**Implementation and performance assessment methods:**

In order to find best performing model, four different types of classification models were assessed:

* K-nearest neighbours
* Multi-layer Perceptron
* Support vectors
* Decision trees

In order to evaluate performance of each model, the following measurements were taken into an account:

* Accuracy
* Precision
* Recall

**Pre-processing:**

Due to the computational limitations, it was difficult to train models with different hyperparameters and test their performance most optimally. In order to adjust computational times, a subset of the original data had to be used. The original file was around 70000 lines long, however, only 10000 arbitrary chosen lines were used when testing hyper-parameters.

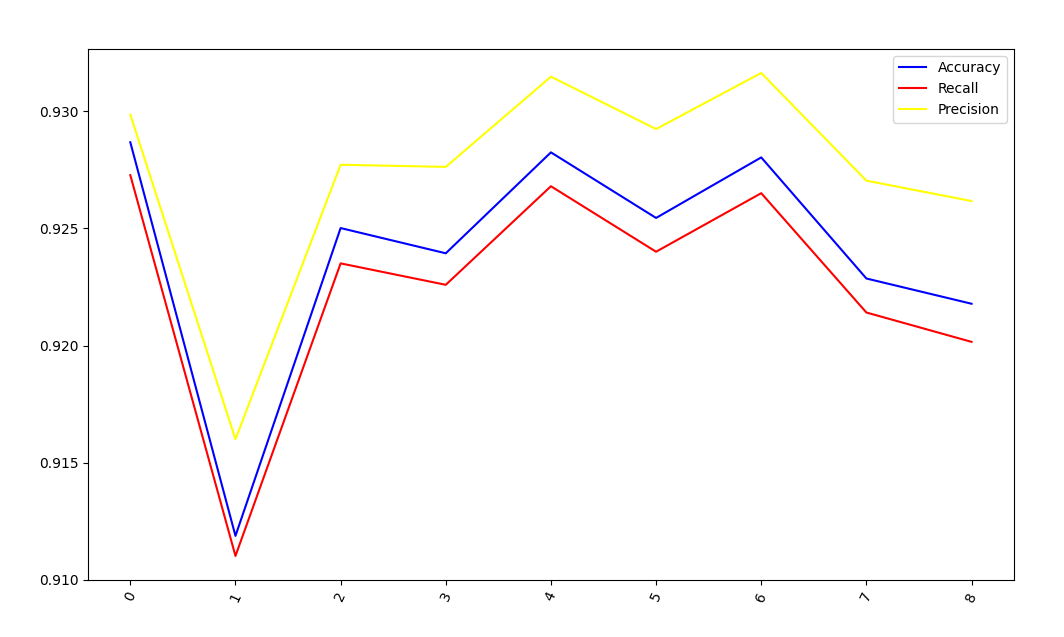
**K-nearest neighbours**

**Choosing hyperparameters and evaluation**

Probably the most important feature to test in the k-nearest neighbours’ algorithm is the number of neighbours taken into a consideration when predicting a point. N\_neighbours values 1 to 9 were tested using 5-fold cross-validation. In addition, two different weighting techniques were tested; uniform and distance based, which favoured points that are closer.

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Accuracy** | **Precision** | **Recall** |
| 1 | **0.9287 (+/- 0.0127)** | **0.9299 (+/- 0.0132)** | **0.9273 (+/- 0.0126)** |
| 2 | 0.9119 (+/- 0.0150) | 0.9160 (+/- 0.0143) | 0.9110 (+/- 0.0154) |
| 3 | 0.9250 (+/- 0.0122) | 0.9277 (+/- 0.0123) | 0.9235 (+/- 0.0131) |
| 4 | 0.9239 (+/- 0.0110) | 0.9276 (+/- 0.0096) | 0.9226 (+/- 0.0122) |
| 5 | 0.9282 (+/- 0.0143) | 0.9315 (+/- 0.0148) | 0.9268 (+/- 0.0151) |
| 6 | 0.9254 (+/- 0.0172) | 0.9292 (+/- 0.0169) | 0.9240 (+/- 0.0179) |
| 7 | 0.9280 (+/- 0.0136) | 0.9316 (+/- 0.0134) | 0.9265 (+/- 0.0142) |
| 8 | 0.9229 (+/- 0.0154) | 0.9270 (+/- 0.0152) | 0.9214 (+/- 0.0166) |
| 9 | 0.9218 (+/- 0.0183) | 0.9262 (+/- 0.0178) | 0.9202 (+/- 0.0189) |

