Salah Mohamed

TP3:Digits Classification

In [3]:

```
pip install scikit-plot
```

```
Requirement already satisfied: scikit-plot in c:\users\hed\an
aconda3\lib\site-packages (0.3.7)
Requirement already satisfied: scipy>=0.9 in c:\users\hed\ana
conda3\lib\site-packages (from scikit-plot) (1.1.0)
Requirement already satisfied: scikit-learn>=0.18 in c:\users
\hed\anaconda3\lib\site-packages (from scikit-plot) (0.20.1)
Requirement already satisfied: matplotlib>=1.4.0 in c:\users
\hed\anaconda3\lib\site-packages (from scikit-plot) (3.1.0)
Requirement already satisfied: joblib>=0.10 in c:\users\hed\a
naconda3\lib\site-packages (from scikit-plot) (0.13.2)
Requirement already satisfied: numpy>=1.8.2 in c:\users\hed\a
naconda3\lib\site-packages (from scikit-learn>=0.18->scikit-p
lot) (1.15.4)
Requirement already satisfied: cycler>=0.10 in c:\users\hed\a
naconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-pl
ot) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users
\hed\anaconda3\lib\site-packages (from matplotlib>=1.4.0->sci
kit-plot) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.
1.6,>=2.0.1 in c:\users\hed\anaconda3\lib\site-packages (from
matplotlib>=1.4.0->scikit-plot) (2.4.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\use
rs\hed\anaconda3\lib\site-packages (from matplotlib>=1.4.0->s
cikit-plot) (2.8.0)
Requirement already satisfied: six in c:\users\hed\anaconda3
\lib\site-packages (from cycler>=0.10->matplotlib>=1.4.0->sci
kit-plot) (1.12.0)
Requirement already satisfied: setuptools in c:\users\hed\ana
conda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>
=1.4.0->scikit-plot) (41.0.1)
Note: you may need to restart the kernel to use updated packa
ges.
```

In [4]:

```python
from sklearn import datasets, neighbors, linear_model
from sklearn.preprocessing import label_binarize
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
from sklearn.metrics import roc_auc_score
from itertools import cycle
```

```
import numpy as np
%matplotlib inline
```

In [5]:

```
X_digits, y_digits = datasets.load_digits(return_X_y=True)
```

In [6]:

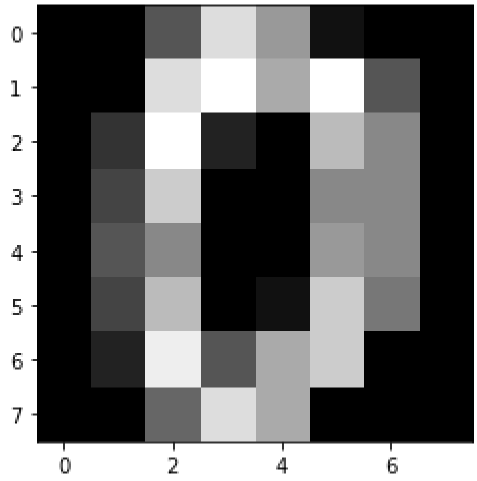
```
print(X_digits.shape)
print(y_digits.shape)
y = label_binarize(y_digits, classes=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
n_classes = y.shape[1]
print(y.shape)
print(n_classes)
```

```
(1797, 64)
(1797,)
(1797, 10)
10
```

Display the first image

In [8]:

```
digit_image=X_digits[0,:].reshape(8,8)
plt.imshow(digit_image,cmap="gray")
plt.show()
```



In [9]:

```
n_samples = len(X_digits)
print(n_samples)
n_samples=X_digits.shape
print(n_samples)
```

```
1797
(1797, 64)
```

In [24]:

```python
from sklearn.model_selection import train_test_split
random_state = np.random.RandomState(0)
digits = datasets.load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.ta
rget)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(1347, 64)
(1347,)
(450, 64)
(450,)
```

