Project Code in jupyter

```
import numpy as np

import pandas as pd

import sklearn

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

from sklearn.datasets import load_iris

import sklearn.metrics as metrics
```

01 Use the iris dataset

```
from sklearn import datasets

iris = datasets.load_iris()
```

```
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Project Code in jupyter

```
from sklearn.model_selection import train_test_split

X= data[['sepal length', 'sepal width', 'petal length', 'petal width']]  # Features
y= data['species']  # Labels

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

02 Using SVM and Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

svm_clf=SVC()
rf_clf=RandomForestClassifier(n_estimators=100)

svm_clf.fit(X_train,y_train)
rf_clf.fit(X_train,y_train)

y_pred_svm=svm_clf.predict(X_test)
y_pred_rf=rf_clf.predict(X_test)
```

```
from sklearn import metrics

print("Accuracy of SVM:",metrics.accuracy_score(y_test, y_pred_svm))
print("Accuracy of Random Forest:",metrics.accuracy_score(y_test, y_pred_rf))
```

```
Accuracy of SVM: 0.9777777777777777
Accuracy of Random Forest: 0.9777777777777777
```

03 Using grid search with cross-validation to select best parameters

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.metrics import recall_score

# Create a dictionary called param_grid and fill out some parameters for C and Gamma
```

Project Code in jupyter

```
param_grid = {'C':[0.1,1,10,100], 'gamma':[1,0.1,0.01,0.001]}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose=3)

scoring = ['precision_macro', 'recall_macro']
scores = cross_validate(grid, X, y, scoring=scoring)
```

```
Output exceeds the size limit. Open the full output data in a text editor
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV 1/5] END .....................C=0.1, gamma=1;, score=1.000 total time=   0.0s
[CV 2/5] END .....................C=0.1, gamma=1;, score=0.917 total time=   0.0s
[CV 3/5] END .....................C=0.1, gamma=1;, score=0.917 total time=   0.0s
[CV 4/5] END .....................C=0.1, gamma=1;, score=1.000 total time=   0.0s
[CV 5/5] END .....................C=0.1, gamma=1;, score=1.000 total time=   0.0s
[CV 1/5] END ...................C=0.1, gamma=0.1;, score=0.958 total time=   0.0s
[CV 2/5] END ...................C=0.1, gamma=0.1;, score=0.875 total time=   0.0s
[CV 3/5] END ...................C=0.1, gamma=0.1;, score=0.833 total time=   0.0s
[CV 4/5] END ...................C=0.1, gamma=0.1;, score=0.917 total time=   0.0s
[CV 5/5] END ...................C=0.1, gamma=0.1;, score=0.917 total time=   0.0s
[CV 1/5] END ..................C=0.1, gamma=0.01;, score=0.958 total time=   0.0s
[CV 2/5] END ..................C=0.1, gamma=0.01;, score=0.917 total time=   0.0s
[CV 3/5] END ..................C=0.1, gamma=0.01;, score=0.875 total time=   0.0s
[CV 4/5] END ..................C=0.1, gamma=0.01;, score=0.958 total time=   0.0s
[CV 5/5] END ..................C=0.1, gamma=0.01;, score=0.875 total time=   0.0s
[CV 1/5] END ................C=0.1, gamma=0.001;, score=0.958 total time=   0.0s
[CV 2/5] END ................C=0.1, gamma=0.001;, score=0.875 total time=   0.0s
[CV 3/5] END ................C=0.1, gamma=0.001;, score=0.875 total time=   0.0s
[CV 4/5] END ................C=0.1, gamma=0.001;, score=0.958 total time=   0.0s
[CV 5/5] END ................C=0.1, gamma=0.001;, score=0.875 total time=   0.0s
[CV 1/5] END .......................C=1, gamma=1;, score=1.000 total time=   0.0s
[CV 2/5] END .......................C=1, gamma=1;, score=0.958 total time=   0.0s
[CV 3/5] END .......................C=1, gamma=1;, score=0.917 total time=   0.0s
[CV 4/5] END .......................C=1, gamma=1;, score=1.000 total time=   0.0s
...
[CV 2/5] END ................C=100, gamma=0.001;, score=1.000 total time=   0.0s
[CV 3/5] END ................C=100, gamma=0.001;, score=1.000 total time=   0.0s
[CV 4/5] END ................C=100, gamma=0.001;, score=0.958 total time=   0.0s
[CV 5/5] END ................C=100, gamma=0.001;, score=0.917 total time=   0.0s
```

```
sorted(scores.keys())
```

```
['fit_time', 'score_time', 'test_precision_macro', 'test_recall_macro']
```

Project Code in jupyter

```
scores['test_recall_macro']
```

```
array([0.96666667, 1.        , 0.96666667, 0.96666667, 1.        ])
```

```
pred_grid = grid.predict(X_test)
print("Accuracy of Grid Search:",metrics.accuracy_score(y_test, pred_grid))
```

```
Accuracy of Grid Search: 0.8888888888888888
```

04 Using two matrix (Confusion_matrix and F1-score)

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
# for SVM
print(confusion_matrix(y_test, y_pred_svm))
print(f1_score(y_test, y_pred_svm, average=None))
```

```
[[13  0  0]
 [ 0 12  0]
 [ 0  6 14]]
[1.         0.8        0.82352941]
```

```
# for Random Forest
print(confusion_matrix(y_test, y_pred_rf))
print(f1_score(y_test, y_pred_rf, average=None))
```

```
[[13  0  0]
 [ 0 11  1]
 [ 0  2 18]]
[1.         0.88       0.92307692]
```

```
# for Grid Search
print(confusion_matrix(y_test, pred_grid))
print(f1_score(y_test, pred_grid, average=None))
```

Project Code in jupyter

```
[[13  0  0]
 [ 0 12  0]
 [ 0  5 15]]
[1.         0.82758621 0.85714286]
```