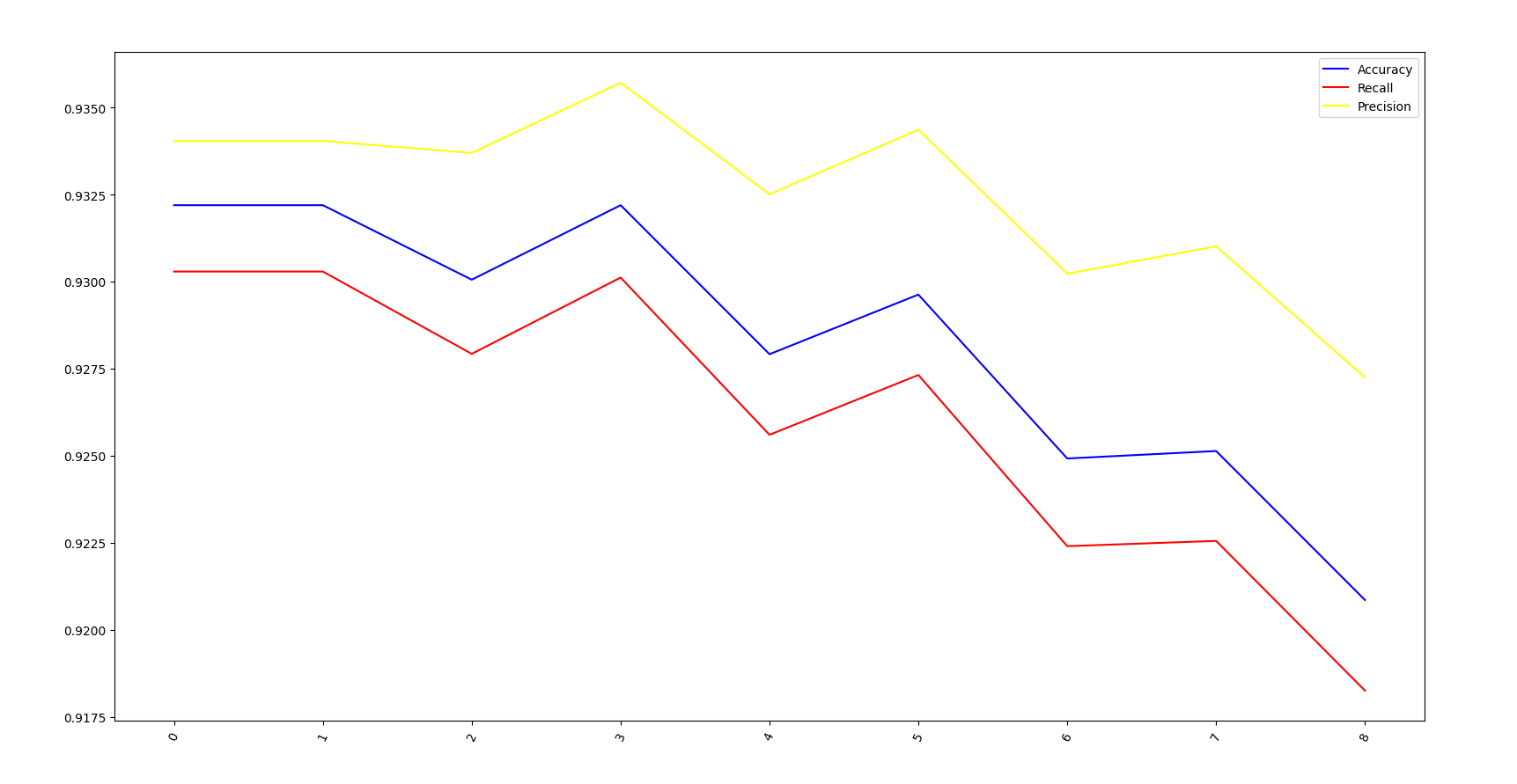
\* (+/- n) refers to the difference between maximum and minimum values received for each fold.



**K-nearest neighbours with distance weights**

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Accuracy** | **Precision** | **Recall** |
| 1 | 0.9322 (+/- 0.0155) | 0.9340 (+/- 0.0137) | 0.9303 (+/- 0.0165) |
| 2 | 0.9322 (+/- 0.0155) | 0.9340 (+/- 0.0137) | 0.9303 (+/- 0.0165) |
| 3 | 0.9301 (+/- 0.0138) | 0.9337 (+/- 0.0121) | 0.9279 (+/- 0.0144) |
| 4 | **0.9322 (+/- 0.0090)** | **0.9357 (+/- 0.0076)** | **0.9301 (+/- 0.0094)** |
| 5 | 0.9279 (+/- 0.0123) | 0.9325 (+/- 0.0099) | 0.9256 (+/- 0.0126) |
| 6 | 0.9296 (+/- 0.0107) | 0.9344 (+/- 0.0105) | 0.9273 (+/- 0.0108) |
| 7 | 0.9249 (+/- 0.0161 | 0.9302 (+/- 0.0138) | 0.9224 (+/- 0.0163) |
| 8 | 0.9251 (+/- 0.0163) | 0.9310 (+/- 0.0141) | 0.9226 (+/- 0.0167) |
| 9 | 0.9209 (+/- 0.0208) | 0.9272 (+/- 0.0171) | 0.9183 (+/- 0.0212) |

\* (+/- n) refers to the difference between maximum and minimum values received for each fold.



