**KFold Cross Validation**

**Option 1: Use all available data for training and test on same dataset**

**Option 2: Split train and test data**

**Option 3: K fold cross validation**

**Option 3: (split 5 training sets) Train 1,2,3,4 Train 2,3,4,5 train 1,3,4,5 train 1,2,4,5 train 1,2,3,5**

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.svm **import** SVC

**from** sklearn.ensemble **import** RandomForestClassifier

**import** numpy **as** np

**from** sklearn.datasets **import** load\_digits

digits **=** load\_digits()

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(digits**.**data,digits**.**target,test\_size**=**0.3)

1. **COMPARA LA EXACTITUD ENTRE METODOS**

**Logistic Regression**

lr **=** LogisticRegression(solver**=**'liblinear',multi\_class**=**'ovr')

lr**.**fit(X\_train, y\_train)

lr**.**score(X\_test, y\_test) # 0.9115

**SVM**

#cross\_val\_score(svm**.**SVC(kernel**=**'linear',C**=**10,gamma**=**'auto'),iris**.**data, iris**.**target, cv**=**5)

#cross\_val\_score(svm**.**SVC(kernel**=**'rbf',C**=**10,gamma**=**'auto'),iris**.**data, iris**.**target, cv**=**5) for no linear, complex grupos

svm **=** SVC(gamma**=**'auto')

svm**.**fit(X\_train, y\_train)

svm**.**score(X\_test, y\_test) # 0.4273

**Random Forest**

rf **=** RandomForestClassifier(n\_estimators**=**40)

rf**.**fit(X\_train, y\_train)

rf**.**score(X\_test, y\_test) # 0.9181

1. **KFold, crea random 3 data sets**

**KFold cross validation**

**from** sklearn.model\_selection **import** KFold

kf **=** KFold(**n\_splits=3**)

kf

Out[6

KFold(n\_splits=3, random\_state=None, shuffle=False)

**for** train\_index, test\_index **in** kf**.**split([1,2,3,4,5,6,7,8,9]):

print(train\_index, test\_index)

[3 4 5 6 7 8] [0 1 2]

[0 1 2 6 7 8] [3 4 5]

[0 1 2 3 4 5] [6 7 8]

**Use KFold for our digits example**

1. **Evalua los 3 modelos con los 3 data set creados por KFold**

Se toma la idea del Random Forest, se crearon 3 data sets en forma random, y generaran 3 valores de exactitud por cada modelo.

Luego se puede calcular un promedio de exactitud por modelo.

**def** **get\_score**(model, X\_train, X\_test, y\_train, y\_test):

model**.**fit(X\_train, y\_train)

**return** model**.**score(X\_test, y\_test)

**from** sklearn.model\_selection **import** StratifiedKFold

folds **=** StratifiedKFold(n\_splits**=**3)

res\_logistic **=** []

res\_svm **=** []

res\_RF **=** []

**for** train\_index, test\_index **in** folds**.**split(digits**.**data,digits**.**target):

X\_train, X\_test, y\_train, y\_test **=** digits**.**data[train\_index],

digits**.**data[test\_index], \

digits**.**target[train\_index],\

digits**.**target[test\_index]

res\_logistic**.**append(**get\_score**(LogisticRegression(solver**=**'liblinear',multi\_class**=**'ovr'), X\_train, X\_test, y\_train, y\_test))

res\_svm**.**append(**get\_score**(SVC(gamma**=**'auto'), X\_train, X\_test, y\_train, y\_test))

res\_RF**.**append(**get\_score**(RandomForestClassifier(**n\_estimators=40**), X\_train, X\_test, y\_train, y\_test))

res\_logistic # [0.89534, 0.949916, 0.9093959]

res\_svm # [0.39368, 0.410684, 0.4597315]

res\_RF # [0.92857, 0.951585, 0.9295302]

El modelo de RF (Random Forest) nos dio mayores valores de exactitud, y el que peor nos dio fue SVM, de modo que procedemos a evaluar con mas detalle el RF.

**NOTA:**

1. Este approach “pudo fallar” porque dependio de el estimador de “40” si hubiera sido un estimador de 5 hubiera dado mas alto el logístico
2. También en el SVM se uso gamma=’auto’ y pudo utilizarse otro y tener una mayor exactitud.
3. **cross\_val\_score Function,**

Cross\_val\_score uses stratifield kfold by default

Se parametriza cuantos grupos a generar cv = No. Grupos

**from** sklearn.model\_selection **import** cross\_val\_score

**grupos = 3**

**Usando cross\_va\_score para Logistic, SVM, RF**

R1 =cross\_val\_score(LogisticRegression(solver**=**'liblinear',multi\_class**=**'ovr'), digits**.**data, digits**.**target,cv**=grupos**)

R2=cross\_val\_score(SVC(gamma**=**'auto'), digits**.**data, digits**.**target,cv**=grupos**)

R3=cross\_val\_score(RandomForestClassifier(n\_estimators**=40**),digits**.**data, digits**.**target,cv**=grupos**)

R1 logistic # [0.89534, 0.949916, 0.9093959]

R2 svm # array([0.393687, 0.410684, 0.459731])

R3 RF # array([0.93521595, 0.94156928, 0.93288591])

El modelo de RF (Random Forest) nos dio mayores valores de exactitud, y el que peor nos dio fue SVM, de modo que procedemos a evaluar con mas detalle el RF.

1. **Parameter tunning for RF, using k fold – cross validation**

**grupos=10 , cross validation 10 times, 10 different datasets generated.**

**from** **sklearn.model\_selection** **import** cross\_val\_score

def evaluar(**arboles**):

scores1 **=** **cross\_val\_score**(RandomForestClassifier(n\_estimators**=arboles**),digits**.**data,

digits**.**target, **cv=grupos**)

print(np.average(scores1))

evaluar(5) 0.8796

evaluar(20) 0.935

evaluar(30) 0.9494

evaluar(40) 0.9482

El mejor score esta entre crear 30 – 40 arboles

Here we used cross\_val\_score to fine tune our random forest classifier and figured that having around 40 trees in random forest gives best result.

Random Forest creates 40 trees, but also for 10 times, so it creates 400 different scenarios.

**Aquí nos ayudo a encontrar que el numero de arboles debe ser entre 30 y 40.**

rf = RandomForestClassifier(**n\_estimators=30**)

rf.fit(X,y)’

Predecimos

valor = X\_test[1]

rf.predict([valor]) #4