Artificial intelligence and Pattern Recognition

D0038E Project

by Magnus Hjörnhede

Group - Py Grp

Task 2. Classification Models on Gesture Data

Done in Python

Table of Contents

1. Introduction

This report details the process and results of Task 2, which involves training, testing, and reflecting on various classification models using gesture data.

2. Methodology

2.1 Models

Several classifiers were trained on the dataset. The training data underwent preprocessing steps as described in Task 1, with some changes. Certain preprocessing steps were omitted because they did not yield satisfactory results, for instance some of the normalization methods. Additionally, missing values in the dataset were filled with its average. The classifiers trained include:

- **Decision Tree**
- **Random Forest**
- **k-NN (k-Nearest Neighbors)**
- **MLP (Multi-layer Perceptron)**
- **SVM** with various kernels:
  - **Linear**: C cost parameter.
  - **Polynomial of degree 3 (SVM-poly-3)**
  - **Polynomial of degree 20 (SVM-poly-20)**
  - **Sigmoid (SVM-sigmoid)**
  - **Radial Basis Function (SVM-rbf)**
- **AdaBoost**
- **Bagging**
- **Extra Trees**
- **Gradient Boosting**

2.2 Classifiers

Below are the setting used for the classifiers

```
classifiers = {
    'k-NN': KNeighborsClassifier(n_neighbors=6),
    'Decision Tree': DecisionTreeClassifier(ccp_alpha=0.01),
    'MLP': MLPClassifier(hidden_layer_sizes=(200,), max_iter=1000),
    'SVM-linear': SVC(kernel='linear'),
    'SVM-poly-3': SVC(kernel='poly', degree=3),
    'SVM-poly-20': SVC(kernel='poly', degree=20),
    'SVM-sigmoid': SVC(kernel='sigmoid'),
    'SVM-rbf': SVC(kernel='rbf', gamma='scale'),
    'Random Forest': RandomForestClassifier(n_estimators=1000),
    'AdaBoost': AdaBoostClassifier(n_estimators=50, learning_rate=1),

    # bagging and boosting
    'Bagging': BaggingClassifier(estimator=SVC(kernel='linear'), n_estimators=10, random_state=0),
    'Extra Trees': ExtraTreesClassifier(n_estimators=100, random_state=0),
    'Gradient Boosting': GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1,
                                                    random_state=0),
}
```

2.3 Crossvalidation

For some classifiers, specific hyperparameters were fine-tuned for performance. For instance, SVM was trained with different kernel functions and cross validation parameter tuning along with polynomial kernels with various degrees.

```
# RandomizedSearchCV cross validation for SVM , might be useless as we use different poly anyway
svm_params = {
    'C': uniform(0.1, 5),
    'gamma': ['scale', 'auto'],
    'shrinking': [True, False],
    'degree': [2, 3, 4, 5, 6, 7]  # more seems useless
}
```

Also cross validation were done to the random forest, with the following settings-

```
# RandomizedSearchCV cross validation for random forest
rf_params = {
    'n_estimators': randint(10, 1000),
    'max_features': ['sqrt', 'log2', None] + list(np.arange(1, X_train.shape[1] + 1, dtype=int)),
    'max_depth': [None] + list(randint(1, 50).rvs(10)),
    'min_samples_split': randint(2, 11),
    'min_samples_leaf': randint(1, 5),
    'bootstrap': [True, False]
}
```

2.4 Metrics and testing

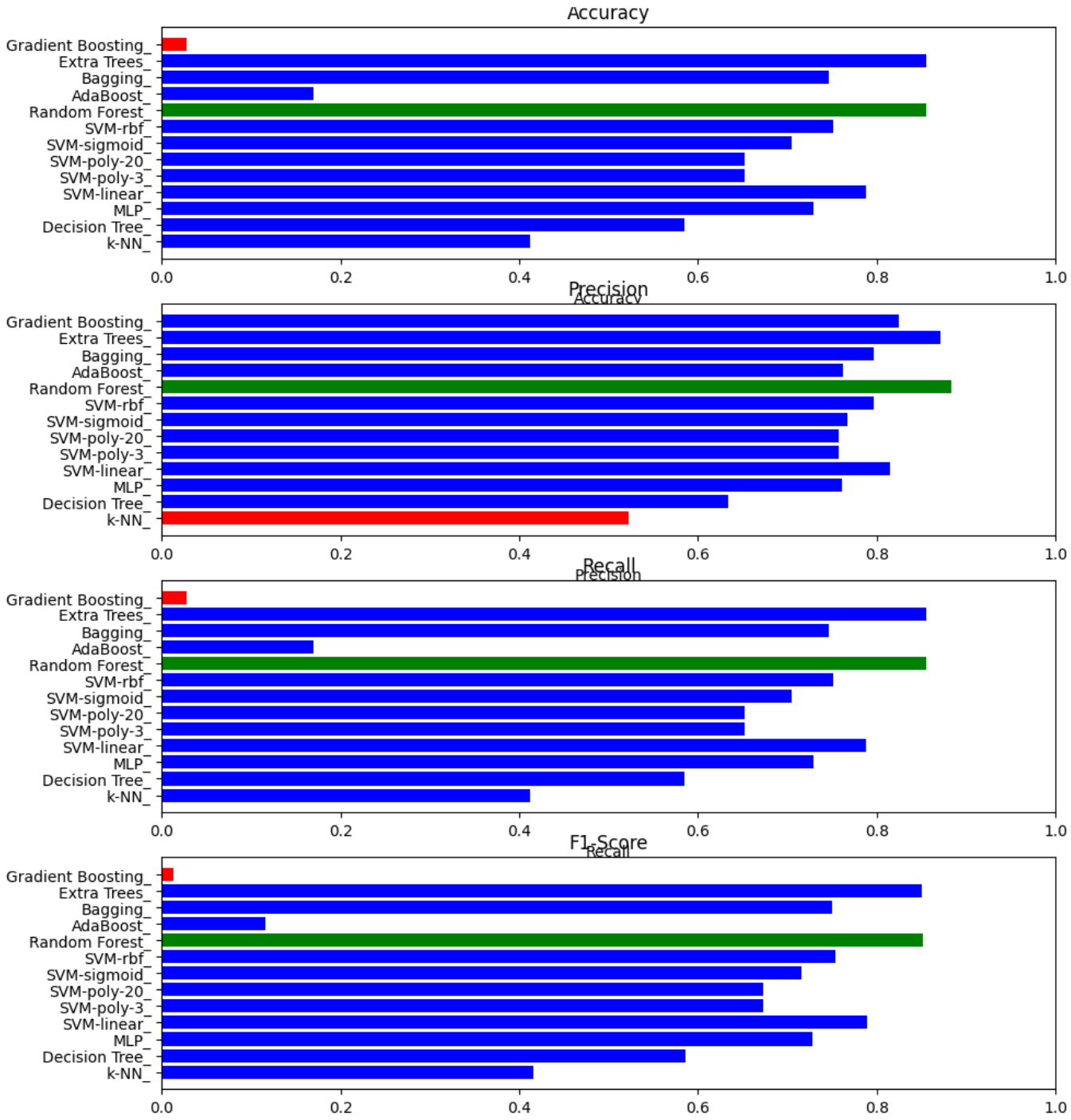
Models were tested against a test set, which was kept separate and unused during the training phase. The evaluation metrics used were accuracy, precision, recall, and F1-score to get a more complete understanding of the results.

