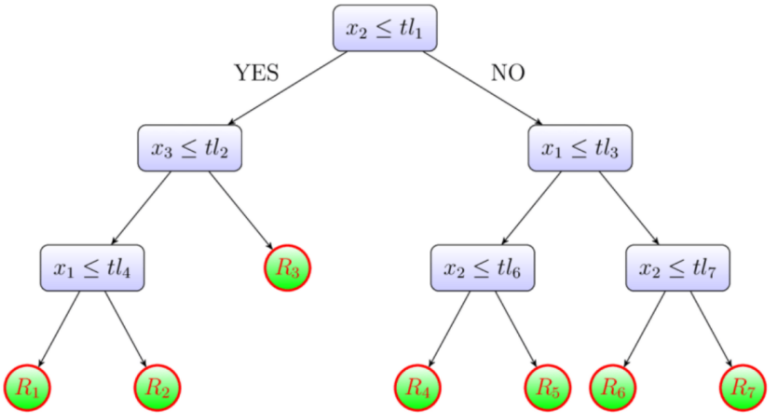
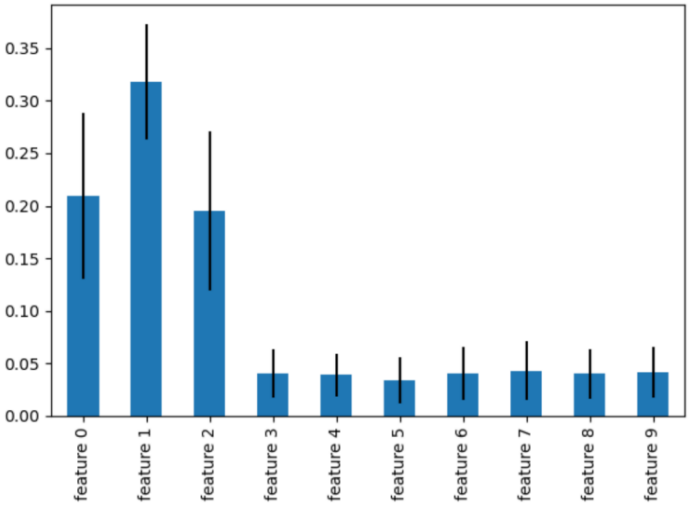
Class-09, June 3rd, Cross-validation- Grid Search- hyperparameter optimization

Hossen Teimoorinia, Uvic, 2023




```
importances = forest.feature_importances_
```



www.nature.com/scientificreports/

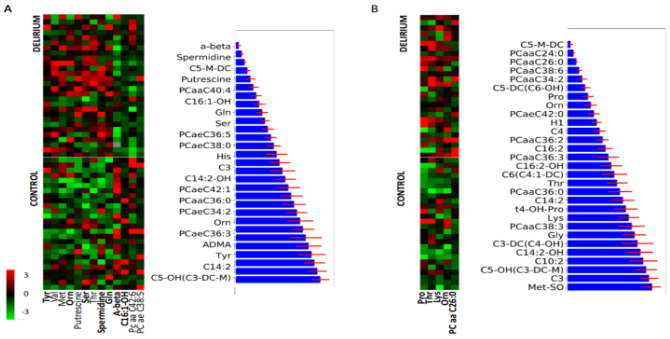


Figure 1. Most significant features distinguishing between control and delirium-prone cohort in (**A**) CSF and (**B**) Blood. SAM subset includes the most relevant metabolites based on statistical analysis (with delta value in TMeV implementation of SAM of 0.5) while the Random Forest method ranks metabolites' relevance for diagnosis between two groups. Metabolic panel was selected from the Random Forest ranking Error bars shown in red represent uncertainties measure obtained using the probabilistic bootstrap method (see Materials and Methods for more details).

Data split? Why?

$(X, Y) \xrightarrow{\text{why}} (X_{tr}, Y_{tr})$ 75% ? & (X_{va}, Y_{va}) 0.25% ?

