y.seq_right, col = 'green')


  predict_df <- rbind(predict_left, predict_middle, predict_right)


  return(list(df_coef = df_coef, predict_df = predict_df))

}


```


```{r}

# Apply the function to each row of the dataframe

output_list <- mapply(process_row, row = split(excel_data, seq_len(nrow(excel_data))), row_number = seq_len(nrow(excel_data)), SIMPLIFY = FALSE)
```

```
# Extract the results
df_coef_list <- lapply(output_list, function(x) x$df_coef)
predict_df_list <- lapply(output_list, function(x) x$predict_df)
```

```{r}
library(openxlsx)

# Initialize empty lists to store df_coef and predict_df
df_coef_combined <- list()
# predict_df_combined <- list()

# Iterate through the output_list
for (i in seq_along(output_list)) {
  # Extract the current df_coef and predict_df
  df_coef <- output_list[[i]]$df_coef

  # Add subject column to df_coef
  df_coef$Subject <- excel_data$Subject[i]



  # Merge with original data to include all columns
  #df_coef <- merge(df_coef, excel_data, by.x = "Subject", by.y = "Subject", all.x = TRUE)

  df_coef <- merge(df_coef, excel_data, by = "Subject", all.x = TRUE, suffixes = c("", ""))
  # Append to combined lists
```

```
  df_coef_combined[[i]] <- df_coef

}

# Combine all df_coef dataframes into one dataframe
df_coef_combined <- do.call(rbind, df_coef_combined)

```

```{r}
# Write to Excel file
write.xlsx(list(df_coef_combined = df_coef_combined),
        file = output_excel)
```

11.3.1 MANOVA R code

MANOVA R code

```{r}
library(readxl)
#data <- read_excel("/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI 5999/output_results.xlsx")
data <- read_excel(output_excel)
```

```{r}
# Select only the relevant columns for the MANOVA
```

data_selected <- data[c("Left_I", "Left_D1", "Left_D2", "Left_D3", "Mid_I", "Mid_D1", "Mid_D2", "Right_I", "Right_D1", "Right_D2", "Right_D3", "race")]

data_selected$race <- as.factor(data_selected$race)

Perform MANOVA using race as the independent variable

manova_results <- manova(cbind(Left_I, Left_D1, Left_D2, Left_D3, Mid_I, Mid_D1, Mid_D2, Right_I, Right_D1, Right_D2, Right_D3) ~ race, data = data_selected)

summary(manova_results)

#Based on a small p-value, there are statistically significant differences in the multivariate means of the dependent variables (like Left_I, Left_D1, etc.) across the different race groups

```

```{r}

#Purpose: to identify which specific race groups differ from each other

library(MASS)  # Load MASS for statistical functions

data$race <- as.factor(data$race)

# Exclude "Hispanic" and "Other" from the race factor

data_filtered <- data[!data$race %in% c("Hispanic", "Other"),]

# Remove rows with any NA values in the columns of interest

data_filtered <- na.omit(data_filtered[, c('Left_I', 'Left_D1', 'Left_D2', 'Left_D3', 'race')])

```
# Function to perform pairwise MANOVA with added checks
pairwise_manova <- function(data, response_cols, group_col) {
 levels <- levels(data[[group_col]])
 combinations <- combn(levels, 2, simplify = FALSE)
 results <- list()

 for (pair in combinations) {
  # Filter data for the current pair of groups
  sub_data <- data[data[[group_col]] %in% pair,]

  # Relevel the factor to drop unused levels
  sub_data[[group_col]] <- factor(sub_data[[group_col]])

  # Ensure we have more than one level in the group factor
  if(length(unique(sub_data[[group_col]])) < 2) {
   results[[paste(pair, collapse = " vs ")]] <- 'Insufficient data for MANOVA'
   next
  }

  # Extract response variables and corresponding group factor
  response_data <- sub_data[, response_cols]
  group_factor <- sub_data[[group_col]]

  # Fit MANOVA
  if(any(colSums(is.na(response_data)) > 0)) {
```

```
    results[[paste(pair, collapse = " vs ")]] <- 'Data contains NAs'
  } else {
    manova_result <- manova(as.matrix(response_data) ~ group_factor)
    summary_result <- summary(manova_result, test = "Pillai")
    results[[paste(pair, collapse = " vs ")]] <- summary_result
  }
 }

 return(results)
}

# Execute the pairwise MANOVA
results <- pairwise_manova(data_filtered, c('Left_I', 'Left_D1', 'Left_D2', 'Left_D3'), 'race')
print(results)



# Check if all expected columns are present
expected_cols <- c('Mid_I', 'Mid_D1', 'Mid_D2', 'Right_I', 'Right_D1', 'Right_D2', 'Right_D3',
'race')
missing_cols <- setdiff(expected_cols, names(data))

if(length(missing_cols) > 0) {
 stop("The following expected columns are missing: ", paste(missing_cols, collapse=", "))
}

# Continue with the existing columns
```

```
data_filtered <- data[!data$race %in% c("Hispanic", "Other"),]
data_filtered <- na.omit(data_filtered[, expected_cols])

# Function to perform pairwise MANOVA with added checks
pairwise_manova <- function(data, response_cols, group_col) {
 levels <- levels(data[[group_col]])
 combinations <- combn(levels, 2, simplify = FALSE)
 results <- list()

 for (pair in combinations) {
  # Filter data for the current pair of groups
  sub_data <- data[data[[group_col]] %in% pair, ]
  sub_data[[group_col]] <- factor(sub_data[[group_col]])

  # Ensure there are more than one level
  if(length(unique(sub_data[[group_col]])) < 2) {
   results[[paste(pair, collapse = " vs ")]] <- 'Insufficient data for MANOVA'
   next
  }

  response_data <- sub_data[, response_cols]
  group_factor <- sub_data[[group_col]]

  if(any(colSums(is.na(response_data)) > 0)) {
   results[[paste(pair, collapse = " vs ")]] <- 'Data contains NAs'
  } else {
```

```
    manova_result <- manova(as.matrix(response_data) ~ group_factor)
    summary_result <- summary(manova_result, test = "Pillai")
    results[[paste(pair, collapse = " vs ")]] <- summary_result
  }
 }

 return(results)
}

# Execute the pairwise MANOVA for Middle and Right parts
middle_results <- pairwise_manova(data_filtered, c('Mid_I', 'Mid_D1', 'Mid_D2'), 'race')
right_results <- pairwise_manova(data_filtered, c('Right_I', 'Right_D1', 'Right_D2', 'Right_D3'), 'race')

# Print results
print("Middle Part Results:")
print(middle_results)

print("Right Part Results:")
print(right_results)
```

```{r}
output_excel <- "/Users/gracezhang/Desktop/STSCI5999/output_results.xlsx"
df <- read_excel(output_excel)
```

```

## 11.4 Machine learning model to predict Race

### 11.4.1 Logistic Regression R code

#logisttic regressoin

```{r}

library(nnet)

library(readxl)

library(caret)

library(pROC)

library(dplyr)



df <- read_excel(output_excel)

df <- df %>%

 ## Both

  select(-c(Subject, Comments, Data_Points, Curve)) %>%

 ##Coefficient

  #select(c( Left_I, Left_D1, Left_D2, Left_D3, Mid_I, Mid_D1, Mid_D2,  Right_I, Right_D1, Right_D2, Right_D3, race))%>%



 ##Numeric

  #select(c( BMI, Max.Hip, Height, `Anterior-posterior.Length`, Depth,

