CS4622
Machine Learning Lab 01
Feature Engineering

Index	190088H
Name	W.R.H.M.Bandara
Date of submission	2023-08-25

Introduction

The data set is created using AudioMNIST with the 256 features and 4 target labels.

- Speaker ID
- speaker age
- speaker gender
- speaker accent

We have to remove the unwanted features using the feature engineering techniques and predict the label values for the test data set.

Preprocessing

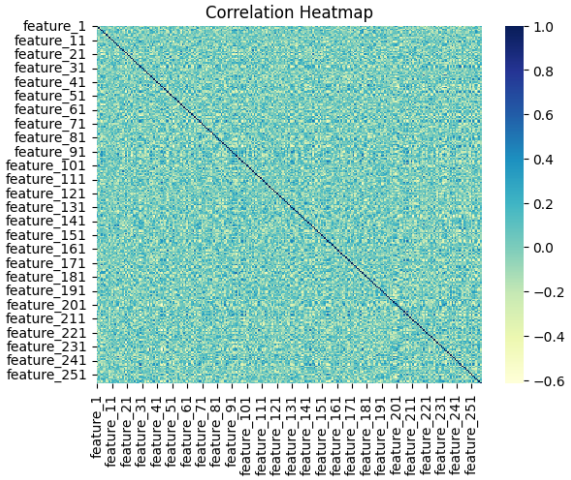
Use scaling for the feature values of the both train and test data sets.

```
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
x_scaled=scaler.fit_transform(x)
#  convert X_scaled back to a pandas DataFrame
x_scaled_df = pd.DataFrame(x_scaled, columns=x.columns)
x_scaled_df

scaler=StandardScaler()
x_test_scaled=scaler.fit_transform(x_test)
#  convert X_scaled back to a pandas DataFrame
x_test_scaled_df = pd.DataFrame(x_test_scaled, columns=x.columns)
```

To get an idea about the features, draw the correlation matrix.

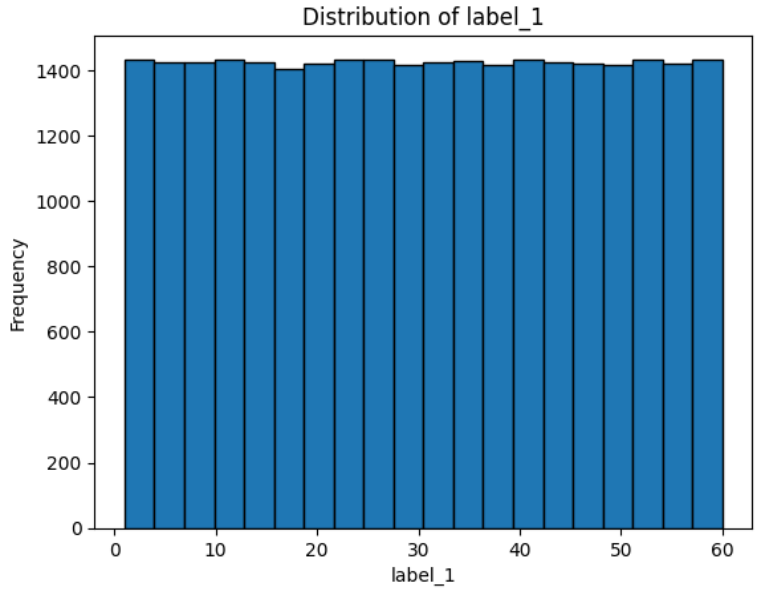


Then remove the rows with NaN values.

Label 1

Colab link- label _1_lab_1.ipynb

First, check the data set to identify any imbalanced situation in the data set.



The frequency of the labels is equally distributed. After rescaling the feature values using a stand scaler from Sklearn. This is a classification task, so I used KNN as the model using the following code and k=5.

```
#use knn without preprocessing
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import hamming_loss, jaccard_score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Initialize the k-NN classifier
knn = KNeighborsClassifier(n_neighbors=5)

# Train the model
knn.fit(x, y)

# Make predictions on the test set
y_pred = knn.predict(x_test)
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred))
```

The result after preprocessing is 0.988.

