Report on Feature Engineering

Lab 01

CS4622 - Machine Learning

Department of Computer Science & Engineering University of Moratuwa

Theshan Wijerathne- 190705B

25/08/2023

Introduction

For this lab, We were given two CSV files. A training data set with 28520 rows and columns with 256 features and 4 target labels, and a validation data set with 750 rows and columns with 256 features and 4 target labels.

The first 256 columns are 256 values of the speaker embedding vector of each audio file in the data set. The last 4 columns are speaker-related labels corresponding to each speaker embedding vector.

- Label 1 - Speaker ID
- Label 2 - Speaker age
- Label 3 - Speaker gender
- Label 4 - Speaker accent

Feature Engineering

Using the Pandas library to read CSV files on Google Drive, I read the CSV files for each label. After that, I processed the data using a series of Pandas DataFrame procedures. I started by making two copies of each DataFrame and choosing each label's y-axis and each feature's x-axes individually. The features in the aforementioned DataFrames were then normalized using the StandardScaler class from the scikit-learn preprocessing module. I was also aware of that label_2 have some missing values, so I removed those values from the dataframe. In this lab, NaN rows were removed for label_2 because this feature contained a significant number of missing values. The data was then scaled using a StandardScaler. This ensures that all features have the same scale and that no one feature dominates the model.

```
labels = ['label_1', 'label_2', 'label_3', 'label_4']
X_train = {}
X_valid = {}
y_train = {}
y_valid = {}
X_test = {}

for label in labels:
    scaler = StandardScaler()
    tr_df = train_df
    val_df = valid_df
    tst_df = test_df
    if label == 'label_2': # Remove NaN rows for label_2
      tr_df = train_df[train_df[label].notna()]
      val_df = valid_df[valid_df[label].notna()]

    X_train[label] = pd.DataFrame(scaler.fit_transform(tr_df.iloc[:, :-4]))
    X_valid[label] = pd.DataFrame(scaler.transform(val_df.iloc[:, :-4]))
    X_test[label] = pd.DataFrame(scaler.transform(tst_df.iloc[:, :-4]))
```

```
    y_train[label] = tr_df[label]
    y_valid[label] = val_df[label]
```

Feature selection is the process of selecting the most relevant features for a machine learning model. This can improve the performance of the model and reduce its overfitting.

In this lab, feature selection was performed using the SelectKBest algorithm. This algorithm selects the k best features based on their f-scores. The f-score is a measure of the statistical significance of the relationship between a feature and the target variable.

The number of features to select was set to 95. This was determined by experimenting with different values of k and selecting the value that resulted in the best performance on the validation set.

```
new_features = 95
selector = SelectKBest(f_classif, k=new_features)

X_train_selected = {}
X_valid_selected = {}
X_test_selected = {}
for label in X_train:
  X_train_selected[label] =
pd.DataFrame(selector.fit_transform(X_train[label], y_train[label]))
  X_valid_selected[label] = pd.DataFrame(selector.transform(X_valid[label]))
  X_test_selected[label] = pd.DataFrame(selector.transform(X_test[label]))
```

Model Training and Evaluation

The SVM classifier was then trained on the train data with the selected features. The model was then evaluated on the validation data.

The following table shows the accuracy scores for each label before and after feature selection:

Label	Accuracy before feature selection	Accuracy after feature selection
label_1	0.9906	0.9746
label_2	23.8256(MSE)	28.8831(MSE)
label_3	0.9986	0.9986
label_4	0.9626	0.9133

Conclusion

Feature engineering is an important technique for improving the performance of machine learning models. By selecting the best features, it is possible to reduce the complexity of the model and improve its accuracy.

In this lab, feature selection improved the accuracy of the SVM classifier for all four labels. This suggests that the selected features are more informative than the original features for predicting the labels.

Potential Improvements

There are a few potential improvements that could be made to this lab:

- Use a more sophisticated feature selection algorithm. The SelectKBest algorithm is a simple but effective feature selection algorithm. However, there are more sophisticated algorithms that may be able to select even better features for the task at hand.
- Use a different machine learning algorithm. The SVM classifier is a powerful machine learning algorithm, but it is not the only algorithm that can be used for classification tasks. Other algorithms, such as random forests and gradient boosting machines, may be able to achieve even better results on this dataset.

Link to Jupyter Notebooks

Jupyter Notebook