CS4622 - Machine Learning

Lab 01 - Feature Engineering

Report

Name: Pasan S. Kalansooriya

Index: 190290U

Colab Notebook

1. Base Accuracies

These are the accuracies I got for predicting each class before applying any feature engineering techniques (using the given 260 features). To get the accuracies I used 2 approaches which are

1. the 80:20 split of the training dataset
2. The validation dataset that was provided

Class label	Base accuracy - on validation dataset	Base accuracy - on train dataset (80:20 train-test split)
Speaker IDs	0.992	0.9921107994389902
Speaker ages	0.9786666666666667	0.9738779803646563
Speaker genders	1.0	0.9992987377279102
Speaker accents	0.9906666666666667	0.9840462833099579

2. Exploratory Data Analysis

First I used .describe() method in pandas library to get a summary of descriptive statistics of the data. It calculates and displays various statistics for each numeric column in the DataFrame, giving us a quick overview of the distribution and central tendencies of the data (refer Fig 1).

	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	feature_9	feature_10	...	feature_251
count	28520.000000	28520.000000	28520.000000	28520.000000	28520.000000	28520.000000	28520.000000	28520.000000	28520.000000	28520.000000	...	28520.000000
mean	-0.225478	-0.616421	-0.212878	0.336330	0.105467	-0.339634	-0.201756	0.742262	0.942325	-1.957584	...	-0.028098
std	0.990632	1.005573	1.075468	1.248919	0.831132	1.073267	0.943505	1.055721	0.940459	0.876939	...	0.989497
min	-4.023911	-5.582544	-4.679888	-4.284380	-3.010300	-5.034355	-4.355222	-3.806574	-2.922614	-5.203347	...	-3.679602
25%	-0.869656	-1.287018	-0.913419	-0.478661	-0.437171	-1.075753	-0.823699	0.064072	0.298269	-2.556436	...	-0.692262
50%	-0.190790	-0.609782	-0.218800	0.295881	0.117310	-0.329571	-0.193263	0.755193	0.940786	-1.949847	...	-0.018218
75%	0.457509	0.063174	0.490262	1.119242	0.657941	0.393339	0.417146	1.461846	1.579200	-1.345999	...	0.644112
max	4.322171	3.876604	4.267163	5.828656	3.884939	3.890622	3.799406	4.817545	4.615069	1.147192	...	3.915582

Fig 1: A summary of descriptive statistics of the data.

Since the original dataset contained class label names as label_1, label_2 likewise, I **renamed** them to have names: speaker_ID, speaker_age, speaker_gender and speaker_accent.

Then I did some further exploratory data analysis on the dataset and found out that there are several **null** values in **'speaker_age'** column (refer Fig 2 heatmap). To overcome this problem I assigned the **mean** of the age column to these missing values in both training and validation datasets.

```
sns.heatmap(train_df.isnull(),yticklabels=False,cbar=False,cmap='viridis')

# note that the missing values represented as yellow bar (top right hand corner in below heatmap)
```

```
<Axes: >
```

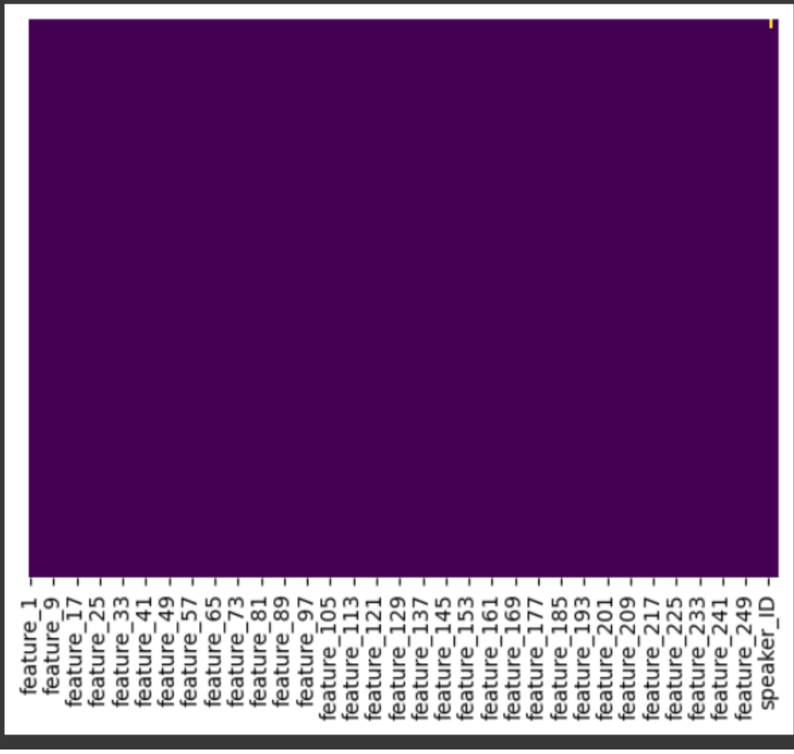

Fig 2

Also, I noticed that "**speaker_accent**" has a **class imbalance issue**. Fig 3 below showcases that (the value '6' has very high frequency compared to other values). To address this issue, I used **class_weight='balanced'** parameter when training the models.

```
train_df.speaker_accent.value_counts()

6     19938
2      1449
0       955
12      954
7       938
13      482
1       481
11      480
10      480
3       479
5       478
9       472
4       469
8       465
Name: speaker_accent, dtype: int64
```

