In [1]:

```
%matplotlib notebook
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits
dataset = load_digits()
X, y = dataset.data, dataset.target
```

```
C:₩Users₩donghyunkim₩anaconda3₩lib₩importlib₩_bootstrap.py:219: RuntimeWarning: nu
mpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C h
eader, got 216 from PyObject
  return f(*args, **kwds)
C:₩Users₩donghyunkim₩anaconda3₩lib₩importlib₩_bootstrap.py:219: RuntimeWarning: nu
mpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C h
eader, got 216 from PyObject
  return f(*args, **kwds)
```

In [10]:

```
y_binary_imbalanced=y.copy()
y_binary_imbalanced[y_binary_imbalanced!=1]=0
```

In [ ]:

```
#SVM
```

In [11]:

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y_binary_imbalanced,
                                                  random_state=0)
svm=SVC().fit(X_train,y_train)
```

In [12]:

```
from sklearn.metrics import confusion_matrix
y_svm_predicted=svm.predict(X_test)
confusion=confusion_matrix(y_test,y_svm_predicted)
print(confusion)
```

```
[[407   0]
 [  2  41]]
```

In [13]:

```
from sklearn.metrics import precision_score,recall_score
print('Precision:{:.2f}'.format(precision_score(y_test,y_svm_predicted)))
print('Recall:{:.2f}'.format(recall_score(y_test,y_svm_predicted)))
```

```
Precision:1.00
Recall:0.95
```

In [ ]:

```
#LR
```

In [14]:

```
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test=train_test_split(X,y_binary_imbalanced,
                                                   random_state=0)
clf=LogisticRegression().fit(X_train,y_train)
```

```
C:₩Users₩donghyunkim₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.p
y:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [15]:

```
y_logreg_predicted = clf.predict(X_test)
confusion_logreg=confusion_matrix(y_test,y_logreg_predicted)
print(confusion_logreg)
```

```
[[401   6]
 [  8  35]]
```

In [16]:

```
print('Precision:{:.2f}'.format(precision_score(y_test,y_logreg_predicted)))
print('Recall:{:.2f}'.format(recall_score(y_test,y_logreg_predicted)))
```

```
Precision:0.85
Recall:0.81
```

In [ ]:

```
#DT
```

In [20]:

```
from sklearn.tree import DecisionTreeClassifier
X_train, X_test, y_train, y_test=train_test_split(X,y_binary_imbalanced,
                                                   random_state=0)
dt=DecisionTreeClassifier().fit(X_train,y_train)
```

In [21]:

```
y_dt_predicted = dt.predict(X_test)
confusion_dt=confusion_matrix(y_test,y_dt_predicted)
print(confusion_dt)
```

```
[[399   8]
 [  8  35]]
```

In [22]:

```
print('Precision:{:.2f}'.format(precision_score(y_test,y_dt_predicted)))
print('Recall:{:.2f}'.format(recall_score(y_test,y_dt_predicted)))
```

```
Precision:0.81
Recall:0.81
```

In [ ]:

```
##Random Forest
```

In [24]:

```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test=train_test_split(X,y_binary_imbalanced,
                                                  random_state=0)
rf=RandomForestClassifier(n_estimators=12,random_state=0).fit(X_train,y_train)
```

In [25]:

```
y_rf_predicted = rf.predict(X_test)
confusion_dt=confusion_matrix(y_test,y_rf_predicted)
print(confusion_dt)
```

```
[[407   0]
 [  6  37]]
```

In [26]:

```
print('Precision:{:.2f}'.format(precision_score(y_test,y_rf_predicted)))
print('Recall:{:.2f}'.format(recall_score(y_test,y_rf_predicted)))
```

