Dataset Description and Preprocessing

This study utilizes single nucleotide polymorphism (SNP) data from the 1000 Genomes Project, comprising genomic data from 3,202 individuals. The raw dataset was derived from chromosome 1 and underwent stringent preprocessing steps to ensure data quality. Variants with a minor allele frequency (MAF) of less than 1% and multi-allelic variants were excluded. The curated dataset is publicly available at https://github.com/d201312151/matlab_AI_kobic.

Superpopulation Assignment

Each sample in the dataset was assigned to one of five superpopulations based on metadata from the 1000 Genomes Project. The **African (AFR)** superpopulation includes individuals from African Ancestry in Southwest US (ASW), African Caribbean in Barbados (ACB), Esan in Nigeria (ESN), Gambian in Western Division, The Gambia - Mandinka (GWD), Luhya in Webuye, Kenya (LWK), Mende in Sierra Leone (MSL), and Yoruba in Ibadan, Nigeria (YRI). The **European (EUR)** superpopulation includes British in England and Scotland (GBR), Finnish in Finland (FIN), Iberian populations in Spain (IBS), Toscani in Italy (TSI), and Utah residents (CEPH) with Northern and Western European ancestry (CEU). The **East Asian (EAS)** superpopulation comprises individuals from Chinese Dai in Xishuangbanna, China (CDX), Han Chinese South (CHS), Han Chinese in Beijing, China (CHB), Japanese in Tokyo, Japan (JPT), and Kinh in Ho Chi Minh City, Vietnam (KHV). The **South Asian (SAS)** superpopulation includes Bengali in Bangladesh (BEB), Gujarati Indians in Houston, TX (GIH), Indian Telugu in the UK (ITU), Punjabi in Lahore, Pakistan (PJL), and Sri Lankan Tamil in the UK (STU). Lastly, the **American (AMR)** superpopulation consists of individuals from Colombian in Medellin, Colombia (CLM), Mexican Ancestry in Los Angeles, California (MXL), Peruvian in Lima, Peru (PEL), and Puerto Rican in Puerto Rico (PUR).

```
load kgp_sample

AFR_list={'African Ancestry in Southwest US'
'African Caribbean in Barbados'
'Esan in Nigeria'
'Gambian in Western Division, The Gambia - Mandinka'
'Luhya in Webuye, Kenya'
'Mende in Sierra Leone'
'Yoruba in Ibadan, Nigeria'}

EUR_list={'British in England and Scotland'
'Finnish in Finland'
'Iberian populations in Spain'
'Toscani in Italy'
'Utah residents (CEPH) with Northern and Western European ancestry'}

EAS_list={'Chinese Dai in Xishuangbanna, China'
'Han Chinese South'
'Han Chinese in Beijing, China'
'Japanese in Tokyo, Japan'
'Kinh in Ho Chi Minh City, Vietnam'}

SAS_list={'Bengali in Bangladesh'
'Gujarati Indians in Houston, TX'
'Indian Telugu in the UK'
'Punjabi in Lahore, Pakistan'
'Sri Lankan Tamil in the UK'}

AMR_list={'Colombian in Medellin, Colombia'
'Mexican Ancestry in Los Angeles, California'
'Peruvian in Lima, Peru'
'Puerto Rican in Puerto Rico'}

AFR_temp_ix = find(ismember(kgp_sample.spec,AFR_list))
EUR_temp_ix = find(ismember(kgp_sample.spec,EUR_list))
EAS_temp_ix = find(ismember(kgp_sample.spec,EAS_list))
SAS_temp_ix = find(ismember(kgp_sample.spec,SAS_list))
AMR_temp_ix = find(ismember(kgp_sample.spec,AMR_list))

AFR_ID = kgp_sample.sampleID(AFR_temp_ix)
EUR_ID = kgp_sample.sampleID(EUR_temp_ix)
EAS_ID = kgp_sample.sampleID(EAS_temp_ix)
SAS_ID = kgp_sample.sampleID(SAS_temp_ix)
AMR_ID = kgp_sample.sampleID(AMR_temp_ix)
```

Dataset Splitting

To create the training and test datasets, 20% of samples from each superpopulation group were randomly selected to form the test set (641 samples). The remaining 2,561 samples were allocated to the training set.

