Gesture UI Programming Project 1 – Fashion Dataset

Contents

Introduction

Student name: Kevin McShane
Student number: G00401808 (random state key 401808)
Project type & dataset chosen: Fashion

This document will report the workings of the first Gesture Based UI project. This project is a task in machine learning and AI using sklearn python library and a chosen dataset provided by Donny Hurley.

The Dataset is labelled and sorted into folders based on the type of article of clothing in the picture. The Dataset is perfectly balanced as each classification has 7000 png files, which is 70,000 files in total.

Methodology

A) The scientific process of making a machine learning model using sklearn.

1. Gather the dataset.
2. Analyse the data format, for example, file type, balanced/unbalanced, is it labelled?
3. Read in the data as the X variable (input) and the y variable (expected output).
4. Split the dataset into a training set and a test set.
5. Choose the appropriate classifiers.
6. Pre-process the data.
7. Conduct a GridSearchCV to find the optimal parameters for the given classifier.
8. Evaluate model performance using the cross_validation method and the score method on the model using the training set.

9. Retrieve final score from the test set / Data visualization.

B) Extracting the data

Getting the X variable included:
Using Glob to read in all the file paths for each image.
Then the images were read in previously all at once but an OS error on local development environments were consistently raised as too many files were open at once, which made processing images in batches of 1000 necessary.

This resulted in X containing image each images features in a 3-dimensional array.
After, the array was reshaped to be 2-dimensional by multiplying y and z in [x, y, z] as such X.reshape(70000, 28*28).

Getting the y variable included:
Simple string manipulation to get the name of the directory the image belonged to and storing it into an NumPy array ensuring it had the same number of rows as the X variable.

C) Data Pre-Processing

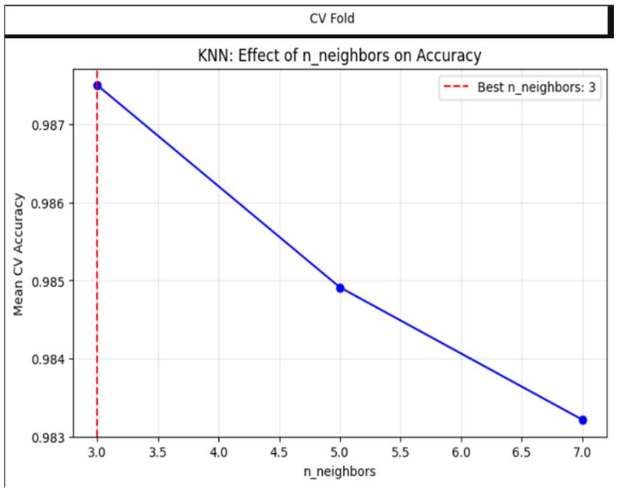
- The images were already grayscale and edited.
- The X variable was normalized by dividing by the maximum value possible np.*divide*(X, 255)
- The dataset was split into a training dataset and test dataset to measure the performance of the model on unseen data at the very end of this project. Cross_val_score from sklearn was used to validate scores to inform which parameters were used.
- The LabelEncoder class and its method fit_transform was used on the y variable as there was a ValueError when using fit () on the KNearestClassifier model.

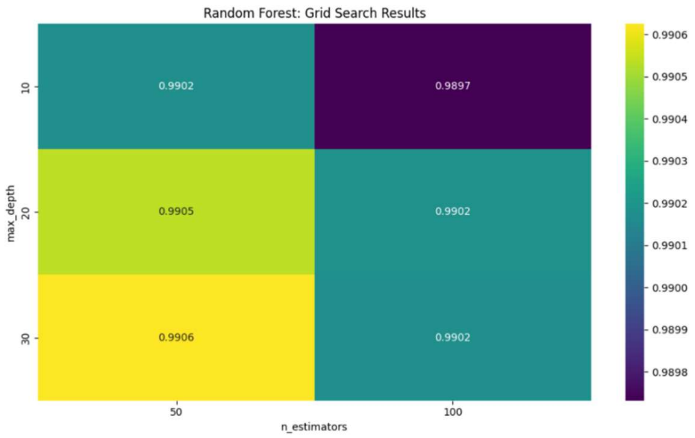
D) Data Analysis and Visualisation

As the dataset was very large it came with a very high computational cost, which served as a challenge.

- Each class has had its most important parameters searched for performance using GridSearchCV, this enhanced my chances at getting the configurations for best fitting models.
- Matplotlib was used to graph the metrics on how each algorithm did in comparison to itself with slightly different features.

- Here are two matplotlib graphs, which compare the parameters relationship to model score.





Experiments and Results

a) Classifier Models

The classifier models chosen were the K-Nearest Classifier, Random Forest classifier and the Logistic Regression Classifier.

Why choose K-Nearest Classifier:
Consistently quick to run in my experience. The model is well capable of being an image classifier.

Why choose Random Forest Classifier:
Random Forest runs efficiently which is required for large datasets causing a very intense computational cost.

Why choose Logistic Regression Classifier:
Minimal hyperparameter tuning makes the need for GridSearchCV a lot less, which greatly reduces computational costs.

The models were chosen to have a clear priority for efficiency in computational cost, this makes it so that any person can view the project and run it with a mid-range device.

b) Results

Final test scores for each model:

K-Nearest: 0.9892857142857143

Random Forest: 0.9914285714285714

Logistic Regression: 0.9910714285714286

Conclusion

The process outlined in the document above was followed successfully, given the fact that the 3 final test scores of each model were above the score 0.98.

This outlined that the process followed was a correct approach to building a machine learning model and that the classification methods chosen were the correct choices.