CS 6316 – *Machine Learning*
Homework 2: Predictive Learning – ***Sample Solutions***

Summary of problems:

		Points
1.	Problem 2.2	**10**
2.	Problem 2.7	**25**
3.	Decision Tree problem	**15**
4.	Problem 2.11	**25**
5.	Problem 2.12	**25**

NOTE TO STUDENTS:

Please note that the following document provides sample solutions. They do not represent the only solution. It is simply an example to give you an idea of what kind of solution and output was expected. It is likely your solutions are correct yet differ from the ones presented, here. Some solutions might not be complete but are sufficient to provide the gist of what was required.

Problem 2.2 (**10**) ~ parts (b) through (f) [part (a) was given as an ex] (2 pts each)
No single 'correct solution' for this problem. Answers will be graded individually. However, example solutions are provided below:

(a) Inputs: person's age, gender and weight-to-height ratio.
Output: remaining life expectancy

Comment: It is indisputable that person's age is one of the most important factors that influence the remaining life expectancy. And generally, women live longer than men. Weight-to-height ratio reflects the health in some ways. Therefore, the dependency can be estimated from data and this estimated model has useful practical value. Therefore, it is appropriate to use predictive learning to estimate a data-analytic model in this case.

(b) Inputs: annual family income, number of years a couple stayed married and number of children.
Output: marriage status in 5 years (Married or Divorced).

Comment: Some research shows that marriage status has correlation with annual family income. Being lower or higher than certain annual family income would result in high possibility of divorce. Besides, number of years a couple stayed married and number of children are also essential factors. Thus, we can reliably estimate these dependency from data, and it is appropriate to use predictive learning to estimate a data-analytic model in this case.

(c) Inputs: person's cell phone number and person's gender.
Output: remaining life expectancy.

Comment: Remaining life expectancy does not depend on person's cell phone number! Even though gender may influence a person's lifespan, it is not the key feature. Thus, the dependency cannot be estimated from data. Therefore this estimated model also has no useful practical value. So it is not appropriate to use predictive learning to estimate a data-analytic model in this case.

(d) Inputs: today's closing price of a stock index.
Output: next (trading) day closing price of this stock index.

Comment: Next day closing price has a correlation with today's closing price of a stock index. Next day closing price may increase or decrease based on today's closing price, which means we can assume that next day closing price has a certain range. However, the model we build from these observations does not have useful practical value because we cannot estimate that the price will increase or decrease, which is most important. So it is not appropriate to use predictive learning to estimate a data-analytic model in this case.

(e) Inputs: time series of stock index price recorded over past 5 days.
Output: the next day price change (Up or Down) of this stock index.

Comment: The price of a certain stock index may have certain trends for consecutive days. However, it may also experience sudden changes due to some events associated with the company, like launching new products or certain affairs. Therefore, it is likely to estimate the price change according to the recorded price over past 5 days. However, it will not be able to predict some unanticipated changes. Since the model can give some predictions on the price change, it has useful practical value. Therefore, it is appropriate to use predictive learning to estimate a data-analytic model in this case.

(f) Inputs: patient's age, brain scan information and results of psychological tests.
Output: presence or absence of Alzheimer's disease diagnosis in the next 5 years.

Comment: Whether a patient will suffer from Alzheimer's disease in the next 5 years or not depends on the patient's age and current health status. There is a higher chance to have Alzheimer's disease when the patient becomes older. The brain scan information and results of psychological tests will show the health status physically and mentally. So the dependency can be estimated from data and the estimated model has useful practical value. Therefore, it is appropriate to use predictive learning to estimate a data-analytic model in this case.

Problem 2.7 (**25**)

The 2000 and 2004 election data are taken from
https://en.wikipedia.org/wiki/United_States_presidential_election,_2000 ~ test data (y-values)
https://en.wikipedia.org/wiki/United_States_presidential_election,_2004 ~ training data

The optimal choice for k via leave-one-out cross-validation (using training data) is *k=17 and 19*, and for 17/19-NN classifier LOO error rate is 15/51=29.412%.

The k-NN classifier (with k=17/19) is tested using 2000 elections data. The result of this prediction is shown in the Table below, where R denotes Republican and D denotes Democrat.

Comparing these predictions with the true outcome for 2000, there are 15 mis-predicted states (marked in red). So the test error rate is 29.412%.