Super population	Number of Training set	Number of Test set	Number of Total samples
AFR	714	179	893
EUR	506	127	633
EAS	468	117	585
SAS	481	120	601
AMR	392	98	490

```
%% Make training set and test set

% data load

ds =
datastore('1k_chr1.recalibrated_variants.maf001_pass_biallelic_snv.number.matrix','TreatAsMissing','
NA',...
'MissingValue',nan,'FileExtensions','.matrix','Type','tabulartext','Delimiter','\t','CommentStyle','
##');

% Superpopulation index

AFR_ix = find(ismember(ds.VariableNames,AFR_ID))
EUR_ix = find(ismember(ds.VariableNames,EUR_ID))
EAS_ix = find(ismember(ds.VariableNames,EAS_ID))
SAS_ix = find(ismember(ds.VariableNames,SAS_ID))
AMR_ix = find(ismember(ds.VariableNames,AMR_ID))

% 20 percent test set, 80 percent test set

AFR_test_indices = AFR_ix(randperm(length(AFR_ix), round(length(AFR_ix)*.2)))
AFR_trainnig_indices = setdiff(AFR_ix,AFR_test_indices)
AFR_other_trainnig_indices =
union(EUR_trainnig_indices,EAS_trainnig_indices),union(SAS_trainnig_indices,AMR_trainnig_indices)

EUR_test_indices = EUR_ix(randperm(length(EUR_ix), round(length(EUR_ix)*.2)))
EUR_trainnig_indices = setdiff(EUR_ix,EUR_test_indices)
EUR_other_trainnig_indices =
union(AFR_trainnig_indices,EAS_trainnig_indices),union(SAS_trainnig_indices,AMR_trainnig_indices)

EAS_test_indices = EAS_ix(randperm(length(EAS_ix), round(length(EAS_ix)*.2)))
EAS_trainnig_indices = setdiff(EAS_ix,EAS_test_indices)
EAS_other_trainnig_indices =
union(AFR_trainnig_indices,EUR_trainnig_indices),union(SAS_trainnig_indices,AMR_trainnig_indices)

SAS_test_indices = SAS_ix(randperm(length(SAS_ix), round(length(SAS_ix)*.2)))
SAS_trainnig_indices = setdiff(SAS_ix,SAS_test_indices)
SAS_other_trainnig_indices =
union(AFR_trainnig_indices,EUR_trainnig_indices),union(EAS_trainnig_indices,AMR_trainnig_indices)

AMR_test_indices = AMR_ix(randperm(length(AMR_ix), round(length(AMR_ix)*.2)))
AMR_trainnig_indices = setdiff(AMR_ix,AMR_test_indices)
AMR_other_trainnig_indices =
union(AFR_trainnig_indices,EUR_trainnig_indices),union(EAS_trainnig_indices,SAS_trainnig_indices)

% feature selection

Total_snv = tall(ds);
opt_1 = 0.6;
opt_2 = 0.4;

Total_snv_AFR = gather(Total_snv(:,AFR_trainnig_indices));
Total_other_snv_AFR = gather(Total_snv(:,AFR_other_trainnig_indices));
```

```
AFR_feature_candidates_1 = af_filter(Total_snv_AFR, Total_other_snv_AFR,
length(AFR_trainnig_indices) * 2, length(AFR_other_trainnig_indices) * 2,  opt_1, opt_2)

Total_snv_EUR = gather(Total_snv(:,EUR_trainnig_indices));
Total_other_snv_EUR = gather(Total_snv(:,EUR_other_trainnig_indices));
EUR_feature_candidates_1 = af_filter(Total_snv_EUR, Total_other_snv_EUR,
length(EUR_trainnig_indices) * 2, length(EUR_other_trainnig_indices) * 2,  opt_1, opt_2)

Total_snv_EAS = gather(Total_snv(:,EAS_trainnig_indices));
Total_other_snv_EAS = gather(Total_snv(:,EAS_other_trainnig_indices));
EAS_feature_candidates_1 = af_filter(Total_snv_EAS, Total_other_snv_EAS,