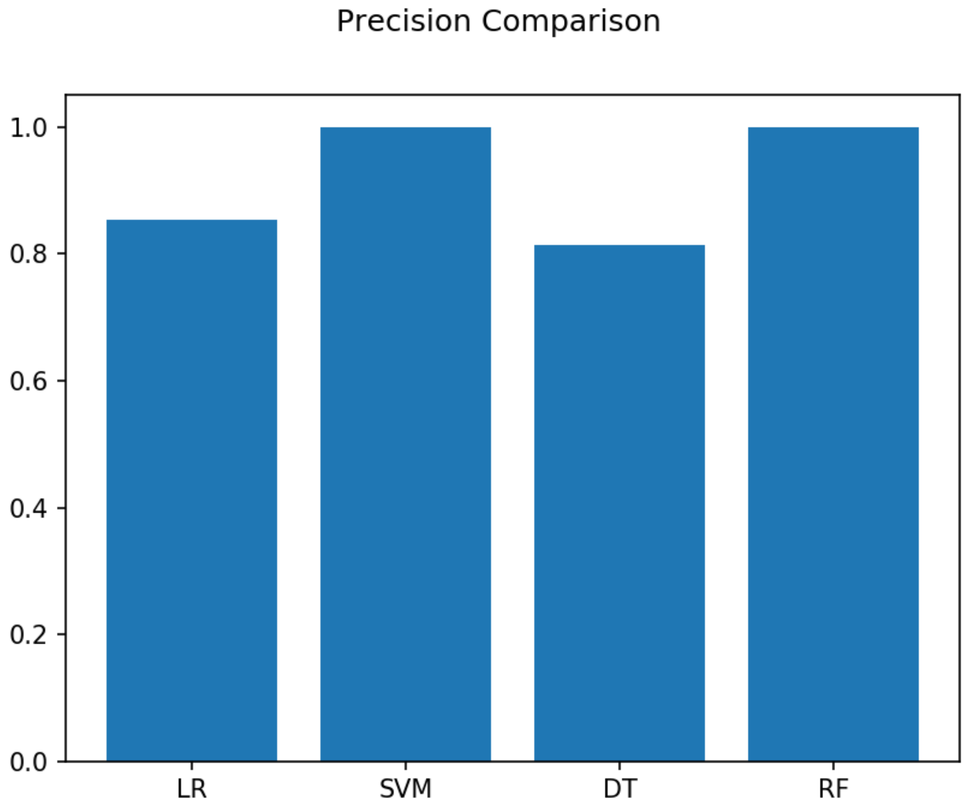
```
Precision:1.00
Recall:0.86
```

In [30]:

```
names=['LR','SVM','DT','RF']
precision_models=[]
precision_models.append(precision_score(y_test,y_logreg_predicted))
precision_models.append(precision_score(y_test,y_svm_predicted))
precision_models.append(precision_score(y_test,y_dt_predicted))
precision_models.append(precision_score(y_test,y_rf_predicted))
recall_models=[]
recall_models.append(recall_score(y_test,y_logreg_predicted))
recall_models.append(recall_score(y_test,y_svm_predicted))
recall_models.append(recall_score(y_test,y_dt_predicted))
recall_models.append(recall_score(y_test,y_rf_predicted))
```

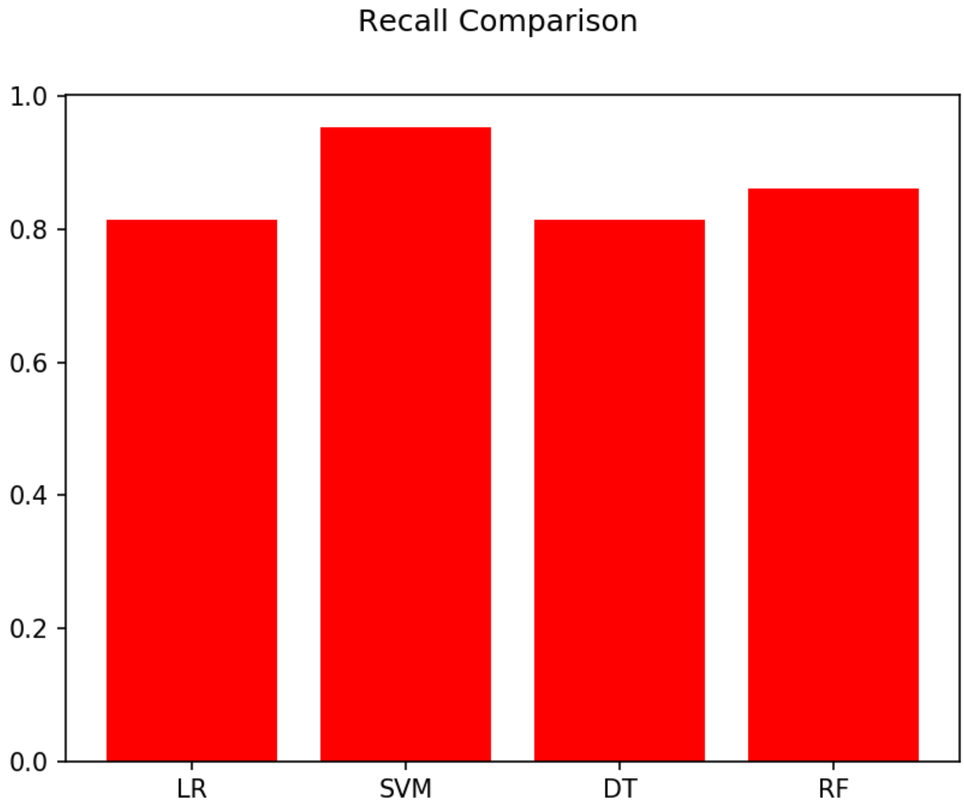
In [44]:

```
fig=plt.figure()
fig.suptitle('Precision Comparison')
plt.bar(names,precision_models)
plt.show()
```



In [45]:

```
fig=plt.figure()
fig.suptitle('Recall Comparison')
plt.bar(names,recall_models,color='red')
plt.show()
```



In [ ]:

```
#Precision은 Spetor 벡터 머신, RandomForest에서 가장 높고
#Recall은 Spetor 벡터 머신,RandomForest에서 가장 높다
```