In [25]:

```python
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier

print(OneVsRestClassifier(DecisionTreeClassifier()).get_params().keys())
print(OneVsRestClassifier(RandomForestClassifier()).get_params().keys())
print(OneVsRestClassifier(KNeighborsClassifier()).get_params().keys())
print(OneVsRestClassifier(LogisticRegression()).get_params().keys())
print(OneVsRestClassifier(GradientBoostingClassifier()).get_params().keys
())
print(OneVsRestClassifier(SVC()).get_params().keys())
print(OneVsRestClassifier(RidgeClassifier()).get_params().keys())
print(OneVsRestClassifier(GaussianNB()).get_params().keys())
print(OneVsRestClassifier(XGBClassifier()).get_params().keys())
```

```
dict_keys(['estimator__class_weight', 'estimator__criterion',
'estimator__max_depth', 'estimator__max_features', 'estimator
__max_leaf_nodes', 'estimator__min_impurity_decrease', 'estim
ator__min_impurity_split', 'estimator__min_samples_leaf', 'es
timator__min_samples_split', 'estimator__min_weight_fraction_
leaf', 'estimator__presort', 'estimator__random_state', 'esti
mator__splitter', 'estimator', 'n_jobs'])
dict_keys(['estimator__bootstrap', 'estimator__class_weight',
'estimator__criterion', 'estimator__max_depth', 'estimator__m
ax_features', 'estimator__max_leaf_nodes', 'estimator__min_im
purity_decrease', 'estimator__min_impurity_split', 'estimator
__min_samples_leaf', 'estimator__min_samples_split', 'estimat
or__min_weight_fraction_leaf', 'estimator__n_estimators', 'es
timator__n_jobs', 'estimator__oob_score', 'estimator__random_
state', 'estimator__verbose', 'estimator__warm_start', 'estim
ator', 'n_jobs'])
dict_keys(['estimator__algorithm', 'estimator__leaf_size', 'e
stimator__metric', 'estimator__metric_params', 'estimator__n_
jobs', 'estimator__n_neighbors', 'estimator__p', 'estimator__
jobs', 'estimator__n_neighbors', 'estimator__p', 'estimator__
weights', 'estimator', 'n_jobs'])
dict_keys(['estimator__C', 'estimator__class_weight', 'estima
tor__dual', 'estimator__fit_intercept', 'estimator__intercept
_scaling', 'estimator__max_iter', 'estimator__multi_class',
'estimator__n_jobs', 'estimator__penalty', 'estimator__random
_state', 'estimator__solver', 'estimator__tol', 'estimator__v
erbose', 'estimator__warm_start', 'estimator', 'n_jobs'])
dict_keys(['estimator__criterion', 'estimator__init', 'estima
tor__learning_rate', 'estimator__loss', 'estimator__max_dept
h', 'estimator__max_features', 'estimator__max_leaf_nodes',
'estimator__min_impurity_decrease', 'estimator__min_impurity_
split', 'estimator__min_samples_leaf', 'estimator__min_sample
s_split', 'estimator__min_weight_fraction_leaf', 'estimator__
n_estimators', 'estimator__n_iter_no_change', 'estimator__pre
sort', 'estimator__random_state', 'estimator__subsample', 'es
timator__tol', 'estimator__validation_fraction', 'estimator__
verbose', 'estimator__warm_start', 'estimator', 'n_jobs'])
dict_keys(['estimator__C', 'estimator__cache_size', 'estimato
r__class_weight', 'estimator__coef0', 'estimator__decision_fu
nction_shape', 'estimator__degree', 'estimator__gamma', 'esti
mator__kernel', 'estimator__max_iter', 'estimator__probabilit
y', 'estimator__random_state', 'estimator__shrinking', 'estim
ator__tol', 'estimator__verbose', 'estimator', 'n_jobs'])
dict_keys(['estimator__alpha', 'estimator__class_weight', 'es
timator__copy_X', 'estimator__fit_intercept', 'estimator__max
_iter', 'estimator__normalize', 'estimator__random_state', 'e
stimator__solver', 'estimator__tol', 'estimator', 'n_jobs'])
dict_keys(['estimator__priors', 'estimator__var_smoothing',
'estimator', 'n_jobs'])
dict_keys(['estimator__objective', 'estimator__use_label_enco
der', 'estimator__base_score', 'estimator__booster', 'estimat
or__colsample_bylevel', 'estimator__colsample_bynode', 'estim
ator__colsample_bytree', 'estimator__gamma', 'estimator__gpu_
id', 'estimator__importance_type', 'estimator__interaction_co
nstraints', 'estimator__learning_rate', 'estimator__max_delta
_step', 'estimator__max_depth', 'estimator__min_child_weigh
t', 'estimator__missing', 'estimator__monotone_constraints',
'estimator__n_estimators', 'estimator__n_jobs', 'estimator__n
um_parallel_tree', 'estimator__random_state', 'estimator__reg
_alpha', 'estimator__reg_lambda', 'estimator__scale_pos_weigh
t', 'estimator__subsample', 'estimator__tree_method', 'estima
tor__validate_parameters', 'estimator__verbosity', 'estimato
r', 'n_jobs'])
```

