Homework 1

Machine Learning

Michele Palma 1849661

Academic Year 2023/2024

Contents

1 Overview

The homework aim is to experiment with different models in order to correctly classify the elements of two different datasets in 10 possible classes.

1.1 Dataset Structure

Each dataset contains 50000 vectors and 10 *labels*. Each *label* has 5000 instances which makes both datasets balanced.

There are two different dimensions for the feature vectors in the datasets:

- 100 for Dataset 1
- 1000 for Dataset 2

1.2 Metrics

1.2.1 Accuracy

Measures the overall correctness of the model, but it is not sufficient if the dataset is unbalanced. In this case both datasets are perfectly balanced with 5000 instances for each class.

$$\textbf{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

1.2.2 Precision

Focuses on the accuracy of positive predictions, indicating how many predicted positives were actually positive

$$\textbf{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

1.2.3 Recall

Measures the ability of the model to capture all positive instances

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

1.2.4 F1 Score

A balance between precision and recall, useful when there is an uneven class distribution

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

1.2.5 Evaluation

In order to evaluate the models the "classic" metrics will be used, more precisely the ones described in the previous paragraph. The first evaluation considers models without any hyperparameter while the second one considers the best hyperparameters found through a grid search technique.

1.3 Hyperparameters Search

In order to find the best hyperparameters for each model a grid search approach has been chosen. A 5 Cross Fold validation was used, choosing between the best values of accuracy since both datasets are balanced. Considering the extreme amount of time required by some models (11.5 hours for Random Forest) only the first dataset, which is smaller, was used in order to obtain the best parameters.

1.4 Time

For each model there was taken into consideration the time needed to find the best hyperparameters. This is an interesting element since it describes how some kind of hyperparameters can influence the amount of time taken for the overall process. Generally, once the best parameters were found the models could be trained in just a few minutes (less than 5).

1.5 Models

In this Homework a few models have been considered in order to classify the data given:

- Logistic Regression
- K Nearest Neighbors
- Random Forest
- Support Vector Machine

1.6 Preprocessing

In order to explore if there were any upgrades in terms of performance, a preprocessing phase on the input data was added for each model. There's an additional comparison between using raw data and preprocessed one. In particular a min-max normalization was applied in order to normalize the feature values in a closed $[0, 1]$ interval.

1.7 Data Split

For splitting the data into training and test set it was decided to go for a 40-60 split. After some testing it seemed like the models kept performing very well and additionally there was a lower chance to overfit in this way. There was some experimenting trying even more conventional splits like 60-40, 70-30 and 80-20, but there were no major differences.

2 First Evaluation

As mentioned in the Overview chapter, in the first step only the strictly necessary hyperparameters were set in order to determine which "base" models could be ideal for the multi-class classification.

All the following models performed extremely well with no selected hyperparameters in Python (sklearn). Moreover the choice of the specific models is based on the very different nature of the approaches for classification and they go from simpler to more sophisticated.

The following results refer to dataset 1, nevertheless the results of dataset 2 are very close, differing only by 1-2% at most.

2.1 Logistic Regression

- **Accuracy**: 0.98813
- **Precision**: 0.98810
- **Recall**: 0.98812
- **F1 Score**: 0.98810

2.2 K Nearest Neighbors

- **Accuracy**: 0.98673
- **Precision**: 0.98680
- **Recall**: 0.98681
- **F1 Score**: 0.98680

2.3 Random Forest

- **Accuracy**: 0.9877
- **Precision**: 0.98768
- **Recall**: 0.98767
- **F1 Score**: 0.98767

2.4 Support Vector Machine

- **Accuracy**: 0.9836
- **Precision**: 0.98368
- **Recall**: 0.98363
- **F1 Score**: 0.98365

3 Second Evaluation

During the second evaluation in order to find the best hyperparameters and to obtain the highest accuracy possible, a grid search has been applied over different kind of parameters. For each model there will be an analysis on the considered parameters, on the preprocessing method and the overall performance on both datasets.

3.1 Logistic Regression

3.1.1 Performance on Dataset 1

For dataset 1 Logistic Regression with no preprocessing performed very well, here's the statistics:

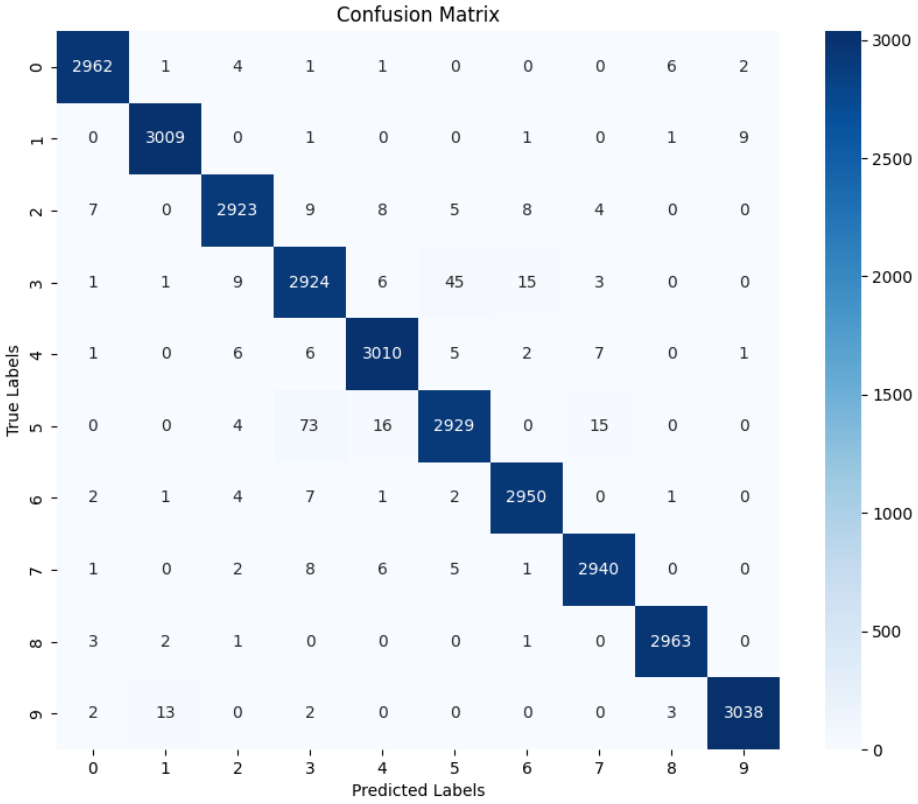

Figure 1: Confusion Matrix for Logistic Regression for Dataset 1

Here's the performance calculated with different kinds of metrics:

