Machine Learning

Lab 2: Regressor & Classifier

Dr. Shuang LIANG

Basic lab instructions

- You may want to *bring your slides* to labs to look up syntax and examples.
- Have a question? *Ask a TA* for help, or look at the lecture slides.
- We encourage you to *talk to your classmates*; it's okay to share code and ideas during lab.
- Make good use of these tools: *search engines, API documentation* and *IDEs*.
- You don't have to finish all of the exercises. Just do as much as you can in the allotted time. You don't need to finish the rest after you leave the lab; there is no homework from lab.

Today’s lab exercises

- The exercise consists of two parts, *Regressor* and *Classifier*
- In the Regressor part, you will complete the following exercises
  - 1.Train linear regression using least squares and gradient descent
  - 2.Evaluation and visualization them
- In the Classifier part, you will complete the following exercises
  - 1.Implement multiple classifiers, including Logistic Regression, KNN, Decision Trees and SVM
  - 2.Evaluation and visualization
- Some extension tasks
  - 1. Observe the changes in the model's training and testing loss, bias and variance as complexity increases
  - 2. Review the content of lab1 using functions from libraries

Part 0
Project Description

Experiment environment

- Python 3.x
- Libraries: Scikit-learn/Matplotlib/Numpy
- Docs
  - Scikit-learn： https://scikitlearn.org/stable/user_guide.html
  - Numpy： https://numpy.org/doc/stable/
  - Matplotlib： https://matplotlib.org/stable/tutorials/index.html

- You can use the operating environment of lab1.

Project structure

- Please design the project and code structure on your own

Part 1 Regressor

Preparing data

- Diabetes dataset
  - https://scikit-learn.org/stable/datasets/toy_dataset.html#diabetes-dataset
  - # diabetes 是一个关于糖尿病的数据集， 该数据集包括442个病人的生理数据及一年以后的病情发展情况

- Provided by Scikit-learn:
  - `from sklearn.datasets import load_diabetes`
- Divide the data set into training and test sets
  - `from sklearn.model_selection import train_test_split`
  - Remember to set random_state

- Observe the data format

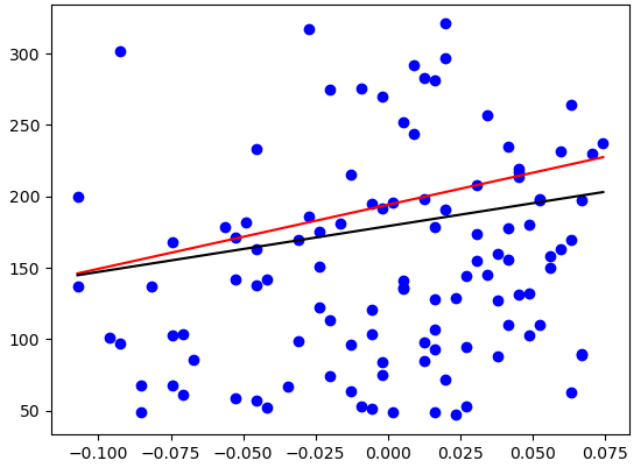
Train a linear regression model

- Least squares
- `from sklearn import linear_model`
  - `Use LinearRegression().fit()`

- Gradient descent
- `from sklearn import linear_model`
  - `Use SGDRegressor().fit()`
- Select appropriate parameters
  - Learning rate
  - Penality (L1, L2)
  - Loss

Evaluation and Visualization

- Evaluation
  - `from sklearn.metrics import mean_squared_error`
  - Can you implement MSE on your own?
- Visualization
  - Matplotlib
  - Data – Scatter
  - Function – Line
- Which model is better?
  - Compare MSE
  - Plot the fitted function in the same figure



Choose which feature to use at your own discretion. For the intuition of the scatter plot, it is recommended to use only one set of feature.

Tasks

- Tasks:
- 1.Train linear regression using least squares and gradient descent
- 2.Evaluation and visualization them

- Basic requirement: (15mins)
- Learn to call functions from libraries `sklearn` to implement them

- Extended requirement: (15mins)
- Independently complete the logic of the Evaluation part.
- Visualization.