40	1.00	1.00	1.00	12
41	1.00	1.00	1.00	10
42	1.00	1.00	1.00	12
43	1.00	1.00	1.00	12
44	1.00	1.00	1.00	19
45	1.00	0.93	0.96	14
46	1.00	1.00	1.00	11
47	1.00	1.00	1.00	8
48	1.00	0.94	0.97	17
49	1.00	1.00	1.00	13
50	0.93	1.00	0.96	13
51	1.00	1.00	1.00	8
52	1.00	0.91	0.95	11
53	1.00	0.87	0.93	15
54	1.00	1.00	1.00	9
55	1.00	1.00	1.00	8
56	1.00	1.00	1.00	10
57	1.00	1.00	1.00	18
58	1.00	1.00	1.00	20
59	1.00	1.00	1.00	10
60	1.00	1.00	1.00	10
accuracy			0.99	750
macro avg	0.99	0.99	0.99	750
weighted avg	0.99	0.99	0.99	750

Then get the correlation matrix between the features. And remove the features that have relations more than 0.4.
Then used the PCA as follows and finally created the 45 features with 0.98 accuracy.

48	0.94	0.94	0.94	17
49	1.00	0.92	0.96	13
50	1.00	1.00	1.00	13
51	1.00	1.00	1.00	8
52	1.00	0.91	0.95	11
53	1.00	0.80	0.89	15
54	1.00	0.89	0.94	9
55	0.89	1.00	0.94	8
56	1.00	1.00	1.00	10
57	1.00	0.94	0.97	18
58	0.95	0.95	0.95	20
59	1.00	1.00	1.00	10
60	1.00	1.00	1.00	10
accuracy			0.98	750
macro avg	0.98	0.98	0.98	750
weighted avg	0.98	0.98	0.98	750

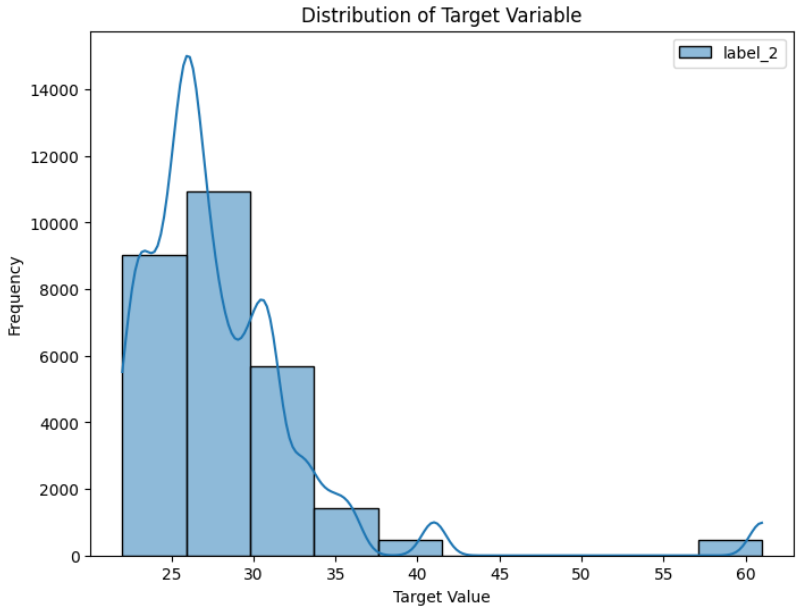
Label 2

Colab link - label_2_lab_1.ipynb
Label 2 is the speaker's age. There are some NaN values in label 2, so removed those columns because there are considerable data sets.

```
# preprocessing data because there are some NaN values\
columns_with_missing = standard_data.columns[standard_data.isna().any()].tolist()
cleaned_standard_data = standard_data.dropna(subset=columns_with_missing)

columns_with_missing = test_data.columns[test_data.isna().any()].tolist()
test_data = test_data.dropna(subset=columns_with_missing)
```

According to the label values and the label description label 2 defines the age of the speaker. age involves predicting a numeric value, it falls under the realm of regression tasks in machine learning.



As the model, I use XGBoost which is a popular machine learning algorithm that is suitable for a wide range of regression and classification tasks. Parameters for the model are defined as follows.

```
import numpy as np
import pandas as pd
import xgboost as xg
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import r2_score
```

```
# Initialize the xg-boost
xgb_r = xg.XGBRegressor(n_estimators=1000, max_depth=7, eta=0.1, subsample=0.7,
colsample_bytree=0.8)

# Train the model
xgb_r.fit(x_train, y_train)

# Make predictions on the test set
y_pred = xgb_r.predict(x_test)
mse = MSE(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Before the preprocessing mean square error was

Then rescale the features using the following code.