- **Accuracy**: 0.98826
- **Precision**: 0.98828
- **Recall**: 0.98830

- **F1 Score**: 0.98828

All values are very close to each other and this suggests that the model it's behaving very well on this classification task.

With preprocessing (min-max normalization) we get slightly different results and more precisely worse results in this case:

- **Accuracy**: 0.9868
- **Precision**: 0.98678
- **Recall**: 0.98686
- **F1 Score**: 0.98680

Of course another preprocessing approach could've improved the performances, but in this case the only one considered was the one chosen for the homework.

```
LogisticRegression(C=1, penalty='l2', solver='newton-cg'),

LogisticRegression(C=0.001, penalty='l2', solver='saga')
```

Figure 2: Best parameters found for Logistic Regression using sklearn. The first instance of Logistic Regression has the best parameters considering preprocessing while the second one has the best ones with no preprocessing

3.2 Time

The total time taken for grid search for dataset 1 with no preprocessing was 20 minutes while for raw data was 33 minutes.

3.2.1 Performance on Dataset 2

The predictions are made for dataset 2 using the same parameters found in the grid search for the first dataset. No preprocessing applied. The overall performance is still very good.

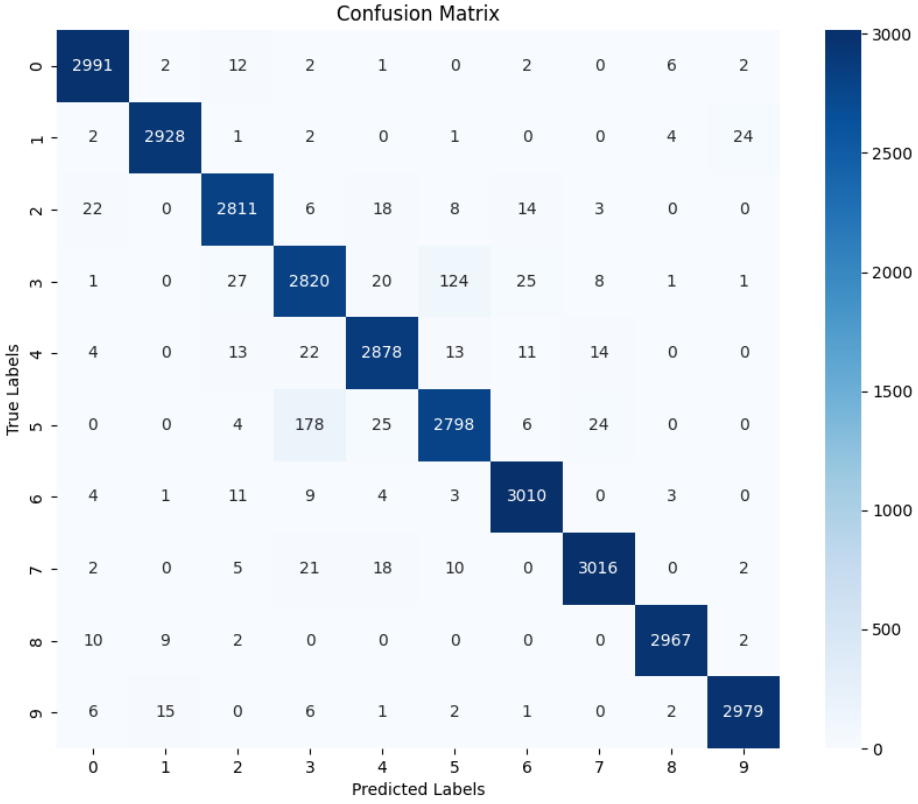

Figure 3: Confusion Matrix for Logistic Regression for Dataset 2

- **Accuracy**: 0.97326
- **Precision**: 0.97332
- **Recall**: 0.97333
- **F1 Score**: 0.97330

Trying to higher the C parameter led to no improvements, in fact the performance got worse. Applying normalization slightly lowered the accuracy, but the other metrics improved. Considering that a good model should be balanced and F1 score represents that, it's acceptable to have a slightly worse accuracy as a tradeoff.

- **Accuracy**: 0.9727
- **Precision**: 0.97257
- **Recall**: 0.97258
- **F1 Score**: 0.97256

3.3 Random Forest

3.3.1 Performance on Dataset 1

For the first dataset Random Forest performed very well with no preprocessing, here's the confusion matrix:

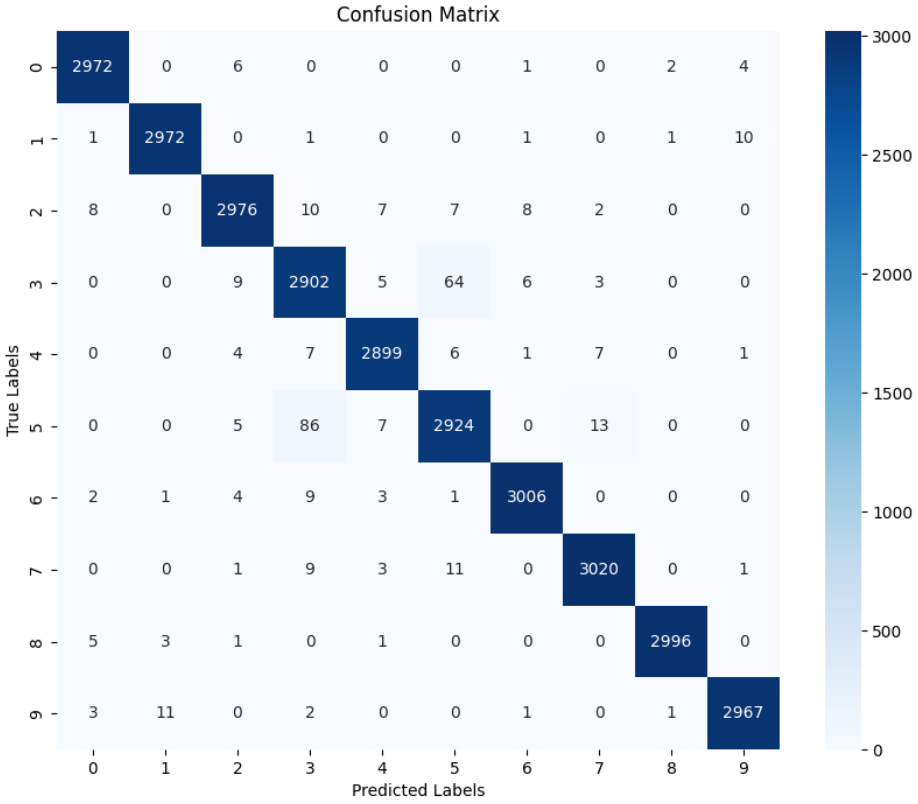

Figure 4: Confusion Matrix for Random Forest for Dataset 1 (no preprocessing)

Here's the different metrics:

- **Accuracy**: 0.9878
- **Precision**: 0.98783
- **Recall**: 0.98782
- **F1 Score**: 0.98782

Very similarly to what happened in Logistic Regression, the results are slighly worse after applying preprocessing:

- **Accuracy**: 0.98706
- **Precision**: 0.98711

- **Recall**: 0.98709
- **F1 Score**: 0.98710

```
RandomForestClassifier(bootstrap= True, criterion= 'entropy', max_depth= None, max_features= 'log2',
                       min_samples_leaf= 4, min_samples_split= 2, n_estimators= 200),

RandomForestClassifier(bootstrap= True, criterion= 'entropy', max_depth= 10, max_features= 'log2',
                       min_samples_leaf= 2, min_samples_split= 10, n_estimators= 100)
```

Figure 5: Best parameters found for Random Forest using sklearn

3.4 Time

The total time taken for grid search for dataset 1 was very similar for preprocessed data and raw data. It was extremely slow with a total of 11.5 hours of search. Compared to other models the amount of parameters was much higher.

3.4.1 Performance on Dataset 2

Even if the best parameters were found for the first dataset they allowed the model to also perform very well on the second one. Considering that the training took around 11.5 hours for the first one and that single folds were taking up to 3 minutes for the second dataset, this was considered the best option since the time required was not feasible.

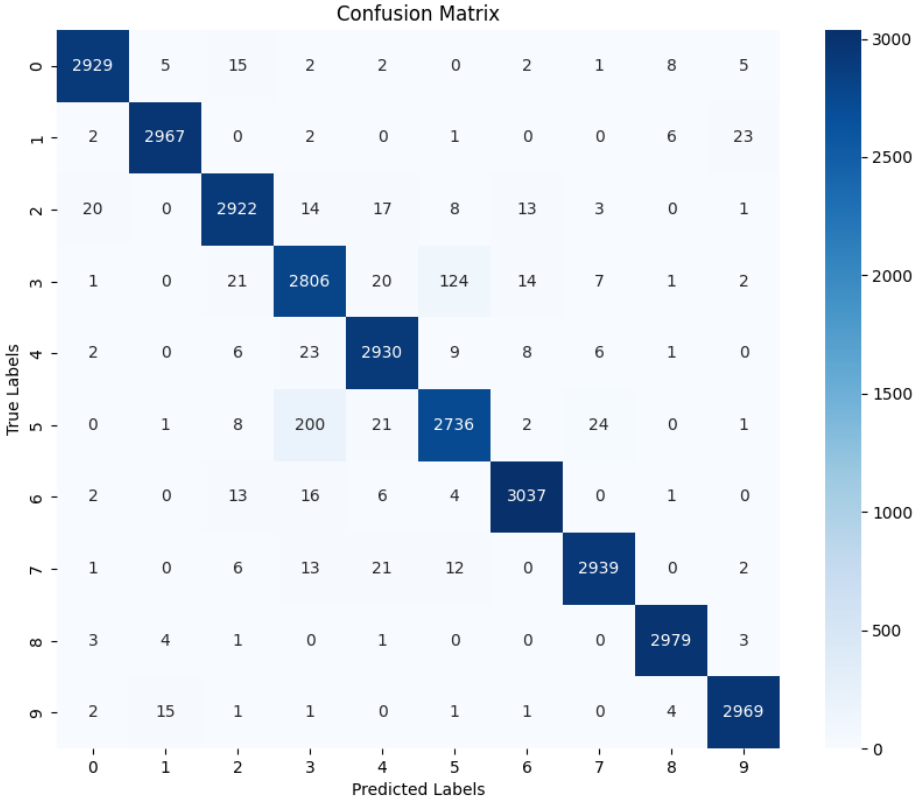

Figure 6: Confusion Matrix for Random Forest for Dataset 2 (no preprocessing)

- **Accuracy**: 0.9738
- **Precision**: 0.97385
- **Recall**: 0.97377
- **F1 Score**: 0.97376

In this case applying preprocessing increased the performance of the model:

- **Accuracy**: 0.9739
- **Precision**: 0.97401
- **Recall**: 0.97399
- **F1 Score**: 0.97398

3.5 Support Vector Machine

3.5.1 Performance on Dataset 1

For the first dataset SVM performed very well with no preprocessing, here's the confusion matrix:

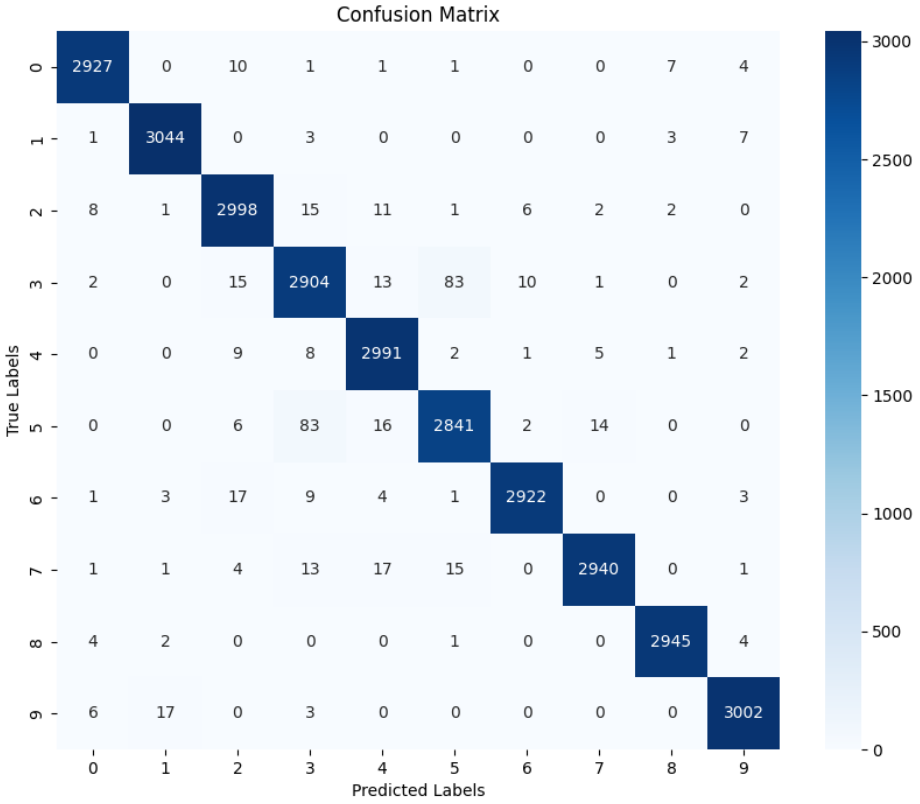

Figure 7: Confusion Matrix for SVM for Dataset 1 (no preprocessing)

Here's the different metrics:

- **Accuracy**: 0.9838
- **Precision**: 0.98385
- **Recall**: 0.98379
- **F1 Score**: 0.98381

In SVM the difference is more noticeable compared to previous models, in fact preprocessing increased all the metrics by 0.45%, which is a very good improvement:

- **Accuracy**: 0.9883
- **Precision**: 0.98837

- **Recall**: 0.98837
- **F1 Score**: 0.98836

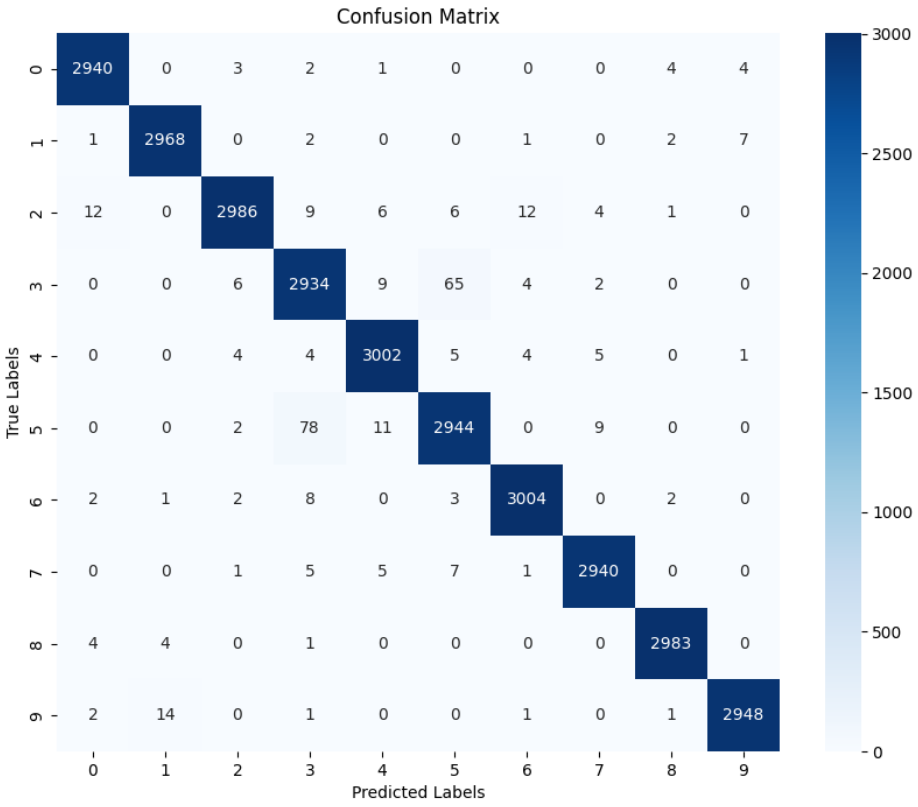

Figure 8: Confusion Matrix for SVM for Dataset 1 (preprocessing)

```
SVC(C=0.1, class_weight='balanced', degree=2, gamma=0.1, kernel='linear'),

SVC(C=1, class_weight=None, degree=2, gamma=0.1, kernel='poly')
```

Figure 9: Best parameters found for SVM using sklearn

3.6 Time

The total time taken for grid search for dataset 1 with no preprocessing was 1 hour and 15 minutes while for preprocessing it took 25 minutes, a substantial difference.