Recall: Performance Measure

MSE for Regression $$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\left(f\left(\boldsymbol{x}_i\right)-y_i\right)^2$$

Accuracy for Classification $$\begin{aligned}\mathrm{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)=y_i\right)\\ &= 1-E(f;D)\ .\end{aligned}$$

Precision $P=\frac{TP}{TP+FP}$

Recall $R=\frac{TP}{TP+FN}$

$$F1=\frac{2*P*R}{P+R}=\frac{2*TP}{the\ number\ of\ samples+TP-TN}$$

Part 2 Classifier

Preparing data

- Iris dataset
  - https://scikit-learn.org/stable/datasets/toy_dataset.html#iris-plants-dataset
  - iris数据集是由三种鸢尾花，各50组数据构成的数据集。每个样本包含4个特征，分别为萼片(sepals)的长和宽、花瓣(petals)的长和宽。

- Provided by Scikit-learn:
  - `from sklearn.datasets import load_iris`
- Divide the data set into training and test sets
  - `from sklearn.model_selection import train_test_split`
  - Remember to set random_state

Expected Format

- The classifier is called through the ***Predict*** function
- Only the first string parameter of the function needs to be adjusted
- The parameter values are KNN, Logistic, DecisionTree and SVM.
- An example (call KNN classifier)

```
y_pred = Predict('KNN',X_test)
```