```
        #Crotch.curve.length.at.back.waist, `Front.Crotch.(Left.side)`,
`Back.Crotch.(Right.Side)`, race))%>%

 filter(race %in% c("White", "AfricanAmer", "Asian")) %>% # Include only the races of
interest

  na.omit()

df$race <- factor(df$race, levels = c("White", "AfricanAmer", "Asian"))

numeric_cols <- sapply(df, is.numeric)

df[numeric_cols] <- scale(df[numeric_cols])

Split data into training and testing sets

set.seed(123)

train_indices <- createDataPartition(df$race, p = 0.8, list = FALSE)

train_data <- df[train_indices,]

test_data <- df[-train_indices,]

Fit a multinomial logistic regression model

model_multinom <- multinom(race ~ ., data = train_data)

predicted_class <- predict(model_multinom, newdata = test_data, type = "class")

predicted_probs <- predict(model_multinom, newdata = test_data, type = "probs")

conf_mat <- confusionMatrix(predicted_class, test_data$race)
```

```
print(conf_mat)
cat("Accuracy Before Stepwise:", conf_mat$overall['Accuracy'], "\n")



# ROC curve and AUC calculation for each class
# Get predicted probabilities for each class
race_categories_both <- levels(test_data$race)
roc_list_both <- list()
auc_list <- list()
colors <- c("red", "green", "blue")  # Colors for each race category
par(pin = c(4, 4), mai = c(1, 1, 0.5, 0.5))
# Calculate ROC curve and AUC for each race
for (i in 1:length(race_categories_both)) {
  response_both <- ifelse(test_data$race == race_categories_both[i], 1, 0)
  roc_obj_both <- roc(response_both, predicted_probs[, i])
  roc_list_both[[i]] <- roc_obj_both
  auc_list [[i]] <- auc(roc_obj_both)
}
colors <- c("red", "green", "blue")  # Colors for each race
for (i in seq_along(levels(test_data$race))) {
  truth <- ifelse(test_data$race == levels(test_data$race)[i], 1, 0)
  roc_obj <- roc(truth, predicted_probs[, i])
  auc_list[[levels(test_data$race)[i]]] <- auc(roc_obj)
  plot(roc_obj, main = paste("ROC for", levels(test_data$race)[i]), col = colors[i], lwd = 2,
print.auc = TRUE)
  cat("AUC for", levels(test_data$race)[i], ":", auc(roc_obj), "\n")
```

```
}

par(mfrow = c(1, 1))


print(summary(model_multinom))



```


Cluster analysis on white Data


```{r}

###If is imbalanced: randomly select 100 data from White to match the amount of data in Asian and AfrianAmer

library(nnet)

library(readxl)

library(caret)

library(pROC)

library(dplyr)


#df <- read_excel("/Users/gracezhang/Desktop/STSCI5999/output_results.xlsx")

df <-  read_excel(output_excel)

df <- df %>%


  ##Both

  select(-c(Subject, Comments, Data_Points, Curve))
```

```
 ##Coefficient
 #select(c( Left_I, Left_D1, Left_D2, Left_D3, Mid_I, Mid_D1, Mid_D2,  Right_I, Right_D1, Right_D2, Right_D3, race))

 ##Numeric
  #select(c( BMI, Max.Hip, Height, `Anterior-posterior.Length`, Depth,
        #Crotch.curve.length.at.back.waist, `Front.Crotch.(Left.side)`, `Back.Crotch.(Right.Side)`, race))

df_filtered <- df %>%
 filter(race %in% c("AfricanAmer", "Asian")) %>%
 bind_rows(
  df %>%
   filter(race == "White") %>%
   sample_n(100)
 ) %>%
  na.omit()


df$race <- factor(df$race, levels = c("White", "AfricanAmer", "Asian"))

numeric_cols <- sapply(df, is.numeric)
df[numeric_cols] <- scale(df[numeric_cols])

set.seed(123)
train_indices <- createDataPartition(df$race, p = 0.8, list = FALSE)
train_data <- df[train_indices, ]
```

```
test_data <- df[-train_indices, ]

model_multinom <- multinom(race ~ ., data = train_data)

predicted_class <- predict(model_multinom, newdata = test_data, type = "class")
predicted_probs <- predict(model_multinom, newdata = test_data, type = "probs")

conf_mat <- confusionMatrix(predicted_class, test_data$race)
print(conf_mat)
cat("Accuracy Before Stepwise:", conf_mat$overall['Accuracy'], "\n")



# ROC curve and AUC calculation for each class
# Get predicted probabilities for each class
race_categories_both <- levels(test_data$race)
roc_list_both <- list()
auc_list <- list()
colors <- c("red", "green", "blue")
par(pin = c(4, 4), mai = c(1, 1, 0.5, 0.5))
# Calculate ROC curve and AUC for each race
for (i in 1:length(race_categories_both)) {
  response_both <- ifelse(test_data$race == race_categories_both[i], 1, 0)
  roc_obj_both <- roc(response_both, predicted_probs[, i])
  roc_list_both[[i]] <- roc_obj_both
  auc_list [[i]] <- auc(roc_obj_both)
}
```

```
colors <- c("red", "green", "blue")

for (i in seq_along(levels(test_data$race))) {

  truth <- ifelse(test_data$race == levels(test_data$race)[i], 1, 0)

  roc_obj <- roc(truth, predicted_probs[, i])

  auc_list[[levels(test_data$race)[i]]] <- auc(roc_obj)

  plot(roc_obj, main = paste("ROC for", levels(test_data$race)[i]), col = colors[i], lwd = 2,
print.auc = TRUE)

  cat("AUC for", levels(test_data$race)[i], ":", auc(roc_obj), "\n")

}

par(mfrow = c(1, 1))



print(summary(model_multinom))
```



```{r}
## Use k-means to cluster White data, see if it vary


library(readxl)

library(dplyr)

library(ggplot2)

library(cluster)

#df <- read_excel("/Users/gracezhang/Desktop/STSCI5999/output_results.xlsx")

df <- read_excel(output_excel)

df_white <- df %>%
```

```
 filter(race == "White") %>%
 select(-race, -Subject, -Comments, -Data_Points, -Curve)

df_white <- na.omit(df_white)
df_white <- df_white[complete.cases(df_white), ]
df_white <- df_white[!apply(df_white, 1, function(x) any(is.infinite(x))), ]

df_white_scaled <- scale(df_white)

set.seed(123)
wss <- sapply(1:10, function(k){kmeans(df_white_scaled, centers = k, nstart =
25)$tot.withinss})
plot(1:10, wss, type = "b", pch = 19, frame = FALSE, xlab = "Number of clusters K", ylab =
"Total within-clusters sum of squares")

set.seed(123)
k <- 2
km_res <- kmeans(df_white_scaled, centers = k, nstart = 25)

df_white$cluster <- as.factor(km_res$cluster)
pca_res <- prcomp(df_white_scaled, scale. = TRUE)

pca_data <- data.frame(PC1 = pca_res$x[, 1], PC2 = pca_res$x[, 2], Cluster =
km_res$cluster)

ggplot(pca_data, aes(x = PC1, y = PC2, color = as.factor(Cluster))) +
```

```
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("#00AFBB", "#E7B800")) +
  theme_minimal() +
  labs(color = "Cluster") +
  ggtitle("PCA and Clustering Results")


```

```{r}
## logistic for both

#df <- read_excel("/Users/gracezhang/Desktop/STSCI5999/output_results.xlsx")
df <- read_excel(output_excel)

df_white$race <- "White"

other_races_data <- df %>%
 filter(race %in% c("AfricanAmer", "Asian")) %>%
 select(-Subject, -Comments, -Data_Points, -Curve) %>%
 na.omit()

set.seed(123)
sampled_data <- df_white %>%
 filter(cluster == 1) %>%
 sample_n(100) %>%
```