3.6.1 Performance on Dataset 2

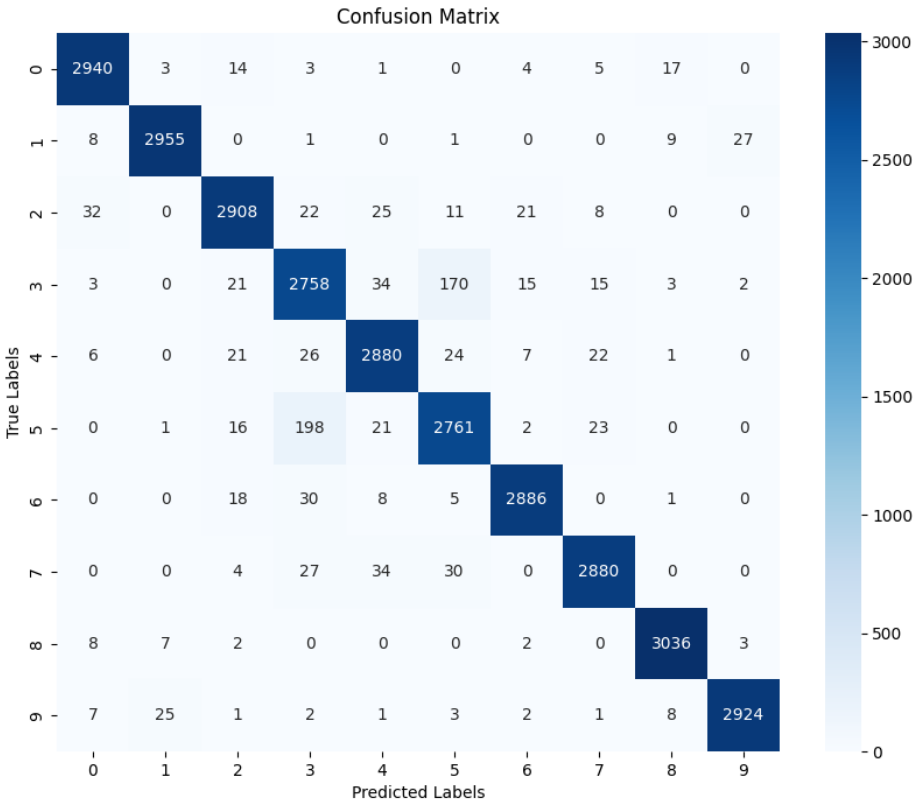

Figure 10: Confusion Matrix for SVM for Dataset 2 (no preprocessing)

- **Accuracy**: 0.96426
- **Precision**: 0.96445
- **Recall**: 0.96434
- **F1 Score**: 0.96438

Respect to dataset 1 the increase in performance by applying preprocessing is greater than 1.3%:

- **Accuracy**: 0.9731
- **Precision**: 0.97297
- **Recall**: 0.97305
- **F1 Score**: 0.97299

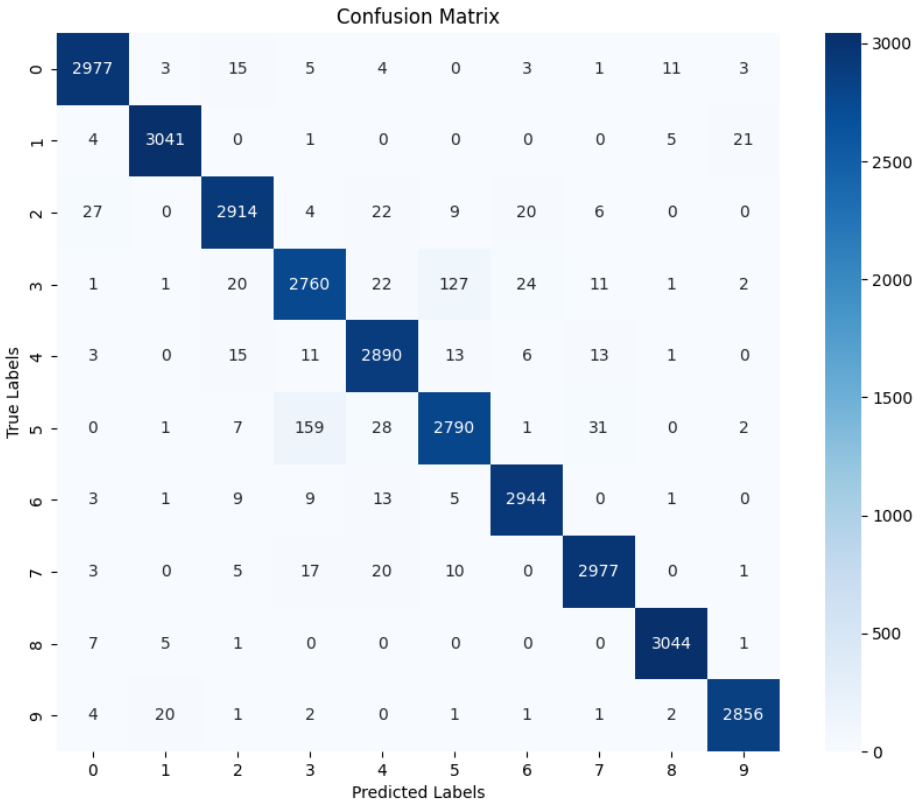

Figure 11: Confusion Matrix for SVM for Dataset 2 (preprocessing)

3.7 K-Nearest Neighbors

3.7.1 Performance on Dataset 1

For the first dataset KNN performed very well with no preprocessing, here's the confusion matrix:

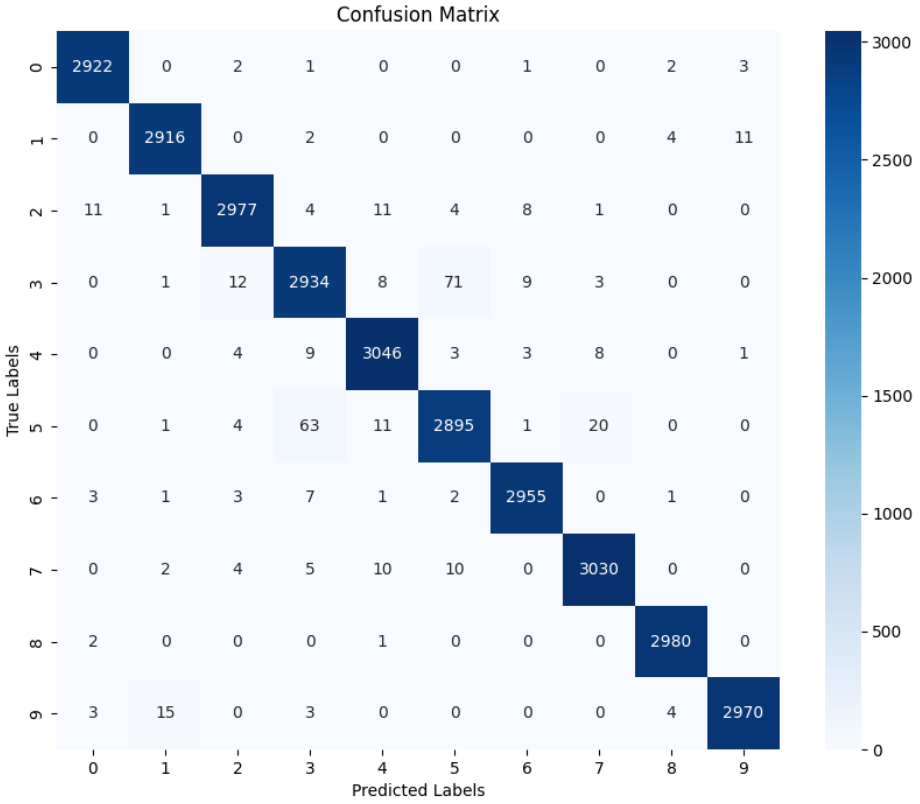

Figure 12: Confusion Matrix for KNN for Dataset 1 (no preprocessing)

Here's the different metrics:

- **Accuracy**: 0.9875
- **Precision**: 0.98754
- **Recall**: 0.98756
- **F1 Score**: 0.98754

In KNN the difference is more noticeable compared to previous models, in fact preprocessing increased all the metrics by 0.45%, which is a very good improvement:

- **Accuracy**: 0.9867
- **Precision**: 0.98664

- **Recall**: 0.98664
- **F1 Score**: 0.98663

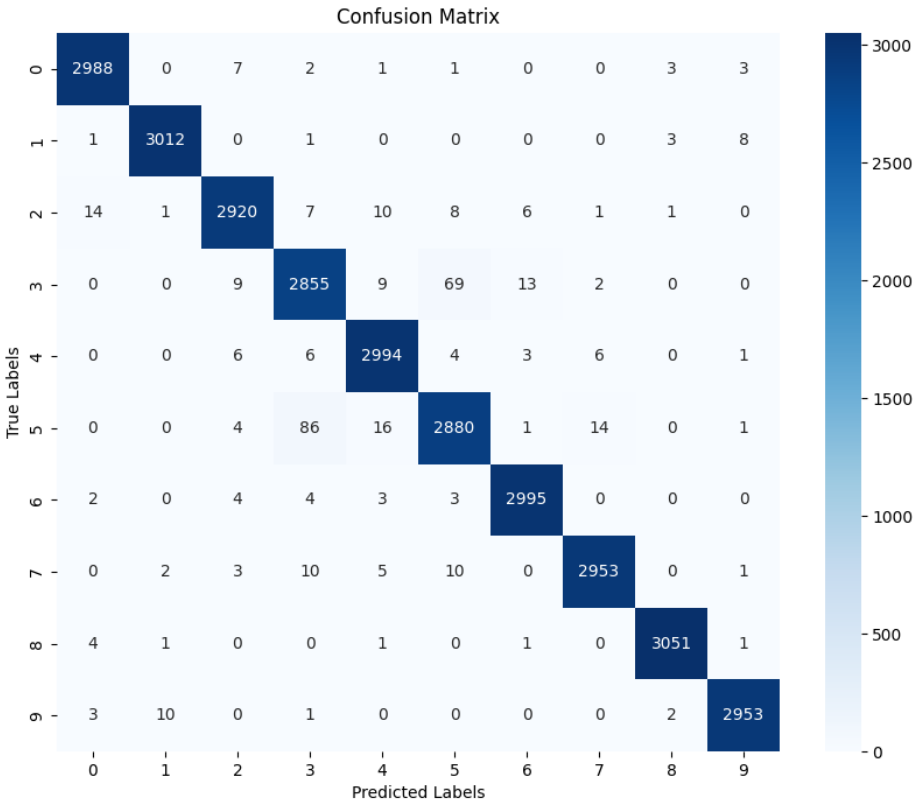

Figure 13: Confusion Matrix for KNN for Dataset 1 (preprocessing)

```
KNeighborsClassifier(metric='euclidean', n_neighbors=13, p=1, weights='distance'),

KNeighborsClassifier(metric='euclidean', n_neighbors=7, p=1, weights='uniform')
```

Figure 14: Best parameters found for KNN using sklearn

Using a number of neighbors equal to $\sqrt{n}$ where n is the number of data points resulted in better performance of the model.