**Results of KNN:**

The best accuracy with uniform weights had number of neighbours equal to one, although the results were very close, which came a bit as a surprise as a higher variety was anticipated.

For the weighted distance, number of neighbours equal to four had the best results, regarding accuracy precision and recall.

Even though there is a slight difference in the depending on the number of neighbours taken into account between different weighting techniques, the end result is quite similar.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Weighting** | **Accuracy** | **Precision** | **Recall** |
| 4 | Distance-based | **0.9322 (+/- 0.0090)** | **0.9357 (+/- 0.0076)** | **0.9301 (+/- 0.0094)** |
| 1 | Uniform | 0.9287 (+/- 0.0127) | 0.9299 (+/- 0.0132) | 0.9273 (+/- 0.0126) |

Hence, the best parameters for KNN algorithm were number of neighbours = 4, distance = weight-based.

**Support vectors**

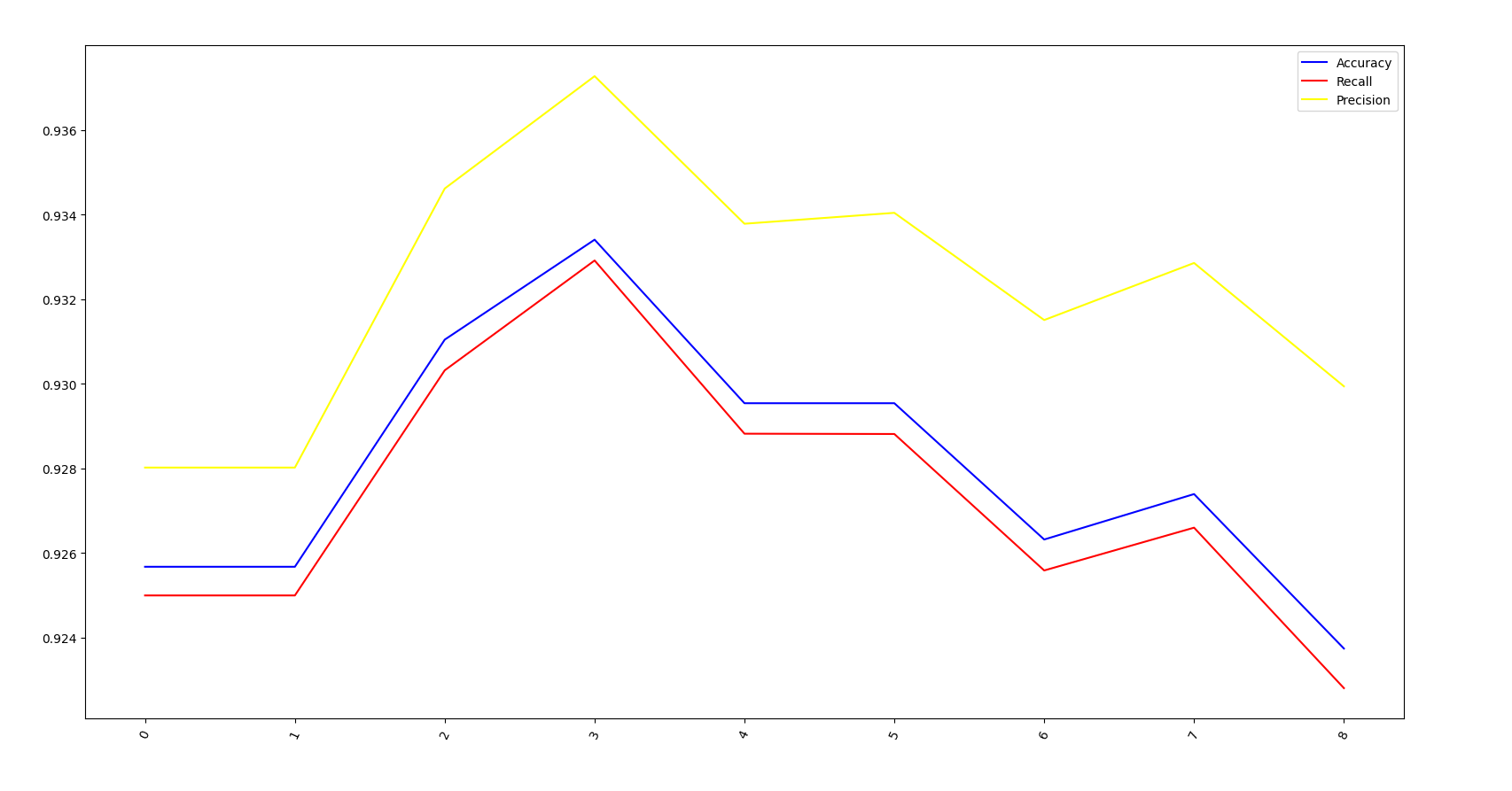
**Choosing hyperparameters and evaluation**