length(EAS_trainnig_indices) * 2, length(EAS_other_trainnig_indices) * 2,  opt_1, opt_2)

Total_snv_SAS = gather(Total_snv(:,SAS_trainnig_indices));
Total_other_snv_SAS = gather(Total_snv(:,SAS_other_trainnig_indices));
SAS_feature_candidates_1 = af_filter(Total_snv_SAS, Total_other_snv_SAS,
length(SAS_trainnig_indices) * 2, length(SAS_other_trainnig_indices) * 2,  opt_1, opt_2)

Total_snv_AMR = gather(Total_snv(:,AMR_trainnig_indices));
Total_other_snv_AMR = gather(Total_snv(:,AMR_other_trainnig_indices));
AMR_feature_candidates_1 = af_filter(Total_snv_AMR, Total_other_snv_AMR,
length(AMR_trainnig_indices) * 2, length(AMR_other_trainnig_indices) * 2,  opt_1, opt_2)

cadidate_union =
union(union(union(union(AFR_feature_candidates_1,EUR_feature_candidates_1),EAS_feature_candidates_1)
,...
    SAS_feature_candidates_1),AMR_feature_candidates_1)

% Make New training set

SNV_N_matrix = [Total_snv_AFR(cadidate_union,:) Total_snv_EUR(cadidate_union,:) ...
    Total_snv_EAS(cadidate_union,:) Total_snv_SAS(cadidate_union,:) Total_snv_AMR(cadidate_union,:)]
Training_set = SNV_N_matrix

% Make New test set

Total_snv_AFR_test = gather(Total_snv(:,AFR_test_indices));
Total_snv_EUR_test = gather(Total_snv(:,EUR_test_indices));
Total_snv_EAS_test = gather(Total_snv(:,EAS_test_indices));
Total_snv_SAS_test = gather(Total_snv(:,SAS_test_indices));
Total_snv_AMR_test = gather(Total_snv(:,AMR_test_indices));

SNV_N_matrix_test = [Total_snv_AFR_test(cadidate_union,:) Total_snv_EUR_test(cadidate_union,:) ...
    Total_snv_EAS_test(cadidate_union,:) Total_snv_SAS_test(cadidate_union,:)
Total_snv_AMR_test(cadidate_union,:)]
Test_set = SNV_N_matrix_test

% Y lables(superpopulation set)

Training_Y = cell(length(Training_set.Properties.VariableNames),1)
Test_Y = cell(length(Test_set.Properties.VariableNames),1)

AFR_train_ix = find(ismember(Training_set.Properties.VariableNames,AFR_ID))
EUR_train_ix = find(ismember(Training_set.Properties.VariableNames,EUR_ID))
EAS_train_ix = find(ismember(Training_set.Properties.VariableNames,EAS_ID))
SAS_train_ix = find(ismember(Training_set.Properties.VariableNames,SAS_ID))
AMR_train_ix = find(ismember(Training_set.Properties.VariableNames,AMR_ID))

AFR_test_ix = find(ismember(Test_set.Properties.VariableNames,AFR_ID))
EUR_test_ix = find(ismember(Test_set.Properties.VariableNames,EUR_ID))
EAS_test_ix = find(ismember(Test_set.Properties.VariableNames,EAS_ID))
SAS_test_ix = find(ismember(Test_set.Properties.VariableNames,SAS_ID))
AMR_test_ix = find(ismember(Test_set.Properties.VariableNames,AMR_ID))

Training_Y(AFR_train_ix)={'AFR'}
Training_Y(EUR_train_ix)={'EUR'}
Training_Y(EAS_train_ix)={'EAS'}
Training_Y(SAS_train_ix)={'SAS'}
Training_Y(AMR_train_ix)={'AMR'}

Test_Y(AFR_test_ix)={'AFR'}
Test_Y(EUR_test_ix)={'EUR'}
Test_Y(EAS_test_ix)={'EAS'}
```

```
Test_Y(SAS_test_ix)={'SAS'}
Test_Y(AMR_test_ix)={'AMR'}
```

