DATA7202 Assignment 2

Longpeng Xu

April 20, 2023

Contents

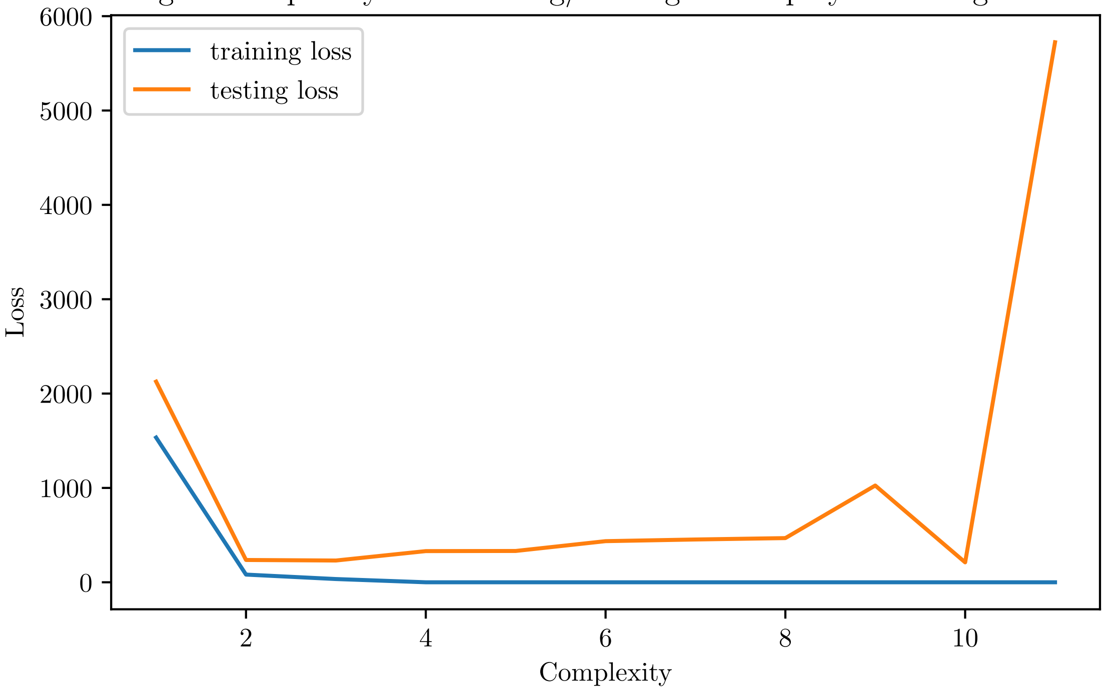
1 Question 1

- Zero training loss can be achieved by a model which over-fits (the training data).
- An example is to generate $\tau = \{(x_i, y_i), i = 1, ..., 6\}$ (x_i's are unique) and varying the complexity (the highest degree k) of polynomial regression. When $k \geqslant 5$, the training loss is constant at zero (See Fig 1 and Code in Appendix).
- However, the model over-fits which is evidenced by the testing loss: When $k = 12$, the testing loss surges to a MSE of 5723.6 (See Fig 1). This is because when the training loss is zero i.e., when $k \geqslant 5$ and keeps increasing, the model becomes increasingly complex and fits to the training data very closely - It 'memorizes' the training data and disables itself to generalize to any test set.
- Hence, in general, for any training set $\tau = \{(\boldsymbol{x_i}, y_i), i = 1, ..., n\}$ with unique $\boldsymbol{x_i}$ values, one can propose a model $f : \boldsymbol{x_i} \rightarrow y_i$ such that $f(\boldsymbol{x_i}) = y_i$ holds for every unique $\boldsymbol{x_i}$. The training loss, of any types,

becomes zero. An example of training loss in the type of MSE:

$$\text{training loss} = \frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x_i}) - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i)^2 = 0.$$

Fig 1: Complexity v.s. Training/Testing loss of polynomial regression



2 Question 2

The squared-error loss is in the form of $\sum_{i=1}^{k}(y_i - u)^2$, set $u = h(\boldsymbol{x})$ be some unknown prediction function. To find the u that minimizes the squared error loss, set the derivative w.r.t. u to be zero.

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}u}\sum_{i=1}^{k}(y_i - u)^2 &= \sum_{i=1}^{k}\frac{\mathrm{d}}{\mathrm{d}u}(y_i - u)^2 \\
&= \sum_{i=1}^{k}2(y_i - u)\frac{\mathrm{d}}{\mathrm{d}u}(y_i - u) \\
&= \sum_{i=1}^{k}2(y_i - u)(-1) \\
&= -2\sum_{i=1}^{k}(y_i - u) \\
&= -2\left(\sum_{i=1}^{k}y_i - ku\right).
\end{aligned}$$

$$\frac{\mathrm{d}}{\mathrm{d}u}\sum_{i=1}^{k}(y_i - u)^2 = -2\left(\sum_{i=1}^{k}y_i - ku\right) = 0 \Rightarrow \sum_{i=1}^{k}y_i - ku = 0 \Rightarrow u^* = \frac{1}{k}\sum_{i=1}^{k}y_i.$$

Since $\frac{\mathrm{d}}{\mathrm{d}u}\sum_{i=1}^{k}(y_i-u)^2 = -2\left(\sum_{i=1}^{k} y_i - ku\right)\begin{cases} <0, & u\in(-\infty,\frac{1}{k}\sum_{i=1}^{k} y_i) \\ >0, & u\in(\frac{1}{k}\sum_{i=1}^{k} y_i,+\infty)\end{cases}$. Hence, u^* is the local and global minimum.

Hence, $h^w(\boldsymbol{x}) := \frac{1}{k}\sum_{i=1}^{k} y_i = u^*$ minimizes the squared-error loss.

3 Question 3

- Note that τ has n points and τ^* has the same size as τ. For bootstrapping, we sample one point each time, by n times which are independent (i.e. with replacement)
- Hence, for every point, the probability that it is sampled in a single time is $\frac{1}{n}$. Hence, the probability that a point is not sampled in a single time is $1-\frac{1}{n}$
- Hence, the probability that a point is not sampled in all the n times (i.e., a point is not in τ^*) is $\left(1-\frac{1}{n}\right)^n$
- For large n,

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n \xlongequal[t\to+\infty \text{ as } n\to-\infty]{t=-n,\ t\to-\infty \text{ as } n\to+\infty,} \lim_{t\to\infty}\left(1+\frac{1}{t}\right)^{-t} = \lim_{t\to\infty}\frac{1}{\left(1+\frac{1}{t}\right)^t} = \frac{1}{e}\approx 0.37,$$

  i.e., the probability that a point is not in τ^* converges to e^{-1}.
- Denote $X_i = \begin{cases} 1 & \text{if the } i\text{th point of } \tau \text{ is not in } \tau^* \\ 0 & \text{otherwise}\end{cases}$, $i=1,...,n$.

  Apparently, $X_i \sim^{\text{approx}} \text{Bern}(e^{-1})$ for large n.

  Set $X=\sum_{i=1}^{n} X_i$ as # of the points not in τ^*. Since X_i's are i.i.d., $X\sim^{\text{approx}}\text{Bin}(n,e^{-1})$ for large n.

  Hence, for large n, the proportion of the points from τ but not in $\tau^* \approx \frac{\mathbb{E}X}{n} = \frac{ne^{-1}}{n} = e^{-1}$

  $\Leftrightarrow$ For large n, τ^* does not contain a fraction of about e^{-1} of the points from τ

4 Question 4

See random forest regressor code in Appendix.

The optimal parameter $m=9$ with the highest $R^2 = 0.724804$.

5 Question 5

See gradient boosting code in Appendix. The plot, Fig 5-2, is in the next page.

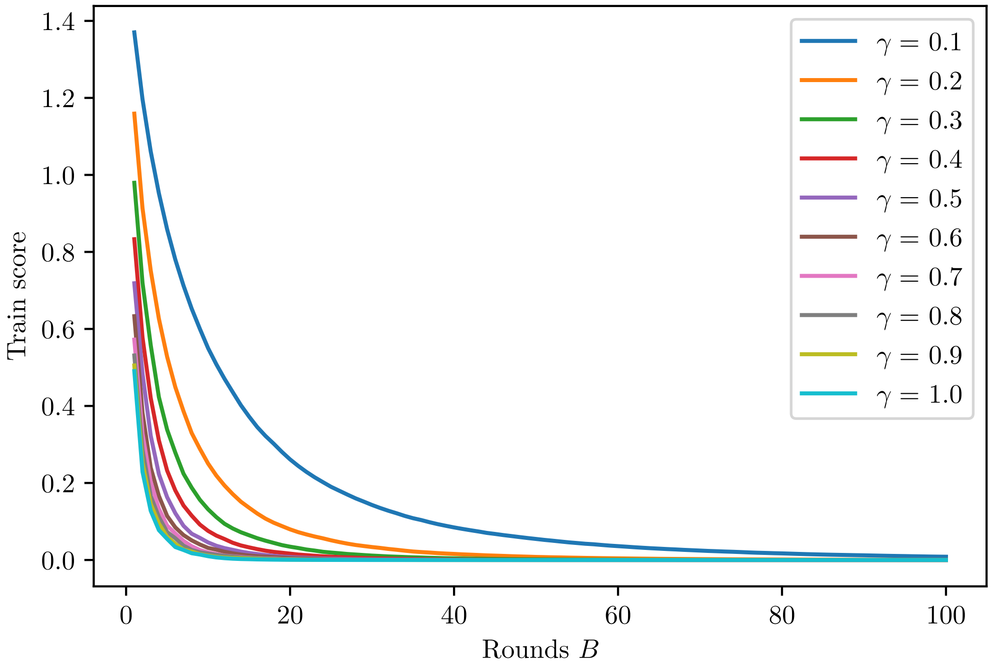
Conclusion: As is shown by Fig 5-2,

- The greater the B, the smaller the train scores (i.e. losses) at all levels of γ, and vice versa.
- The greater the γ, the smaller the train scores at all levels of B, and vice versa.

Note In `GradientBoostingClassifier`, `max_depth` is set so that there is no over-fit to the training set. If not setting `max_depth`, it is default to be 3, which can lead to a sudden increase in train score where γ

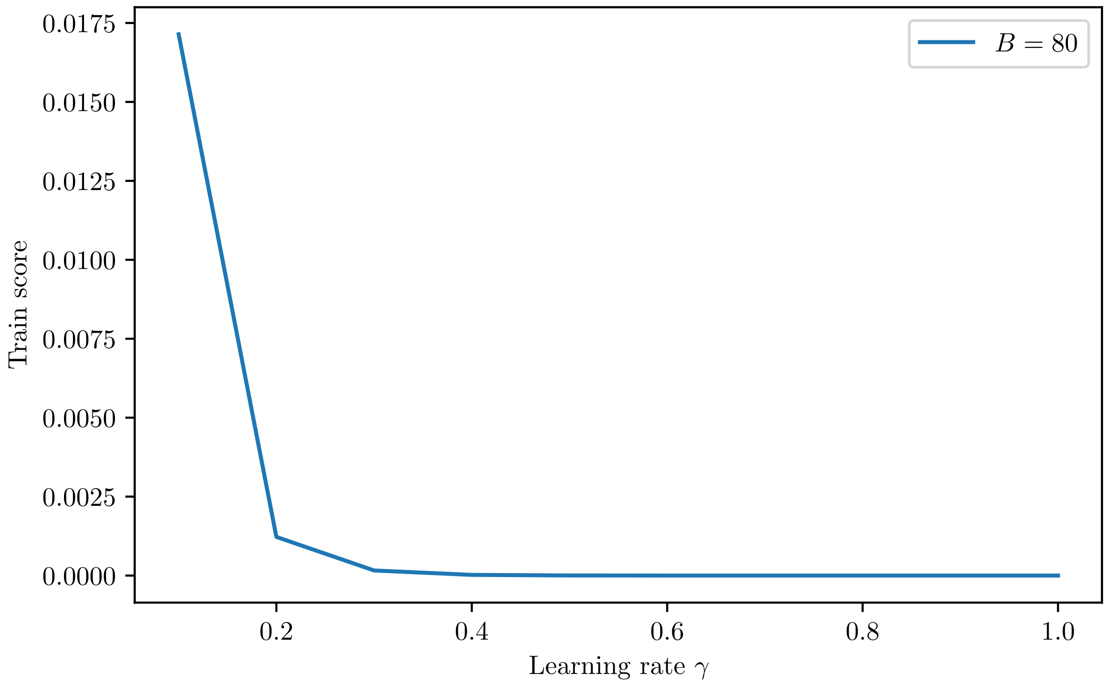
is large in this case.

Fig 5-2: Round B v.s. Train score, at different learning rate γ



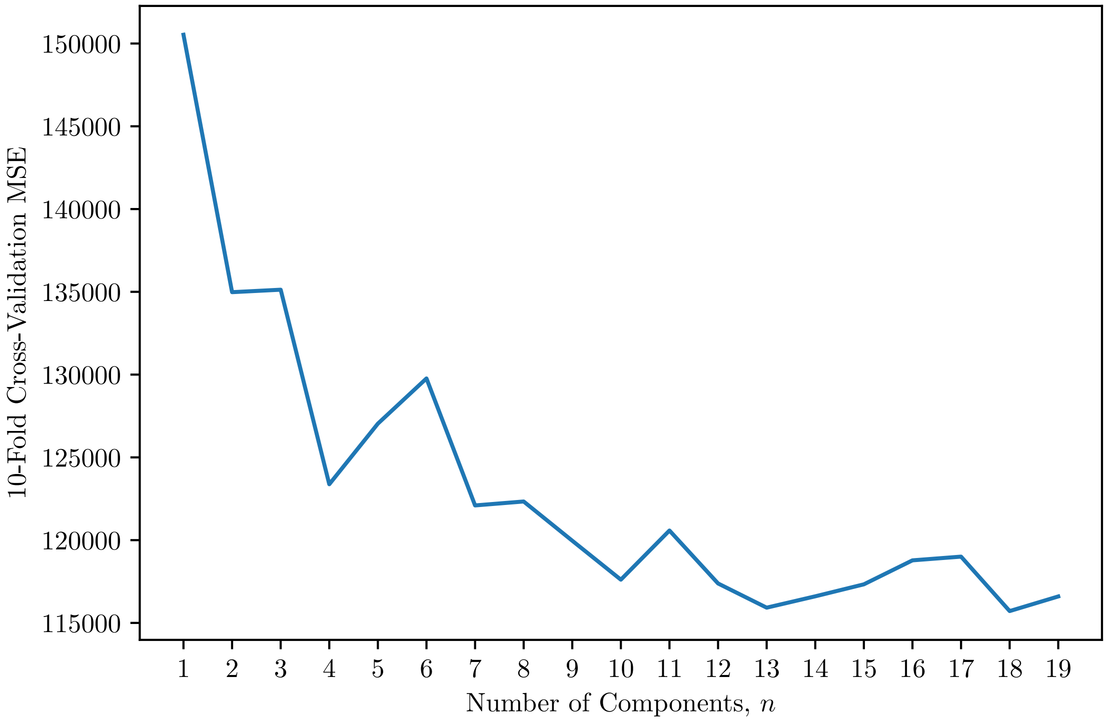
When B is fixed to 80 rounds, the plot of γ v.s. train score is shown in Fig 5-1. (Not sure whether this is required.)

Fig 5-1: Learning rate γ v.s. Train score, $B = 80$ rounds



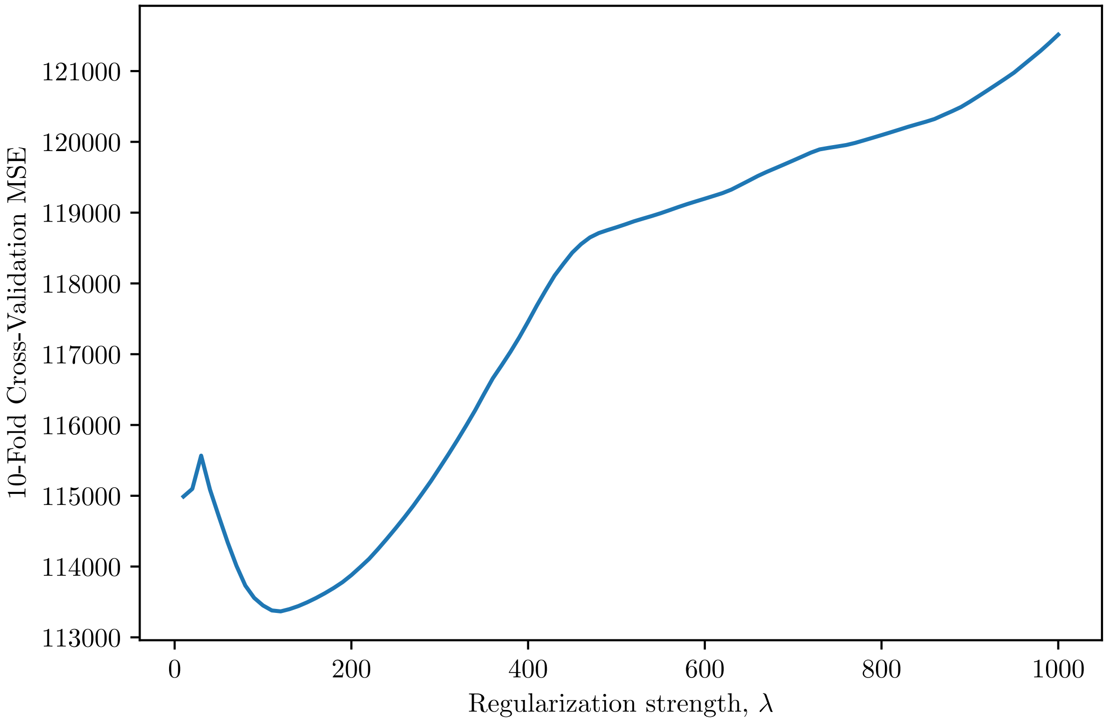
6 Question 6

(a) See PCR and 10-Fold Cross-Validation code in Appendix.



The optimal number of component is $n = 18$.

(b) See Lasso and 10-Fold Cross-Validation code in Appendix.



The best λ is 120, with the lowest MSE 113366.62.

Note The problem does not specify whether a constant should be included. However, if not including the constant, the 10-Fold CV MSEs will be extremely high. Hence, it remains for Question 6.

7 Question 7

(a) See Poisson regression code in Appendix.

	coef	lower bound (0.025)	upper bound (0.975)
type	-0.2237	-0.317	-0.130
construction	0.3714	0.255	0.488
operation	0.7680	0.567	0.969
months	8.095×10^{-5}	7.54×10^{-5}	8.65×10^{-5}

Note The problem does not specify whether a constant should be included. However, the added constant term can be regarded as insignificant (p-value > 0.1). Hence, it is removed for Question 7.

(b) See bootstrapping code (with coefficients, standard errors and normal 95% CIs) in Appendix.
Difference: CIs with bootstrapping are narrower than CIs without bootstrapping.
Reason: Bootstrapping, the sampling with replacement, makes the bootstrapped sampled datasets similar to the original dataset. Hence, the coefficients of each bootstrapped dataset are similar to the coefficients of the original dataset. (Also, possibly, bootstrapping 34 observations for building 1000 bootstrapped datasets brings repeated, identical observations, i.e. even more similarity, to these datasets). By averaging the coefficients across the 1000 bootstrapped datasets, the estimates of the four coefficients are even more stable, leading to the lower standard deviations of the coefficients and hence, narrower CIs.
The CIs with bootstrapping:

	lower bound (0.025)	upper bound (0.975)
type	-0.215245	-0.199097
construction	0.393864	0.414539
operation	0.574860	0.621786
months	0.000096	0.000101

8 Question 8

(a) See code for multiple regression in Appendix.
The fitted model is $Time = 2.3412 + 1.6159 \cdot Cases + 0.0144 \cdot Distance$
The estimated residual standard deviation is 3.259473
The p-value for overall model is 4.69×10^{-16};
The p-value for Cases is 3.254932×10^{-9};
The p-value for Distance is 6.312469×10^{-4}

Note The problem does not specify whether a constant should be included. However, the added constant term can be regarded as significant (p-value < 0.05). Hence, it remains for Question 8.

(b) See code for residual plot/histogram in Appendix.
The residual plots (Fig 8-1, ..., 8-5) are shown below.

- The linear relationship between the explanatory variables and the explained variable fails to be guaranteed, because the residuals does not show a random scatter of points around zero (the dashed horizontal line). In fact, some residuals significantly deviates from zero (Fig 8-1, 8-2, 8-3).
- The constant variance for the residuals fails to be guaranteed, since a as the predicted Y increases, the residuals are increasingly divergent, i.e. a funnel shape in Fig 8-3 which indicates heteroscedasticity.
- Independence among the residuals fail to be guaranteed, since the residuals plotted, by the order in which y_i were measured, are not randomly scattered but show some degree of correlation in Fig 8-4.
- The residuals fail to follow a normal distribution, since the residual histogram is not normally distributed (Fig 8-5).

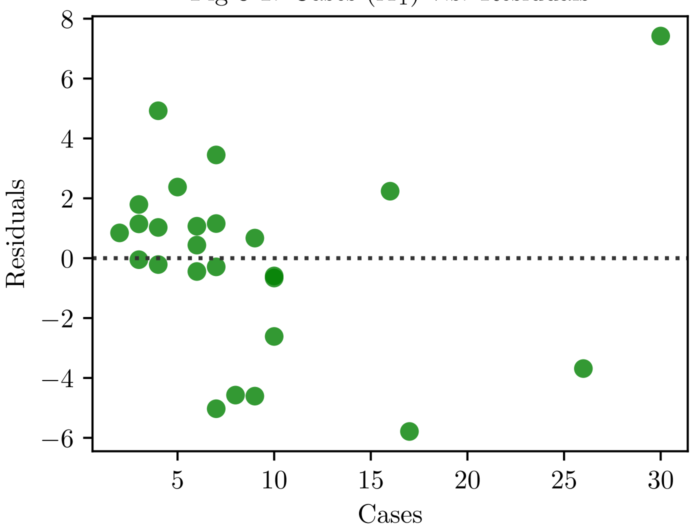


Fig 8-1: Cases (X_1) v.s. Residuals

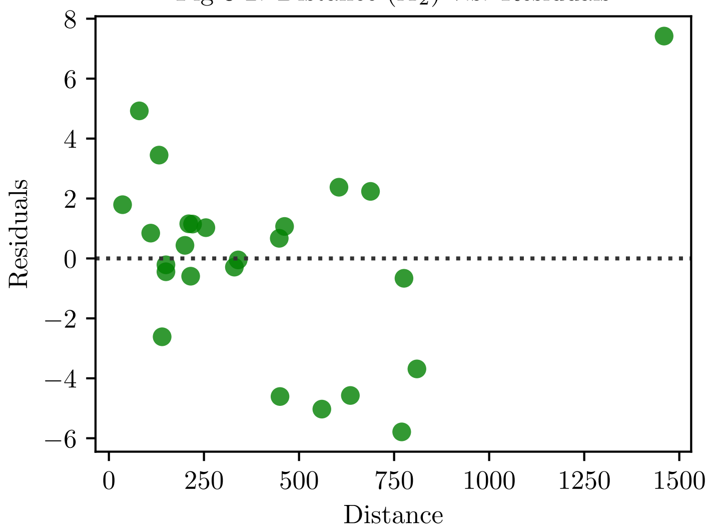


Fig 8-2: Distance (X_2) v.s. Residuals

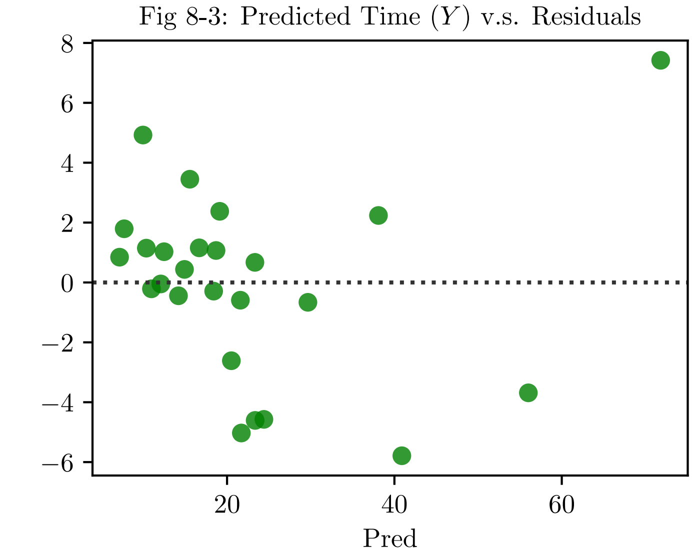
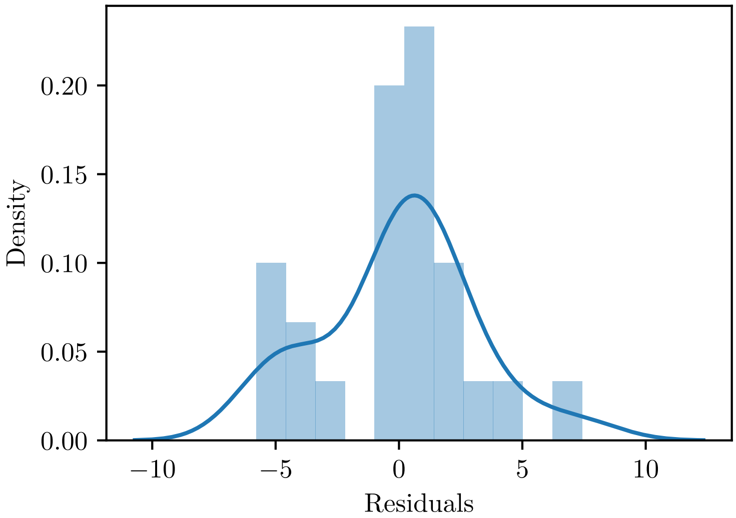


Fig 8-3: Predicted Time (Y) v.s. Residuals

Fig 8-4: Residuals by the order in which y_i were measured

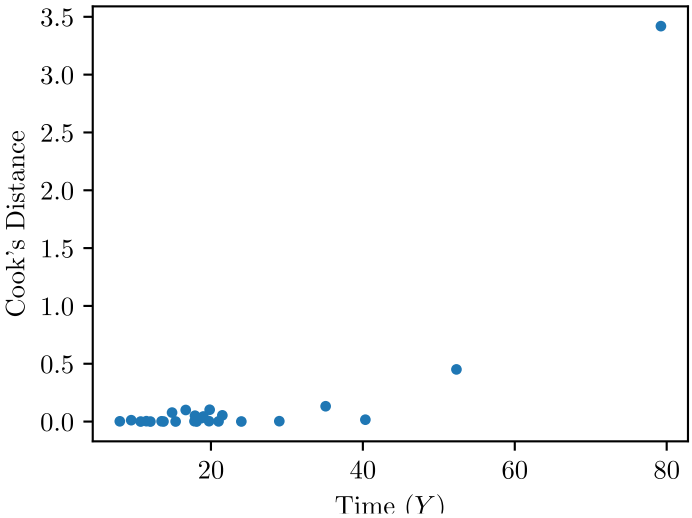
Fig 8-5: Residuals histogram



(c) See code for Cook's distance plot in Appendix.

The Cook's distance plot is below.

Fig 8-6: Time (Y) v.s. Cook's Distance



- The extremely influential observation, with Cook's distance 3.419318, is the row index 8 of `softdrink.csv`, with $Time = 79.24,\ Cases = 30,\ Distance = 1460$
- The next most influential, with Cook's distance 0.451045, is the row index 21 of `softdrink.csv`, with $Time = 52.32,\ Cases = 26,\ Distance = 810$

9 Appendix

Listing 1: **Code for Questions**

```
# ---------------------------------------
# Q1
# ---------------------------------------

import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

np.random.seed(seed=1)
x_tr = np.random.normal(0, 1, 6)
y_tr = 1*x_tr**6 + 2*x_tr**5 + 3*x_tr**4 + 4*x_tr**3 + 5*x_tr**2 + 6*x_tr
np.random.seed(seed=2)
x_te = np.random.normal(0, 1, 6)
y_te = 1.5*x_te**6 + 2*x_te**5 + 3*x_te**4 + 4*x_te**3 + 5*x_te**2 + 6*x_te + np.random.uniform(-2,2,6)

train_loss = []
test_loss = []

for i in  range(1,12):
    poly = PolynomialFeatures(degree=i, include_bias=False)
    poly_features_tr = poly.fit_transform(x_tr.reshape(-1, 1))
    poly_features_te = poly.fit_transform(x_te.reshape(-1, 1))
    poly_reg_model = LinearRegression()
    pr = poly_reg_model.fit(poly_features_tr, y_tr)
    train_loss += [mean_squared_error(y_tr, pr.predict(poly_features_tr))]
    test_loss += [mean_squared_error(y_te, pr.predict(poly_features_te))]

print('train_loss: ', train_loss, '\n test_loss: ', test_loss)

plt.rcParams.update({'text.usetex': True, 'font.family': 'serif'})
fig, ax = plt.subplots(figsize=(6, 4), tight_layout=True, dpi=600, facecolor='white')

ax.plot( range(1,12), train_loss)
ax.plot( range(1,12), test_loss)

ax.legend(labels = ['training loss','testing loss'], loc='best')
ax.set_xlabel('Complexity')
ax.set_ylabel('Loss')
ax.set_title('Fig 1: Complexity v.s. Training/Testing loss of polynomial regression')
plt.savefig('7202a2fig1.pdf')
plt.show()



# ---------------------------------------
```

```
# Q4
# ---------------------------------------

import numpy as np
from sklearn.datasets import make_friedman1
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
# create regression problem
n_points = 1000
x, y = make_friedman1(n_samples = n_points, n_features = 20 ,
                      noise = 2.0 , random_state = 100)
# split to train /test set
x_train , x_test , y_train , y_test = \
train_test_split(x, y, test_size = 0.33 , random_state = 100)

def best_pred_size(m):
    for i in m:
        rf = RandomForestRegressor(n_estimators=500, #oob_score = True,
                                   max_features=i, random_state=100)
        rf.fit(x_train,y_train)
        yhatrf = rf.predict(x_test)
        print('max_features = ', i, ' '*4,
              'RF R^2 score = ',  round(r2_score(y_test, yhatrf), 6))
    return None

best_pred_size(m= list( range(1,21)))



# ---------------------------------------
# Q5
# ---------------------------------------

from sklearn.datasets import make_blobs
from sklearn.metrics import zero_one_loss
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier
import matplotlib.pyplot as plt
import numpy as np

if __name__ == "__main__":
    X_train, y_train = make_blobs(n_samples=1000, n_features=20, centers=5,
                                  random_state=100, cluster_std=6)

def grad_b_c_1(gamma, round_list):
    train_score = []
    for i in round_list:
        breg = GradientBoostingClassifier(learning_rate = gamma,
                                          n_estimators = i,
```

```
                                          max_depth=2, random_state=100)
        breg.fit(X_train,y_train)
        train_score += [breg.train_score_[-1]]
    return round_list, train_score

gamma_list = np.arange(0.1, 1.1, 0.1)
round_list = np.arange(1,101)

plt.figure(#figsize=(4,3),
           dpi=600, facecolor='white')
for j in gamma_list:
    xpoints, ypoints = grad_b_c_1(gamma=j, round_list= round_list)
    plt.plot(xpoints, ypoints)
plt.legend(labels=['$\gamma =\ $' +  str( round(i,1)) for i in gamma_list], loc='best')
plt.xlabel('Rounds $B$')
plt.ylabel('Train score')
plt.title('Fig 5-2: Round $B$ v.s. Train score, at different learning rate $\gamma$')
plt.savefig('7202a2fig5-2.pdf')
plt.show()

def grad_b_c(gamma, rounds):
    train_score = []
    for i in gamma:
        breg = GradientBoostingClassifier(learning_rate = i,
                                          n_estimators = rounds,
                                          max_depth=2, random_state=100)
        breg.fit(X_train,y_train)
        train_score += [breg.train_score_[-1]]
    return gamma, train_score

grad_b_c(gamma_list, 80)

plt.rcParams.update({'text.usetex': True, 'font.family': 'serif'})
fig, ax = plt.subplots(figsize=(6, 4), tight_layout=True, dpi=600, facecolor='white')
xpoints, ypoints = grad_b_c(gamma_list, 80)
ax.plot(xpoints, ypoints)
ax.legend(labels = ['$B = 80$'], loc='best')
ax.set_xlabel('Learning rate $\gamma$')
ax.set_ylabel('Train score')
ax.set_title('Fig 5-1: Learning rate $\gamma$ v.s. Train score, $B=80$ rounds')
plt.savefig('7202a2fig5-1.pdf')
plt.show()



# ---------------------------------------
# Q6
# ---------------------------------------

import pandas as pd
import numpy as np
from sklearn import preprocessing
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn import linear_model

hitters = pd.read_csv('Hitters.csv', header=0)
le = preprocessing.LabelEncoder()
change = ['League', 'Division', 'NewLeague']
for colname in change:
    hitters[colname] = le.fit_transform(hitters[colname])
X_ = hitters.drop(['Salary'], axis=1)
y_ = hitters['Salary']

def Validate(X,Y):
    kf = KFold(n_splits = 10)
    kf.get_n_splits(X)
    mse_10fold = []
    for i in  range(1, X_.shape[1] + 1):
        mse_fold = []
        for train_index, test_index in kf.split(X):
            X_train, X_test = X.iloc[train_index, :], X.iloc[test_index, :]
            y_train, y_test = Y.iloc[train_index], Y.iloc[test_index]
            X_train_pca, X_test_pca = pca_user(X_train=X_train, X_test=X_test, ncomp=i)
            lm = LinearRegression().fit(X_train_pca, y_train)
            y_pred = lm.predict(X_test_pca)
            mse_fold.append(mean_squared_error(y_test, y_pred))
        mse_10fold.append( round(np.mean(mse_fold),2))
    return pd.DataFrame({'ncomp': range(1, X_.shape[1] + 1), 'mse_10fold': mse_10fold})

def pca_user(X_train, X_test, ncomp):
    pca = PCA(n_components = ncomp)
    X_train_pca = pca.fit_transform(X_train)
    X_test_pca = pca.transform(X_test)
    return X_train_pca, X_test_pca

Q6a = Validate(X_, y_)
Q6a

plt.rcParams.update({'text.usetex': True, 'font.family': 'serif'})
fig, ax = plt.subplots(figsize=(6, 4), tight_layout=True, dpi=600, facecolor='white')

xpoints, ypoints = Q6a.iloc[:,0], Q6a.iloc[:,1]
ax.plot(xpoints, ypoints)

ax.set_xlabel('Number of Components, $n$')
ax.set_ylabel('10-Fold Cross-Validation MSE')
plt.xticks(xpoints, xpoints)
plt.savefig('7202a2fig3.pdf')
```

```
plt.show()

Q6a[Q6a.mse_10fold == Q6a.mse_10fold. min()]

def Validate_1(X,Y):
    kf = KFold(n_splits = 10)
    kf.get_n_splits(X)
    mse_10fold = []
    for i in np.arange(10,1010,10):
        mse_fold = []
        lasso = linear_model.Lasso(fit_intercept=True, alpha=i)
        for train_index, test_index in kf.split(X):
            X_train, X_test = X.iloc[train_index, :], X.iloc[test_index, :]
            y_train, y_test = Y.iloc[train_index], Y.iloc[test_index]
            lasso.fit(X_train, y_train)
            y_pred = lasso.predict(X_test)
            mse_fold.append(mean_squared_error(y_test, y_pred))
        mse_10fold.append( round(np.mean(mse_fold),2))
    return pd.DataFrame({'Lambda': np.arange(10,1010,10), 'Ten_Fold_CV_MSE': mse_10fold})

Q6b = Validate_1(X_, y_)
Q6b

plt.rcParams.update({'text.usetex': True, 'font.family': 'serif'})
fig, ax = plt.subplots(figsize=(6, 4), tight_layout=True, dpi=600, facecolor='white')

xpoints, ypoints = Q6b.iloc[:,0], Q6b.iloc[:,1]
ax.plot(xpoints, ypoints)

ax.set_xlabel(r'Regularization strength, $\lambda$')
ax.set_ylabel('10-Fold Cross-Validation MSE')
plt.savefig('7202a2fig4.pdf')
plt.show()

Q6b[Q6b.Ten_Fold_CV_MSE == Q6b.Ten_Fold_CV_MSE. min()]



# ---------------------------------------
# Q7
# ---------------------------------------

import pandas as pd
import statsmodels.api as sm

ships = pd.read_csv('ships.csv', header=0)
ships

exog, endog = ships.drop('damage', axis=1), ships['damage']
pois = sm.GLM(endog, exog, family=sm.families.Poisson())
res = pois.fit()
res.summary()
```

```
n_estimators = 1000
params = pd.DataFrame(columns=['type','construction','operation','months'])

for i in  range(n_estimators):
    np.random.seed(seed=i)
    ids = np.random.choice( range( len(ships)), size =  len(ships), replace=True)
    exog, endog = ships.loc[ids,:].drop('damage', axis=1), ships.loc[ids,'damage']
    pois_res = sm.GLM(endog, exog, family=sm.families.Poisson()).fit()
    params.loc[i,:] =  list(pois_res.params)

params

lower_bound =  list(params.mean(axis=0) - 1.96 * params.std(axis=0) / np.sqrt(1000))
upper_bound =  list(params.mean(axis=0) + 1.96 * params.std(axis=0) / np.sqrt(1000))

params_1 = pd.DataFrame(index = ['type','construction','operation','months'],
                        columns=['coef','std err','[0.025','0.975]'])
params_1.loc[:,'coef'] =  list(params.mean(axis=0))
params_1.loc[:,'std err'] =  list(params.std(axis=0))
params_1.loc[:,'[0.025'] = lower_bound
params_1.loc[:,'0.975]'] = upper_bound
params_1



# ---------------------------------------
# Q8
# ---------------------------------------

import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns

softdrink = pd.read_csv('softdrink.csv', header=0)
softdrink

X = sm.add_constant(softdrink.drop('Time',axis=1))
Y = softdrink['Time']
lr = sm.OLS(Y, X).fit()
print(lr.summary())

lr.params
lr.pvalues
lr.resid

softdrink['Pred'] = lr.predict(X)
softdrink['Residuals'] = softdrink['Time'] - softdrink['Pred']
softdrink

# first method to calculate the estimated residual standard deviation
```

```
lr.mse_resid**0.5
# second method to calculate the estimated residual standard deviation
( sum(softdrink['Residuals']**2) / (25-2-1))**0.5

plt.rcParams.update({'text.usetex': True, 'font.family': 'serif'})
plt.figure(figsize=(4,3),dpi=600, facecolor='white')
sns.residplot(x=softdrink['Cases'], y=lr.resid, lowess=False, color='g', label='a')
plt.title('Fig 8-1: Cases ($X_1$) v.s. Residuals', fontsize=10)
plt.savefig('7202a2fig8-1.pdf')
plt.show()

plt.figure(figsize=(4,3),dpi=600, facecolor='white')
sns.residplot(x=softdrink['Distance'], y=lr.resid, lowess=False, color='g', label='a')
plt.title('Fig 8-2: Distance ($X_2$) v.s. Residuals', fontsize=10)
plt.savefig('7202a2fig8-2.pdf')
plt.show()

plt.figure(figsize=(4,3),dpi=600, facecolor='white')
sns.residplot(x=softdrink['Pred'], y=lr.resid, lowess=False, color='g', label='a')
plt.title('Fig 8-3: Predicted Time ($Y$) v.s. Residuals', fontsize=10)
plt.savefig('7202a2fig8-3.pdf')
plt.show()

plt.figure(figsize=(4,3),dpi=600, facecolor='white')
sns.residplot(x= list( range(1, len(softdrink)+1)), y=lr.resid, lowess=False, color='g', label='a')
plt.title('Fig 8-4: Residuals by the order in which $y_i$ were measured', fontsize=8)
plt.savefig('7202a2fig8-4.pdf')
plt.show()

plt.figure(figsize=(4.2,3),dpi=600, facecolor='white')
sns.distplot(softdrink['Residuals'])
plt.title('Fig 8-5: Residuals histogram', fontsize=10)
plt.savefig('7202a2fig8-5.pdf')
plt.show()

cooks = lr.get_influence().cooks_distance
cooks
Q8c = pd.DataFrame({'Y': softdrink['Time'], 'Cooks': cooks[0]})
Q8c

plt.figure(figsize=(4,3),dpi=600, facecolor='white')
plt.scatter(Q8c.iloc[:,0], Q8c.iloc[:,1], marker='.')
plt.xlabel('Time ($Y$)')
plt.ylabel('Cook\'s Distance')
plt.title('Fig 8-6: Time ($Y$) v.s. Cook\'s Distance')
plt.savefig('7202a2fig8-6.pdf')
plt.show()

Q8c.sort_values('Cooks',ascending=False)
softdrink.iloc[Q8c.sort_values('Cooks',ascending=False).index[:2],:]
```