```
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
x_scaled=scaler.fit_transform(x_train)
#  convert X_scaled back to a pandas DataFrame
x_train_scaled_df = pd.DataFrame(x_scaled, columns=x_train.columns)
x_train_scaled_df=x_train

scaler=StandardScaler()
x_test_scaled=scaler.fit_transform(x_test)
#  convert X_scaled back to a pandas DataFrame
x_test_scaled_df = pd.DataFrame(x_test_scaled, columns=x_train.columns)
x_test_scaled_df=x_test
```

Using the correlation matrix, get an idea about the relationship between the features and the target label.
Finally use the PCA to reduce the features by choosing 0.97 as the variance. As a result, the final data set has only 79 Features with the mean squared error of 10.36.

Label 3

Colab link- label_3_lab_1.ipynb

Before the preprocessing of the data set, using the k-NN classifier I got the following accuracy 1.

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       142
           1       1.00      1.00      1.00       608

    accuracy                           1.00       750
   macro avg       1.00      1.00      1.00       750
weighted avg       1.00      1.00      1.00       750
```

Then rescale the features and identify the correlation between features.
After that remove 181 features that have a correlation of more than 0.4. Then the accuracy was 0.9986 for the valid data set.
Then use the PCA with a 0.97 variance and finally get the 52 features.

```
from sklearn.decomposition import PCA

pca=PCA(.97, svd_solver='full')
pca=pca.fit(x_train_scaled_df)
x_train_pca=pca.transform(x_train_scaled_df)
x_test_pca=pca.transform(x_test_scaled_df)
x_test_pca.shape
```

The final result for the valid data set is given below.

```
# Train the model
knn.fit(x_train_pca, y_train)

# Make predictions on the test set
y_pred = knn.predict(x_test_pca)
y_pred_after=y_pred
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_
  return self._fit(X, y)
0.9986666666666667
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       142
           1       1.00      1.00      1.00       608

    accuracy                           1.00       750
   macro avg       1.00      1.00      1.00       750
weighted avg       1.00      1.00      1.00       750
```

Label 4

Colab link - label_4_lab_1.ipynb

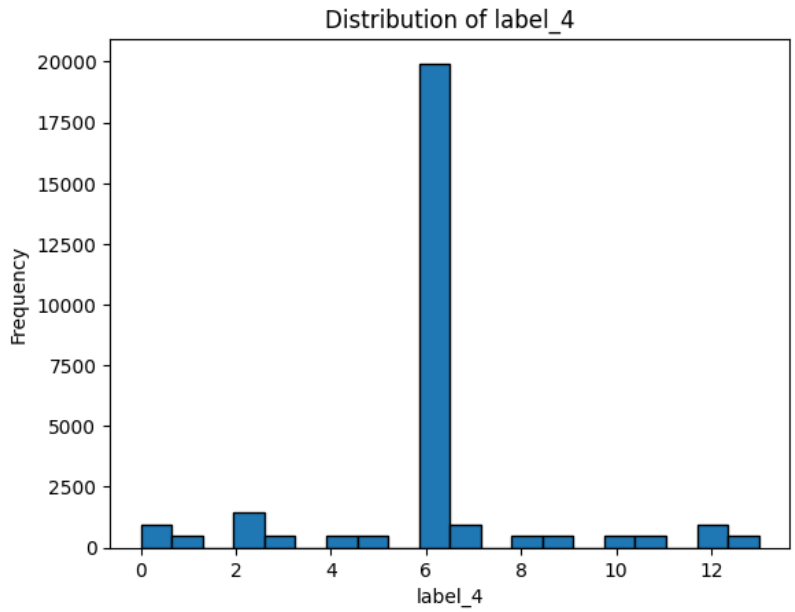
Before the preprocessing of the data set, using the k-NN classifier I got the following accuracy.

```
0.9933333333333333
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       1.00      0.91      0.95        11
           2       1.00      1.00      1.00        27
           3       1.00      1.00      1.00         8
           4       1.00      1.00      1.00        15
           5       1.00      0.91      0.95        11
           6       0.99      1.00      1.00       532
           7       1.00      0.97      0.98        32
           8       1.00      0.95      0.97        19
           9       1.00      1.00      1.00        17
          10       1.00      1.00      1.00        10
          11       1.00      1.00      1.00        11
          12       1.00      0.96      0.98        26
          13       1.00      1.00      1.00        10

    accuracy                           0.99       750
   macro avg       1.00      0.98      0.99       750
weighted avg       0.99      0.99      0.99       750
```

First, visualize the label values distribution using the hist graph that is shown below. It is clear that data are highly imbalanced.

