Modelling and Learning From Data Lab Report 1

WENRUI XIE

October 6, 2024

This report uses various machine learning techniques for classification tasks using a provided music dataset, evaluating performance on a test set of 200 songs. First, we will talk about standard preprocessing steps, such as normalization, deleting outlier data and evaluate by k-fold cross-validation. Then we will talk about specific methods including KNN, Random forest and deep learning method. Lastly, the predictive outcomes are printed and submitted to an online platform for comparison with the accuracy of models from other classmates.

1. Data Processing

1.1 Loading of dataset and test set

Read the provided dataset and test set using functions from the Pandas package:

```
data=pd.read_csv('training_data.csv', sep=',')
test_data=pd.read_csv('songs_to_classify.csv', sep=',')
k = 10

# select which features to use
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

test_data = test_data.iloc[:].values
```

The dataset consists of 750 rows and 14 columns, each row represents a song. The first 13 columns contain features, and the final column contains the labels. To prepare the data, the dataset is first partitioned by separating the features from the labels.

1.2 Delete outlier data

The process of identifying values in a dataset that significantly deviates from many observations. This technique is used for data cleaning to remove noise and anomalies, ultimately improving the performance of machine learning models. Following the detection of outliers, this step involves eliminating the identified anomalous data points from the dataset. (From ChatGPT)

```
Iso = IsolationForest(contamination=0.02)
outliers = Iso.fit_predict(X)
outlier_indices = np.where(outliers == -1)
X = np.delete(X, outlier_indices, axis=0)
y = np.delete(y, outlier_indices, axis=0)
```

1.3 Normalization

Normalization is required due to the different scales of the dataset's features. By rescaling the values of each feature to a similar range, the performance of machine learning models can be improved. To accomplish this, the MinMaxScaler from Scikit-Learn is used to scale the features to a range between 0 and 1.

```
# Normalize input data
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
test_data = scaler.fit_transform(test_data)
```

1.4 Divide the dataset and validation set

The rows in the dataset are randomly split, with 20% allocated as the test set and the remaining 80% used for training.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. Different Method for Prediction

2.1 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a supervised learning algorithm utilized for both classification and regression tasks. The core principle of KNN is that if the majority of a sample's K nearest neighbors in the feature space

belong to a specific category, the sample is also classified into that category. Essentially, this algorithm employs a voting mechanism for classification.

KNN is commonly applied in practical situations involving small datasets, where samples are evenly distributed, and where strong interpretability of the model is required.

```
model = KNeighborsClassifier(n_neighbors=k, weights='distance')
model.fit(X_train, y_train)
```

By selecting an appropriate value for k and using k-fold cross-validation to evaluate the model, obtain the training results as shown in Figure 1.

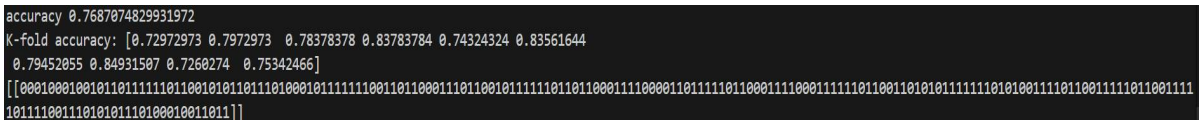

Figure 1 : KNN Result

2.2 Random Forest

Random Forest is an ensemble learning algorithm designed for classification and regression tasks. It improves model performance and generalization by combining multiple decision trees. The process begins by randomly selecting a subset of samples (with replacement) and a specific number of features (usually the square root of the total features) from the original training dataset to construct each decision tree. The CART (Classification and Regression Trees) algorithm is typically utilized for this construction.

This process is repeated to create a collection of decision trees, forming a Random Forest. For classification tasks, the class of a sample is determined using a majority voting method among the individual trees.

```
# randomforest
model = RandomForestClassifier(n_estimators=110, oob_score=True, random_state=1234)
model.fit(X_train, y_train)
```

By selecting an appropriate value for k and using k-fold cross-validation to evaluate the model, obtain the training results as shown in Figure 2.

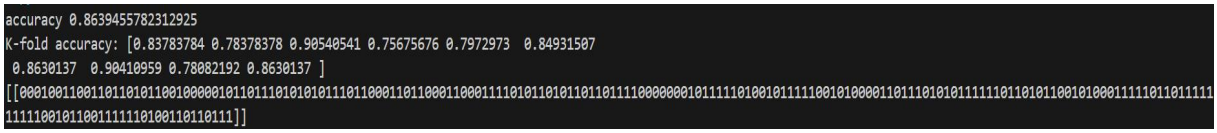

Figure 2 : Random Forest Result

2.3 Deep Learning Method

Neural networks offer significant advantages due to their capability to learn and represent complex nonlinear relationships, making them particularly effective for handling intricate data patterns and features. In contrast, traditional machine learning algorithms, such as logistic regression and support vector machines, are typically linear models, which limits their ability to capture nonlinear relationships. Well-regularized neural networks often demonstrate strong generalization capabilities, performing well on unseen data.

Binary classification methods utilizing fully connected neural networks excel in addressing complex problems and large-scale datasets, although they require more data and computational resources, leading to increased computational complexity and longer training times. Traditional machine learning algorithms, on the other hand, are generally easier to interpret and provide insights into feature importance. In comparison, neural networks are often seen as "black box" models, making their internal decision-making processes more challenging to interpret.

The training process for a fully connected neural network involves several key steps. First, the dataset is loaded and preprocessed, followed by randomization to ensure an effective training process. (from ChatGPT)

build the neural network model, define the loss function, and select the optimizer.

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=1, out_channels=32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv1d(in_channels=32, out_channels=64, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(64 * 13, 128)  # Adjust to match the number of features
        self.fc2 = nn.Linear(128, 1)  # Binary output
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = x.unsqueeze(1)  # Add channel dimension
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = x.view(x.size(0), -1)  # Flatten
        x = self.relu(self.fc1(x))
        x = self.sigmoid(self.fc2(x))
        return x
# Instantiate model, define loss function and optimizer
model = CNN()
criterion = nn.BCELoss()  # Binary Cross-Entropy Loss
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop
epochs = 250
batch_size = 64

for epoch in range(epochs):
    model.train()
    optimizer.zero_grad()

    # Forward pass
    outputs = model(X_train)
    loss = criterion(outputs, y_train)

    # Backward pass and optimization
    loss.backward()
    optimizer.step()
```

By selecting an appropriate seed to train the model, obtain the training results as shown in Figure 3.

Figure 3 : Deep Learning Result

3. Comparison For Different Method and Conclusion

After testing and analyzing the model, the following conclusions have been drawn in this study:

In traditional machine learning methods, Random Forest performs well on the test set, with an accuracy rate of up to 85%, and accuracy rate on scoreboard can reach 79.5%, However, the classification accuracy of KNN is relatively poor, only reaching around 80%, and accuracy rate on scoreboard is 77.5%

And deep learning methods also perform well on the test set, with an accuracy rate of up to 85% and accuracy rate on scoreboard can reach 81%. However, overfitting is prone to occur, and the number of training steps needs to be controlled, with significant variations depending on different random seeds.

Based on the comprehensive consideration of the above test results, random forest was ultimately selected for this study.

5	100	0.81	dl	2024-10-05

(a)

8	100	0.795	randomforest	2024-10-03

(b)

12	100	0.775		2024-09-30

(c)

Figure 4: prediction for test data, show in scoreboard,(a): accuracy rate for deep learning method;(b): accuracy for random forest method (c) accuracy rate for K-neighbors method