Feature Selection

Using the training dataset, SNP markers were filtered based on population allele frequency (AF) criteria. Specifically, markers with an AF of ≥0.6 in one superpopulation and <0.4 in all other superpopulations were retained. This filtering process yielded 41,017 SNP markers, which were used as features for subsequent analyses.

Dimensionality Reduction via Autoencoder

To reduce the dimensionality of the SNP feature set while retaining critical information, an autoencoder architecture was implemented. The autoencoder consists of two components: an encoder that maps the input data into a compressed latent space, and a decoder that reconstructs the original input from the latent space. The encoder network includes a feature input layer of size 41,017, followed by fully connected layers with 5,000 and 2,000 neurons, respectively, each activated by ReLU functions. The bottleneck layer represents the embedding space with 1,000 neurons. The decoder mirrors the encoder, with fully connected layers of size 2,000 and 5,000 neurons, followed by an output layer that reconstructs the original input size. The network was trained using the *adam* optimizer with a learning rate of 0.001 for 100 epochs and a mini-batch size of 64. A cross-entropy loss function was used to optimize reconstruction accuracy. The resulting embedding space, derived from the encoder's bottleneck layer, was extracted as a 1,000-dimensional feature set for downstream classification tasks.

```
A = [table2array(Training_set) table2array(Test_set)];

% Autoencoder
inputSize = size(A, 1);
embeddingSize = 1000;
hiddenSize1 = 5000;
hiddenSize2 = 2000;

% Encoder
encoderLayers = [
    featureInputLayer(inputSize, 'Name', 'input')
    fullyConnectedLayer(hiddenSize1, 'Name', 'encoder_fc1')
    reluLayer('Name', 'encoder_relu1')
    fullyConnectedLayer(hiddenSize2, 'Name', 'encoder_fc2')
    reluLayer('Name', 'encoder_relu2')
    fullyConnectedLayer(embeddingSize, 'Name', 'embedding')
];

% Decoder
decoderLayers = [
    fullyConnectedLayer(hiddenSize2, 'Name', 'decoder_fc1')
    reluLayer('Name', 'decoder_relu1')
    fullyConnectedLayer(hiddenSize1, 'Name', 'decoder_fc2')
    reluLayer('Name', 'decoder_relu2')
    fullyConnectedLayer(inputSize, 'Name', 'output')
];

% Autoencoder
layers = [
    encoderLayers
    decoderLayers
];

options = trainingOptions('adam', ...
    'MaxEpochs', 100, ...
    'MiniBatchSize', 64, ...
    'InitialLearnRate', 0.001, ...
    'Shuffle', 'every-epoch', ...
    'Verbose', false)

% Autoencoder learning
% autoencoderNet = trainNetwork(A, A, layers, options);

autoencoderNet = trainnet(A',A',layers,"crossentropy",options);

% Encoder
encoderNet = layerGraph(layers);
```

```
encoderNet = encoderNet.Layers(1:6);

% Embedding
embeddedFeatures = predict(dlnetwork(encoderNet), A');
embeddedFeatures = embeddedFeatures';
```

Classification Using Machine Learning Models

Using the 1,000-dimensional feature set derived from the autoencoder, we performed classification to predict the superpopulation of each sample. For this task, we employed four widely used machine learning models. First, a random forest classifier was trained with 200 decision trees using the *classification* method. Second, a k-nearest neighbors (k-NN) classifier was implemented with k=5 neighbors to determine the class of each test sample based on proximity to the training samples. Third, a support vector machine (SVM) model was trained using an error-correcting output codes (ECOC) framework for multi-class classification. Lastly, a Gaussian Naïve Bayes (NB) classifier was used to model the probabilistic relationships between features and class labels. Each model was trained on the training dataset and evaluated on an independent test dataset. All four models demonstrated high classification accuracy, highlighting the discriminative power of the reduced feature set in accurately predicting superpopulation categories.