To avoid overfitting / underfitting

$(X, Y) \longrightarrow (X_{tr}, Y_{tr})$ 70% + (X_{va}, Y_{va}) 0.15% + (X_{test}, Y_{test}) 0.15%

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

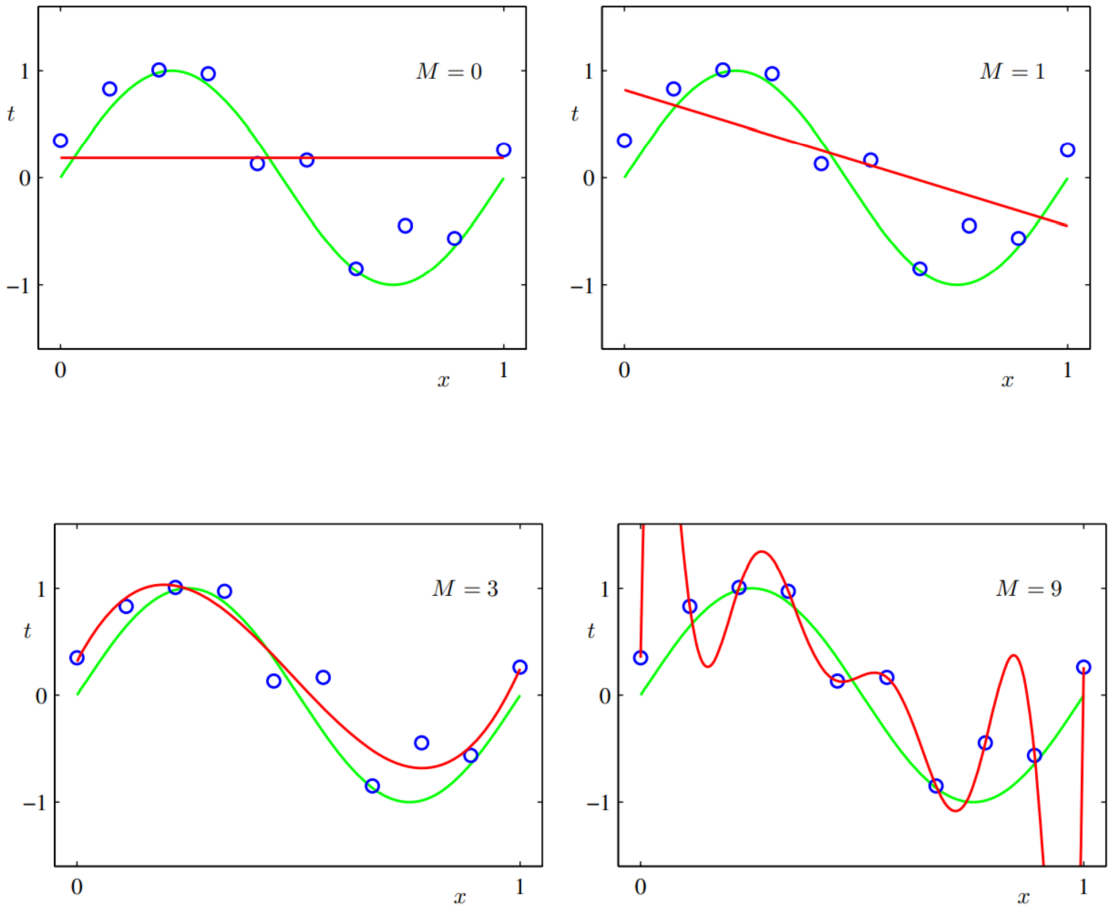
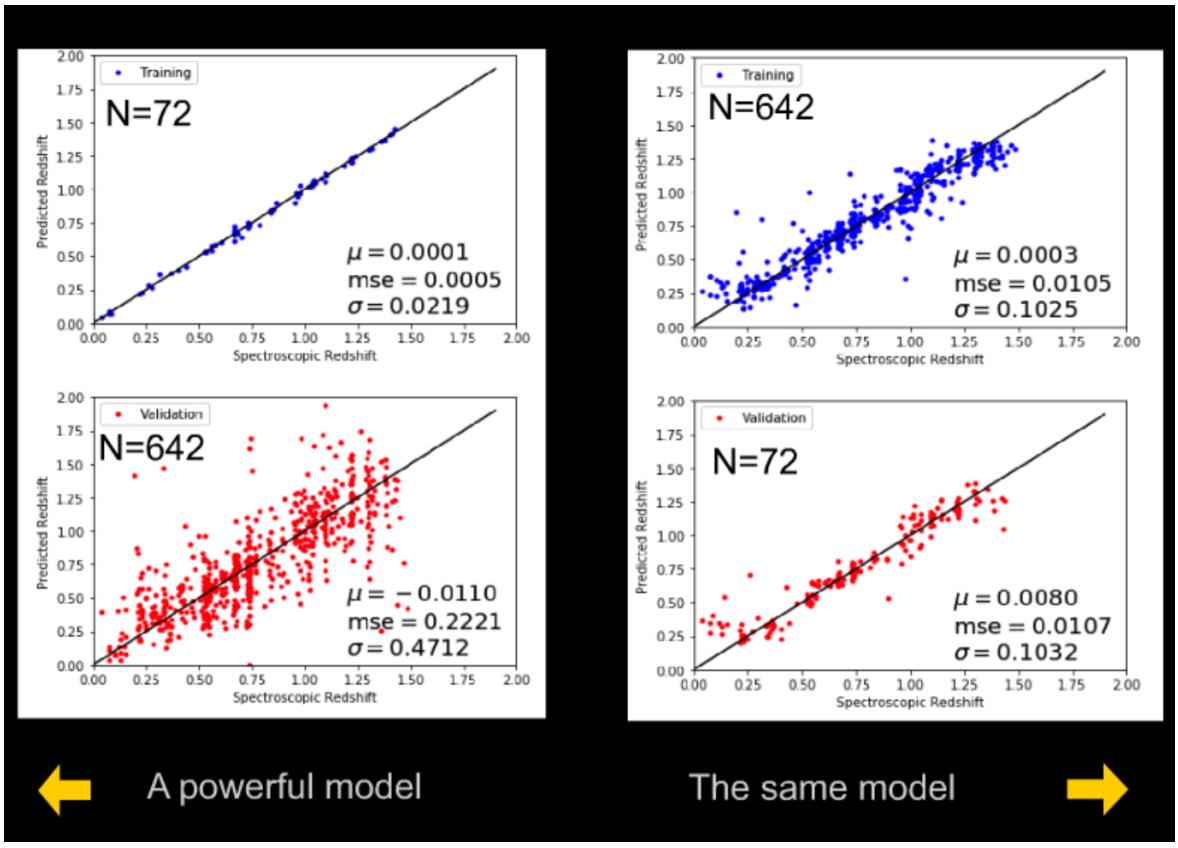