No preprocessing applied with 141 neighbors:

- **Accuracy**: 0.98796
- **Precision**: 0.98792
- **Recall**: 0.98793

- **F1 Score**: 0.98792

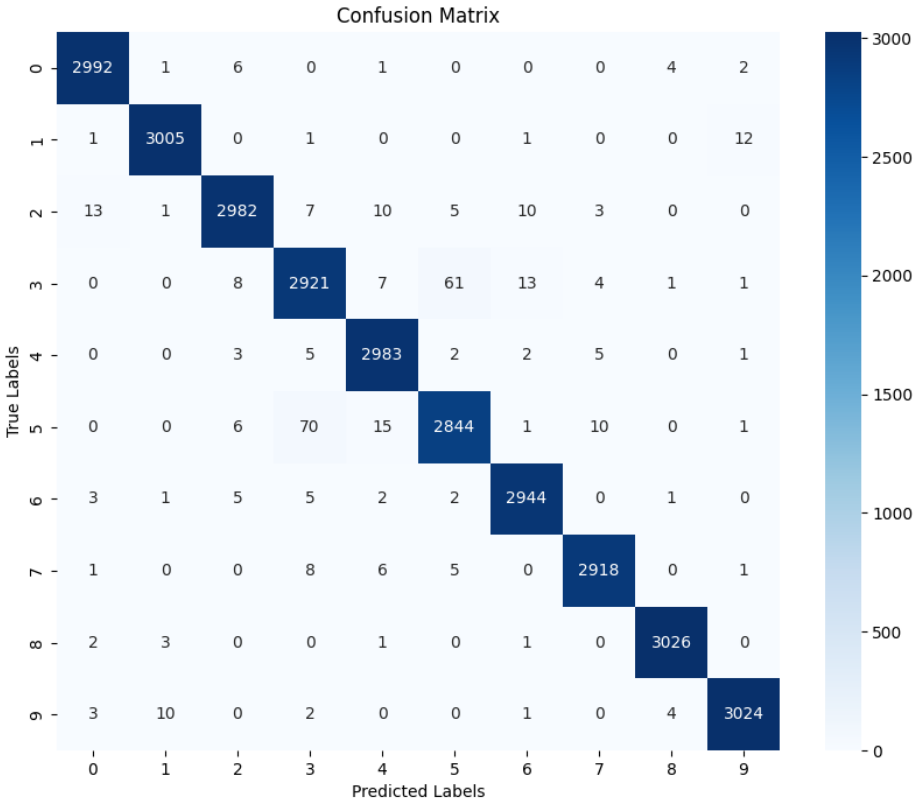

Figure 15: Confusion Matrix for KNN for Dataset 1 (no preprocessing and using sqrt(n) neighbors)

Preprocessed data with 141 neighbors:

- **Accuracy**: 0.988
- **Precision**: 0.98797
- **Recall**: 0.98799
- **F1 Score**: 0.98797

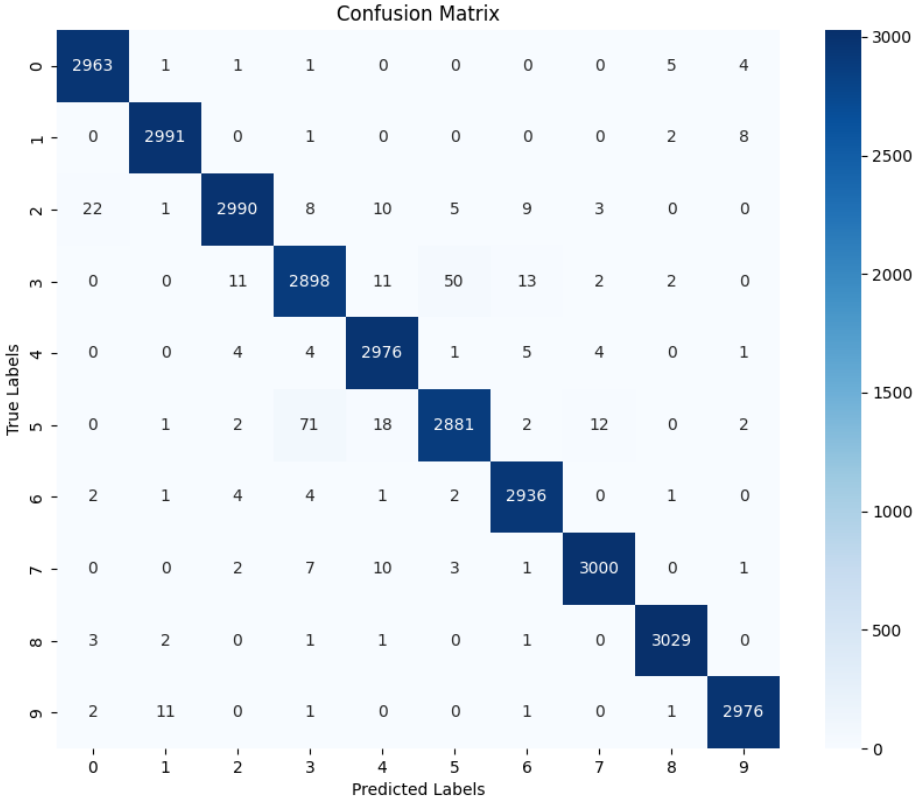

Figure 16: Confusion Matrix for KNN for Dataset 1 (preprocessing and using sqrt(n) neighbors)

3.8 Time

The total time taken for grid search for dataset 1 was similar between raw and preprocessed data, in fact it took 3 hours.