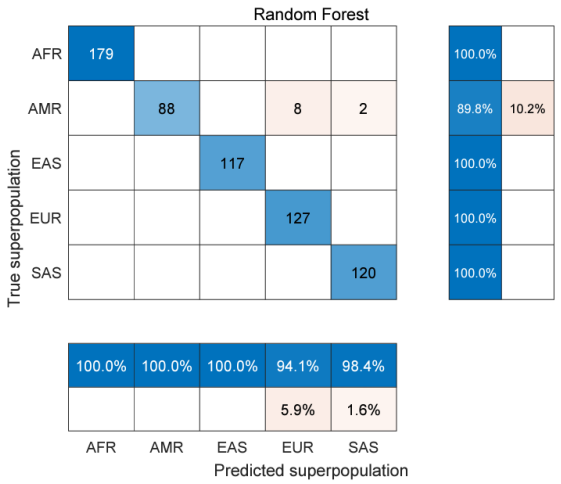
```
%% Prediction model construction

Train_new = embeddedFeatures(1:2561,:)
Test_new = embeddedFeatures(2562:end,:)

rfModel = TreeBagger(200, Train_new, Training_Y, 'Method', 'classification');
rf_result = predict(rfModel, Test_new);

knnModel = fitcknn(Train_new, Training_Y, 'NumNeighbors', 5);
knn_result = predict(knnModel, Test_new);

svmModel = fitcecoc(Train_new, Training_Y);
svm_result = predict(svmModel, Test_new);

nbModel = fitcnb(Train_new, Training_Y);
nb_result = predict(nbModel, Test_new);
```

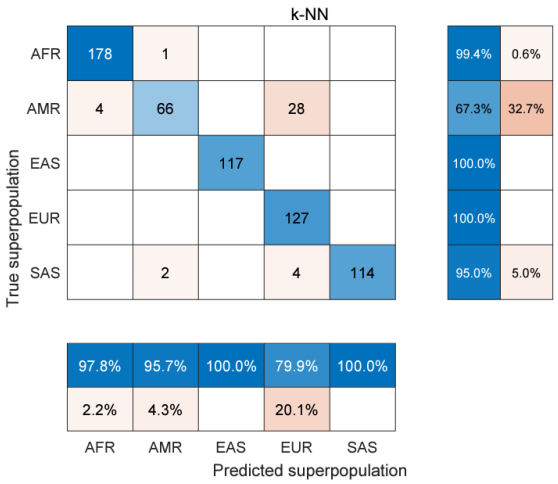
Results

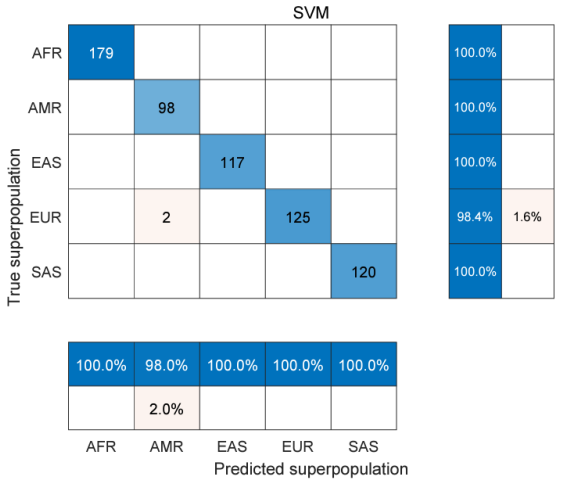
Each of the four models demonstrated impressive classification performance. The overall accuracies achieved were 0.9844 for the random forest model, 0.9392 for the k-nearest neighbors classifier, 0.9969 for the support vector machine, and 0.9594 for the Naïve Bayes classifier, indicating robust predictive capabilities across all approaches. Furthermore, the accuracy across individual superpopulation groups was consistently high, confirming that the reduced feature set effectively captured the genetic diversity and unique characteristics of each population. This uniformity in performance highlights the models' ability to generalize well across diverse groups, ensuring equitable classification outcomes.