In [26]:

```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
knn = neighbors.KNeighborsClassifier()
logistic = OneVsRestClassifier(linear_model.LogisticRegression(max_iter=10
00))
naive_bayes= OneVsRestClassifier(GaussianNB())
svmm = OneVsRestClassifier(SVC())
Des_tree=OneVsRestClassifier(DecisionTreeClassifier(max_depth=5))
Ran_forest=RandomForestClassifier(max_depth=5)
Gradient_boost = OneVsRestClassifier(GradientBoostingClassifier())
xgboost=OneVsRestClassifier(XGBClassifier())
AdaBoost=OneVsRestClassifier(AdaBoostClassifier())
```

```python
print('KNN score: %f' % knn.fit(X_train, y_train).score(X_test, y_test))
print('LogisticRegression score: %f' % logistic.fit(X_train, y_train).score(X_test, y_test))
print('Naive Byes score: %f' % naive_bayes.fit(X_train, y_train).score(X_test, y_test))
print('SVM: %f'  % svmm.fit(X_train, y_train).score(X_test, y_test))
print('GradientBoosting: %f' % Gradient_boost.fit(X_train, y_train).score(X_test, y_test))
print('DecisionTreeClassifier: %f' %Des_tree.fit(X_train, y_train).score(X_test, y_test))
print('RandomForestClassifier: %f'%Ran_forest.fit(X_train, y_train).score(X_test, y_test))
print('XGboost: %f'%xgboost.fit(X_train, y_train).score(X_test, y_test))
# Model Selection : The best model with highest performances: accuracy
# confusion matrix, ROC curves, AUC-ROC, Precision, Recall,
# F-score, PR curves, AUC-PR

# Deploy the best model

# OPTIONAL : develop a web image for selecting an image from a local path
# display the image and its number (returned by the backend)
```

```
KNN score: 0.991111
LogisticRegression score: 0.984444
Naive Byes score: 0.531111
SVM: 0.724444
GradientBoosting: 0.971111
DecisionTreeClassifier: 0.864444
RandomForestClassifier: 0.900000
```

```
C:\Users\hed\Anaconda3\lib\site-packages\xgboost\sklearn.py:8
88: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_enco
der=False when constructing XGBClassifier object; and 2) Enco
de your labels (y) as integers starting with 0, i.e. 0, 1, 2,


..., [num_class - 1].
```

```
[21:04:44] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:04:47] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:04:56] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:04:57] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
```

```
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:05:01] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:05:04] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:05:07] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:05:09] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:05:13] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
[21:05:16] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Expl
icitly set eval_metric if you'd like to restore the old behav
ior.
XGboost: 0.984444
```