```
  select(-cluster)

df <- bind_rows(sampled_data, other_races_data)
df$race <- factor(df$race, levels = c("White", "AfricanAmer", "Asian"))

numeric_cols <- sapply(df, is.numeric)
df[numeric_cols] <- scale(df[numeric_cols])



set.seed(123)
train_indices <- createDataPartition(df$race, p = 0.8, list = FALSE)
train_data <- df[train_indices, ]
test_data <- df[-train_indices, ]

model_multinom <- multinom(race ~ ., data = train_data)




predicted_class <- predict(model_multinom, newdata = test_data, type = "class")
predicted_class <- factor(predicted_class, levels = levels(test_data$race))

conf_mat <- confusionMatrix(predicted_class, test_data$race)
print(conf_mat)
cat("Accuracy:", conf_mat$overall['Accuracy'], "\n")

# ROC curve and AUC calculation for each class
```

```
race_categories <- levels(test_data$race)

predicted_probs <- predict(model_multinom, newdata = test_data, type = "probs")

colors <- c("red", "green", "blue")  # Colors for each race category

par(mfrow = c(1, length(race_categories)), mai = c(1, 1, 0.5, 0.5))


for (i in seq_along(race_categories)) {

  response <- ifelse(test_data$race == race_categories[i], 1, 0)

  roc_obj <- roc(response, predicted_probs[, i])

  plot(roc_obj, main = paste("ROC for", race_categories[i]), col = colors[i], lwd = 2,
print.auc = TRUE)

  cat("AUC for", race_categories[i], ":", auc(roc_obj), "\n")

}

par(mfrow = c(4, 4))


print(summary(model_multinom))


```

Density plot only white data -variation check

```{r}

library(nnet)

library(MASS)

library(readxl)

library(caret)

library(pROC)

library(dplyr)
```

```
#df <-
read_excel("C:/Users/lilyb/Box/Capstone_Proj_Internal_Team/Data/output_results.xlsx")

df <- read_excel(output_excel)


#Exclude unnecessary columns (e.g., comments, data points, curve)

df <- select(df, -c(Comments, Data_Points, Curve))


# Exclude 'Other' and 'Hispanic' from the race column

df <- filter(df, !race %in% c("Other", "Hispanic"))


# Convert 'race' to a factor

df$race <- as.factor(df$race)


# Check for NA values and remove or impute

df <- na.omit(df)  # Removes all rows with any NA values

# Normalize only numeric columns

numeric_cols <- sapply(df, is.numeric)

df_normalized <- as.data.frame(lapply(df[, numeric_cols], function(x) (x - min(x)) / (max(x) -
min(x))))



# # Find non-numeric columns

# non_numeric_cols <- !numeric_cols

#

# # Print non-numeric columns

# non_numeric_names <- names(df)[non_numeric_cols]

# print(non_numeric_names)
```

```
#
# # Print numeric columns
# non_numeric_names <- names(df)[numeric_cols]
# print(non_numeric_names)



# Combine normalized numeric columns with non-numeric columns
df_normalized <- cbind(df[, !numeric_cols], df_normalized)

# Separate predictors (X) and target variable (Y)
X <- df_normalized[, 2:ncol(df_normalized)]  # Exclude the first column (race)
Y <- df_normalized[, 1]



set.seed(123)
# Filter the data for only White, Asian, and AfricanAmer races
filtered_data <- df
filtered_data$race <- as.factor(df$race)

train_indices <- createDataPartition(filtered_data$race, p = 0.8, list = FALSE)
train_data <- filtered_data[train_indices, ]
test_data <- filtered_data[-train_indices, ]

train_data$race <- relevel(train_data$race, ref = "White")
# train_data_last <- names(train_data)
```

```
# print(train_data_last)


```


```{r}
# Plots to see distrubution of each feature in subcategory of Race (white and AfricanAmerican)


# Load necessary libraries
if (!require(ggplot2)) {
  install.packages("ggplot2")
}
if (!require(gridExtra)) {
  install.packages("gridExtra")
}
library(ggplot2)
library(gridExtra)


# Function to plot histograms or density plots for each variable
plot_variable <- function(data_white, data_african_american, variable, race_label) {
  p1 <- ggplot() +
    geom_histogram(data = data_white, aes(x = !!sym(variable), fill = "White"), color = "white", bins = 30) +
    geom_histogram(data = data_african_american, aes(x = !!sym(variable), fill = "African American"), color = "white", bins = 30) +
    labs(title = paste("Histogram for", variable, "(", race_label, ")"), x = variable, y = "Frequency") +
```

```
  scale_fill_manual(values = c("White" = "lightblue", "African American" = "lightgreen"))


 p2 <- ggplot() +
  geom_density(data = data_white, aes(x = !!sym(variable), fill = "White"), color = "blue") +
  geom_density(data = data_african_american, aes(x = !!sym(variable), fill = "African American"), color = "darkgreen") +
  labs(title = paste("Density Plot for", variable, "(", race_label, ")"), x = variable, y = "Density") +
  scale_fill_manual(values = c("White" = "lightblue", "African American" = "lightgreen"))


 return(list(p1, p2))
}

# List of variables excluding the "race" variable
variables <- setdiff(names(df), "race")

# Subset the data for "White" and "African American" categories
df_white <- subset(df, race == "White")
df_african_american <- subset(df, race == "AfricanAmer")

# Plot histograms and density plots for each variable separately for white and African American races
for (var in variables) {
 p <- plot_variable(df_white, df_african_american, var, "White vs African American")
 grid.arrange(p[[1]], p[[2]], nrow = 2, top = paste("Variable:", var))
}
```

11.4.2 SVM, XGBOOST, Random Forest

Ml model for Numeric Coeff

```{r}
library(readxl)
library(caret)
library(xgboost)
#library(catboost)
library(lightgbm)
library(adabag)
library(pROC)
```

```{r}
# Load required libraries
library(randomForest)
library(caret)
library(pROC)

# data <- read_excel("F:/Graduate/Project/output_results.xlsx")

library(readxl)
# Read the data from Excel
#data <- read_excel("F:/Graduate/Project/output_results.xlsx")
data <- read_excel(output_excel)
```

```
# Exclude "Hispanic" and "Other" from race
data <- subset(data, race != "Hispanic" & race != "Other")

# Convert race to factor
data$race <- factor(data$race, levels = c("White", "Asian", "AfricanAmer"))

# Select relevant columns and remove missing values
df <- data[, c(1:20, 24)]
df <- na.omit(df)

# Normalize only numeric columns
numeric_cols <- sapply(df, is.numeric)
df_normalized <- as.data.frame(lapply(df[, numeric_cols], function(x) (x - min(x)) / (max(x) - min(x))))

# Combine normalized numeric columns with non-numeric columns
df_normalized <- cbind(df[, !numeric_cols], df_normalized)

# Separate predictors (X) and target variable (Y)
X <- df_normalized[, 2:ncol(df_normalized)]  # Exclude the first column (race)
Y <- df_normalized[, 1]

# Set seed for reproducibility
set.seed(123)

# Create indices for the training set
```

```
trainIndex <- sample(1:nrow(df), size = 0.8 * nrow(df))

# Split the data into training and testing sets
train_X <- X[trainIndex, ]
test_X <- X[-trainIndex, ]
train_Y <- Y[trainIndex]  # Corrected indexing
test_Y <- Y[-trainIndex]  # Corrected indexing