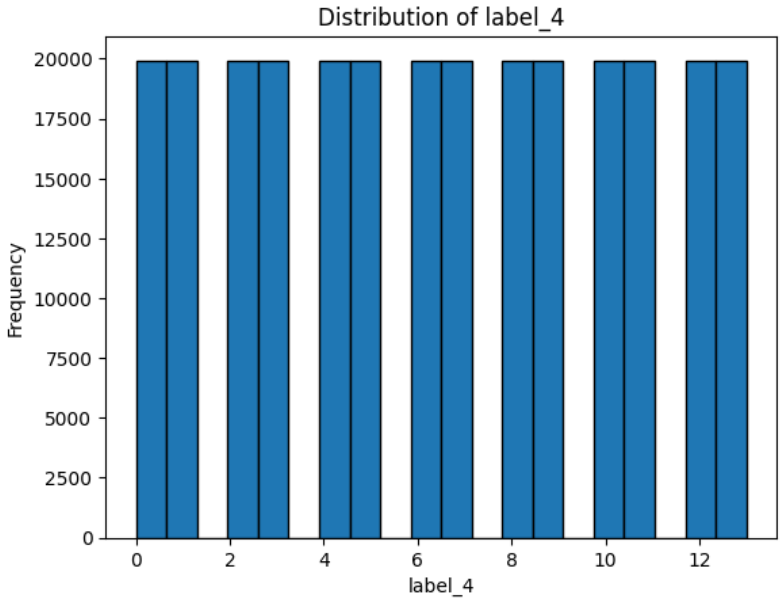


Then we used sampling techniques to rescale the data set.

```
# Apply SMOTE to oversample the minority classes
smote = SMOTE(sampling_strategy='auto', random_state=42)
```

```
X_train_resampled, y_train_resampled = smote.fit_resample(x_train, y_train)
```

After oversampling the data set.



Then scale the data set as described in the preprocessing part.
To identify the correlation between the features, get the correlation matrix and remove the features that have more than 0.4 correlation between the two features. As a result removed the 181 features.

Then take the PCA and create a new feature set with the variance of 0.95.

```
from sklearn.decomposition import PCA

pca=PCA(.95, svd_solver='full')
pca=pca.fit(x_train_scaled_df)
x_train_pca=pca.transform(x_train_scaled_df)
x_test_pca=pca.transform(x_test_scaled_df)
```

As a result created 88 Features with an accuracy of 0.988.
The accuracy for the final features set is given below.

0.988

	precision	recall	f1-score	support
0	0.95	1.00	0.98	21
1	1.00	0.91	0.95	11
2	1.00	1.00	1.00	27
3	0.89	1.00	0.94	8
4	1.00	0.93	0.97	15
5	1.00	0.91	0.95	11
6	0.99	1.00	0.99	532
7	1.00	0.94	0.97	32
8	1.00	0.95	0.97	19
9	1.00	1.00	1.00	17
10	1.00	1.00	1.00	10
11	1.00	1.00	1.00	11
12	1.00	0.96	0.98	26
13	1.00	1.00	1.00	10
accuracy			0.99	750
macro avg	0.99	0.97	0.98	750
weighted avg	0.99	0.99	0.99	750

Summary

	Label 1	Label 2	Label 3	Label 4
Features	256	256	256	256
Model	K-Nearest Neighbors Algorithm	xgboost	K-Nearest Neighbors Algorithm	K-Nearest Neighbors Algorithm
Type	classification	regression	classification	classification
Accuracy for valid data set before preprocessing	0.988	Mean Squared Error: 7.03	1	0.9933
Accuracy for valid data set after preprocessing	0.9893	7.0313	0.998	0.996
Final features	**45**	**79**	**52**	**88**
Final accuracy for valid data set	0.976	10.36	0.9986	0.9
For the test data set	0.9706	10.36	0.9986	0.98