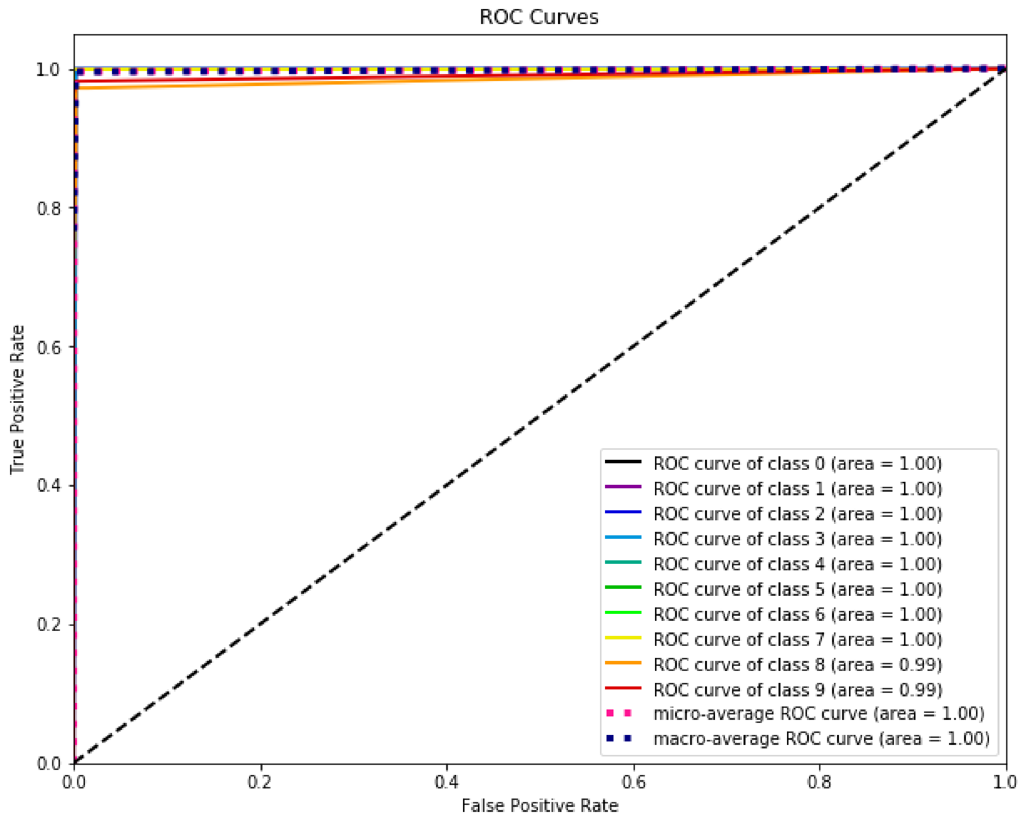
In [28]:

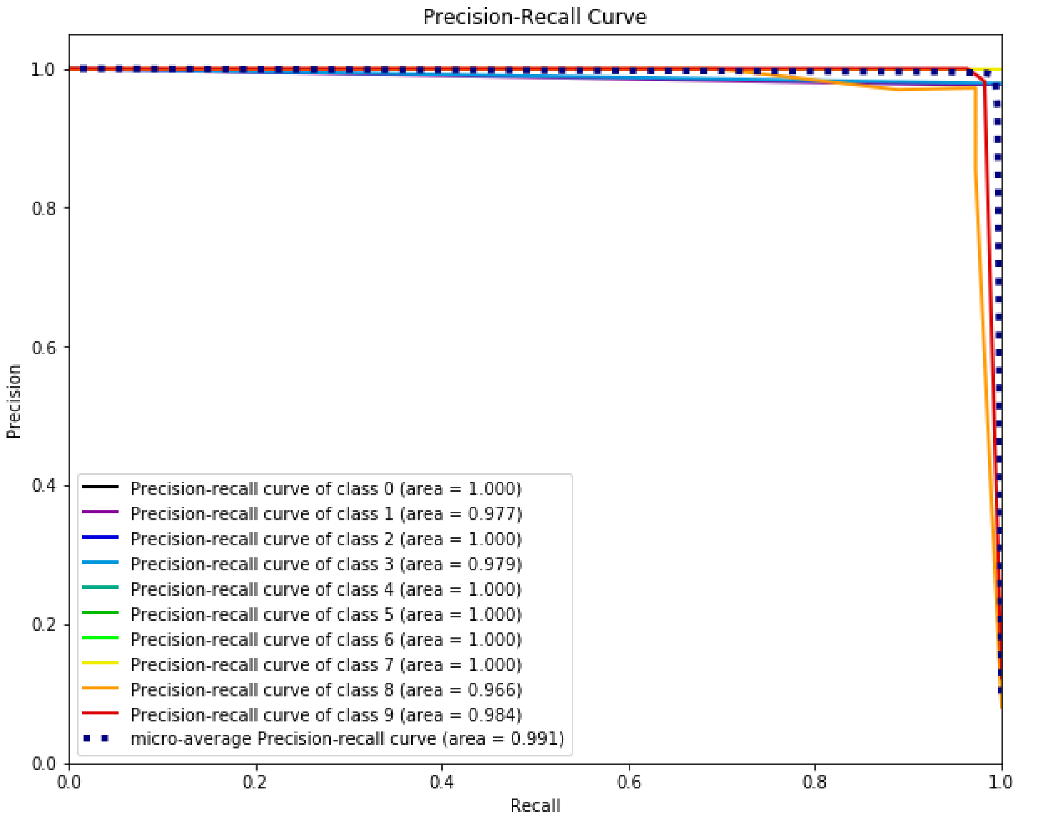
```
# Add other classifiers : Naive Bayes, SVM, Decision Tree
# Random Forest, Gradient Boosting and XGBoost using piplines of models
from sklearn import svm
from sklearn.pipeline import Pipeline
import xgboost as xgb
knn = neighbors.KNeighborsClassifier()
logistic = LogisticRegression()#max_iter=1000
classifier = svm.SVC()#gamma=0.001
clf = RandomForestClassifier()#max_depth=2, random_state=0
disgn = DecisionTreeClassifier()#random_state=0
xgb_model = xgb.XGBClassifier()#objective="binary:logistic", random_state=
42
pipelines = []
params = []
names = []
pipelines.append(Pipeline([('clf',neighbors.KNeighborsClassifier())]))
pipelines.append(Pipeline([('clf', LogisticRegression())]))
pipelines.append(Pipeline([('clf', svm.SVC())]))
pipelines.append(Pipeline([('clf', RandomForestClassifier())]))
pipelines.append(Pipeline([('clf', DecisionTreeClassifier())]))
pipelines.append(Pipeline([('clf',xgb.XGBClassifier())]))

# Using GridSearchCV with Kfolds = 5 to fine-tuning hyper
# and to find each best model of piplines
from sklearn.model_selection import KFold, GridSearchCV, cross_val_score
print('KNN score: %f' % knn.fit(X_train, y_train).score(X_test, y_test))
print('LogisticRegression score: %f'
      % logistic.fit(X_train, y_train).score(X_test, y_test))

params.append({'clf__n_neighbors':[5]})
names.append('KNeighborsClassifier')
params.append({'clf__penalty':['l2']})
names.append('LogisticRegression')
params.append({'clf__gamma':[0.001]})
names.append('SVC')
params.append({'clf__n_estimators': [50,100,200]})
names.append('RandomForestRegressor')
params.append({'clf__max_depth':np.linspace(5, 15, 5)})
names.append('DecisionTreeRegressor')
params.append({'clf__random_state':[42]})
names.append('XGBClassifier')
# Using GridSearchCV with Kfolds = 5 to fine-tuning hyperparameters
# and to find each best model of piplines
from sklearn.model_selection import KFold, GridSearchCV, cross_val_score

def model(pipeline, parameters, name, X, y):
    cv = KFold(n_splits=5, shuffle=True, random_state=32)
    grid_obj = GridSearchCV(estimator=pipeline, param_grid=parameters, cv=