Evaluation

- Use Accuracy, Precision, Recall, F1 Score for evaluation
  - ```
    from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
    ```

- Think：
- Is the result the same?
- This is a multi-classification problem. You can set the average precision type parameter ‘average’. Recall **Homework 1**.
- There are many parameters in the classifier. Try to tune them to get a better model.
    ```

Homework 1

4. Consider a triple classification problem that requires the recognition of three classes, A, B and C.

		Predicted		
		A	**B**	**C**
Actual	**A**	40	20	10
	B	35	85	40
	C	0	10	20

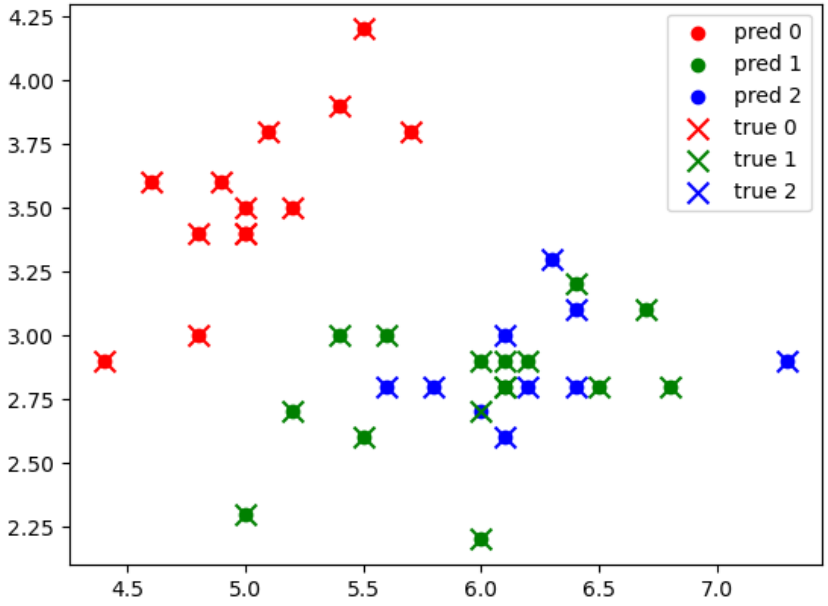
Given the confusion matrix, please complete the following calculations.

(1) Calculate the precision and recall for each class respectively.

(2) Use both Macro-average and Weighted-average to calculate the precision and recall of the whole classifier. If you are not sure about these concepts, use a search engine. Retain 4 decimal places (only for (2)).

Visualization

- Visualize the data with a scatter plot
- Different classes of data are labeled with different colors
- Implement ***Vis*** function in the template



Tasks

- Basic requirement: (20mins)
  - Learn to call functions from libraries `sklearn` to implement :
  - 1. Classification using KNN, Logistic Regression, Decision Tree and SVM (10mins)
    - Although some classifiers have not yet been taught in class, you can still use them easily
  - 2. Evaluation: Accuracy, Precision, Recall, F1 Score (10mins)
    - Combining with Assignment 1, try to understand micro avg, macro avg, and weighted avg.

- Extended requirement:
  - 3. Visualization: classification results (10mins)

Another Extension Task

- 1. You can adjust the complexity of the classifier and observe the changes in their training and testing loss, bias and variance. (15mins)
- Taking SVM as an example, the gamma parameter represents its complexity.
- You can use a larger dataset, for example, `sklearn.datasets.load_digits()`.
  - *手写数字识别数据集*

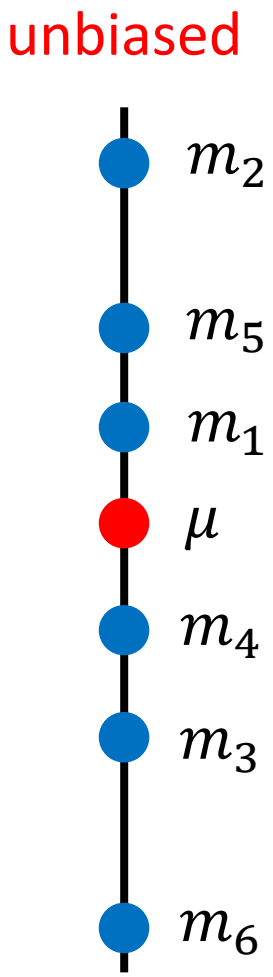
- 2. You can review the content of lab1 using functions from libraries `sklearn` : (15mins)
  - Try K-fold Cross Validation to selecting the best model
    - Hint: *sklearn.model_selection. cross_val_score*
  - Try to plot P-R Curve
    - Hint: *sklearn.metrics.precision_recall_curve*
  - Try to plot P-R Curve
    - Hint: *sklearn.metrics. roc_curve*

Bias and Variance of Estimator

- Estimate the mean of a variable x
  - assume the mean of x is μ
  - assume the variance of x is σ^2
- Estimator of mean μ
  - Sample N points: $\{x^1, x^2, \ldots, x^N\}$

$$m = \frac{1}{N}\sum_n x^n \neq \mu$$

$$E[m] = E\left[\frac{1}{N}\sum_n x^n\right] = \frac{1}{N}\sum_n E[x^n] = \mu$$



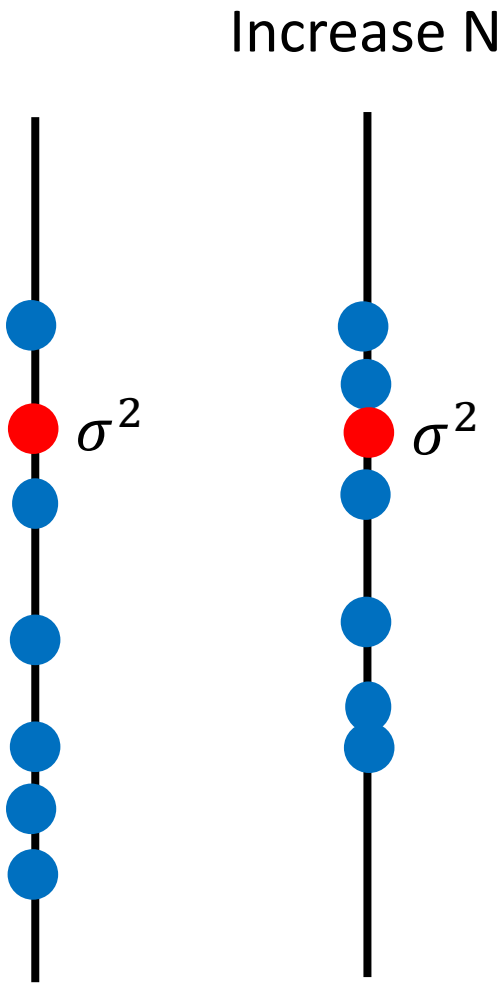
Bias and Variance of Estimator

- Estimate the mean of a variable x
  - assume the mean of x is μ
  - assume the variance of x is σ^2
- Estimator of variance σ^2
  - Sample N points: $\{x^1, x^2, \ldots, x^N\}$

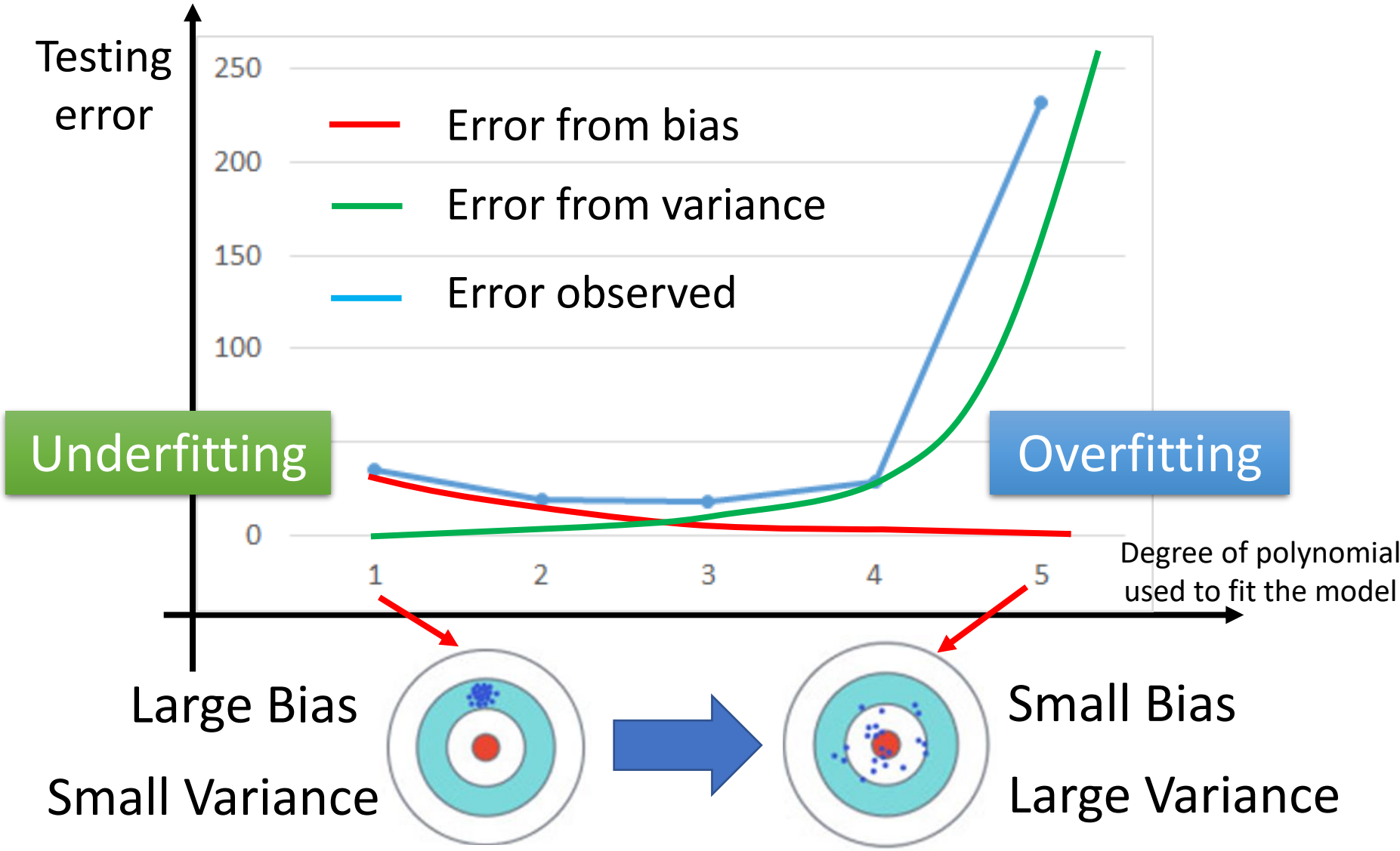
$$m = \frac{1}{N}\sum_n x^n \quad s^2 = \frac{1}{N}\sum_n (x^n - m)^2$$

Biased estimator

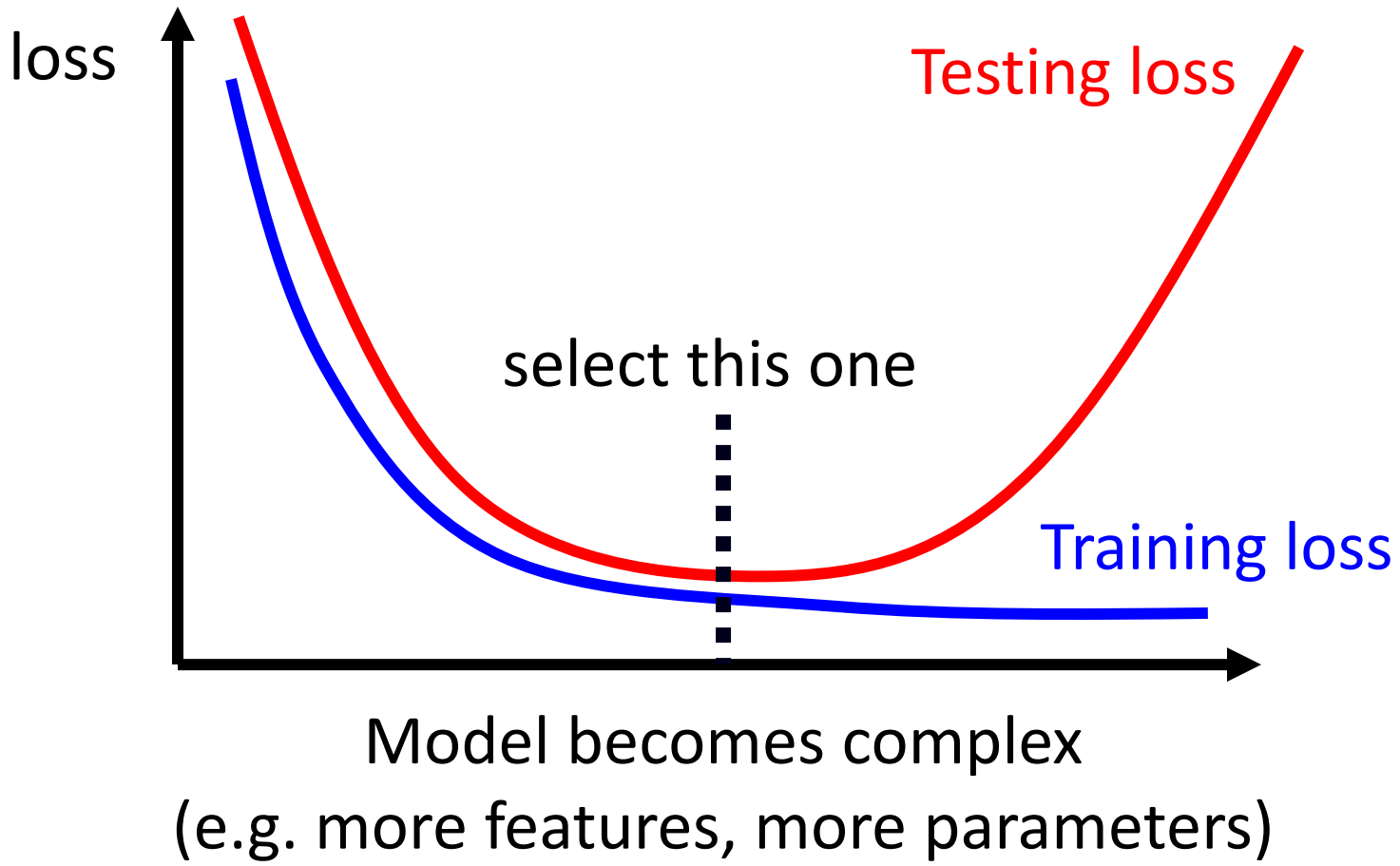
$$E[s^2] = \frac{N-1}{N}\sigma^2 \neq \sigma^2$$



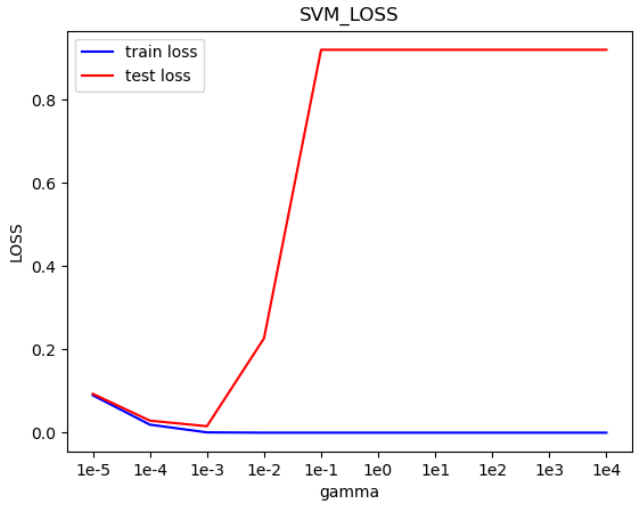
Bias v.s. Variance

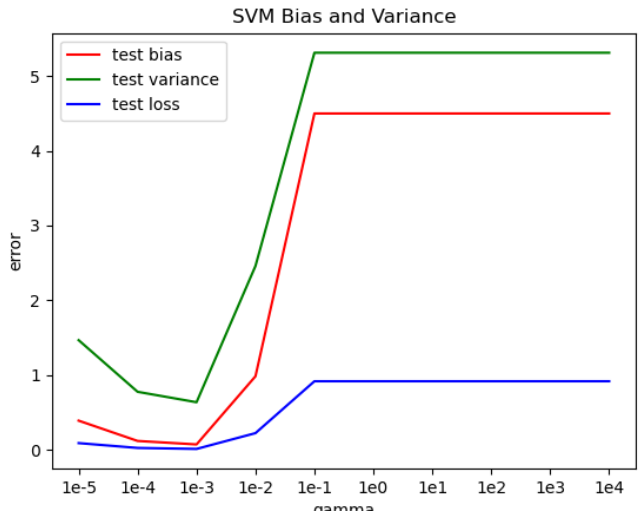


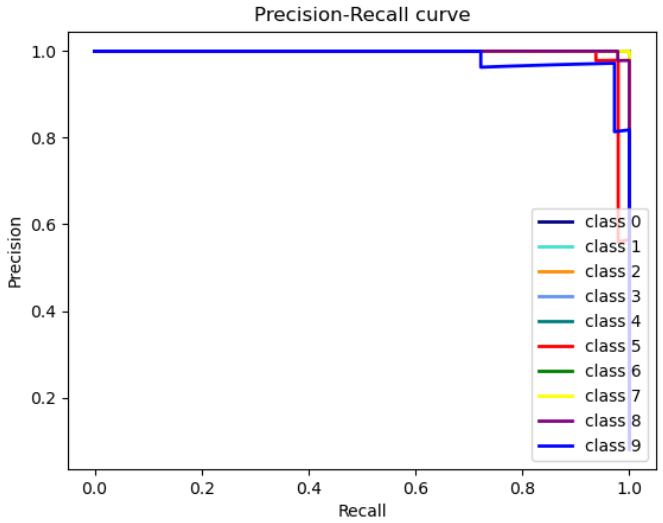
Bias-Complexity Trade-off

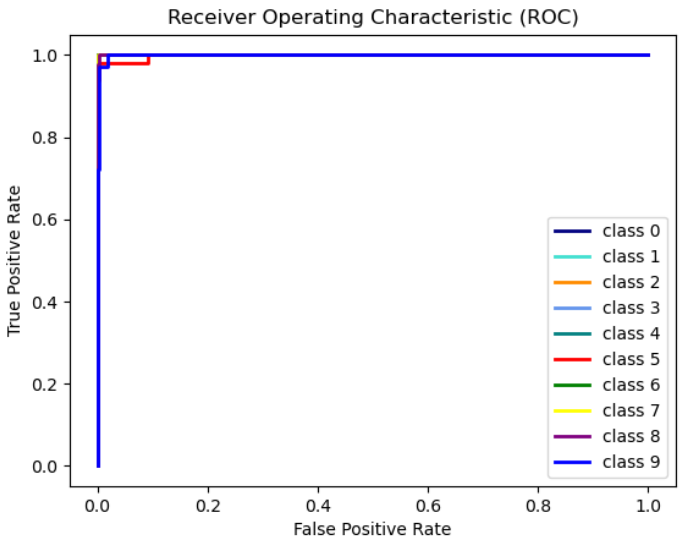


Expected Result









Thank you for your attention!