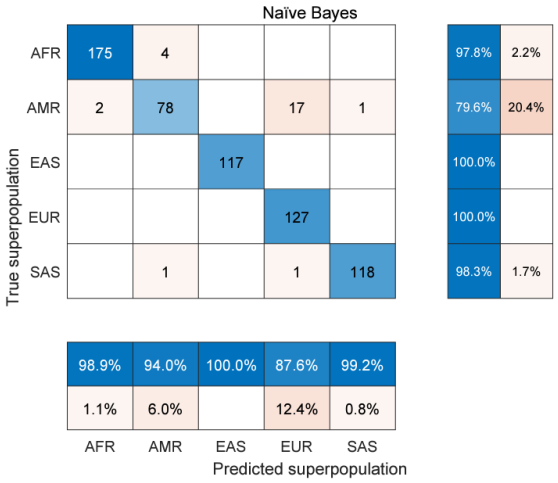
```
subplot(2,2,1)
confMatrix = confusionmat(Test_Y, rf_result);
f1 = confusionchart(confMatrix, {'AFR','EUR','EAS','SAS','AMR'}, ...
    'RowSummary', 'row-normalized', 'ColumnSummary', 'column-normalized');
f1.XLabel = 'Predicted superpopulation'
f1.YLabel = 'True superpopulation'
f1.Title = 'Random Forest'

subplot(2,2,2)
confMatrix = confusionmat(Test_Y, knn_result);
f2 = confusionchart(confMatrix, {'AFR','EUR','EAS','SAS','AMR'}, ...
    'RowSummary', 'row-normalized', 'ColumnSummary', 'column-normalized');
f2.XLabel = 'Predicted superpopulation'
f2.YLabel = 'True superpopulation'
f2.Title = 'k-NN'
subplot(2,2,3)
confMatrix = confusionmat(Test_Y, svm_result);
f3=confusionchart(confMatrix, {'AFR','EUR','EAS','SAS','AMR'}, ...
    'RowSummary', 'row-normalized', 'ColumnSummary', 'column-normalized');
f3.XLabel = 'Predicted superpopulation'
f3.YLabel = 'True superpopulation'
f3.Title = 'SVM'
subplot(2,2,4)
```

```
confMatrix = confusionmat(Test_Y, nb_result);
f4=confusionchart(confMatrix, {'AFR','EUR','EAS','SAS','AMR'}, ...
    'RowSummary', 'row-normalized', 'ColumnSummary', 'column-normalized');
f4.XLabel = 'Predicted superpopulation'
f4.YLabel = 'True superpopulation'
f4.Title = 'Naïve Bayes'
```





