Artificial intelligence and Pattern Recognition

D0038E Project

by Magnus Hjörnhede

Group - Py Grp

Task 3. Ensemble Models and PCA

Done in Python

Table of Contents

1. Introduction

This report details the process and results of Task 3, which involves training, testing, and reflecting on various classification models performance using some gesture data with ensemble methods along with PCA.

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**
- **k-NN (k-Nearest Neighbors)**
- **MLP (Multi-layer Perceptron)**
- **SVM** with various kernels:
  - **Linear**
  - **Polynomial of degree 3 (SVM-poly-3)**
  - **Polynomial of degree 20 (SVM-poly-20)**
  - **Sigmoid (SVM-sigmoid)**
- **AdaBoost**
- **Bagging**
- **Extra Trees**
- **Gradient Boosting**
- **Random Forest**

2.2 Classifiers

Below are the setting used for the classifiers

```
classifiers = {
    'k-NN': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'MLP': MLPClassifier(max_iter=1000),
    'SVM-linear': SVC(kernel='linear'),
    'SVM-poly-3': SVC(kernel='poly', degree=3),
    'SVM-poly-7': SVC(kernel='poly', degree=7),

    'Random Forest': RandomForestClassifier(n_estimators=1000),
    'AdaBoost': AdaBoostClassifier(),
    'Bagging': BaggingClassifier(estimator=SVC(kernel='linear')),
    'Extra Trees': ExtraTreesClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(),
}
```

2.3 PCA

When it came to select the PCA threshold, we opted for 95% variance, as it keep a good balance between reducing the dimensionality at the same time keeping most of the information.

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, 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:

3.1 Performance

Ensemble vs non-ensemble results

Method	Type	Accuracy	Precision	F1-Score
k-NN	Non-Ensemble	0.3889	0.5976	0.3676
Decision Tree	Non-Ensemble	0.5093	0.5637	0.4962
MLP	Non-Ensemble	0.7037	0.8031	0.7003
SVM-linear	Non-Ensemble	0.7315 🌟	0.8297	0.7405
SVM-poly-3	Non-Ensemble	0.3241	0.8917	0.3461
SVM-poly-7	Non-Ensemble	0.1019	0.9568	0.0908
Random Forest	Ensemble	0.7778 🌟	0.8703	0.7676
AdaBoost	Ensemble	0.1574	0.8429	0.1158
Bagging	Ensemble	0.7037	0.7954	0.6862
Extra Trees	Ensemble	0.7963 🌟	0.8778	0.7929
Gradient Boosting	Ensemble	0.6111	0.6961	0.6059

3.2 PCA Performance

PCA impact on results

Model	Accuracy	Precision	F1-Score
k-NN	0.3889	0.5976	0.3676
With PCA k-NN	0.4098	0.4972	0.4126

Model	Accuracy	Precision	F1-Score
Decision Tree	0.5093	0.5637	0.4962
With PCA Decision Tree	0.2538	0.2750	0.2551
MLP	0.7037	0.8031	0.7003
With PCA MLP	0.7180	0.7484	0.7191
SVM-linear	0.7315	0.8297 🌟	0.7405
With PCA SVM-linear	0.7726 🌟	0.8010	0.7730
SVM-poly-3	0.3241	0.8917 🌟	0.3461
With PCA SVM-poly-3	0.3289	0.8687	0.3757
SVM-poly-7	0.1019	0.9568 🌟	0.0908
With PCA SVM-poly-7	0.1316	0.9281	0.1594
Random Forest	0.7778 🌟	0.8703 🌟	0.7676
With PCA Random Forest	0.6673	0.7610	0.6740
AdaBoost	0.1574	0.8429	0.1158
With PCA AdaBoost	0.0545	0.9346 🌟	0.0185
Bagging	0.7037	0.7954	0.6862
With PCA Bagging	0.7350	0.7747	0.7389
Extra Trees	0.7963 🌟	0.8778 🌟	0.7929 🌟
With PCA Extra Trees	0.6015	0.7238	0.6091
Gradient Boosting	0.6111	0.6961	0.6059
With PCA Gradient Boosting	0.2669	0.3355	0.2734

🌟 Top 3 Accuracy: With PCA SVM-linear, Random Forest, and Extra Trees. 🌟 Top 3 Precision: SVM-poly-7, With PCA AdaBoost, and SVM-poly-3. 🌟 Top 3 F1-Score: Extra Trees, With PCA SVM-linear, and Random Forest.

Comments on individual methods

Algorithm	Impact of PCA
k-NN	Slightly improved Accuracy and F1-Score but reduced Precision.

Algorithm	Impact of PCA
Decision Tree	Performance decreased across all three categories
MLP	Modest increase in all three metrics.
SVM-linear	Some improvement in all metrics.
SVM-poly-3	Slightly better with PCA, notice the F1-Score.
SVM-poly-7	Slight improvements in Accuracy and F1-Score.
Random Forest	Reduction in performance across all metrics.
AdaBoost	Worse in all performance metrics.
Bagging	Improvements in all three metrics.
Extra Trees	Decreased performance across all metrics.
Gradient Boosting	Significant decline.

4. Conclusion

In overall performance, the ensemble methods generally gave superior performance over non-ensemble methods , which is evident in algorithms such as Random Forest and Extra Trees, that gave the highest accuracy and precision. When looking on non-ensemble we could see SVM-linear also provides good overall performance, with particular high precision.

Looking at the results of the PCA's impact on the models performance varies based on the specific algorithm. For some models like SVM-linear and Bagging, we can se that PCA led to some improvements. On the other side, for others like Decision Tree and AdaBoost, PCA was detrimental to their performance. This shows the importance of evaluating the impact context in which PCA is applied. While PCA can help out with the dimensionality and computational complexity, it can also discard potentially important information from the dataset, leading to varying impacts on different algorithms.

Lastly, both data preprocessing and hyperparameter tuning can play a important role in the outcomes of models. Due to time constraints, an exploration of these aspects was not undertaken in this study and remains a subject for future.

5. Overall Results and reflections

We knew from the very beginning that ensemble methods are powerful, so it wasn't surprising to see the results, but what maybe were interesting was the PCAs impact on certain algorithms, and you have to be careful using this on the dataset.

Finally, some reflections from this project. It has been quite a workload to finish this project. Perhaps some early mistakes where that I made almost a superclass to handle everything and that was in the end very ineffective and hard to work with in the long run. After pondering what to do about it, I've decided to make it very simple instead and sometimes even hardcode some of the results directly into eg. to plotting methods, this helped out alot and make it very easy to keep track over all the adjustments you can make while trying to work with every method. And the best of all you got what I believe a very good understanding how to make model pipelines from the beginning to the end.

Thank you for taking time reading this :)