The most important parameter to choose is the kernel. The following kernels were tested:

* poly
* rbf
* sigmoid

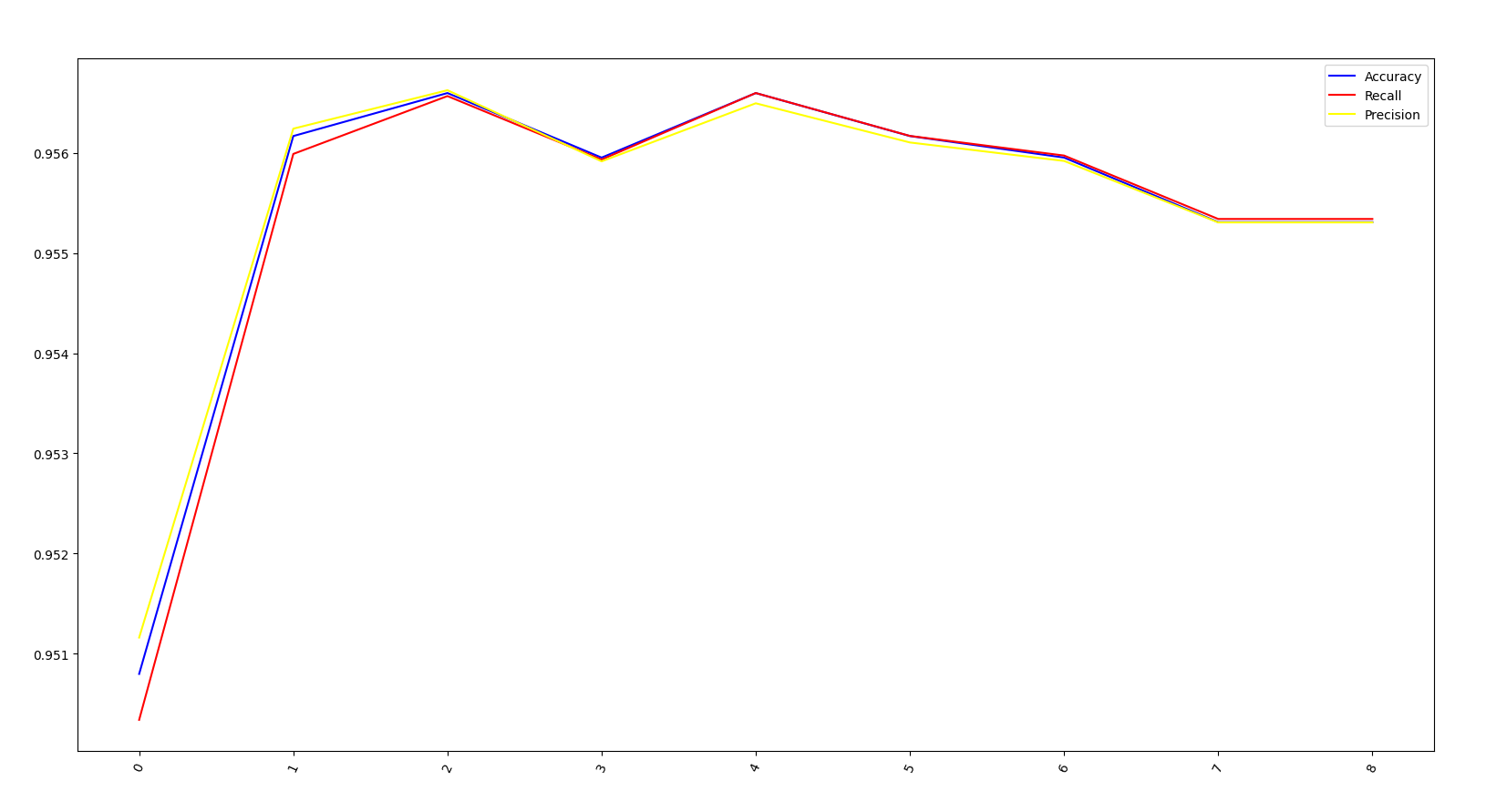
**Polynomial**

|  |  |  |  |
| --- | --- | --- | --- |
| **Degree** | **Accuracy** | **Precision** | **Recall** |
| 1 | 0.9257 (+/- 0.0205) | 0.9280 (+/- 0.0199) | 0.9250 (+/- 0.0204) |
| 2 | 0.9257 (+/- 0.0205) | 0.9280 (+/- 0.0199) | 0.9250 (+/- 0.0204) |
| 3 | 0.9310 (+/- 0.0090) | 0.9346 (+/- 0.0092) | 0.9303 (+/- 0.0089) |
| 4 | **0.9334 (+/- 0.0171)** | **0.9373 (+/- 0.0156)** | **0.9329 (+/- 0.0172)** |
| 5 | 0.9295 (+/- 0.0131) | 0.9338 (+/- 0.0108) | 0.9288 (+/- 0.0130) |
| 6 | 0.9295 (+/- 0.0119) | 0.9340 (+/- 0.0096) | 0.9288 (+/- 0.0116) |