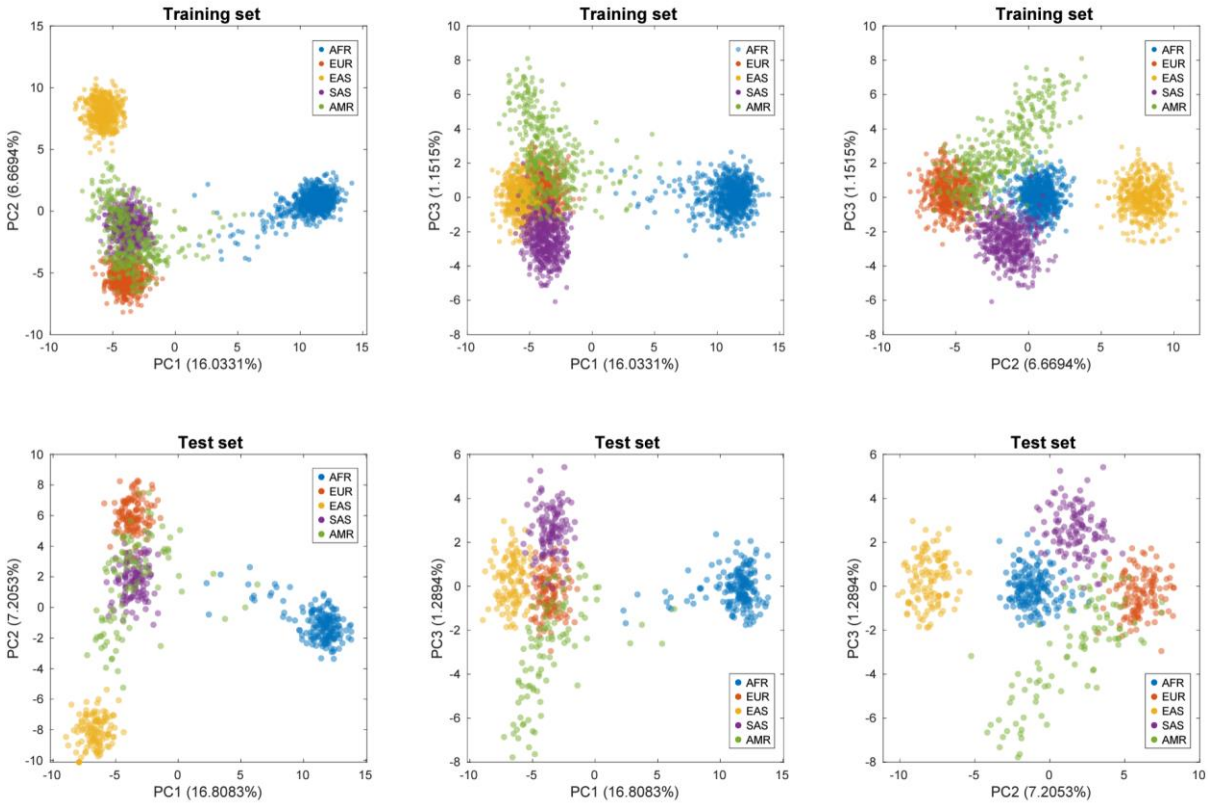


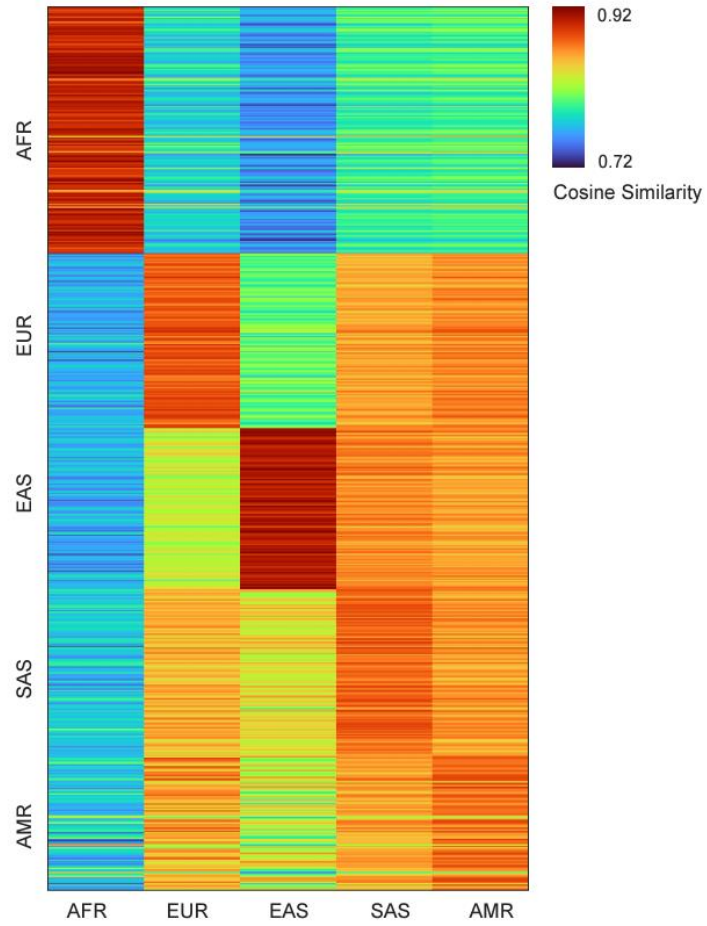


Feature Space Evaluation

To assess the population-specific characteristics captured by the constructed feature space, we performed principal component analysis (PCA). The results demonstrated clear clustering of both training and test samples based on the 1,000 selected markers, effectively grouping individuals according to their respective superpopulations. Additionally, we evaluated the proximity of test samples to the centroids representing each superpopulation in the 1,000-feature space. Using cosine similarity as a metric, we observed that test samples exhibited very high similarity to the centroids of their actual superpopulations. This finding underscores the robustness of the feature space in preserving population-specific genetic signatures and its effectiveness for classification tasks.

```
%% PCA plot

embeddedFeatures = predict(dlnetwork(encoderNet), table2array(Training_set)');

[coefs,score,~,~,expl] = pca((embeddedFeatures)) ;

subplot(2,3,1)
gscatter(score(:,1),score(:,2),Training_Y)
title('Training set')
xlabel(['PC1 (' num2str(expl(1)) '%)'])
ylabel(['PC2 (' num2str(expl(2)) '%)'])
subplot(2,3,2)
gscatter(score(:,1),score(:,3),Training_Y)
title('Training set')
xlabel(['PC1 (' num2str(expl(1)) '%)'])
ylabel(['PC3 (' num2str(expl(3)) '%)'])
subplot(2,3,3)
gscatter(score(:,2),score(:,3),Training_Y)
title('Training set')
xlabel(['PC2 (' num2str(expl(2)) '%)'])