3.8.1 Performance on Dataset 2

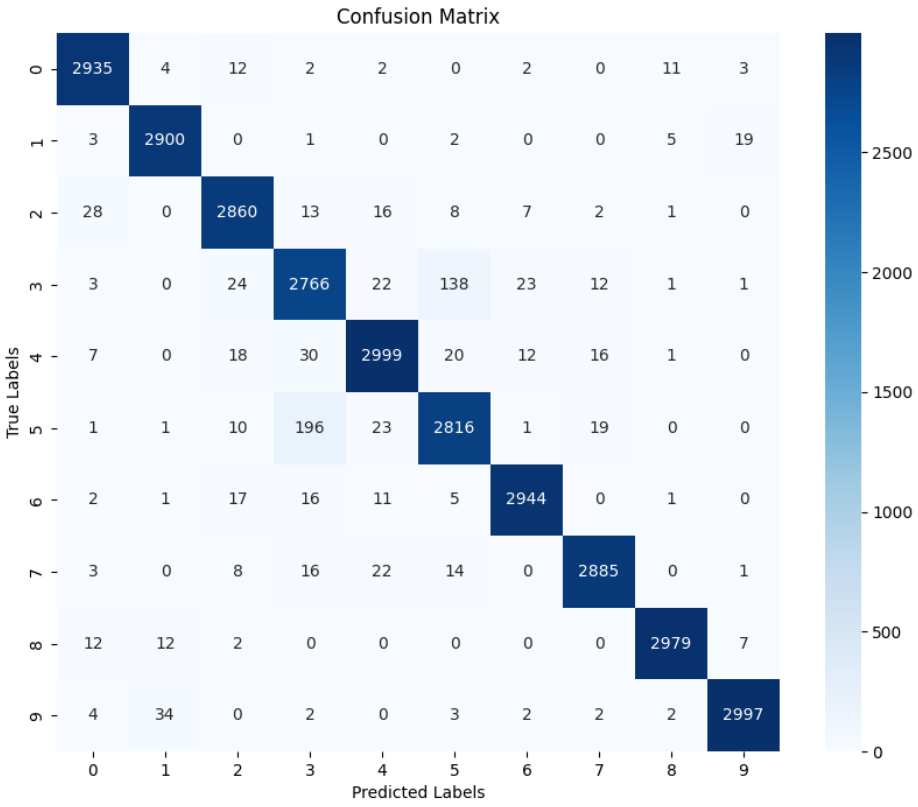

Figure 17: Confusion Matrix for KNN for Dataset 2 (no preprocessing)

Compared to dataset 1, the parameters chosen are still performing well, but 2.1% worse than dataset 1.

- **Accuracy**: 0.96936
- **Precision**: 0.96949
- **Recall**: 0.96953
- **F1 Score**: 0.96949

Respect to the no preprocessing version there's an increase in performance by 1.1%, however it performs 1% worse on average compared to dataset 1:

- **Accuracy**: 0.97123
- **Precision**: 0.97131
- **Recall**: 0.97139

- **F1 Score**: 0.97135

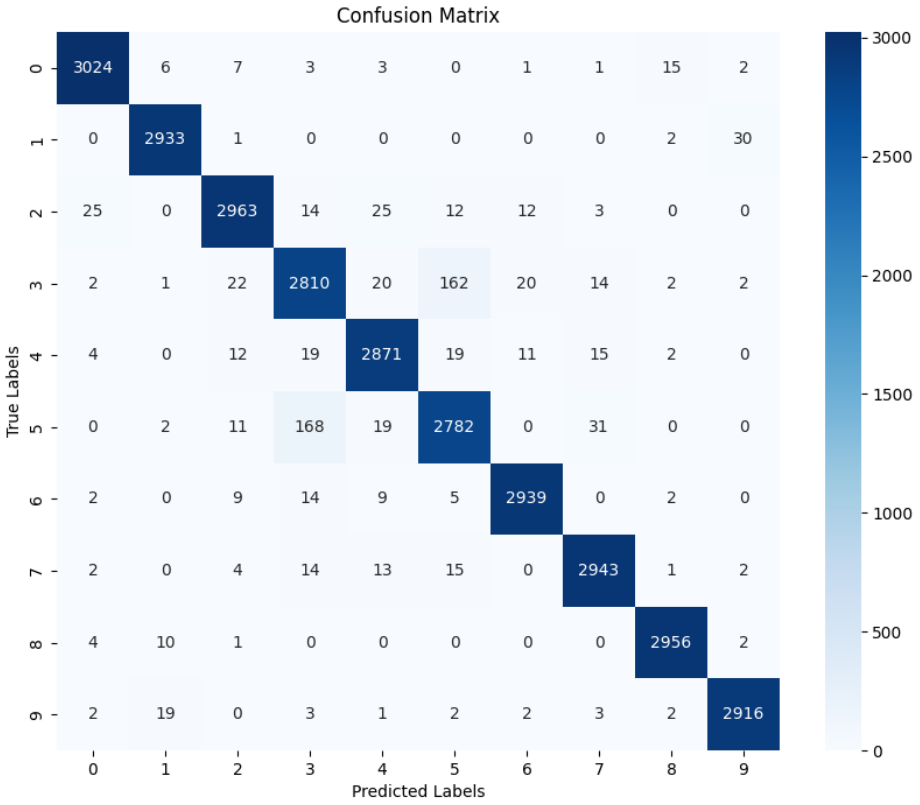

Figure 18: Confusion Matrix for KNN for Dataset 2 (preprocessing)

4 Blind Tests

In order to make predictions on the blind tests the models with the best $F1$ score were chosen. For blind test 1 with $d = 100$ the chosen model is:

- SVM (0.98836)

While for the blind test 2 with $d = 1000$ the chosen model is:

- Random Forest (0.97398)

The predictions made can be found in the blind test csv files.

5 Conclusions

In conclusion it was possible to observe that for dataset 1 the models had the best $F1$ values in combination with preprocessing. Out of the 4 models the best individual performances are obtained by Logistic Regression and SVM by applying preprocessing, while for the remaining two it was by training on raw data. The situation is completely opposite for feature vectors of d equal to 1000 for dataset 2. In fact all models with the best value of F1 score had preprocessed the data. This shows the tendency of obtaining better results with growing values of d with processed data.