cv, scoring='accuracy', n_jobs=-1)
    grid_obj.fit(X,y)
    print(name, 'Accuracy:', grid_obj.best_score_,'Best Parametre',grid_ob
j.best_params_)
    estimator = grid_obj.best_estimator_
    estimator.fit(X,y) # training sur tout training dataset
    return estimator
estimators = []
for i in range(len(pipelines)):
    estimators.append(model(pipelines[i], params[i], names[i], X_train, y_
train))

# Model Selection : The best model with highest performances: accuracy
# confusion matrix, ROC curves, AUC-ROC, Precision, Recall,
# F-score, PR curves, AUC-PR
    knn.fit(X_train,y_train)
    y_pred = knn.predict(X_test)
    from sklearn.metrics import recall_score
    recall_score(y_test, y_pred, pos_label='positive',average='micro')
    from sklearn.metrics import precision_score
    precision_score(y_test, y_pred,pos_label='positive',average='micro')
    from sklearn.metrics import f1_score
    f1_score(y_test, y_pred,pos_label='positive',average='micro')

    import scikitplot as skplt
    y_probas=knn.predict_proba(X_test)
    skplt.metrics.plot_roc(y_test, y_probas, figsize=(10, 8))
    #import scikitplot.plotters as skplt
    #skplt.plot_roc_curve(y_test, y_probas,figsize=(10, 8))
    #plt.show()
```

```
import scikitplot.plotters as skplt
skplt.plot_precision_recall_curve(y_test, y_probas,figsize=(10, 8))
plt.show()
```

```
KNN score: 0.991111
LogisticRegression score: 0.984444
KNeighborsClassifier Accuracy: 0.9829250185597624 Best Parame
tre {'clf__n_neighbors': 5}
```

```
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\metrics\clas
sification.py:1052: UserWarning: Note that pos_label (set to
'positive') is ignored when average != 'binary' (got 'micr
o'). You may use labels=[pos_label] to specify a single posit
ive class.
  % (pos_label, average), UserWarning)
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:77: DeprecationWarning: Function plot_precision_reca
ll_curve is deprecated; This will be removed in v0.4.0. Pleas
e use scikitplot.metrics.plot_precision_recall_curve instead.
  warnings.warn(msg, category=DeprecationWarning)
```