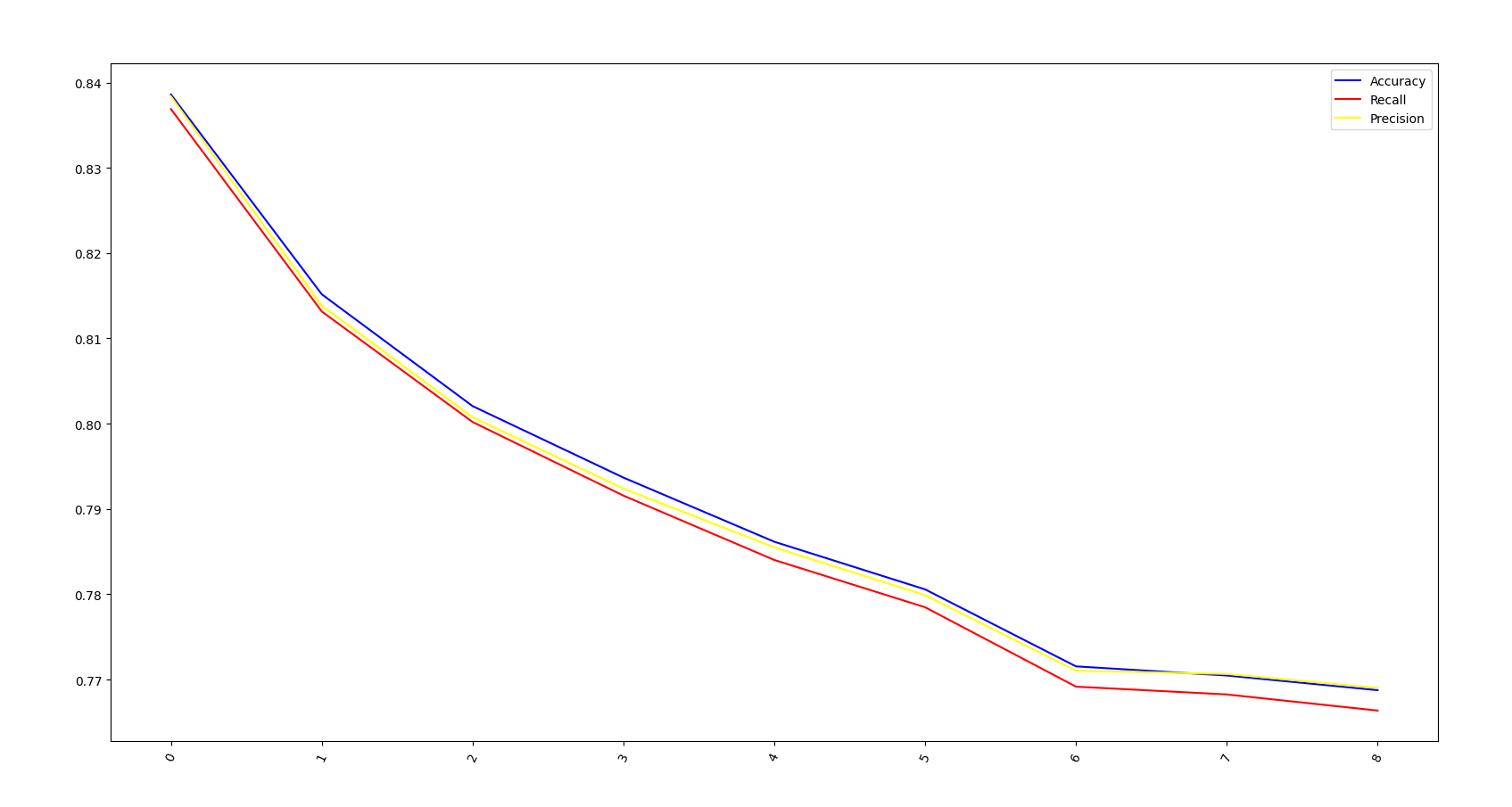
**RBF**

|  |  |  |  |
| --- | --- | --- | --- |
| **Regularization** | **Accuracy** | **Precision** | **Recall** |
| 1 | 0.9508 (+/- 0.0228) | 0.9512 (+/- 0.0227) | 0.9503 (+/- 0.0224) |
| 2 | 0.9562 (+/- 0.0175) | 0.9562 (+/- 0.0169) | 0.9560 (+/- 0.0171) |
| 3 | **0.9566 (+/- 0.0166)** | **0.9566 (+/- 0.0163)** | **0.9566 (+/- 0.0161)** |
| 4 | 0.9560 (+/- 0.0143) | 0.9559 (+/- 0.0141) | 0.9559 (+/- 0.0138) |
| 5 | 0.9566 (+/- 0.0144) | 0.9565 (+/- 0.0144) | 0.9566 (+/- 0.0139) |
| 6 | 0.9562 (+/- 0.0148 | 0.9561 (+/- 0.0150) | 0.9562 (+/- 0.0144) |



**Sigmoid**

|  |  |  |  |
| --- | --- | --- | --- |
| **Regularization** | **Accuracy** | **Precision** | **Recall** |
| 1 | **0.8386 (+/- 0.0068)** | **0.8384 (+/- 0.0060)** | **0.8369 (+/- 0.0070)** |