# Check the number of rows in each set
nrow(train_X)
nrow(test_X)

# Combine train_X and train_Y
train_data <- cbind(train_X, train_Y)
```

SVM

```{r}
ctrl <- trainControl(method = "cv", number = 5, verboseIter = TRUE, classProbs = TRUE)

tuned_svm <- train(train_Y ~ ., data = data.frame(train_X, train_Y),
                   method = "svmRadial",
                   trControl = ctrl,
                   preProcess = "scale",
                   tuneLength = 3)
```

```
# Predict using the tuned SVM model
predictions_svm <- predict(tuned_svm, newdata = test_X)
# Evaluation metrics
confusion_matrix_svm <- confusionMatrix(predictions_svm, test_Y)
roc_curve_svm <- roc(response = test_Y, predictor = as.numeric(predictions_svm))
auc_svm <- auc(roc_curve_svm)

# Output results
print(confusion_matrix_svm)
print(paste("AUC:", auc_svm))
plot(roc_curve_svm, main="ROC Curve for SVM Model")
```

Random Forest

```{r}
# Set seed for reproducibility
set.seed(123)
# Define cross-validation control
ctrl <- trainControl(method = "cv", number = 5, verboseIter = TRUE)

# Tune the random forest model
tuned_rf <- train(train_Y ~ ., data = train_data, method = "rf",
                  trControl = ctrl, tuneLength = 3)
```

```
# Get the best model
best_model <- tuned_rf$finalModel

# Make predictions on the test set
predictions <- predict(best_model, newdata = test_X)

# Evaluate the model using various metrics
confusion_matrix <- confusionMatrix(predictions, test_Y)
roc_curve <- roc(test_Y, as.numeric(predictions))
auc <- auc(roc_curve)

# Print evaluation metrics
print(confusion_matrix)
print(paste("AUC:", auc))
plot(roc_curve, main="ROC Curve for Random Forest", col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)



library(pROC)

# Compute AUC for each class separately
auc_by_class <- lapply(levels(test_Y), function(class) {
  class_predictions <- as.numeric(predictions == class)
  class_actual <- as.numeric(test_Y == class)
  roc_curve <- roc(class_actual, class_predictions)
  auc <- auc(roc_curve)
```

```
 # Plot ROC curve for each class separately
 plot(roc_curve, main = paste("ROC Curve for", class), col = "blue", lwd = 2,
     print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)

 return(auc)
})

# Print AUC for each class
names(auc_by_class) <- levels(test_Y)
print(auc_by_class)
```

```

## XGboost

```{r}
# Load necessary library
library(xgboost)
library(caret)
library(pROC)

# Convert the data to DMatrix object which is optimized for XGBoost
dtrain <- xgb.DMatrix(data = as.matrix(train_X), label = as.numeric(train_Y) - 1)
dtest <- xgb.DMatrix(data = as.matrix(test_X), label = as.numeric(test_Y) - 1)
```
```

```
# Set parameters for XGBoost


param_grid = expand.grid(
  booster = "gbtree",
  objective = "multi:softprob",
  num_class = length(levels(factor(Y))),
  eval_metric = "mlogloss",


  eta = seq(from = 0.01, to = 1, length.out = 10),  # Learning rate
  max_depth = c(3, 6, 9),   # Depth of trees
  #min_child_weight = c(1, 5, 10),  # Minimum sum of instance weight (hessian) needed in a child
  min_child_weight = 5,
  subsample = c(0.5, 0.75, 1),  # Subsample ratio of the training instances
  #subsample = 0.5,
  #colsample_bytree = c(0.5, 0.75, 1)  # Subsample ratio of columns when constructing each tree
  colsample_bytree = 0.8
)



best_auc = 0
best_params = NULL
prediction = NULL


# Loop over each row in the parameter grid
for (i in 1:nrow(param_grid)) {
```

```
  params <- list(
  booster = "gbtree",
  objective = "multi:softprob",
  num_class = 3,
  eval_metric = "mlogloss",
  eta = param_grid[i, "eta"],
  max_depth =  param_grid[i, "max_depth"],
  min_child_weight = param_grid[i, "min_child_weight"],
  subsample = param_grid[i, "subsample"],
  colsample_bytree = param_grid[i, "colsample_bytree"]
 )

  nrounds <- 100

# Train the model
  xgb_model <- xgb.train(params = params, data = dtrain, nrounds = nrounds,
                         watchlist = list(eval = dtrain, test = dtest),                         verbose = 1)


  # Make predictions
  predictions <- predict(xgb_model, newdata = dtest)
  if (params$objective == "multi:softprob") {
    max_prob_indices <- max.col(matrix(predictions, ncol = length(levels(factor(Y))), byrow
= TRUE))  # Convert probabilities to class predictions for multiclass
    predictions <- max_prob_indices - 1
  }
```

```
   # Confusion Matrix and AUC calculation using caret and pROC



   #conf_matrix <- confusionMatrix(as.factor(predictions), as.factor(as.numeric(test_Y) -
1))


   #conf_matrix$overall['Accuracy']


   roc_curve <- roc(as.numeric(test_Y) - 1, predictions, multi.class = "ovr")
   auc_score <- auc(roc_curve)



  # Check if this is the best AUC
  if (auc_score > best_auc) {
   best_auc <- auc_score
   best_params <- params
   prediction <- predictions
  }
}



# Output the best parameters
print(paste("Best AUC:", best_auc))
print("Best Parameters:")
print(best_params)
```

```
print(confusionMatrix(as.factor(prediction), as.factor(as.numeric(test_Y) - 1)))

roc_curve <- roc(as.numeric(test_Y) - 1, prediction, multi.class = "ovr")

plot(roc_curve, main="ROC Curve for XGboost")
```

```{r}
confusion_matrix = confusionMatrix(as.factor(prediction), as.factor(as.numeric(test_Y) - 1))


auc = roc(as.factor(as.numeric(test_Y) - 1), as.numeric(as.factor(prediction)))

# Print evaluation metrics

print(confusion_matrix)

print(paste("AUC:", auc))

plot(roc_curve, main="ROC Curve for XGboost", col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)



auc_by_class <- lapply(levels(as.factor(as.numeric(test_Y) - 1)), function(class) {
 class_predictions <- as.numeric(as.factor(prediction) == class)
 class_actual <- as.numeric(as.factor(as.numeric(test_Y) - 1) == class)
 roc_curve <- roc(class_actual, class_predictions)
 auc <- auc(roc_curve)

 # Plot ROC curve for each class separately
 plot(roc_curve, main = paste("ROC Curve for", class), col = "blue", lwd = 2,
     print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)

 return(auc)
```

```
})

# Print AUC for each class
names(auc_by_class) <- levels(test_Y)
print(auc_by_class)
```

Ml model for Numeric

```{r}
# Load required libraries
library(randomForest)
library(caret)
library(pROC)

#data <- read_excel("F:/Graduate/Project/output_results.xlsx")
data <- read_excel(output_excel)
# Exclude "Hispanic" and "Other" from race
data <- subset(data, race != "Hispanic" & race != "Other")

# Convert race to factor
data$race <- factor(data$race, levels = c("White", "Asian", "AfricanAmer"))

# Select relevant columns and remove missing values
df <- data[, c(13:20, 24)]
df <- na.omit(df)
```

```
# Normalize only numeric columns
numeric_cols <- sapply(df, is.numeric)
df_normalized <- as.data.frame(lapply(df[, numeric_cols], function(x) (x - min(x)) / (max(x) - min(x))))

# Combine normalized numeric columns with non-numeric columns
df_normalized <- cbind(df[, !numeric_cols], df_normalized)

# Separate predictors (X) and target variable (Y)
X <- df_normalized[, 2:ncol(df_normalized)]  # Exclude the first column (race)
Y <- df_normalized[, 1]

