Bonus Question

How bias and variance would change when model complexity changes?

1. Analyze from definition

- Bias: how much the prediction value obtained by the model differs from the real value.

  sampling and *estimation* could both introduce bias. As the discussion is mainly about the impace of model complexity, more focus will be on *estimation error* (for sampling error, resampling and repeat the model building precess then derive the average of prediction values could halp, however, no change in model complexity).

  So the causes for a larger bias could be a simpler model to do a simpler approximation, as a more complicated model usually could do a better fit. The risk of underfitting exists. Because with simpler model, less assumptions are made. For example, with the simplest model I could think (simple linear regression)

- Variance: with the same model, if a different training set is used, how much the estimate of the function will be changed. Intuitively, when a more comlicated model is selected, more parameters need to be calculated. It would be much easier to obtain a result with a large variance compared to the former results. High-variance might fit the training data well (as with different training data, large change in estimate), but the risk of overfitting exist (fit too good to predict the future value). Large noise might be fitted by the complicated model.

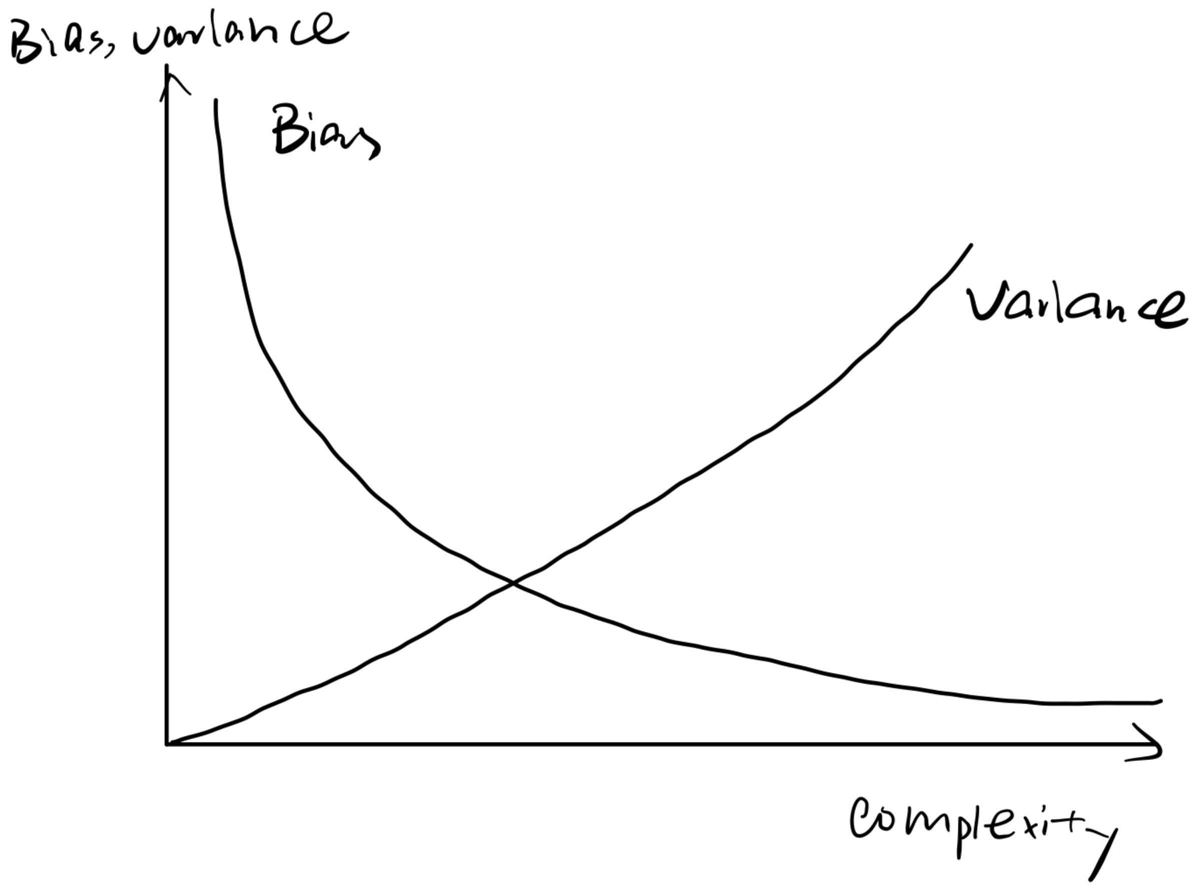
So, there exists bias-variance tradeoff. From the lecture slides, $\mathrm{MSE(m)} = \mathbb{E}[(\mathrm{Y}-\mathrm{m})^2] = \mathrm{Var[Y]} + (\mathbb{E}[\mathrm{Y}-\mathrm{m}])^2$,

Model with small variance and high bias *underfit* the truth target, while with high variance and small bias *overfit* the truth target.

2. Answer for the trend

This answer is mainly based on the intuition in 1. and the fitted results shown in 3.

	Bias	**Variance**
monotone	yes	yes
change in rate	first change fast, then slower	roughly the same rate, do not differ much
Increase / decrease	Decrase	Increase

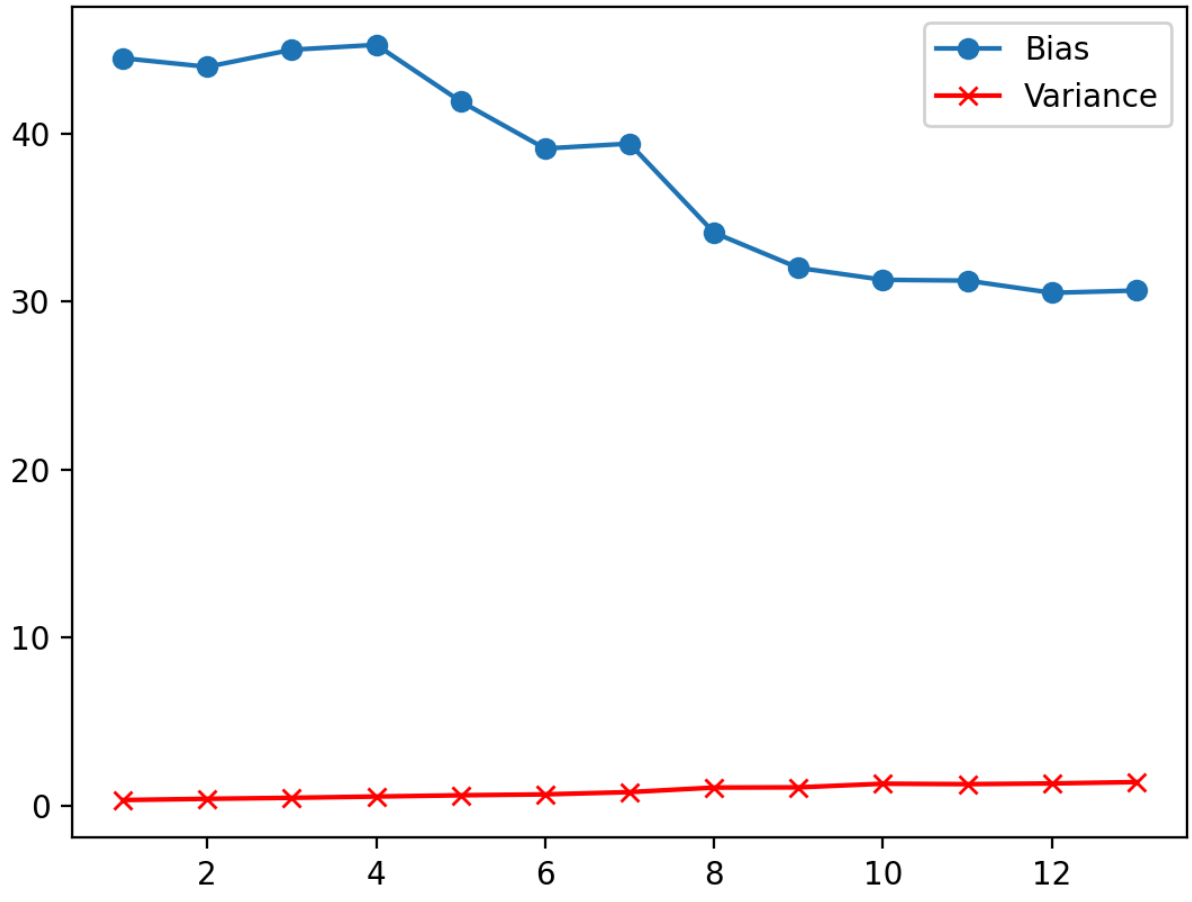


3. The fit is performed through Python and the raw data obtained by url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'. The head of the imported data set is

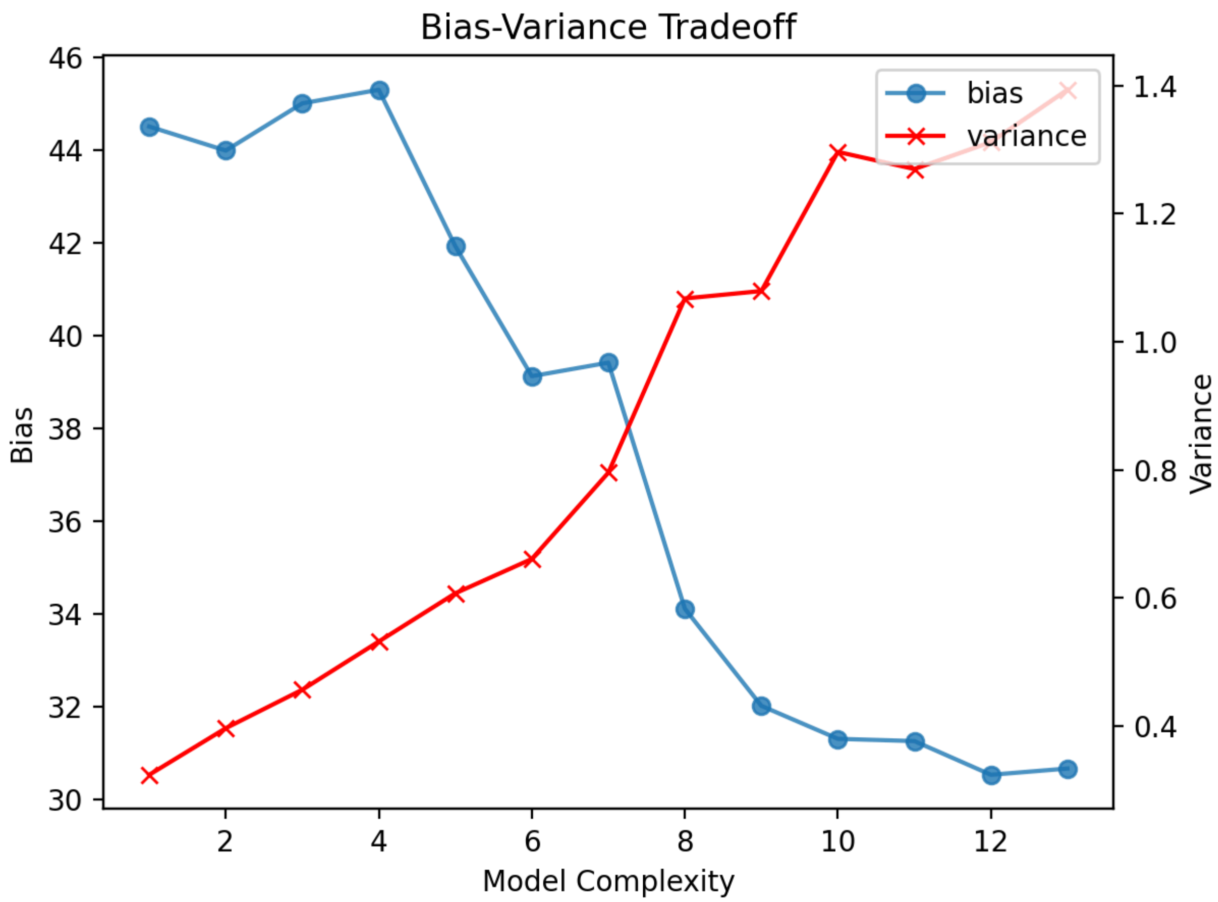
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

The last column is the y value we need to fit. So there are 13 parameters. The model complexity is increased by adding more features in the linear regression mode. After fitting, we have 13 models with the correspinding bias and variance.

The line plot of bias and variance that have the smae y-axis shows below.



To have a better view, different axes are used. Then the different pattern of bias and variance are obvious. Because of the limitation of the real dataset and simple method to increase the model complexity, it has some fluctuations. But the overall pattern meets the assumptions in 1 and 2. Bias will decrease and variance will increase when the model complexity increases.



The code its attached at the end.

```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import cross_val_score
from mlxtend.evaluate import bias_variance_decomp

# estimate the bias and variance for a regression model
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from mlxtend.evaluate import bias_variance_decomp
# load dataset
url =
'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housin
g.csv'
```

```
dataframe = read_csv(url, header=None)
# separate into inputs and outputs
data = dataframe.values
y = data[:, -1]
biasList = []
varList = []
for i in range(12,0, -1):
    X= data[:, i:-1]
    X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
    linreg = LinearRegression().fit(X_train, y_train)
    mse, bias, var = bias_variance_decomp(
        linreg, X_train, y_train, X_test, y_test, loss = 'mse')
    biasList.append(bias)
    varList.append(var)

x_num = [i for i in range(1,13)]
x_num

fig = plt.figure()
plt.plot(x_num, biasList, '-o', x_num, varList, '-xr')
plt.legend(['Bias', 'Variance'])

plt.show()

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(x_num, biasList, '-o',alpha=0.8, label = 'bias')
ax1.set_ylabel('Bias')
ax1.set_xlabel('Model Complexity')
ax1.set_title('Bias-Variance Tradeoff')

ax2 = ax1.twinx()
ax2.plot(x_num, varList, '-xr', label = 'variance')
ax2.set_ylabel('Variance')
fig.legend(loc=1, bbox_to_anchor=(1,1),
bbox_transform=ax1.transAxes)
plt.show()
```

Also, residuals vs. fitted values are plotted for further analysis. We could see that it has been more close to the mean 0, which meets our assumption.