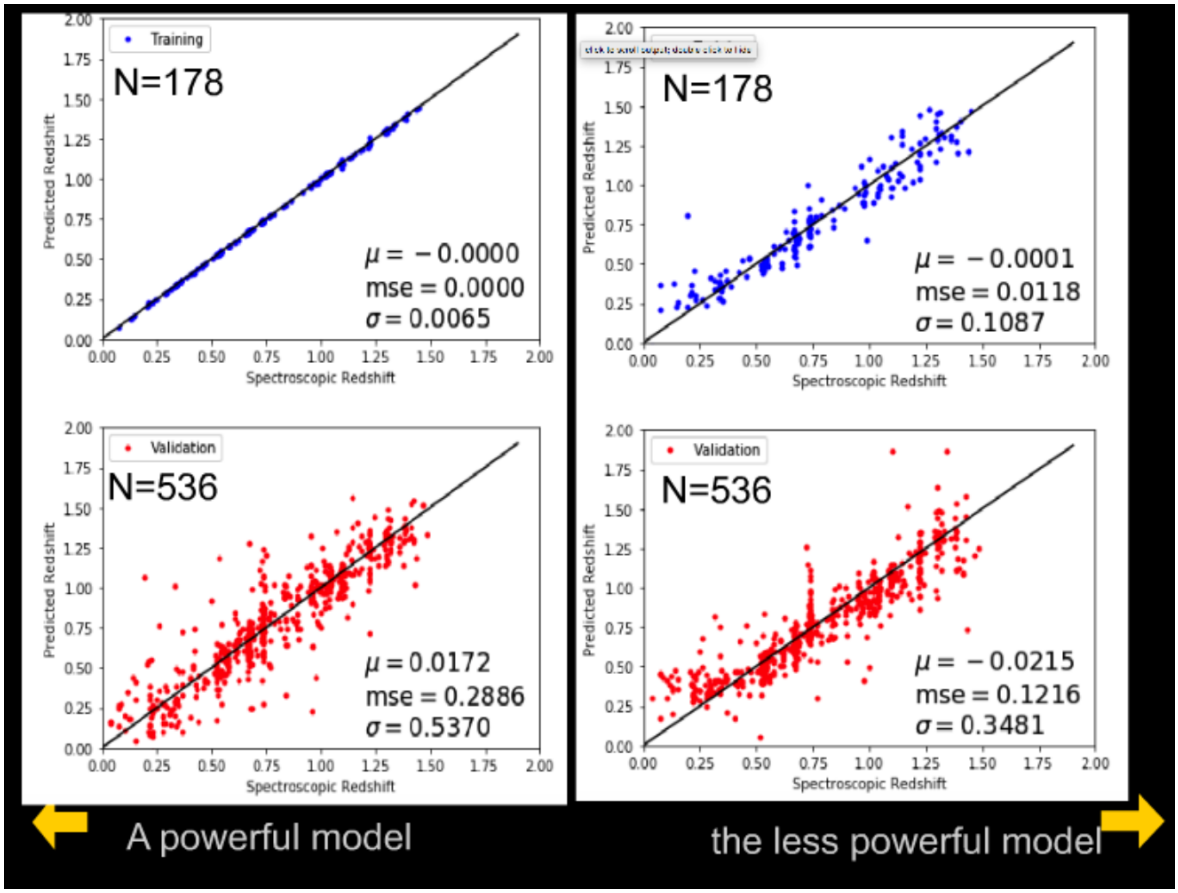
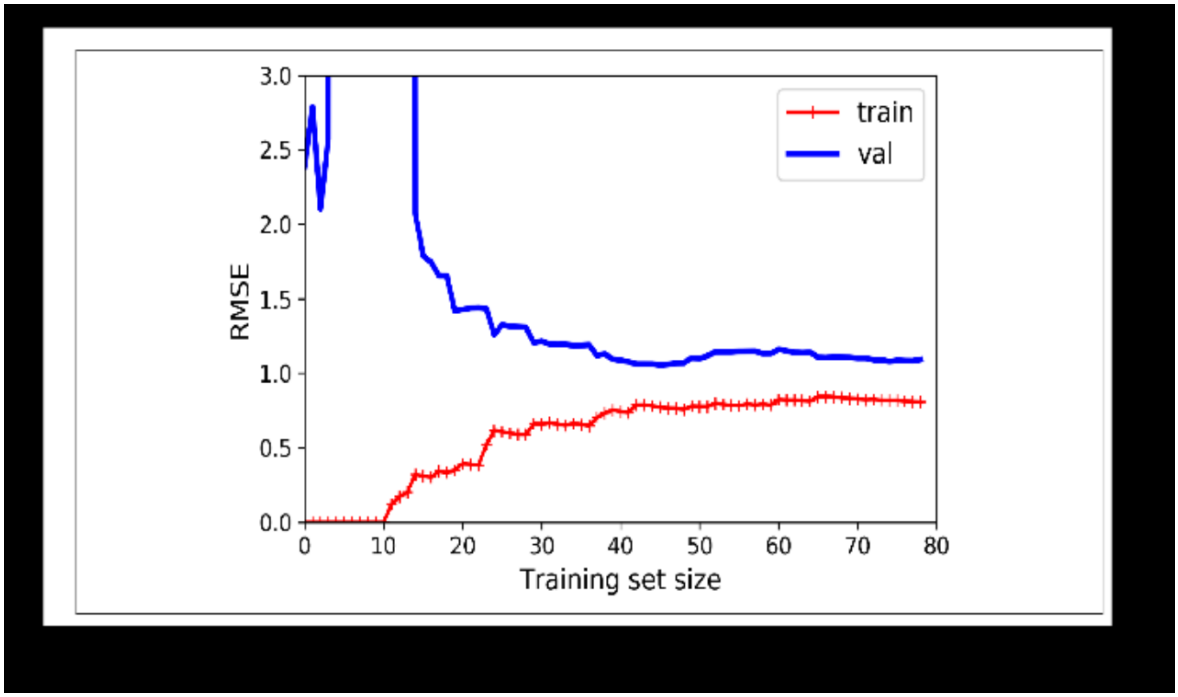
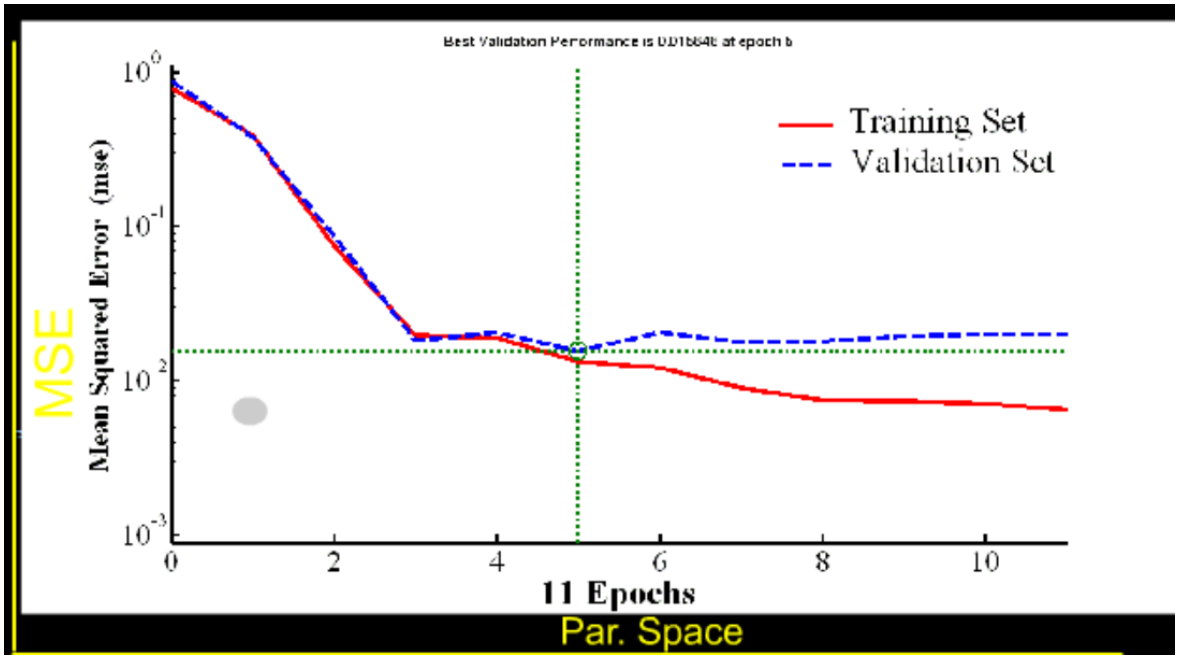