Fig 3

3. Feature Engineering

Note: For the training I used SVC, GaussianNB, LogisticRegression and RandomForestClassifier models and considered the highest accuracy. (Also for all the instances SVC gave the best result).

3.1 Standardization

As the first step, I standardized the data using **Z-score standardisation** to check whether it will help to improve the base accuracies. The results showed that it **improved the accuracy** scores in all the combinations:

Class label	Accuracy after standardization - on validation dataset — (1)	Accuracy after standardization - on train dataset (80:20 train-test split) — (2)	Result (with respect to the base accuracies of (1) and (2))
Speaker IDs	`0.9946666666666667`	`0.9938639551192145`	`Both increased`
Speaker ages	`0.9853333333333333`	`0.9850981767180925`	`Both increased`
Speaker genders	`1.0`	`0.9996493688639552`	`(1) No change` `(2) increased`
Speaker accents	`0.9933333333333333`	`0.9898316970546984`	`Both increased`

Therefore for the rest of the techniques, I used the standardized version of the features.

3.2 Techniques that didn't reduce the number of features

As per the requirements of the lab exercise we are supposed to reduce the number of features in the given dataset while achieving acceptable accuracy scores. I then applied some techniques for feature reduction. Some of them couldn't reduce the features while some were able to reduce the feature while maintaining acceptable accuracy scores. First let's look at some of the techniques that **didn't reduce** the number of features.

1. Check for constant and Quasi-constant features

- Constant Features:
  - A constant feature is a column in a dataset where all of its values are the same for every observation.
  - Constant features provide no information to the model since they don't vary across observations.
  - These features are typically removed during the data preprocessing phase to avoid adding unnecessary complexity to the model.

- Quasi-Constant Features:
  - A quasi-constant feature is a column in a dataset where the majority of its values are the same for most observations, but there might be a small proportion of different values.
  - Unlike constant features, quasi-constant features have some variability, but this variability is very limited.
  - These features might still contain some information, especially if the variable with limited variability has a meaningful impact on the target variable.
  - In practice, it's common to identify quasi-constant features and decide whether to keep or remove them based on domain knowledge and the specific modeling task.
- **No constant or quasi-constant features were found** in the dataset.

2. Check for highly correlated features

- This technique is used to identify pairs of variables that are strongly correlated with each other. Highly correlated features can lead to multicollinearity issues in models and might not provide much additional information, so identifying and handling them can be important for improving model performance and interpretability. After runing
- No highly correlated features were also found in the dataset (at the threshold of 0.8)

3. Univariate MSE

- This technique involves evaluating each individual feature's predictive power by training a separate Decision Tree Regressor for each feature and measuring the Mean Squared Error (MSE) of predictions made by each individual regressor.
- I checked with several threshold values and found out the accuracies do not get improved even though the number of features can be reduced. For example when I run the technique for ‘train_speaker_IDs’ class label, ⇒ The features got reduced to **204** (from 256) but the **accuracies dropped** compared to base accuracies (Fig 4).

```
[58] run_models_and_check_with_test_dataset(selected_features_train_X, train_speaker_IDs, selected_features_valid_X, valid_speaker_IDs)

     #Though the features got reduced to 204 (from 256), the accuracy dropped to 0.9906666666666667 from 0.992

     Best accuracy out of SVM, NB, Logigstic Reg, and Random Forest is: 0.9946666666666667 ( SVM )

     run_models_and_check_with_train_test_split_on_train_dataset(selected_features_train_X, train_speaker_IDs)

     #Though the features got reduced to 204 (from 256), the accuracy dropped to 0.9919354838709677 from 0.9921107994389902

     Best accuracy out of SVM, NB, Logigstic Reg, and Random Forest is: 0.9935133239831697 ( SVM )
```

Fig 4

3.3 Reduce number of Features based on Mutual information measure

Mutual information is a statistical measure that quantifies the amount of information shared between two random variables. In the context of feature selection, mutual information is used to evaluate the relationship between each individual feature and the target variable. It measures how much information the presence or absence of a feature contributes to making accurate predictions about the target variable. Higher mutual information between a feature and the target variable implies that the feature contains valuable information for predicting the target. Conversely, lower mutual information indicates that the feature may not contribute significantly to predicting the target variable.