Discussion: Note that the LOO resampling error (estimated on the training data) is the same as test error. This may be due to:
(a) using the same value of obesity index for both training and test data set. This modeling assumption can be questioned. It may be more reasonable to use actual accurate obesity index measured for 2004, if such information can be found. Yet, common sense suggests that states' obesity rates remain the same or change only slightly over a 4-year period.
(b) using a single predictor (obesity index). Clearly, the model can be possibly improved by using additional input variables for each state. For example, we may add variables reflecting average educational level, such as % of high school graduates and/or college graduates in each state.

Arguably the best prediction rule is to use each state's voting outcome during training period (Republican or Democrat) as optimal prediction during test period. This will result in just 4 mispredictions for this problem, e.g. test error 4/51.

Table 1. The results of predicting elections.

State	Obesity	2004	Prediction	[Ground Truth] 2000
Alabama	**0.301**	**R**	**R**	**R**
Alaska	**0.273**	**R**	**R**	**R**
Arizona	**0.233**	**R**	**D**	**R**
Arkansas	**0.281**	**R**	**R**	**R**
California	**0.231**	**D**	**D**	**D**

Colorado	**0.21**	**R**	**D**	**R**
Connecticut	**0.208**	**D**	**D**	**D**
Delaware	**0.221**	**D**	**D**	**D**
DC	**0.259**	**D**	**R**	**D**
Florida	**0.233**	**R**	**D**	**R**
Georgia	**0.275**	**R**	**R**	**R**
Hawaii	**0.207**	**R**	**D**	**D**
Idaho	**0.246**	**R**	**D**	**R**
Illinois	**0.253**	**D**	**D**	**D**
Indiana	**0.275**	**R**	**R**	**R**
Iowa	**0.263**	**R**	**R**	**D**
Kansas	**0.258**	**R**	**R**	**R**
Kentucky	**0.284**	**R**	**R**	**R**
Louisiana	**0.295**	**R**	**R**	**R**
Maine	**0.237**	**D**	**D**	**D**
Maryland	**0.252**	**D**	**D**	**D**
Massachusetts	**0.209**	**D**	**D**	**D**
Michigan	**0.277**	**D**	**R**	**D**
Minnesota	**0.248**	**D**	**D**	**D**
Mississippi	**0.344**	**R**	**R**	**R**
Missouri	**0.274**	**R**	**R**	**R**
Montana	**0.217**	**R**	**D**	**R**
Nebraska	**0.265**	**R**	**R**	**R**
Nevada	**0.236**	**R**	**D**	**R**
New Hampshire	**0.236**	**D**	**D**	**R**
New Jersey	**0.229**	**D**	**D**	**D**

New Mexico	**0.233**	**R**	**D**	**D**
New York	**0.235**	**D**	**D**	**D**
North Carolina	**0.271**	**R**	**R**	**R**
North Dakota	**0.259**	**R**	**R**	**R**
Ohio	**0.269**	**R**	**R**	**R**
Oklahoma	**0.281**	**R**	**R**	**R**
Oregon	**0.25**	**D**	**D**	**D**
Pennsylvania	**0.257**	**D**	**R**	**D**
Rhode Island	**0.214**	**D**	**D**	**D**
South Carolina	**0.292**	**R**	**R**	**R**
South Dakota	**0.261**	**R**	**R**	**R**
Tennessee	**0.29**	**R**	**R**	**R**
Texas	**0.272**	**R**	**R**	**R**
Utah	**0.218**	**R**	**D**	**R**
Vermont	**0.211**	**D**	**D**	**D**
Virginia	**0.252**	**R**	**D**	**R**
Washington	**0.245**	**D**	**D**	**D**
West Virginia	**0.306**	**R**	**R**	**R**
Wisconsin	**0.255**	**D**	**R**	**D**
Wyoming	**0.24**	**R**	**D**	**R**

=====

Sample Code:

The data used in the code.

https://www.dropbox.com/s/fqtxg0jsi77qedh/raw_data.csv?dl=0

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score, LeaveOneOut
import matplotlib.pyplot as plt
from tabulate import tabulate
```