| 2 | 0.8152 (+/- 0.0190) | 0.8138 (+/- 0.0172) | 0.8132 (+/- 0.0182) |
| 3 | 0.8021 (+/- 0.0156) | 0.8008 (+/- 0.0151) | 0.8002 (+/- 0.0146) |
| 4 | 0.7937 (+/- 0.0162) | 0.7924 (+/- 0.0160) | 0.7916 (+/- 0.0151) |
| 5 | 0.7862 (+/- 0.0171) | 0.7855 (+/- 0.0186) | 0.7840 (+/- 0.0161) |
| 6 | 0.7806 (+/- 0.0208) | 0.7799 (+/- 0.0226) | 0.7785 (+/- 0.0202) |



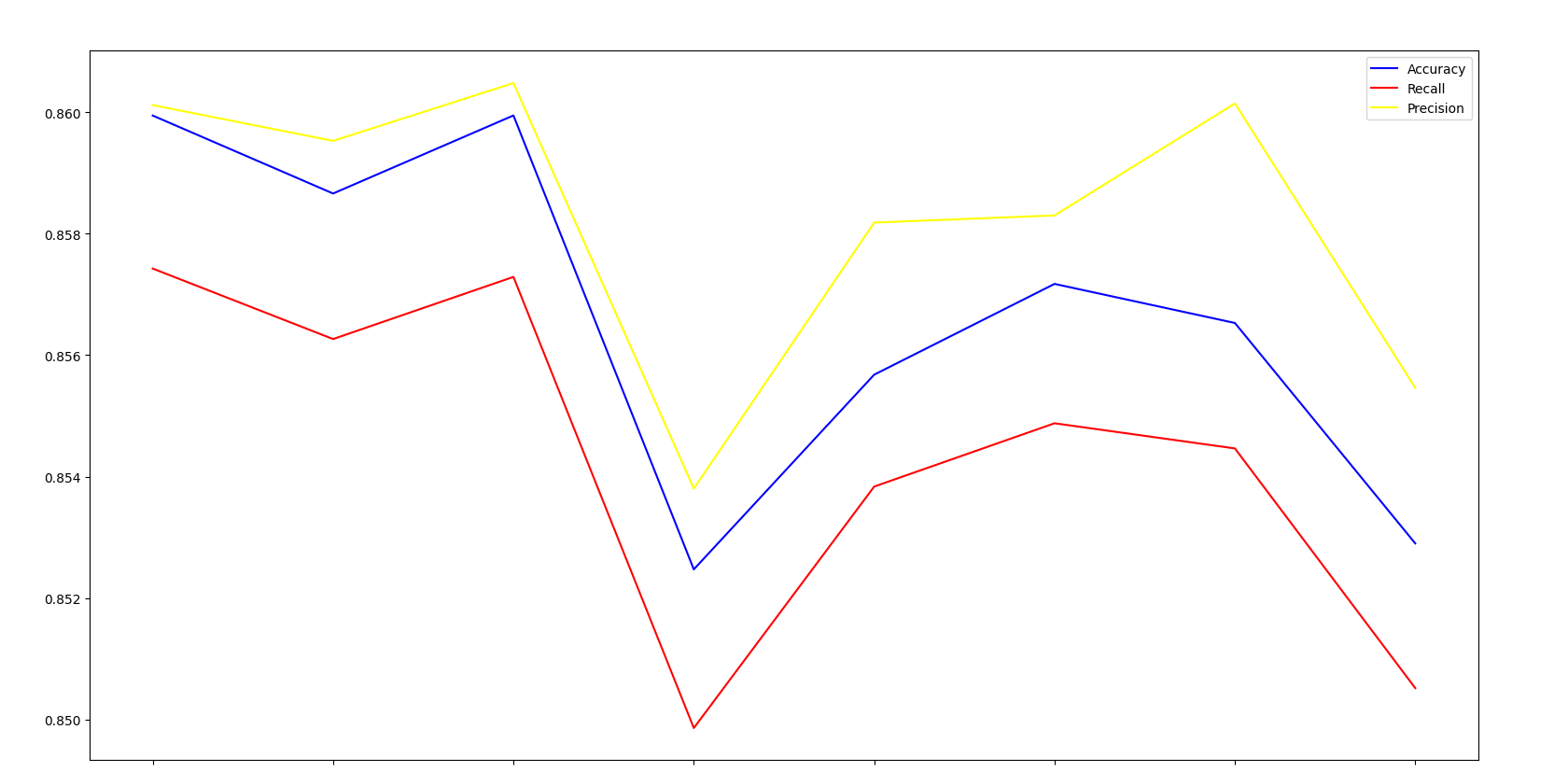
**Results of support vector classification:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Degree/C** | **Kernel** | **Accuracy** | **Precision** | **Recall** |
| 4 | Polynomial | 0.9334 (+/- 0.0171) | 0.9373 (+/- 0.0156) | 0.9329 (+/- 0.0172) |
| 3 | RBF | **0.9566 (+/- 0.0166)** | **0.9566 (+/- 0.0163)** | **0.9566 (+/- 0.0161)** |
| 1 | Sigmoid | 0.8386 (+/- 0.0068) | 0.8384 (+/- 0.0060) | 0.8369 (+/- 0.0070) |