# Set seed for reproducibility
set.seed(123)

# Create indices for the training set
trainIndex <- sample(1:nrow(df), size = 0.8 * nrow(df))

# Split the data into training and testing sets
train_X <- X[trainIndex, ]
test_X <- X[-trainIndex, ]
train_Y <- Y[trainIndex]  # Corrected indexing
test_Y <- Y[-trainIndex]  # Corrected indexing

# Check the number of rows in each set
```

```
nrow(train_X)

nrow(test_X)


# Combine train_X and train_Y

train_data <- cbind(train_X, train_Y)


```

SVM

```{r}

ctrl <- trainControl(method = "cv", number = 5, verboseIter = TRUE, classProbs = TRUE)


tuned_svm <- train(train_Y ~ ., data = data.frame(train_X, train_Y),

                   method = "svmRadial",

                   trControl = ctrl,

                   preProcess = "scale",

                   tuneLength = 3)




# Predict using the tuned SVM model

predictions_svm <- predict(tuned_svm, newdata = test_X)

# Evaluation metrics

confusion_matrix_svm <- confusionMatrix(predictions_svm, test_Y)

roc_curve_svm <- roc(response = test_Y, predictor = as.numeric(predictions_svm))

auc_svm <- auc(roc_curve_svm)
```

```
# Output results
print(confusion_matrix_svm)
print(paste("AUC:", auc_svm))
plot(roc_curve_svm, main="ROC Curve for SVM Model")
```

```{r}
auc_svm
```

Random Forest

```{r}
# Set seed for reproducibility
set.seed(123)
# Define cross-validation control
ctrl <- trainControl(method = "cv", number = 5, verboseIter = TRUE)

# Tune the random forest model
tuned_rf <- train(train_Y ~ ., data = train_data, method = "rf",
                  trControl = ctrl, tuneLength = 3)

# Get the best model
best_model <- tuned_rf$finalModel

# Make predictions on the test set
predictions <- predict(best_model, newdata = test_X)
```

```
# Evaluate the model using various metrics
confusion_matrix <- confusionMatrix(predictions, test_Y)
roc_curve <- roc(test_Y, as.numeric(predictions))
auc <- auc(roc_curve)

# Print evaluation metrics
print(confusion_matrix)
print(paste("AUC:", auc))
plot(roc_curve, main="ROC Curve for Random Forest", col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)



library(pROC)

# Compute AUC for each class separately
auc_by_class <- lapply(levels(test_Y), function(class) {
 class_predictions <- as.numeric(predictions == class)
 class_actual <- as.numeric(test_Y == class)
 roc_curve <- roc(class_actual, class_predictions)
 auc <- auc(roc_curve)

 # Plot ROC curve for each class separately
 plot(roc_curve, main = paste("ROC Curve for", class), col = "blue", lwd = 2,
     print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)
```

```
  return(auc)
})

# Print AUC for each class
names(auc_by_class) <- levels(test_Y)
print(auc_by_class)



```


XGboost
```{r}
# Load necessary library
library(xgboost)
library(caret)
library(pROC)

# Convert the data to DMatrix object which is optimized for XGBoost
dtrain <- xgb.DMatrix(data = as.matrix(train_X), label = as.numeric(train_Y) - 1)
dtest <- xgb.DMatrix(data = as.matrix(test_X), label = as.numeric(test_Y) - 1)

# Set parameters for XGBoost
param_grid = expand.grid(
  booster = "gbtree",
  objective = "multi:softprob",
```

```
  num_class = 3,
  eval_metric = "mlogloss",

  eta = seq(from = 0.01, to = 1, length.out = 10),  # Learning rate
  max_depth = c(3, 6, 9),   # Depth of trees
  #min_child_weight = c(1, 5, 10),  # Minimum sum of instance weight (hessian) needed in a child
  min_child_weight = 5,
  subsample = c(0.5, 0.75, 1),  # Subsample ratio of the training instances
  #subsample = 0.5,
  #colsample_bytree = c(0.5, 0.75, 1)  # Subsample ratio of columns when constructing each tree
  colsample_bytree = 0.8
)

best_auc = 0
best_params = NULL
prediction = NULL

# Loop over each row in the parameter grid
for (i in 1:nrow(param_grid)) {
   params <- list(
   booster = "gbtree",
   objective = "multi:softprob",
   num_class = 3,
   eval_metric = "mlogloss",
```

```
  eta = param_grid[i, "eta"],
  max_depth =  param_grid[i, "max_depth"],
  min_child_weight = param_grid[i, "min_child_weight"],
  subsample = param_grid[i, "subsample"],
  colsample_bytree = param_grid[i, "colsample_bytree"]
 )

  nrounds <- 100

# Train the model
  xgb_model <- xgb.train(params = params, data = dtrain, nrounds = nrounds,
                         watchlist = list(eval = dtrain, test = dtest),                         verbose = 1)


  # Make predictions
  predictions <- predict(xgb_model, newdata = dtest)
  if (params$objective == "multi:softprob") {
   max_prob_indices <- max.col(matrix(predictions, ncol = length(levels(factor(Y))), byrow
= TRUE))
   predictions <- max_prob_indices - 1
  }

  # Confusion Matrix and AUC calculation using caret and pROC


  #conf_matrix <- confusionMatrix(as.factor(predictions), as.factor(as.numeric(test_Y) -
1))
```

```
  #conf_matrix$overall['Accuracy']

  roc_curve <- roc(as.numeric(test_Y) - 1, predictions, multi.class = "ovr")
  auc_score <- auc(roc_curve)


 # Check if this is the best AUC
 if (auc_score > best_auc) {
  best_auc <- auc_score
  best_params <- params
  prediction <- predictions
 }
}

# Output the best parameters
print(paste("Best AUC:", best_auc))
print("Best Parameters:")
print(best_params)
print(confusionMatrix(as.factor(prediction), as.factor(as.numeric(test_Y) - 1)))
roc_curve <- roc(as.numeric(test_Y) - 1, prediction, multi.class = "ovr")
plot(roc_curve, main="ROC Curve for XGboost")
```

```{r}
```

```
confusion_matrix = confusionMatrix(as.factor(prediction), as.factor(as.numeric(test_Y) - 1))


auc = roc(as.factor(as.numeric(test_Y) - 1), as.numeric(as.factor(prediction)))

# Print evaluation metrics

print(confusion_matrix)

print(paste("AUC:", auc))

plot(roc_curve, main="ROC Curve for XGboost", col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)



auc_by_class <- lapply(levels(as.factor(as.numeric(test_Y) - 1)), function(class) {
 class_predictions <- as.numeric(as.factor(prediction) == class)
 class_actual <- as.numeric(as.factor(as.numeric(test_Y) - 1) == class)
 roc_curve <- roc(class_actual, class_predictions)
 auc <- auc(roc_curve)

 # Plot ROC curve for each class separately
 plot(roc_curve, main = paste("ROC Curve for", class), col = "blue", lwd = 2,
     print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)

 return(auc)
})

# Print AUC for each class

names(auc_by_class) <- levels(test_Y)

print(auc_by_class)
```

```

```{r}
# Evaluate the model using various metrics
confusion_matrix <- confusionMatrix(prediction, test_Y)
roc_curve <- roc(test_Y, as.numeric(predictions))
auc <- auc(roc_curve)

# Print evaluation metrics
print(confusion_matrix)
print(paste("AUC:", auc))
plot(roc_curve, main="ROC Curve for Random Forest", col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)



library(pROC)

# Compute AUC for each class separately
auc_by_class <- lapply(levels(test_Y), function(class) {
  class_predictions <- as.numeric(predictions == class)
  class_actual <- as.numeric(test_Y == class)
  roc_curve <- roc(class_actual, class_predictions)
  auc <- auc(roc_curve)
```
```