```
filename = 'raw_data.csv'
with open(filename, 'rb') as csvfile:
   x = np.array([[]])
   y_train = []
   y_test = []
   states = []
   csvlines = np.genfromtxt(csvfile, delimiter='\r', dtype=str)
   firstline = True
   for row in csvlines:
       if firstline:  # skip first line
           firstline = False
           continue
       value = row.split(',')
       states.append(value[0])
       x = np.append(x, np.array([[float(value[1])]]), axis=0) if x.size else
np.array([[float(value[1])]])
       if(float(value[2]) > float(value[3])):
           y_train.append('R')
       else:
           y_train.append('D')
       if(float(value[4]) > float(value[5])):
           y_test.append('R')
       else:
           y_test.append('D')

neighbors = [i for i in range(1,50,2)]
optimal_k = 0
best_acc = 0
MSE = []
Acc = []
for k in neighbors:
   knn = KNeighborsClassifier(n_neighbors=k)
   result = cross_val_score(knn, x, y_train, cv=LeaveOneOut(), scoring='accuracy')
   acc = result.mean()
   if acc > best_acc:
       optimal_k = k
       best_acc = acc
   Acc.append(acc)
   MSE.append(1-acc)

#print the results and plot
print("K    Accracy     MSE")
for i in range(len(neighbors)):
   print("{}   {}   {}".format(neighbors[i], round(Acc[i], 5), round(MSE[i], 5)))

print("The optimal K is %d" % optimal_k)

plt.plot(neighbors, MSE)
plt.xlabel('K')
plt.ylabel('MSE')
plt.show()
```

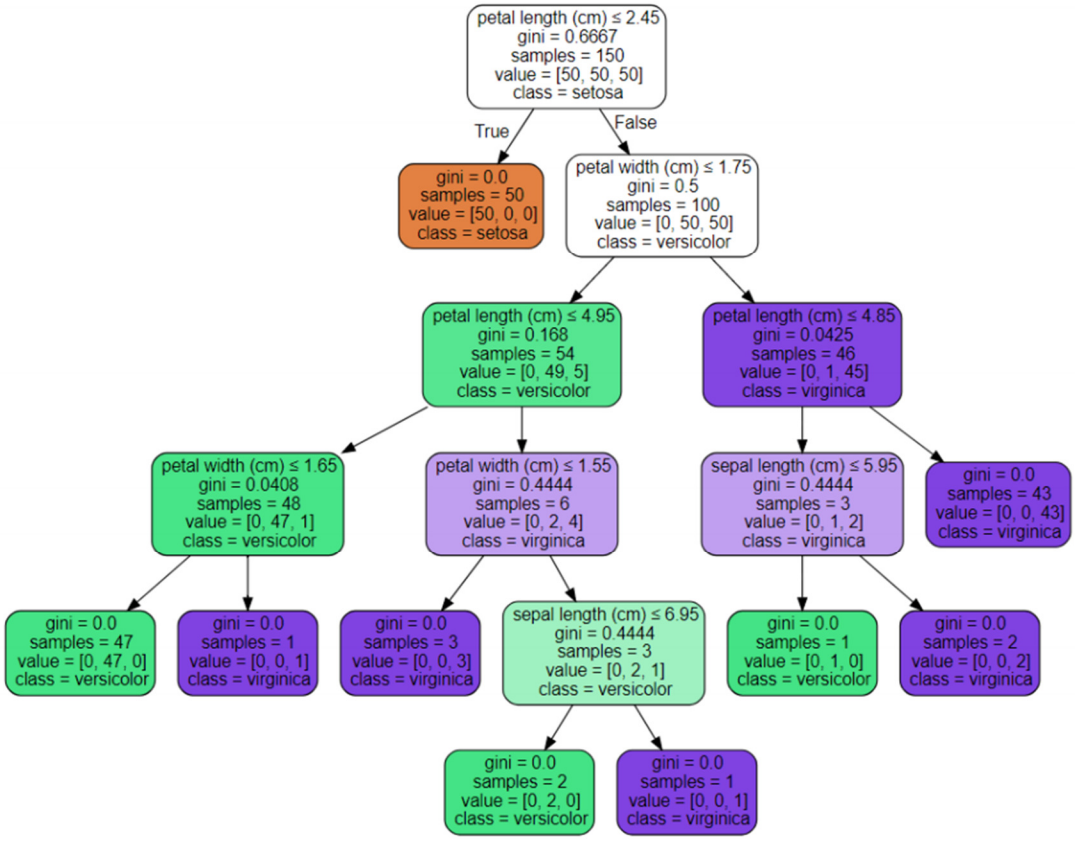
```
#train model
knn = KNeighborsClassifier(n_neighbors=optimal_k).fit(x,y_train)

#test model
predictions = knn.predict(x)