3. Results

The performance metrics of the models on the test set were as follows:

Classifier	Accuracy	Precision	Recall	F1-Score
Gradient Boosting	0.0282	0.8243	0.0282	0.0126
k-NN (6)	0.4117	0.5224	0.4117	0.4158
SVM-linear	0.7876	0.8155	0.7876	0.7890
SVM-poly-3	0.6523	0.7576	0.6523	0.6728
SVM-poly-20	0.6523	0.7576	0.6523	0.6728

Classifier	Accuracy	Precision	Recall	F1-Score
SVM-sigmoid	0.7049	0.7679	0.7049	0.7164
SVM-rbf	0.7519	0.7968	0.7519	0.7535
AdaBoost	0.1692	0.7630	0.1692	0.1156
Bagging	0.7462	0.7965	0.7462	0.7507
MLP	0.7293	0.7613	0.7293	0.7281
Decision Tree	0.5846	0.6335	0.5846	0.5865
Random Forest	0.8553	0.8833	0.8553	0.8512
Extra Trees	0.8553	0.8714	0.8553	0.8506



3.1 Model Performance

Looking at the results, it's evident that the **Random Forest** along with **Extra trees** classifiers outshines the others with an accuracy of around 0.85 and an F1-Score of 0.85. This is not unexpected since these are known to perform well due to its ensemble nature.

Now, classifiers like **k-NN** and **Gradient Boosting** had remarkably low accuracies, with the latter having an accuracy of **only 0.0282**. This poor performance from Gradient Boosting might suggest that its parameters were not optimally set, or it might be a consequence of the data distribution or feature set.

Something else that is worth mentioning is that classifiers like **SVM-linear** and **SVM-rbf** performed quite good as well, as for SVMs are known for their ability to handle high dimensional spaces well, which is the case with

the gesture data.

3.2 Metrics

When evaluating a classifier performance, relying on a single metric might be misleading in some cases, therefore a combination of metrics like Accuracy, Precision, Recall, and F1-Score can become handy. As for accuracy, it measures the overall correctness of a model but this can be deceptive, especially with imbalanced datasets. Precision looks at the accuracy of positive predictions. Looking at Gradient Boosting's low accuracy but rather high precision can suggest it's usually right when predicting positives but its overlooking too many at the same time. Therefore, Recall and F1-Score are valuable metrics to get a more balanced score. A good F1-score shows a good balance between precision and recall, as shown for the Random Forest and Extra Trees.

3.3 Hyperparameter Tuning

Hyperparameters can play a vital role in the performance of classifiers. For instance, the kernel choice and C value in SVM, the number of trees in Random Forest, or the depth of the trees in Decision Trees can drastically alter results.

For SVM, we used different kernels to try out different decision boundaries especially in higher dimensions. The SVM all used The **linear** kernel were tried out along with polynomial kernels of various degrees, sigmoid and rbf. The SVM were using randomized search cross validation for tuning, to randomly sample a given number of samples. Finding these optimal settings for these parameters can be tricky depending on the nature and distribution of the data, and be rather time-consuming.

4. Conclusion

With this hands-on experience tinkering with various classification models, we've concluded the importance of understanding the data along with intricacies of each model. Additionally, mastering the art of hyperparameter tuning has proven to be very time-consuming. Furthermore, sometimes the most "advanced" model doesn't necessarily yield the best results. It's also proven essential to pay attention to different kinds of metrics to get a comprehensive understanding of how the classifiers truly perform. In future explorations, try to improve the hyperparameter even more tuning to better the performance.

And finally thank you for taking time reading this :)