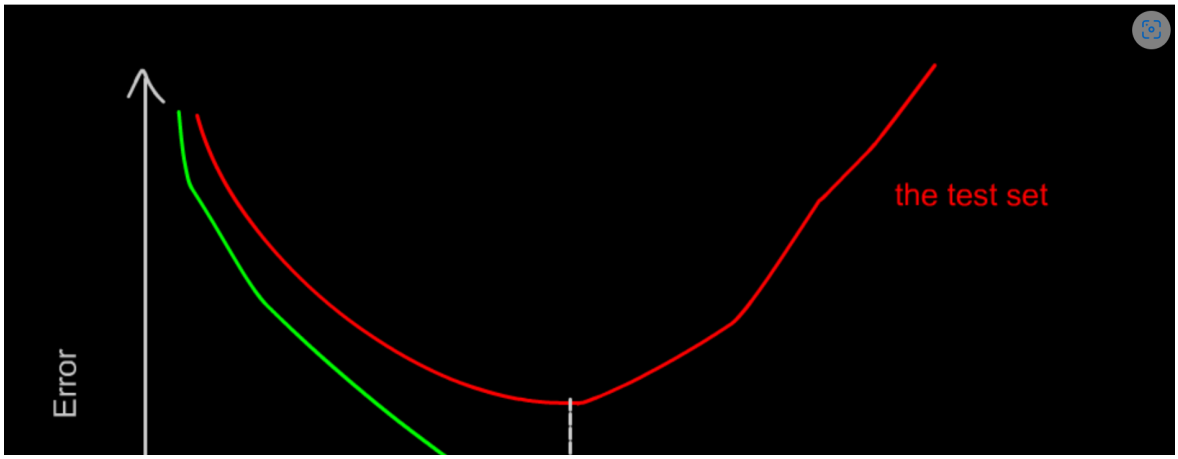
Figure 1.4 Plots of polynomials having various orders M, shown as red curves, fitted to the data set shown in Figure 1.2.