**MLP**

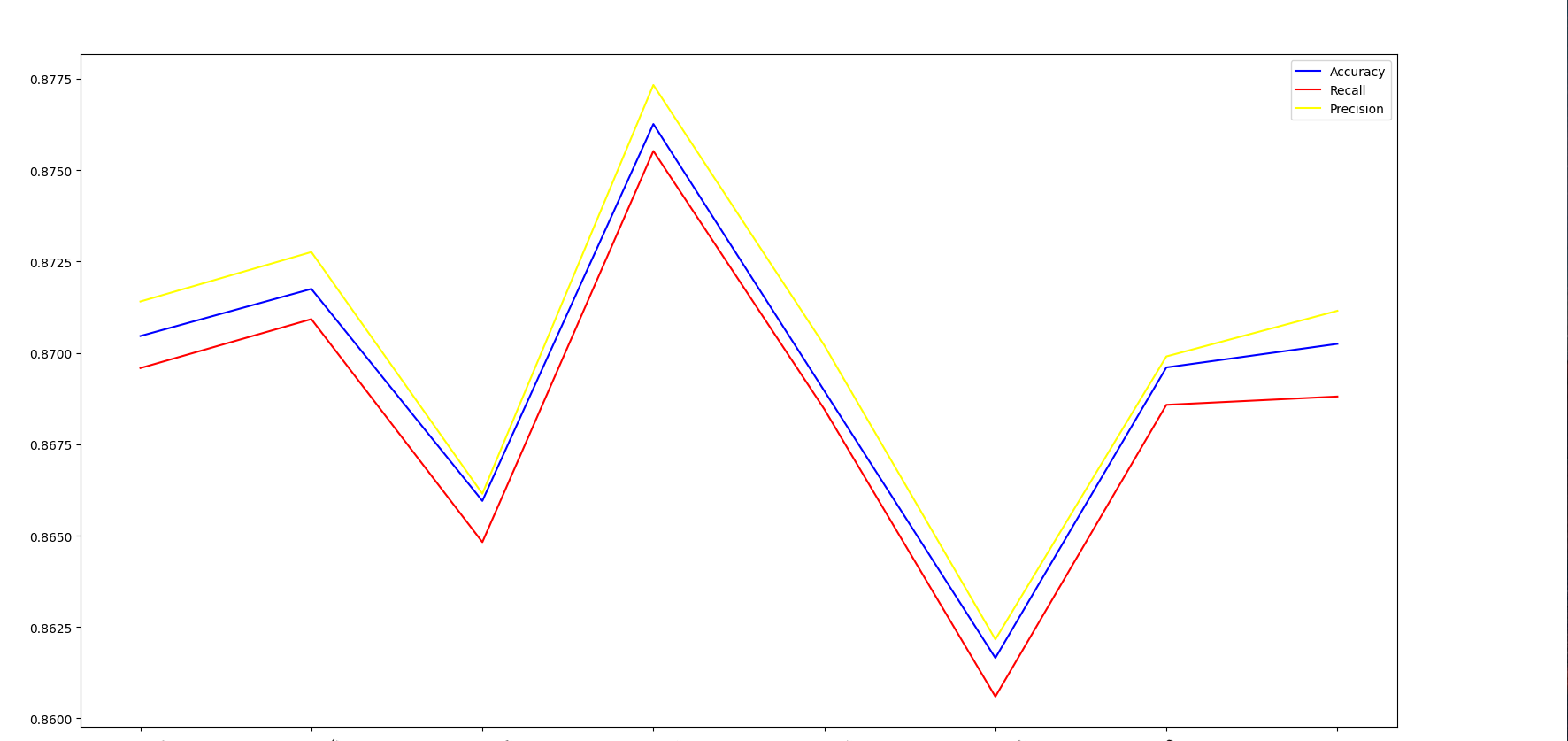
**Activation: “identity”**

|  |  |  |  |
| --- | --- | --- | --- |
| **Max iterations** | **Accuracy** | **Precision** | **Recall** |
| 50 | 0.8599 (+/- 0.0284) | 0.8601 (+/- 0.0284) | 0.8574 (+/- 0.0294) |
| 71 | 0.8587 (+/- 0.0416) | 0.8595 (+/- 0.0407) | 0.8563 (+/- 0.0418) |
| 92 | **0.8599 (+/- 0.0245)** | **0.8605 (+/- 0.0251)** | **0.8573 (+/- 0.0251)** |
| 114 | 0.8525 (+/- 0.0267) | 0.8538 (+/- 0.0235) | 0.8499 (+/- 0.0268) |
| 135 | 0.8557 (+/- 0.0355) | 0.8582 (+/- 0.0345) | 0.8538 (+/- 0.0352) |
| 157 | 0.8572 (+/- 0.0214) | 0.8583 (+/- 0.0194) | 0.8549 (+/- 0.0208) |
| 178 | 0.8565 (+/- 0.0217) | 0.8601 (+/- 0.0198) | 0.8545 (+/- 0.0205) |



**Activation: “relu”**

|  |  |  |  |
| --- | --- | --- | --- |
| **Max iterations** | **Accuracy** | **Precision** | **Recall** |
| 50 | 0.8705 (+/- 0.0122) | 0.8714 (+/- 0.0133) | 0.8696 (+/- 0.0121) |
| 71 | 0.8718 (+/- 0.0142) | 0.8728 (+/- 0.0147) | 0.8709 (+/- 0.0138) |
| 92 | 0.8660 (+/- 0.0262) | 0.8661 (+/- 0.0244) | 0.8648 (+/- 0.0270) |
| 114 | **0.8763 (+/- 0.0202)** | **0.8773 (+/- 0.0208)** | **0.8755 (+/- 0.0199)** |