#print table
make_format = []
for i in range(len(predictions)):
   make_format.append([states[i], x[i][0], y_test[i], predictions[i], y_train[i]])

print tabulate(make_format, headers=['State', 'Obesity', '2004', 'Prediction', '[Ground
Truth]2000'])

count = 0
for i in range(len(predictions)):
   if predictions[i] == y_test[i]:
       count += 1

print("The accuracy is %f" % (count*1.0/len(predictions)))
```

Decision Tree problem (**15**)

(a)

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import seaborn as sns; sns.set(style="ticks", color_codes=True)
import pydotplus
from sklearn.externals.six import StringIO

names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
df = pd.read_csv('iris.data.txt', header=None, names=names)
df.head()

features = list(df.columns[:4])

X = df[features]
y = df["class"]
dt = DecisionTreeClassifier()
dt.fit(X, y)
dot_data = StringIO()
export_graphviz(dt,out_file=dot_data,
feature_names=names[:4],class_names=names[4],filled=True, rounded=True,
special_characters=True)

tree=pydotplus.graph_from_dot_data(dot_data.getvalue())
tree.write_png("treeAnswer.png")
```

Comment: The code can be different from the usage of python library.



(b) As seen in the above picture, the first decision node is "petal length"

(c) Since IG for iris depends on students' choice, we will show the calculation of Lenses data here for age split.

After the conversion of the data, the dataset looks like:

1 1 1 1 N
1 1 1 2 Y
1 1 2 1 N
1 1 2 2 Y
1 2 1 1 N
1 2 1 2 Y
1 2 2 1 N
1 2 2 2 Y
2 1 1 1 N
2 1 1 2 Y

2 1 2 1 N
2 1 2 2 Y
2 2 1 1 N
2 2 1 2 Y
2 2 2 1 N
2 2 2 2 N
3 1 1 1 N
3 1 1 2 N
3 1 2 1 N
3 1 2 2 Y
3 2 1 1 N
3 2 1 2 Y
3 2 2 1 N
3 2 2 2 N

Count of Y is 9
Count of N is 15

Split by age:
Age = 1, Y = 4, N = 4, sum = 8
Age = 2, Y = 3, N = 5, sum = 8
Age = 3, Y = 2, N = 6, sum = 8

IG(age,Label) = H(Label) – H(Label | age)

H(Label) = $-(\frac{9}{24}\log_2\frac{9}{24}) - (\frac{15}{24}\log_2\frac{15}{24})$ = 0.9544

H(Label | age)
= P(age = 1) H(Label | age = 1)
+ P(age = 2) H(Label | age = 2)
+ P(age = 2) H(Label | age = 2)
= $\frac{8}{24}(-(\frac{4}{8}\log_2\frac{4}{8}) - (\frac{4}{8}\log_2\frac{4}{8}))$
+ $\frac{8}{24}(-(\frac{3}{8}\log_2\frac{3}{8}) - (\frac{5}{8}\log_2\frac{5}{8}))$
+ $\frac{8}{24}(-(\frac{2}{8}\log_2\frac{2}{8}) - (\frac{6}{8}\log_2\frac{6}{8}))$
= 0.9219

IG(age, Label) = 0.9544-0.9219 = 0.0325

(d) Comment: Answer can be different based on the answer of (c)

(e) Comment: Student can choose from different perspective. The IG for continuous attributes can be calculated with differential entropy.

Problem 2.11 (**25**)

This problem illustrates the use of analytic methods for model complexity control. The available data consists of n = 10 samples, (x, y), where x is uniformly distributed in [0,1] and y= x^2+0.1*x+ ξ, where ξ is noise. The noise has Gaussian distribution N(0, 0.25) [N(mean, variance)]. Note that the noise has variance 0.25 or st. dev. 0.5.

The modeling results using algebraic polynomial regression are shown in **Table 2**. Although Rpen (estimated prediction risk) has the minimum value when **m=8** (marked in blue in **Table 2**), apparently we are *overfitting* the training data. Therefore, instead of selecting the model according to the global minima of the prediction risks, we should choose a *simpler model* (smaller m) which yields local minimum prediction risk. In this problem, we select **m=2** (marked in red in **Table 2**). See **Figure 1** showing the empirical and (estimated) prediction risk as a function of model complexity. **Figure 2** shows the training data, estimated model and the true function.

Discussion: Note that when m=8 the fitting error (empirical risk) is very close to zero, as the model with 9 parameters is used to fit/explain perfectly 10 data points. This may suggest that Schwartz criterion does not provide sufficiently high penalization for large values of DoF (m).

Table 2. Model selection for the *algebraic polynomial regression*. The optimal model is m = 2 with complexity (degree of freedom) 3.

m	DOF (m+1)	Remp	Penalized factor (r)	Rpen = r*Remp
0	1	0.1779	1.2558	0.2234
1	2	0.0329	1.5756	0.0518
2	3	0.0224	1.9868	0.0446
3	4	0.0214	2.5351	0.0543
4	5	0.0209	3.3026	0.0689
5	6	0.0154	4.4539	0.0684
6	7	0.0154	6.3727	0.0979
7	8	0.0076	10.2103	0.0775
8	9	0.0006	21.7233	0.0129
9	10	0.0000	Inf	Inf

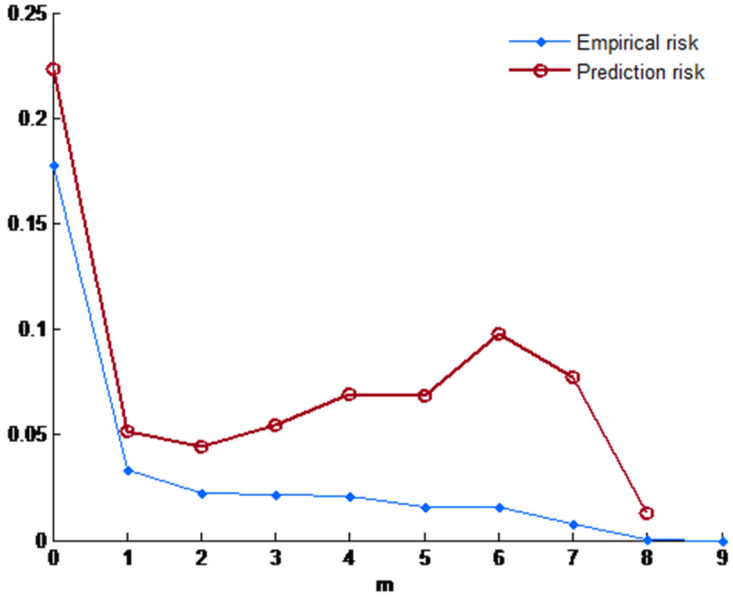


Figure 1. Empirical and (estimated) prediction risk as a function of m (DoF-1).

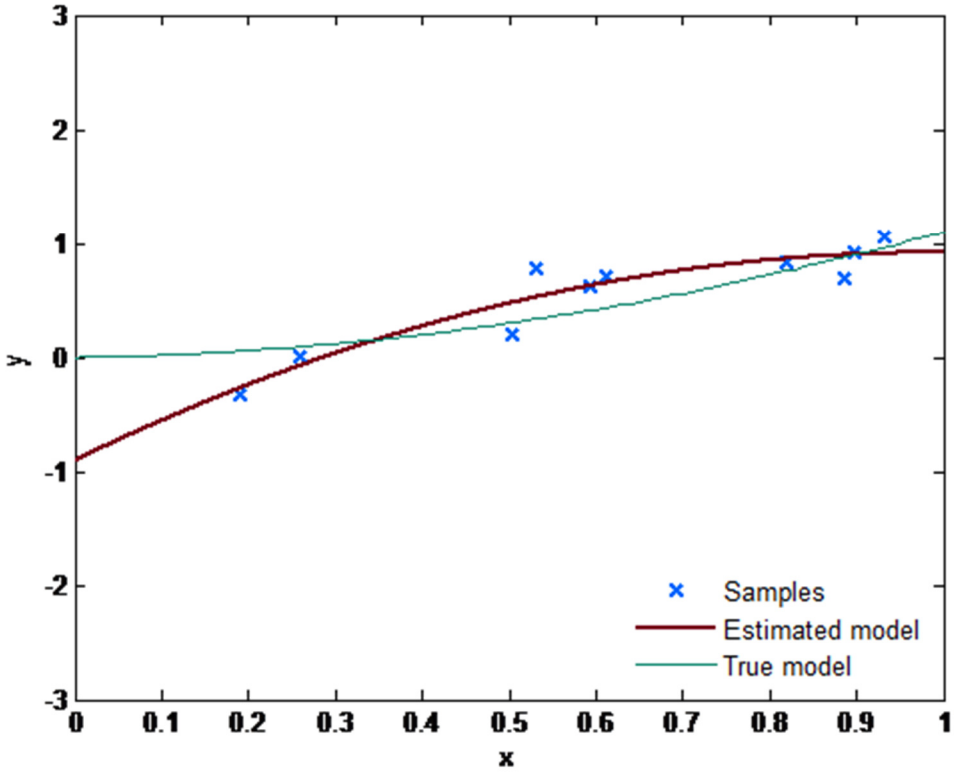


Figure 2. Algebraic polynomial model (m = 2) estimated from 10 training samples. True target function is shown in green.

=====

```
import random as rd
import numpy as np
import matplotlib.pyplot as plt
import math
import pdb

# Generate 10 X values in uniform distribution (0,1)
x = []
for i in range (0, 10):
   r = rd.uniform(0,1)
   x.append(r)
print("X:")
print(x)

# Generate 10 noise values in normal distribution (0,0.5)
noise = []
for j in range (0, 10):
   n = rd.normalvariate(0,0.5)
   noise.append(n)
print('-----------------------------------------------------------')
print("Noise:")
print(noise)

# Genearte true y using the formula given
y = []
for i in range(0, 10):
    y_i = x[i]*x[i] + 0.1*x[i] + noise[i]
    y.append(y_i)



# Calculate MSE for each polynomial degree
MSE = []

for i in range (0, 9):
       polyfit = np.polyfit(x, y, i)
       polyval = np.polyval(polyfit, x)
       MSE.append(np.sum((polyval - y)**2)/10)
print('-----------------------------------------------------------')
print("MSE: ")
print(MSE)

# Penalized Factors and Rpen
def penalized_factor(DoF, n):
    p = DoF/n
    return ( 1 + ( p / ( 1 - p ) ) * math.log( n ) )

def Rpen(p_Factor, rEmp):
    return p_Factor * rEmp
```

```
penalized_factors = []
rPens = []

for m in range(0, 9):
    penalized_factors.append(penalized_factor(m + 1, 10))
    rPens.append(Rpen(penalized_factor(m + 1, 10), MSE[m]))

print('-----------------------------------------------------------')
print("penalized_factors: ")
print(penalized_factors)
print('-----------------------------------------------------------')
print("rPens: ")
print(rPens)

# Make plots
m = range(9)
plt.plot(m, MSE, label='Empirical Risk')
plt.plot(m, rPens, label='Prediction Risk')
plt.legend()
plt.show()

fit = np.polyfit(x, y, 2)
fit_true = [1,0.1,0]
x_new = np.linspace(min(x), max(x), num=len(x)*10)

plt.plot(x, y,'g^', label='Samples')
plt.plot(x_new, np.polyval(fit, x_new), label='Estimated model')
plt.plot(x_new, np.polyval(fit_true, x_new), label='true model')
plt.legend()
plt.show()
```

Problem 2.12 (**25**)

The prediction errors (Normalized Root Mean Squared Error, NRMS) from the algebraic polynomial model is summarized in Table 3 below.

Discussion: Results in Table 3 indicate

(a) High variability of optimal model complexity selected (m) in different folds (due to variability of small training sample used to estimate a model in each fold). This variability also results in high variability of the test error (NRMS).

(b) Poorly chosen model complexity (large m-values) has quite high impact on the quality of algebraic polynomial models. The algebraic polynomials (of high degree) may exhibit very large values near the boundary of the input (x) domain.

Table 3. Prediction errors (NRMS) of algebraic polynomial model in different folds.

Fold	***Algebraic***	
	Optimal m	NRMS
1	1	0.3532
2	5	10.0803
3	3	1.0646
4	5	3.4262
5	2	0.5682
Average		3.0985

=====

Sample Code

```
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
#Load Data
X = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
Y = np.array(
   [0.40173564, -0.12671902, 0.55487409, 0.4708342, 0.575, 0.34285828, 0.33246747, 1.00781095,
0.91699119, 0.79823413])

#  KFold
kf = KFold(n_splits=5)
fold_num = 1
# Split Data
for train_index, test_index in kf.split(X):
   X_train, Y_train = X[train_index], Y[train_index]
```

```
   X_test, Y_test = X[test_index], Y[test_index]
   mse = 1.00
   optimal = -1
#Apply different degree
   for d in range(9):
       z = np.polyfit(X_train,Y_train,d)
       p = np.poly1d(z)
       Y_pred = p(X_test)
       if mse > mean_squared_error(Y_test, Y_pred):
           mse = mean_squared_error(Y_test, Y_pred)
           optimal = d
   print('Fold:', fold_num, 'Optimal:', optimal, mse)
   fold_num += 1
```

Output:

Fold	Optimal	MSE
1	1	0.0020966677582472484
2	0	0.004788505975436997
3	5	0.039555846951988646
4	6	0.039555846951988646
5	1	0.012099401443848407