ylabel(['PC3 (' num2str(expl(3)) '%)'])

embeddedFeatures_test = predict(dlnetwork(encoderNet), table2array(Test_set)');

[coefs,score,~,~,expl] = pca((embeddedFeatures_test)) ;

subplot(2,3,4)
gscatter(score(:,1),score(:,2),Test_Y)
title('Test set')
xlabel(['PC1 (' num2str(expl(1)) '%)'])
ylabel(['PC2 (' num2str(expl(2)) '%)'])
subplot(2,3,5)
gscatter(score(:,1),score(:,3),Test_Y)
title('Test set')
xlabel(['PC1 (' num2str(expl(1)) '%)'])
ylabel(['PC3 (' num2str(expl(3)) '%)'])
subplot(2,3,6)
gscatter(score(:,2),score(:,3),Test_Y)
title('Test set')
xlabel(['PC2 (' num2str(expl(2)) '%)'])
ylabel(['PC3 (' num2str(expl(3)) '%)'])

%% cosine similarity

AFR_centroid = mean(Train_new(ismember(Training_Y,'AFR'),:))
EUR_centroid = mean(Train_new(ismember(Training_Y,'EUR'),:))
EAS_centroid = mean(Train_new(ismember(Training_Y,'EAS'),:))
SAS_centroid = mean(Train_new(ismember(Training_Y,'SAS'),:))
AMR_centroid = mean(Train_new(ismember(Training_Y,'AMR'),:))

Z=[];
for i = 1 : length(Test_Y)

Z = [Z; cosine_sim(Test_new(i,:) ,AFR_centroid), cosine_sim(Test_new(i,:) ,EUR_centroid), ...

cosine_sim(Test_new(i,:) ,EAS_centroid),cosine_sim(Test_new(i,:) ,SAS_centroid),cosine_sim(Test_new(
i,:) ,AMR_centroid)]
end

a = heatmap(Z)
grid off
a.YDisplayLabels=repmat({''},641,1)
```


Training set
Test set
AFR
EUR
EAS
SAS
AMR
PC1 (16.0331%)
PC2 (6.6694%)
PC3 (1.1515%)
PC1 (16.8083%)
PC2 (7.2053%)
PC3 (1.2894%)


0.92
0.72
Cosine Similarity
AFR
EUR
EAS
SAS
AMR