Training
N=178
$\mu = -0.0000$
mse = 0.0000
$\sigma = 0.0065$
Validation
N=536
$\mu = 0.0172$
mse = 0.2886
$\sigma = 0.5370$
N=178
$\mu = -0.0001$
mse = 0.0118
$\sigma = 0.1087$
Validation
N=536
$\mu = -0.0215$
mse = 0.1216
$\sigma = 0.3481$
Predicted Redshift
Spectroscopic Redshift
A powerful model
the less powerful model


train
val
RMSE
Training set size

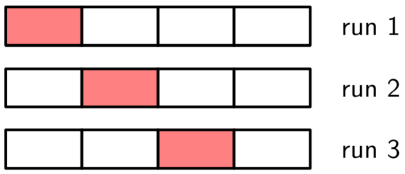




cross-validation (wiki)

Cross-validation,[2][3][4] sometimes called **rotation estimation**[5][6][7] or **out-of-sample testing**, is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of *known data* on which training is run (*training dataset*), and a dataset of *unknown data* (or *first seen* data) against which the model is tested (called the validation dataset or *testing set*).[8][9] The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias[10] and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

Figure 1.18 The technique of S-fold cross-validation, illustrated here for the case of $S = 4$, involves taking the available data and partitioning it into S groups (in the simplest case these are of equal size). Then $S - 1$ of the groups are used to train a set of models that are then evaluated on the remaining group. This procedure is then repeated



Load the general packages you might need

In [3]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
%matplotlib inline
from sklearn.decomposition import PCA
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from pandas.plotting import scatter_matrix
from sklearn.neighbors import KNeighborsRegressor
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn import linear_model
import sys
from sklearn.neighbors import KNeighborsClassifier
from scipy.interpolate import interp1d
from matplotlib import pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection, svm
print('Done!')
```

```
Done!
```

load your data

In [4]:
```
X=np.load('inp_alfalfa.npy')  # load the input
Y=np.load('tar_alfalfa.npy')  # load the target
print(np.shape(X),np.shape(Y))
print ('Done!')
```

```
(13674, 15) (13674,)
Done!
```

Split the data

In [5]:
```
from sklearn.model_selection import train_test_split

X_tr,X_va,Y_tr, Y_va = train_test_split(X,Y ,test_size=0.1 )

print ('training set == ',np.shape(X_tr),np.shape(Y_tr),',, validation set ==
```

```
training set ==  (12306, 15) (12306,) ,, validation set ==  (1368, 15) (136
8,)
```

Normalize the data

In [6]:
```

scaler_S= StandardScaler().fit(X_tr)  # Line #2
X_tr_Norm= scaler_S.transform(X_tr) # Line # 3

X_va_Norm= scaler_S.transform(X_va)  # Line #4

print('Done!')
```

```
Done!
```

Grid Search (load new packages)

In [7]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler


# set up th pipeline
knn_pipeline = Pipeline([('KNN', KNeighborsClassifier())])

# choose the parameters that you would like to optimize:

params = [{'KNN__n_neighbors': [ 9,11,13,15,30,50,100],
          'KNN__p': [1, 2],
           }]
# # use GridSearchCV and the right meteric for classification or regression
gs_knn = GridSearchCV(knn_pipeline,
                      param_grid=params,
                      scoring='accuracy',
                      cv=5)
# run pipleline and fit the model
gs_knn.fit(X_tr_Norm, Y_tr)

#The best parameters of the optimization
gs_knn.best_params_
```

