# Machine Learning –KernelSVM Assignment

Name: Nour Mohamed Hussein Mahmoud Mohamed Kamaly

ID: 20191700701

Department: Scientific Computing Level 3

Section: 5

I was training on google colab as some libraries couldn't be installed due to version conflicts, so if it's possible please review the notebook first.

## Feature selection:

- According to LaGrange interpolation techniques for finding a unique polynomial that passes through the observations, if the number of features (estimators/predictors) was more than the number of observations (data points), then the predictive line is not unique and the model is prone to overfitting, and this is the case here, 3780 features and 2000 observations, so some features had to be removed
- Two methods were used
    - Selecting k best features: the problem with this one is that the maximum k is 10 (which is very small number of features relatively) and when I tried different models, the accuracy was hardly over 60.5%
    - Hypothesis testing using p value:
      My null hypothesis is that the model will make use of the feature otherwise the label is independent on that feature.
      So, a p value < 0.05 proves the null hypothesis is true and a p value larger than it proves otherwise. 1754 features were selected in the end. This improved the model's performance.

## Experiments:

- 6 different models were chosen
    - Linear SVM
    - SVM with linear kernel
    - SVM with rbf kernel
    - SVM with polynomial kernel
    - Stacking Classifier that takes the above 4 models and stacks them according to best performing to worst performing, 10 folds (cross validation), and logistic regression (2 classes only) as a final estimator
    - Convolution neural network (due to the small number of training set the model resulted in overfitting)
      2 convolution layers and 2 dense layers
      Optimizer: stochastic gradient descent (learning rate = 0.01 & momentum = 0.8 – used high momentum value as the model used to be stuck at a local minimum {0.5})
      Loss function: binary cross entropy
      Batch size = 32

## Models' results:

| Model | Training accuracy | Testing accuracy |
|---|---|---|
| Linear SVM | 0.976 | 0.725 |
| Linear kernel function | 0.9185 | 0.73 |
| RBF kernel function | 0.9485 | 0.725 |
| Polynomial kernel function | 1.0 | 0.735 |
| Stacking Classifier | 0.986 | 0.75 |
| Convolutional neural network | 1.0 | 0.5818 |

Selecting K Best features:

```
[57] print(accuracy_score(svc.predict(x_test),y_test))
     print(accuracy_score(lin_svc.predict(x_test),y_test))
     print(accuracy_score(rbf_svc.predict(x_test),y_test))
     print(accuracy_score(poly_svc.predict(x_test),y_test))

     0.605
     0.6
     0.585
     0.595
```

## Stacking Classifier:

```
[36] from sklearn.ensemble import StackingClassifier
     from sklearn.linear_model import LogisticRegression

[37] supportVectorMachines = [("polyKernel",svm.SVC(kernel='poly')),("linearKernel",svm.SVC(kernel='linear')),("rbfKernel",svm.SVC(kernel='rbf')),("linear",svm.LinearSVC())]

[38] classifier = StackingClassifier(estimators=supportVectorMachines,final_estimator=LogisticRegression(),cv=10).fit(x_train,y_train)

[39] print(accuracy_score(classifier.predict(x_train),y_train))
     print(accuracy_score(classifier.predict(x_test),y_test))

     0.986
     0.75
```

## Support Vector Machines:

```
[32] svc = svm.SVC(kernel='linear').fit(x_train, y_train)
     print(accuracy_score(svc.predict(x_train),y_train))
     print(accuracy_score(svc.predict(x_test),y_test))

     0.9185
     0.73
```

```
[33] lin_svc = svm.LinearSVC().fit(x_train, y_train)
     print(accuracy_score(lin_svc.predict(x_train),y_train))
     print(accuracy_score(lin_svc.predict(x_test),y_test))

     0.976
     0.725
```

```
[34] rbf_svc = svm.SVC(kernel='rbf').fit(x_train, y_train)
     print(accuracy_score(rbf_svc.predict(x_train),y_train))
     print(accuracy_score(rbf_svc.predict(x_test),y_test))

     0.9485
     0.725
```

```
[35] poly_svc = svm.SVC(kernel='poly').fit(x_train, y_train)
     print(accuracy_score(poly_svc.predict(x_train),y_train))
     print(accuracy_score(poly_svc.predict(x_test),y_test))

     1.0
     0.735
```

## Convolution Neural Network:

```
Epoch 47/50
62/62 [==============================] - 2s 26ms/step - loss: 8.8718e-04 - accuracy: 1.0000
Epoch 48/50
62/62 [==============================] - 2s 25ms/step - loss: 7.7315e-04 - accuracy: 1.0000
Epoch 49/50
62/62 [==============================] - 2s 26ms/step - loss: 6.7748e-04 - accuracy: 1.0000
Epoch 50/50
62/62 [==============================] - 2s 26ms/step - loss: 6.0569e-04 - accuracy: 1.0000
<keras.callbacks.History at 0x7f167c1da050>
```

```
[164] model.evaluate(x_test,y_test)
```

```
7/7 [==============================] - 0s 12ms/step - loss: 2.5243 - accuracy: 0.5818
[2.5243170261383057, 0.581818163394928]
```