```
  # Plot ROC curve for each class separately
  plot(roc_curve, main = paste("ROC Curve for", class), col = "blue", lwd = 2,
       print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)

  return(auc)
})

# Print AUC for each class
names(auc_by_class) <- levels(test_Y)
print(auc_by_class)
```

```{r}
library(xgboost)
library(caret)
library(pROC)

# Convert labels to a consistent factor across training and testing datasets
levels_all <- sort(unique(c(train_Y, test_Y)))  # Gather all unique levels and sort them

# Prepare training and testing data as DMatrix objects
dtrain <- xgb.DMatrix(data = as.matrix(train_X), label = as.numeric(factor(train_Y, levels = levels_all)) - 1)
dtest <- xgb.DMatrix(data = as.matrix(test_X), label = as.numeric(factor(test_Y, levels = levels_all)) - 1)
```

```
# Parameters for XGBoost
param_grid <- expand.grid(
  eta = seq(from = 0.01, to = 1, length.out = 10),
  max_depth = c(3, 6, 9),
  min_child_weight = 5,
  subsample = c(0.5, 0.75, 1),
  colsample_bytree = 0.8
)

best_auc = 0
best_params = NULL
prediction = NULL

# Loop over parameter grid
for (i in 1:nrow(param_grid)) {
  params <- list(
    booster = "gbtree",
    objective = "multi:softprob",
    num_class = length(levels_all),
    eval_metric = "mlogloss",
    eta = param_grid[i, "eta"],
    max_depth = param_grid[i, "max_depth"],
    min_child_weight = param_grid[i, "min_child_weight"],
    subsample = param_grid[i, "subsample"],
    colsample_bytree = param_grid[i, "colsample_bytree"]
```

```
)

nrounds <- 100
watchlist <- list(train = dtrain, test = dtest)

# Train the model
xgb_model <- xgb.train(params = params, data = dtrain, nrounds = nrounds,
                       watchlist = watchlist, verbose = 1)

# Make predictions
predictions <- predict(xgb_model, newdata = dtest)
predictions_matrix <- matrix(predictions, ncol = length(levels_all), byrow = TRUE)
colnames(predictions_matrix) <- levels_all  # Set column names as sorted unique levels

# AUC Calculation with properly named predictors
response_factor <- factor(test_Y, levels = levels_all)
roc_result <- multiclass.roc(response = response_factor, predictor = predictions_matrix)

# Calculate overall AUC and per-class AUCs
overall_auc <- auc(roc_result)
aucs_per_class <- sapply(roc_result$rocs, function(x) auc(x))

# Update best model tracking
if (overall_auc > best_auc) {
  best_auc <- overall_auc
  best_params <- params
```

```
   prediction <- predictions_matrix
  }
}

# Print results
print(paste("Best Overall AUC: ", best_auc))
print("Best Parameters:")
print(best_params)
print("AUC for each class comparison:")
print(aucs_per_class)
```

ML model for Coeff

```{r}
# Load required libraries
library(randomForest)
library(caret)
library(pROC)

#data <- read_excel("F:/Graduate/Project/output_results.xlsx")
data <- read_excel(output_excel)

# Exclude "Hispanic" and "Other" from race
```

```
data <- subset(data, race != "Hispanic" & race != "Other")

# Convert race to factor
data$race <- factor(data$race, levels = c("White", "Asian", "AfricanAmer"))

# Select relevant columns and remove missing values
df <- data[, c(2:12, 24)]
df <- na.omit(df)

# Normalize only numeric columns
numeric_cols <- sapply(df, is.numeric)
df_normalized <- as.data.frame(lapply(df[, numeric_cols], function(x) (x - min(x)) / (max(x) - min(x))))

# Combine normalized numeric columns with non-numeric columns
df_normalized <- cbind(df[, !numeric_cols], df_normalized)

# Separate predictors (X) and target variable (Y)
X <- df_normalized[, 2:ncol(df_normalized)]  # Exclude the first column (race)
Y <- df_normalized[, 1]

# Set seed for reproducibility
set.seed(123)

# Create indices for the training set
trainIndex <- sample(1:nrow(df), size = 0.8 * nrow(df))
```

```
# Split the data into training and testing sets
train_X <- X[trainIndex, ]
test_X <- X[-trainIndex, ]
train_Y <- Y[trainIndex]  # Corrected indexing
test_Y <- Y[-trainIndex]  # Corrected indexing

# Check the number of rows in each set
nrow(train_X)
nrow(test_X)

# Combine train_X and train_Y
train_data <- cbind(train_X, train_Y)
```

```
## SVM
```{r}
ctrl <- trainControl(method = "cv", number = 5, verboseIter = TRUE, classProbs = TRUE)

tuned_svm <- train(train_Y ~ ., data = data.frame(train_X, train_Y),
                   method = "svmRadial",
                   trControl = ctrl,
                   preProcess = "scale",
                   tuneLength = 3)

# Predict using the tuned SVM model
```
```

```
predictions_svm <- predict(tuned_svm, newdata = test_X)
# Evaluation metrics
confusion_matrix_svm <- confusionMatrix(predictions_svm, test_Y)
roc_curve_svm <- roc(response = test_Y, predictor = as.numeric(predictions_svm))
auc_svm <- auc(roc_curve_svm)

# Output results
print(confusion_matrix_svm)
print(paste("AUC:", auc_svm))
plot(roc_curve_svm, main="ROC Curve for SVM Model")
```

Random Forest

```{r}
# Define cross-validation control
ctrl <- trainControl(method = "cv", number = 5, verboseIter = TRUE)

# Tune the random forest model
tuned_rf <- train(train_Y ~ ., data = train_data, method = "rf",
                  trControl = ctrl, tuneLength = 3)

# Get the best model
best_model <- tuned_rf$finalModel

# Make predictions on the test set
predictions <- predict(best_model, newdata = test_X)
```

```
# Evaluate the model using various metrics
# confusion_matrix <- confusionMatrix(predictions, test_Y)
# roc_curve <- roc(test_Y, as.numeric(predictions))
# auc <- auc(roc_curve)
#
# # Print evaluation metrics
# print(confusion_matrix)
# print(paste("AUC:", auc))
# plot(roc_curve, main="ROC Curve for Random Forest")



##################
# Evaluate the model using various metrics
confusion_matrix <- confusionMatrix(predictions, test_Y)
roc_curve <- roc(test_Y, as.numeric(predictions))
auc <- auc(roc_curve)

# Print evaluation metrics
print(confusion_matrix)
print(paste("AUC:", auc))
plot(roc_curve, main="ROC Curve for Random Forest", col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)



library(pROC)
```

```
# Compute AUC for each class separately
auc_by_class <- lapply(levels(test_Y), function(class) {
 class_predictions <- as.numeric(predictions == class)
 class_actual <- as.numeric(test_Y == class)
 roc_curve <- roc(class_actual, class_predictions)
 auc <- auc(roc_curve)

 # Plot ROC curve for each class separately
 plot(roc_curve, main = paste("ROC Curve for", class), col = "blue", lwd = 2,
     print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)

 return(auc)
})

# Print AUC for each class
names(auc_by_class) <- levels(test_Y)
print(auc_by_class)
```

XGboost

```{r}
# Load necessary library
library(xgboost)
```

```
library(caret)
library(pROC)

# Convert the data to DMatrix object which is optimized for XGBoost
dtrain <- xgb.DMatrix(data = as.matrix(train_X), label = as.numeric(train_Y) - 1)
# Assuming your Y is a factor starting from 1
dtest <- xgb.DMatrix(data = as.matrix(test_X), label = as.numeric(test_Y) - 1)

# Set parameters for XGBoost

param_grid = expand.grid(
  booster = "gbtree",
  objective = "multi:softprob",
  num_class = length(levels(factor(Y))),
  eval_metric = "mlogloss",

  eta = seq(from = 0.01, to = 1, length.out = 10),  # Learning rate
  max_depth = c(3, 6, 9),   # Depth of trees
  #min_child_weight = c(1, 5, 10),  # Minimum sum of instance weight (hessian) needed in a child
  min_child_weight = 5,
  subsample = c(0.5, 0.75, 1),
  colsample_bytree = 0.8
)
```

```
best_auc = 0
best_params = NULL
prediction = NULL

# Loop over each row in the parameter grid
for (i in 1:nrow(param_grid)) {
  params <- list(
  booster = "gbtree",
  objective = "multi:softprob",
  num_class = 3,
  eval_metric = "mlogloss",
  eta = param_grid[i, "eta"],
  max_depth =  param_grid[i, "max_depth"],
  min_child_weight = param_grid[i, "min_child_weight"],
  subsample = param_grid[i, "subsample"],
  colsample_bytree = param_grid[i, "colsample_bytree"]
 )

  nrounds <- 100

# Train the model
  xgb_model <- xgb.train(params = params, data = dtrain, nrounds = nrounds,
                         watchlist = list(eval = dtrain, test = dtest),                         verbose = 1)


  # Make predictions
```

```
  predictions <- predict(xgb_model, newdata = dtest)
  if (params$objective == "multi:softprob") {
    max_prob_indices <- max.col(matrix(predictions, ncol = length(levels(factor(Y))), byrow = TRUE))
    predictions <- max_prob_indices - 1
  }

  # Confusion Matrix and AUC calculation using caret and pROC



  #conf_matrix <- confusionMatrix(as.factor(predictions), as.factor(as.numeric(test_Y) - 1))


  #conf_matrix$overall['Accuracy']


  roc_curve <- roc(as.numeric(test_Y) - 1, predictions, multi.class = "ovr")
  auc_score <- auc(roc_curve)



 # Check if this is the best AUC
 if (auc_score > best_auc) {
  best_auc <- auc_score
  best_params <- params
  prediction <- predictions
 }
}
```

```
# Output the best parameters
print(paste("Best AUC:", best_auc))
print("Best Parameters:")
print(best_params)
print(confusionMatrix(as.factor(prediction), as.factor(as.numeric(test_Y) - 1)))

roc_curve <- roc(as.numeric(test_Y) - 1, prediction, multi.class = "ovr")
plot(roc_curve, main="ROC Curve for XGboost")
```

```{r}
confusion_matrix = confusionMatrix(as.factor(prediction), as.factor(as.numeric(test_Y) - 1))

auc = roc(as.factor(as.numeric(test_Y) - 1), as.numeric(as.factor(prediction)))
# Print evaluation metrics
print(confusion_matrix)
print(paste("AUC:", auc))
plot(roc_curve, main="ROC Curve for XGboost", col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)

auc_by_class <- lapply(levels(as.factor(as.numeric(test_Y) - 1)), function(class) {
  class_predictions <- as.numeric(as.factor(prediction) == class)
  class_actual <- as.numeric(as.factor(as.numeric(test_Y) - 1) == class)
  roc_curve <- roc(class_actual, class_predictions)
```

```
  auc <- auc(roc_curve)


  # Plot ROC curve for each class separately
  plot(roc_curve, main = paste("ROC Curve for", class), col = "blue", lwd = 2,
       print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)


  return(auc)
})


# Print AUC for each class
names(auc_by_class) <- levels(test_Y)
print(auc_by_class)
```

11.4.3 Gradient Boosting Model and AUC curve

```
# Gradient Boosting Model
##Spline Coefficient
```

```{r}
library(xgboost)
library(readxl)
library(dplyr)
library(caret)
library(pROC)
#data <- read_excel("/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI 5999/output_results.xlsx")
data <- read_excel(output_excel)
```

```
# Exclude 'Other' and 'Hispanic' from the race column
data <- filter(data, !race %in% c("Other", "Hispanic"))

# Convert 'race' to a factor if it's not already
data$race <- as.factor(data$race)

# Normalize only numeric columns
numeric_cols <- sapply(data, is.numeric)
data_normalized <- as.data.frame(lapply(data[, numeric_cols], function(x) (x - min(x)) / (max(x) - min(x))))

# Combine normalized numeric columns with non-numeric columns
data_normalized <- cbind(data[, !numeric_cols], data_normalized)

# Separate predictors (X) and target variable (Y)
X <- data_normalized[, -which(names(data_normalized) == "race")]
Y <- data_normalized$race

# Split data into training and test sets
set.seed(123)  # for reproducibility
train_index <- createDataPartition(Y, p = 0.8, list = FALSE)
train_data <- data_normalized[train_index, ]
test_data <- data_normalized[-train_index, ]

# Combine all predictors
all_predictors <- c("Left_I", "Left_D1", "Left_D2", "Left_D3",
```

```
            "Mid_I", "Mid_D1", "Mid_D2",
            "Right_I", "Right_D1", "Right_D2", "Right_D3")

# Define a function to fit and evaluate a model
fit_and_evaluate_model <- function(predictors) {
  X_train <- model.matrix(~., data = train_data[, predictors])
  X_test <- model.matrix(~., data = test_data[, predictors])
  y_train <- as.integer(train_data$race) - 1
  y_test <- as.integer(test_data$race) - 1

  param <- list(max_depth = 3, eta = 0.1, silent = 1, nthread = 2,
          num_class = length(levels(data$race)), objective = "multi:softprob")
  model <- xgboost(data = X_train, label = y_train, params = param, nrounds = 100)

  # Predict on test set
  y_pred <- predict(model, X_test)
  y_pred_max <- max.col(matrix(y_pred, ncol = length(levels(data$race)), byrow = TRUE)) - 1

  # Calculate accuracy
  accuracy <- mean(y_pred_max == y_test)
  print(paste("Accuracy:", accuracy))
}

# Fit and evaluate the model using all predictors combined
fit_and_evaluate_model(all_predictors)
```

```{r}

#Further dive into ROC Curve For Each Specific Race


predictors <- c("Left_I", "Left_D1", "Left_D2", "Left_D3", "Mid_I", "Mid_D1", "Mid_D2", "Right_I", "Right_D1", "Right_D2", "Right_D3")


# Create model matrix for predictors

X_train <- model.matrix(~., data = train_data[, predictors])

X_test <- model.matrix(~., data = test_data[, predictors])

y_train <- train_data$race

y_test <- test_data$race


# Fit xgboost model

param <- list(max_depth = 3, eta = 0.1, silent = 1, nthread = 2,

        num_class = length(levels(data$race)), objective = "multi:softprob")

model <- xgboost(data = X_train, label = as.integer(y_train) - 1, params = param, nrounds = 100)


# Predict on test set and calculate probabilities

y_pred <- predict(model, X_test)

y_pred_matrix <- matrix(y_pred, ncol = length(levels(data$race)), byrow = TRUE)


# Plot ROC and calculate AUC for each class

par(mfrow = c(2, 2))

auc_values <- numeric(length(levels(data$race)))  # initialize vector to store AUC values

for (i in 1:length(levels(data$race))) {

  roc_curve <- roc(response = as.numeric(y_test == levels(data$race)[i]),
```

```
            predictor = y_pred_matrix[, i],
            plot = TRUE, print.auc = TRUE,
            main = paste("ROC for", levels(data$race)[i]))
  auc_values[i] <- auc(roc_curve)
  print(paste("AUC for", levels(data$race)[i], ":", auc_values[i]))
}
```

##numeric columns

```{r}
#data <- read_excel("/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI 5999/output_results.xlsx")
data <- read_excel(output_excel)
# Exclude unnecessary columns (e.g., comments, data points, curve)
data <- select(data, -c(Comments, Data_Points, Curve))

# Exclude 'Other' and 'Hispanic' from the race column
data <- filter(data, !race %in% c("Other", "Hispanic"))

# Convert 'race' to a factor
data$race <- as.factor(data$race)

# Check for NA values and remove or impute
data <- na.omit(data)  # Removes all rows with any NA values

# Define the predictors as specified
predictors <- c("BMI", "Height", "Max.Hip", "Anterior-posterior.Length", "Depth",
```

```
        "Crotch.curve.length.at.back.waist", "Front.Crotch.(Left.side)",
"Back.Crotch.(Right.Side)")


# Normalize only numeric columns

numeric_cols <- sapply(data[predictors], is.numeric)

predictors_to_normalize <- predictors[numeric_cols]  # select only numeric predictors for
normalization

data_normalized <- data  # make a copy of data to retain non-numeric data intact

data_normalized[predictors_to_normalize] <- scale(data[predictors_to_normalize])  #
apply normalization


# Split data into training and test sets

set.seed(123)

train_index <- createDataPartition(data_normalized$race, p = 0.8, list = FALSE)

train_data <- data_normalized[train_index, ]

test_data <- data_normalized[-train_index, ]


# Create model matrix for predictors

X_train <- model.matrix(~., data = train_data[, predictors])

X_test <- model.matrix(~., data = test_data[, predictors])

y_train <- train_data$race

y_test <- test_data$race


# Fit xgboost model

param <- list(max_depth = 3, eta = 0.1, silent = 1, nthread = 2,

      num_class = length(levels(data$race)), objective = "multi:softprob")
```

```
model <- xgboost(data = X_train, label = as.integer(y_train) - 1, params = param, nrounds = 100)


# Predict on test set and calculate probabilities
y_pred <- predict(model, X_test)
y_pred_matrix <- matrix(y_pred, ncol = length(levels(data$race)), byrow = TRUE)


# plot ROC and calculate AUC for each class
par(mfrow = c(2, 2))
for (i in 1:length(levels(data$race))) {
  roc_curve <- roc(response = as.numeric(y_test == levels(data$race)[i]),
                   predictor = y_pred_matrix[, i],
                   plot = TRUE, print.auc = TRUE,
                   main = paste("ROC for", levels(data$race)[i]))
  auc_value <- auc(roc_curve)
  print(paste("AUC for", levels(data$race)[i], ":", auc_value))
}
# Calculate overall accuracy
y_pred_labels <- max.col(y_pred_matrix)  # gets the index of the max probability for each row
accuracy <- mean(y_pred_labels == as.integer(y_test))
print(paste("Overall accuracy:", accuracy))
```


##Model Coefficient+ Numeric(Roc Curve over specific race)

```{r}
```

```
#data <- read_excel("/Users/zhangyanan/Desktop/Cornell/Spring 2024/STSCI 5999/output_results.xlsx")

data <- read_excel(output_excel)

# Exclude unnecessary columns (e.g., comments, data points, curve)

data <- select(data, -c(Comments, Data_Points, Curve))


# Exclude 'Other' and 'Hispanic' from the race column

data <- filter(data, !race %in% c("Other", "Hispanic"))


# Convert 'race' to a factor

data$race <- as.factor(data$race)


# Check for NA values and remove or impute

data <- na.omit(data)  # Removes all rows with any NA values


# Define the predictors as specified

predictors <- c("Left_I","Left_D1",   "Left_D2","Left_D3","Mid_I", "Mid_D1","Mid_D2","Right_I","Right_D1","Right_D2", "Right_D3","BMI", "Height", "Max.Hip", "Anterior-posterior.Length", "Depth", "Crotch.curve.length.at.back.waist", "Front.Crotch.(Left.side)", "Back.Crotch.(Right.Side)")


# Normalize only numeric columns

numeric_cols <- sapply(data[predictors], is.numeric)

predictors_to_normalize <- predictors[numeric_cols]  # select only numeric predictors for normalization

data_normalized <- data  # make a copy of data to retain non-numeric data intact

data_normalized[predictors_to_normalize] <- scale(data[predictors_to_normalize])  # apply normalization
```

```
# Split data into training and test sets
set.seed(123)
train_index <- createDataPartition(data_normalized$race, p = 0.8, list = FALSE)
train_data <- data_normalized[train_index, ]
test_data <- data_normalized[-train_index, ]

# Create model matrix for predictors
X_train <- model.matrix(~., data = train_data[, predictors])
X_test <- model.matrix(~., data = test_data[, predictors])
y_train <- train_data$race
y_test <- test_data$race

# Fit xgboost model
param <- list(max_depth = 3, eta = 0.1, silent = 1, nthread = 2,
        num_class = length(levels(data$race)), objective = "multi:softprob")
model <- xgboost(data = X_train, label = as.integer(y_train) - 1, params = param, nrounds =
100)

# Predict on test set and calculate probabilities
y_pred <- predict(model, X_test)
y_pred_matrix <- matrix(y_pred, ncol = length(levels(data$race)), byrow = TRUE)

# Optionally plot ROC and calculate AUC for each class
par(mfrow = c(2, 2))  # adjust depending on the number of classes
for (i in 1:length(levels(data$race))) {
```

```
  roc_curve <- roc(response = as.numeric(y_test == levels(data$race)[i]),
                   predictor = y_pred_matrix[, i],
                   plot = TRUE, print.auc = TRUE,
                   main = paste("ROC for", levels(data$race)[i]))
  auc_value <- auc(roc_curve)
  print(paste("AUC for", levels(data$race)[i], ":", auc_value))
}
# Calculate overall accuracy
y_pred_labels <- max.col(y_pred_matrix)  # gets the index of the max probability for each row
accuracy <- mean(y_pred_labels == as.integer(y_test))
print(paste("Overall accuracy:", accuracy))
```

##Roc Curve overall

```{r}
predictors <- c("Left_I", "Left_D1", "Left_D2", "Left_D3", "Mid_I", "Mid_D1", "Mid_D2",
                "Right_I", "Right_D1", "Right_D2", "Right_D3", "BMI", "Height", "Max.Hip",
                "Anterior-posterior.Length", "Depth", "Crotch.curve.length.at.back.waist",
                "Front.Crotch.(Left.side)", "Back.Crotch.(Right.Side)")

# Create model matrix for predictors
X_train <- model.matrix(~., data = train_data[, predictors])
X_test <- model.matrix(~., data = test_data[, predictors])
y_train <- as.numeric(train_data$race == "White")  # Binary conversion

# Fit xgboost model for binary classification
```

```
params <- list(max_depth = 3, eta = 0.1, verbosity = 0, nthread = 2, objective =
"binary:logistic")

model <- xgboost(data = X_train, label = y_train, params = params, nrounds = 100)


# Predict on test set and calculate probabilities for the 'White' class

predicted_probs <- predict(model, X_test)


# Compute the ROC curve

roc_obj <- roc(test_data$race == "White", predicted_probs)


# Plot the ROC curve

plot(roc_obj, main = "ROC Curve for Gradient Boosting (Both)", col = "blue", lwd = 2,
print.auc = TRUE, print.auc.x = 0.6, print.auc.y = 0.4)
```

**********End of the Document**********