Out[7]:
```
{'KNN__n_neighbors': 50, 'KNN__p': 1}
```

In [8]:
```
knn_pipeline = Pipeline([('PCA', PCA()),
                         ('KNN', KNeighborsClassifier() )])

params = [{'PCA__n_components':[.7, .9, .999], 'KNN__n_neighbors': [15,30,50,
          'KNN__p': [1, 2],
           }]

gs_knn = GridSearchCV(knn_pipeline,
                      param_grid=params,
                      scoring='accuracy',
                      cv=5)

gs_knn.fit(X_tr_Norm, Y_tr)

gs_knn.best_params_
```

Out[8]:
```
{'KNN__n_neighbors': 50, 'KNN__p': 1, 'PCA__n_components': 0.999}
```

In [9]:
```
knn_pipeline = Pipeline([('PCA', PCA()),
                         ('KNN', KNeighborsClassifier() )])

params = [{'PCA__n_components':[.7, .9, .999], 'KNN__n_neighbors': [15,30,50,
          'KNN__p': [1, 2],
           }]

gs_knn = GridSearchCV(knn_pipeline,
                      param_grid=params,
                      scoring='accuracy',
                      cv=5)

gs_knn.fit(X_tr_Norm, Y_tr)

gs_knn.best_params_
```

Out[9]:
```
{'KNN__n_neighbors': 50, 'KNN__p': 1, 'PCA__n_components': 0.999}
```

Regression with ANN

In [10]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
```

In [11]:

```
# reg = MLPRegressor(hidden_layer_sizes=(100,50,), max_iter=50000,
#                         activation='relu',solver='adam',tol=.00000001,alpha=
#                         validation_fraction=0.1, learning_rate= 'constant',l

ANN_cls_pipeline = Pipeline([('ANNcls', MLPClassifier())])

params = [{'ANNcls__solver':['adam','sgd','lbfgs']}]

gs_ann_cls = GridSearchCV(ANN_cls_pipeline,
                          param_grid=params,
                          scoring='accuracy',
                          cv=5)

gs_ann_cls.fit(X_tr_Norm, Y_tr)

gs_ann_cls.best_params_
```

```
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:559: ConvergenceWarning: lbfgs failed to converge (s
tatus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:559: ConvergenceWarning: lbfgs failed to converge (s
tatus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:559: ConvergenceWarning: lbfgs failed to converge (s
tatus=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:559: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:559: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[11]:
```
{'ANNcls__solver': 'adam'}
```

In [12]:
```
X=np.load('inp_redshift.npy')  # Load the input
Y=np.load('tar_redshift.npy')  # Load the target
print(np.shape(X),np.shape(Y))
print ('Done!')
```

```
(713, 7) (713,)
Done!
```

In [13]:
```
#clf = MLPClassifier(solver='adam', alpha=1e-5,
#                     hidden_layer_sizes=(100,50), max_iter=1,
#                    early_stopping=False, validation_fraction=0.1 )


ANN_reg_pipeline = Pipeline([('STD', StandardScaler()),('ANNreg', MLPRegresso

params = [{'ANNreg__alpha':[1, .1, .01, .001,.0001],'ANNreg__solver': ['adam'

gs_ann_reg = GridSearchCV(ANN_reg_pipeline,
                      param_grid=params,
                      scoring='neg_root_mean_squared_error',
                      cv=5)
#,'ANNreg__solver': 'lbfgs', 'ANNreg__hidden_layer_sizes':(20,20)
gs_ann_reg.fit(X_tr_Norm, Y_tr)

gs_ann_reg.best_params_
```

Out[13]:
```
{'ANNreg__alpha': 0.1,
 'ANNreg__hidden_layer_sizes': (20, 20),
 'ANNreg__solver': 'adam'}
```

In [ ]:

Use ANN regression and find the best alpha and solver

Do the same with ANN classification

Do classification with KNN and finf the best K and P

Do the same with KNN regession

In [ ]: