SVM with sklearn library for classification using Iris and Moon datasets

BEIJING INSTITUTE OF TECHNOLOGY

PIMENOV GLEB

1820243077

I hereby promise that the experimental report and codes written by me are independently completed without any plagiarism or copying of others' homework. All the views and materials involving other classmates have been annotated. If there is any plagiarism or infringement of others' intellectual property rights, I will bear all the consequences arising from it.

SVM with sklearn library for classification using Iris and Moon datasets

1 Experiment Introduction

In this experiment, we aim to utilize the SVM (Support Vector Machine) algorithm implemented in the scikit-learn library for classification tasks. We will apply SVM to two different datasets: Iris dataset and Make_moons dataset. SVM is a powerful supervised learning method commonly used for classification and regression tasks.

2 Experiment Objectives

Utilize the Iris dataset: The Iris dataset is a classic dataset in machine learning used for classification tasks. It consists of measurements of iris flowers from three different species. We will use SVM to classify iris flowers into their respective species based on the provided measurements.

Utilize the Make_moons dataset: The Make_moons dataset is a synthetic dataset useful for binary classification tasks. It generates moon-shaped clusters of points, making it a suitable dataset for testing non-linear classification algorithms like SVM.

Apply the SVM method: Support Vector Machine (SVM) is a supervised learning algorithm that can be used for both classification and regression tasks. SVM works by finding the hyperplane that best separates the classes in the feature space. We will apply SVM to both datasets and evaluate its performance.

3 Relevant Theories and Knowledge

Support Vector Machine (SVM) Method Theory:

SVM is a supervised learning algorithm used for classification and regression tasks.

In classification, SVM finds the hyperplane that best separates the classes in the feature space. The goal is to maximize the margin, i.e., the distance between the hyperplane and the nearest data points from each class.

SVM can handle both linear and non-linear classification tasks by using different kernel functions such as linear, polynomial, radial basis function (RBF), etc.

SVM is effective in high-dimensional spaces and is memory efficient because it only uses a subset of training points (support vectors) in the decision function.

SVM performs well in practice even with relatively small datasets. However, it may not be suitable for very large datasets due to its computational complexity.

In this experiment, we will apply SVM to classify the Iris and Make_moons datasets and evaluate its performance in terms of accuracy and generalization.

4 Experimental Tasks and Grading Criteria

No.	**Task Name**	**Specific Requirements**	**Grading Criteria (100-point scale)**
1	SVM with sklearn library for classification using Iris and Moon datasets	Development language: Python	100

5 Experimental Conditions and Environment

Requirements	Name	Version	Remarks
Programming Language	Python	3.12	
Development Environment	windows	11	

Third-party toolkits/libraries/plugins	Numpy Matplotlib Sklearn Seaborn Joblib		
Other Tools	Jupyter notebook		
Hardware Environment	I5 12XXX 8GB RAM		

6 Experimental Data and Description

Attribute (Entry)	Content
Dataset Name	**Iris, Moons**
Dataset Origin	Made famous by the British statistician and biologist Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis,
Main Contents of the Dataset	sepal length, sepal width, petal length, petal width and species
Dataset File Format	sklearn
Dataset Name	**Make moons**
Dataset Origin	A simple toy dataset to visualize clustering and classification algorithms
Main Contents of the Dataset	X,Y
Dataset File Format	sklearn

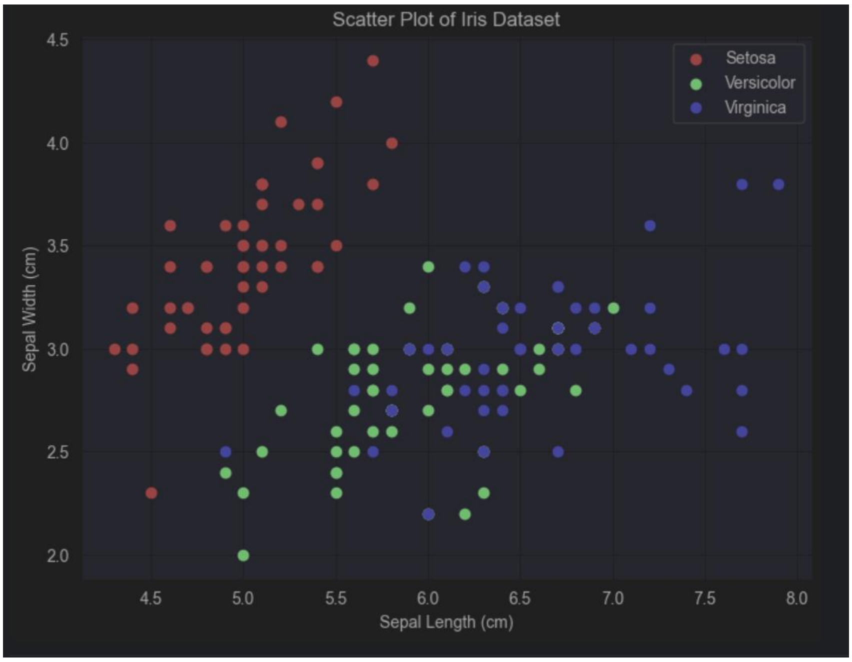
7 Experimental Steps and Corresponding Codes

Step number	1

Step Name	Importing
Step Description	import the required libraries
Code and Explanation	#%% import seaborn as sns from sklearn.metrics import confusion_matrix from sklearn.metrics import classification_report from matplotlib import pyplot as plt #%% md # Datasets #%% from sklearn.datasets import load_iris from sklearn.datasets import make_moons #%% md # Models #%% from sklearn.svm import LinearSVC from sklearn.svm import SVC from sklearn.model_selection import train_test_split from sklearn.model_selection import GridSearchCV #%% md # Saving #%% from joblib import dump

Step number	**2**
Step Name	Looking at the iris dataset
Step Description	Loading iris dataset, checking data shapes, getting features and targets, draw a

	plot by 2 params
Code and Explanation	iris = load_iris() row_count = iris.data.shape[0] column_count = iris.data.shape[1] print(f'The DataFrame has {row_count} rows.') print(f'The DataFrame has {column_count} columns.') #%% md # Getting features and targets #%% x = iris.data y = iris.target #%% md # Define colors and labels for each class #%% colors = ['red', 'green', 'blue'] labels = ['Setosa', 'Versicolor', 'Virginica'] #%% md # Create scatter plot #%% plt.figure(figsize=(8, 6)) for i in range(len(colors)): plt.scatter(x[y == i, 0], x[y == i, 1], c=colors[i], label=labels[i]) # Add labels and legend plt.xlabel('Sepal Length (cm)') plt.ylabel('Sepal Width (cm)') plt.title('Scatter Plot of Iris Dataset') plt.legend() # Show plot

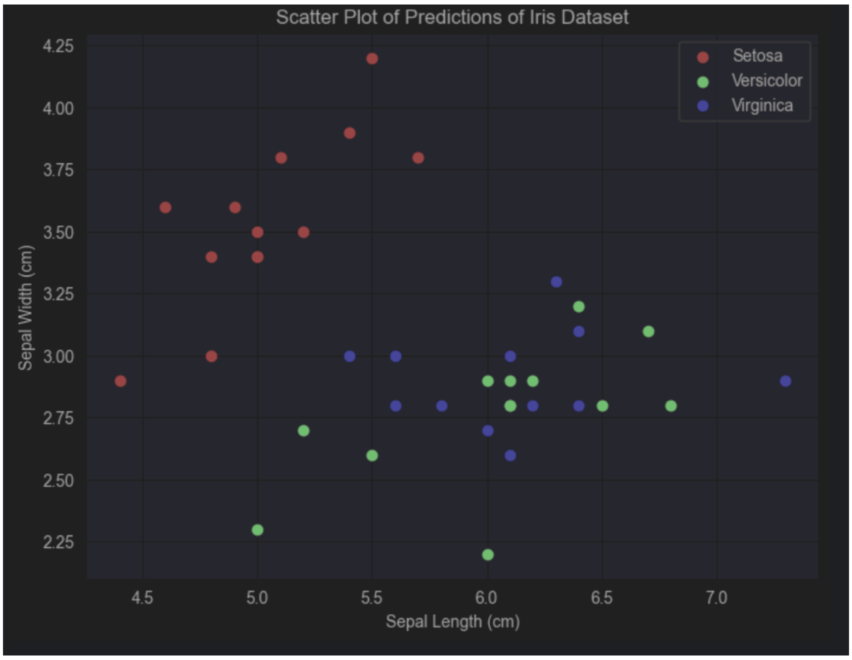
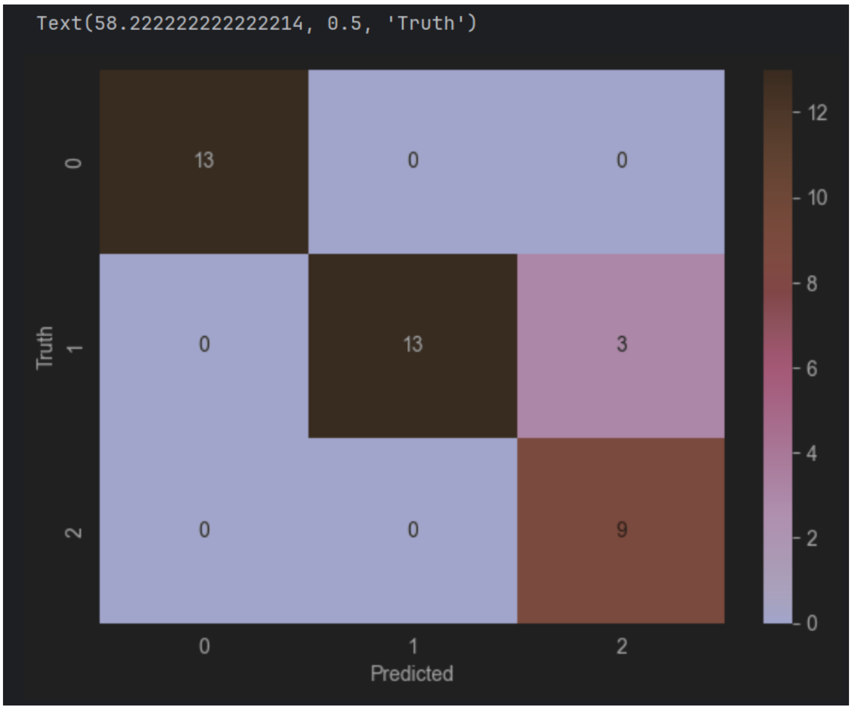
	plt.grid(True) plt.show()
Output results and Interpretation	

Step number	3
Step Name	Training LinearSVC model for iris dataset
Step Description	Splitting data into training and splitting samples, define the hyperparameters grid, fitting the model, checking the best params and saving the model into the file
Code and Explanation	# Splitting data on train and test #%% x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, shuffle=True) #%% md # Define the parameter grid (for the iterating hyperparameters) #%% param_grid = { 'C': [1, 2, 2.1, 2.2, 2.5, 3],

	'dual' : ['auto'] } #%% md # Create the grid search #%% model = GridSearchCV(LinearSVC(), param_grid, scoring='precision_macro', n_jobs=-1, verbose=3) #%% md # Fitting (training) the model with different parameters and printing the best parameters #%% model.fit(x_train, y_train) model.best_params_ #%% md # Saving model #%% dump(model, "svm_model_iris.joblib")
Output results and Interpretation	Fitting 5 folds for each of 6 candidates, totalling 30 fits {'C': 2.5, 'dual': 'auto'}

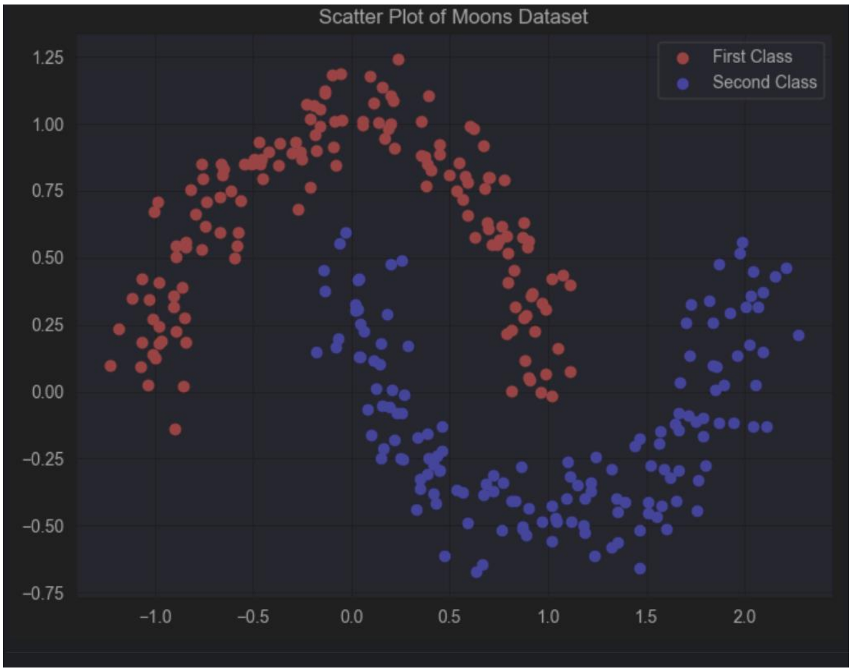
Step number	**4**
Step Name	Testing model
Step Description	Make predictions with test data, make confusion matrix, draw plot by two params, print classification report that includes different metrics
Code and Explanation	# Predictions #%% y_pred = model.predict(x_test)

	matrix = confusion_matrix(y_test, y_pred) #%% md # Create scatter plot #%% plt.figure(figsize=(8, 6)) for i in range(len(colors)): plt.scatter(x_test[y_pred == i, 0], x_test[y_pred == i, 1], c=colors[i], label=labels[i]) # Add labels and legend plt.xlabel('Sepal Length (cm)') plt.ylabel('Sepal Width (cm)') plt.title('Scatter Plot of Predictions of Iris Dataset') plt.legend() # Show plot plt.grid(True) plt.show() #%% md # Classification report #%% plt.figure(figsize=(7, 5)) sns.heatmap(matrix, annot=True) plt.xlabel("Predicted") plt.ylabel("Truth") #%% print(classification_report(y_test, y_pred))

Output results and Interpretation	 

	`` precision recall f1-score support 0 1.00 1.00 1.00 13 1 1.00 0.81 0.90 16 2 0.75 1.00 0.86 9 accuracy 0.92 38 macro avg 0.92 0.94 0.92 38 weighted avg 0.94 0.92 0.92 38 ```

Step number	**5**
Step Name	Looking at the moons dataset
Step Description	Getting moons dataset, checking shapes, drawing plot with full dataset
Code and Explanation	moons = make_moons(n_samples=300, noise=0.1, random_state=42) row_count = moons[0].shape[0] column_count = moons[0].shape[1] print(f'The DataFrme has {row_count} rows.') print(f'The DataFrame has {column_count} columns.') #%% md # Setting features and targets #%% x = moons[0] y = moons[1] #%% md # Defining colours #%% colors = ['red', 'blue'] labels = ['First Class', 'Second Class'] #%% md # Plotting full dataset

	#%% plt.figure(figsize=(8, 6)) for i in range(len(colors)): plt.scatter(x[y == i, 0], x[y == i, 1], c=colors[i], label=labels[i]) # Add labels and legend plt.title('Scatter Plot of Moons Dataset') plt.legend() # Show plot plt.grid(True) plt.show()
Output results and Interpretation	

Step number	**6**
Step Name	Training SVC model for moons dataset
Step Description	Splitting moons dataset into train and test, setting param grid, defining model, fitting and saving model
Code and Explanation	x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, shuffle=True)

<table>
<tr><td></td><td>

```
#%% md
# Setting param grid for the gridsearch (iterating hyperparams)
#%%
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'gamma': ['scale', 'auto'],    # gamma is relevant for 'rbf', 'poly', and 'sigmoid' kernels
    'degree': [2, 3, 4],   # degree is relevant for 'poly' kernel
}
#%% md
# Defining model
#%%
model = GridSearchCV(SVC(), param_grid, scoring='precision_macro', n_jobs=-1, verbose=3)
#%% md
# Training (fitting) model
#%%
model.fit(x_train, y_train)
model.best_params_
#%% md
# Saving model
#%%
dump(model, "svm_model_moons.joblib")
```

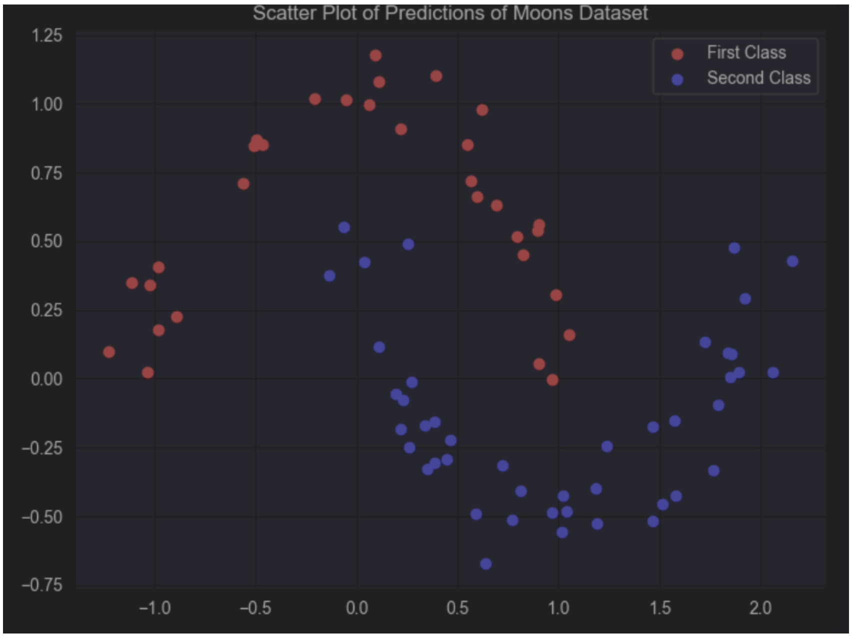
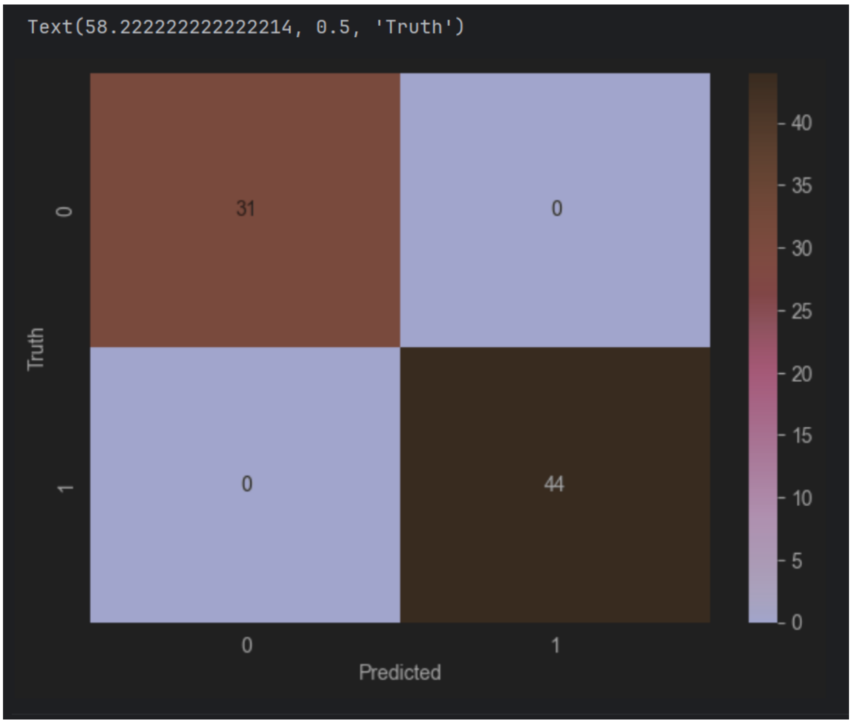
</td></tr>
<tr><td>Output results and Interpretation</td><td>

```
Fitting 5 folds for each of 96 candidates, totalling 480 fits
{'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}
```

</td></tr>
</table>

Step number	**7**

Step Name	Testing model
Step Description	Making predictions, building confusion matrix, making plot for testing data, printing classification report with different metrics
Code and Explanation	y_pred = model.predict(x_test) matrix = confusion_matrix(y_test, y_pred) #%% md # Plotting predicted data #%% plt.figure(figsize=(8, 6)) for i in range(len(colors)): plt.scatter(x_test[y_pred == i, 0], x_test[y_pred == i, 1], c=colors[i], label=labels[i]) # Add labels and legend plt.title('Scatter Plot of Predictions of Moons Dataset') plt.legend() # Show plot plt.grid(True) plt.show() #%% md # Classification report #%% plt.figure(figsize=(7, 5)) sns.heatmap(matrix, annot=True) plt.xlabel("Predicted") plt.ylabel("Truth") #%% print(classification_report(y_test, y_pred))

Output results and Interpretation	 

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        31
           1       1.00      1.00      1.00        44

    accuracy                           1.00        75
   macro avg       1.00      1.00      1.00        75
weighted avg       1.00      1.00      1.00        75
```

The metrics show us that the models have done their job perfectly

8 Experiment Difficulties and Precautions

9 Experiment Results and Interpretation

Iris dataset:

Metrics:

Precision: Precision measures the accuracy of positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives. In this case:

Class 0 (setosa) has a precision of 1.00, indicating that all instances classified as setosa are indeed setosa.

Class 1 (versicolor) has a precision of 1.00, indicating that all instances classified as versicolor are indeed versicolor.

Class 2 (virginica) has a precision of 0.75, indicating that 75% of instances classified as virginica are indeed virginica.

Recall: Recall measures the ability of the classifier to find all positive instances. It is the ratio of correctly predicted positive observations to the all observations in actual class. In this case:

Class 0 (setosa) has a recall of 1.00, indicating that all actual setosa instances were correctly classified as setosa.
Class 1 (versicolor) has a recall of 0.81, indicating that 81% of actual versicolor instances were correctly classified as versicolor.
Class 2 (virginica) has a recall of 1.00, indicating that all actual virginica instances were correctly classified as virginica.
F1-score: The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall. In this case:

Class 0 (setosa) has an F1-score of 1.00, indicating excellent precision and recall.
Class 1 (versicolor) has an F1-score of 0.90, indicating a good balance between precision and recall.
Class 2 (virginica) has an F1-score of 0.86, indicating a reasonably good balance between precision and recall.
Support: Support is the number of actual occurrences of the class in the specified dataset.

Accuracy: Accuracy measures the overall correctness of the classification model. It is the ratio of correctly predicted instances to the total instances. In this case, the overall accuracy is 0.92, indicating that the model correctly classified 92% of the instances.

Macro Avg and Weighted Avg: These are the averages of precision, recall, and F1-score across all classes. Macro Avg gives equal weight to each class, while Weighted Avg accounts for class imbalance by weighting each class's score by its support.

The results suggest that the classification model performs well overall, with high precision and recall for setosa and virginica classes, and slightly lower precision and recall for the versicolor class. The choice of hyperparameters for the Linear SVM model (C=2.5, dual='auto') seems to produce good results based on the evaluation metrics.

Moons dataset:

Metrics:

Precision: Perfect precision (1.00) for both classes (0 and 1) indicates that all instances classified as a particular class are indeed members of that class.

Recall: Perfect recall (1.00) for both classes (0 and 1) indicates that the classifier correctly identifies all instances of each class.

F1-score: The harmonic mean of precision and recall is also perfect (1.00) for both classes, indicating a perfect balance between precision and recall.

Support: The number of actual instances of each class in the dataset.

Accuracy: The overall accuracy of the classification model is 1.00, indicating that all instances are correctly classified.

Macro Avg and Weighted Avg: Both are perfect (1.00) since there is no class imbalance.

These results suggest that the SVM classifier with the given hyperparameters (C=10, degree=2, gamma='scale', kernel='rbf') performs extremely well on the moons dataset, achieving perfect classification accuracy with high precision and recall for both classes.

10 References

11 Experiment-related Metadata

Metadata Item	Content
Case name	
Applicable course name	Machine learning Fundamentals
Keyword/Search Term	Iris, Moons, SVM, SVC
AliTianchi URI	

12 Remarks and Others