| 135 | 0.8690 (+/- 0.0124) | 0.8702 (+/- 0.0143) | 0.8685 (+/- 0.0125) |
| 157 | 0.8617 (+/- 0.0172) | 0.8622 (+/- 0.0158) | 0.8606 (+/- 0.0177) |
| 178 | 0.8696 (+/- 0.0246) | 0.8699 (+/- 0.0262) | 0.8686 (+/- 0.0237) |



**Results of support MLP classification:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iterations** | **Activation Function** | **Accuracy** | **Precision** | **Recall** |
| 114 | Relu | **0.8763 (+/- 0.0202)** | **0.8773 (+/- 0.0208)** | **0.8755 (+/- 0.0199)** |
| 92 | Identity | 0.8599 (+/- 0.0245) | 0.8605 (+/- 0.0251) | 0.8573 (+/- 0.0251) |

**Model performance comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Hyper-parameters** | **Accuracy** | **Precision** | **Recall** |
| KNN | Weight-based distance,  number of neighbours = 4 | 0.9322 (+/- 0.0090) | 0.9357 (+/- 0.0076) | 0.9301 (+/- 0.0094) |
| MLP | Activation function = relu,  Max iterations = 114 | 0.8599 (+/- 0.0245) | 0.8605 (+/- 0.0251) | 0.8573 (+/- 0.0251) |
| SVC | Kernel = RBF,  Normalization parameter = 3 | **0.9566 (+/- 0.0166)** | **0.9566 (+/- 0.0163)** | **0.9566 (+/- 0.0161)** |