Based on the mutual information measure (threshold as 95%) I could **reduce the number of features to 243** with respective to each class label. Then I checked the accuracy score and got these results:

Class label	**Accuracy after standardization - on validation dataset — (1)**	**Accuracy after standardization - on train dataset (80:20 train-test split) — (2)**	**Result (with respect to the base accuracies of (1) and (2))**
Speaker IDs	0.9933333333333333	0.9938639551192145	Both increase
Speaker ages	0.9853333333333333	0.9850981767180925	Both increase
Speaker genders	1.0	0.9996493688639552	(1) Same (2) Increased
Speaker accents	0.9946666666666667	0.989656381486676	Both increased

3.4 Model-Based Feature Selection

This technique is commonly used in machine learning to automatically select the most important features from a dataset based on a model's evaluation of their importance for a given task. It's a way to improve model performance, reduce overfitting, and enhance interpretability by focusing on the most relevant features. It leverages the predictive power of a chosen machine learning algorithm (in my code I used logistic regression) to identify and retain the most important features.

After applying this technique I was able to reduce the features as below.

- Speaker IDs : **255**
- Speaker ages : **249**
- Speaker genders : **40**
- Speaker accents : **244**

The below table shows the summary after running the technique:

Class label	**Accuracy after standardization - on validation dataset — (1)**	**Accuracy after standardization - on train dataset (80:20 train-test split) — (2)**	**Result (with respect to the base accuracies of (1) and (2))**
Speaker	0.9946666666666667	0.993688639551192	Both increase

IDs		1	
Speaker ages	0.9853333333333333	0.9856241234221599	Both increased
Speaker genders	1.0	0.9989481065918654	(1) Same (2) Decreased
Speaker accents	0.9933333333333333	0.9901823281907434	Both increased

3.5 Random Forest Importance

Random Forest is an ensemble machine learning algorithm that uses multiple decision trees to make predictions. It naturally assigns importance scores to features based on their contribution to the model's performance. The higher the importance score, the more significant the feature is in predicting the target variable(s).

After applying this technique I was able to reduce the features as below.

- Speaker IDs : **250**
- Speaker ages : **255**
- Speaker genders : **65**
- Speaker accents : **253**

The below table shows the summary after running the technique:

Class label	Accuracy after standardization - on validation dataset — (1)	Accuracy after standardization - on train dataset (80:20 train-test split) — (2)	Result (with respect to the base accuracies of (1) and (2))
Speaker IDs	0.9933333333333333	0.9935133239831697	Both Increased
Speaker ages	0.9853333333333333	0.9849228611500701	Both increased
Speaker genders	1.0	0.9987727910238429	(1) same (2) Decreased
Speaker accents	0.9946666666666667	0.989656381486676	Both increased

3.6 PCA - Principle Component Analysis

PCA is a technique that aims to capture the most important patterns of variance in the original dataset by transforming it into a new set of uncorrelated variables, known as principal components. The code begins by creating a PCA model with a specified explained **variance threshold** of **0.96**, indicating that the transformed data should retain 96% of the original dataset's variance. The PCA model is then fitted to the standardized training data. Subsequently, the training and validation data are transformed using the learned PCA model, resulting in reduced-dimensional representations of the data.

After applying this technique the features got **reduced** to **73 (from 256)** with respect to all the class labels.

The results after running are shown below.

Class label	Accuracy after standardization - on validation dataset — (1)	Accuracy after standardization - on train dataset (80:20 train-test split) — (2)	Result (with respect to the base accuracies of (1) and (2))
Speaker IDs	0.9933333333333333	0.9924614305750351	Both increased
Speaker ages	0.9826666666666667	0.9798387096774194	Both increased
Speaker genders	1.0	0.9996493688639552	(1) Same (2) Increased
Speaker accents	0.9893333333333333	0.986851332398317	(1) Decreased (2) Increased

4. Conclusions

Based on above results of all the techniques, these are the optimal ways to reduce the features based on each class label.

Class label	Technique	Accuracy after standardization - on validation dataset — (1)	Accuracy after standardization - on train dataset (80:20 train-test split) — (2)	Result (with respect to the base accuracies of (1) and (2))	# features/ justification
Speaker IDs	PCA	0.9933333333333333	0.9924614305750351	Both increased	73
Speaker	PCA	0.9826666	0.97983870967	Both	73

ages		666666667	74194	increase d	
Speaker genders	Model-ba sed	1.0	0.99894810659 18654	(1) Same (2) Decrease d	40 / Though the accuracy in (2) reduced, it is a ver small difference (-0.000350631 136)
Speaker accents	PCA	0.9893333 333333333	0.98685133239 8317	(1) Decrease d (2) Increase d	73