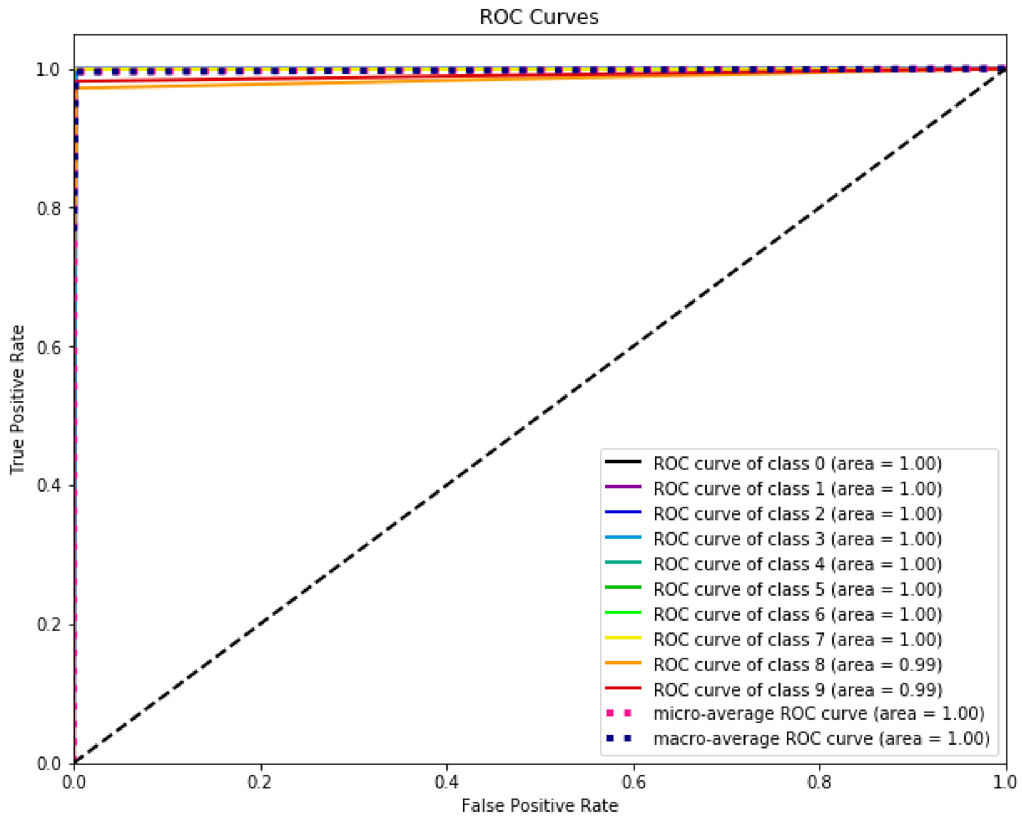


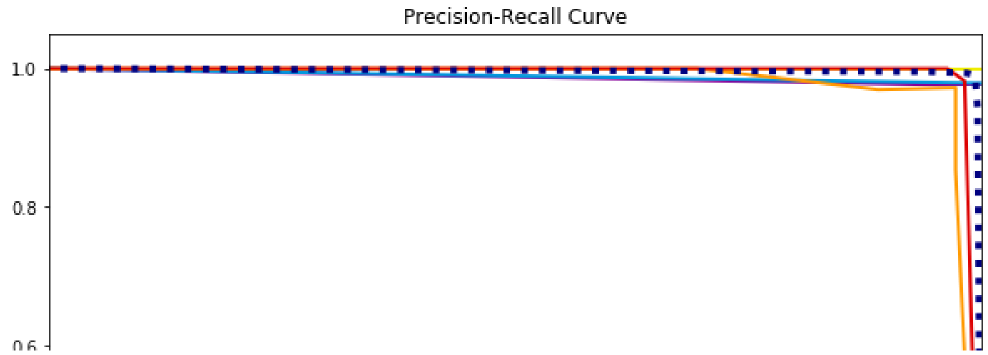


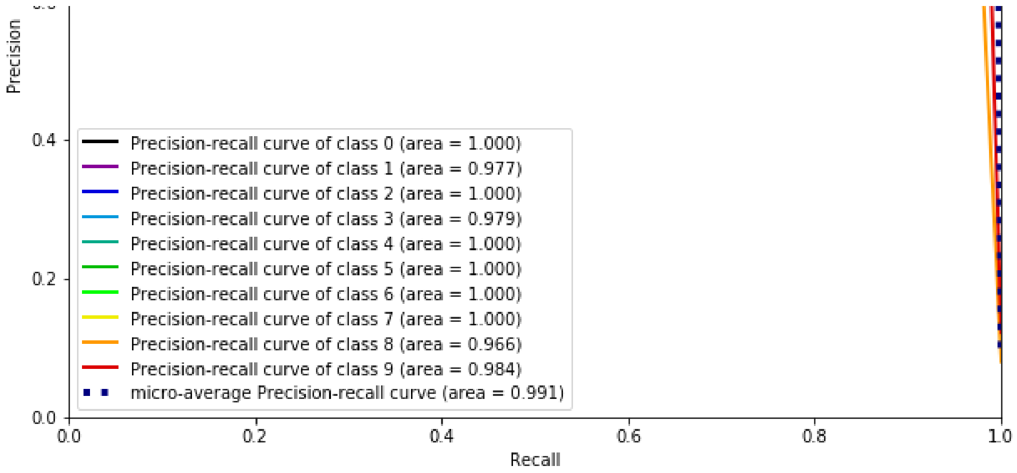
```
LogisticRegression Accuracy: 0.9517446176688938 Best Parametr
e {'clf__penalty': 'l2'}
```

```
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\metrics\clas
sification.py:1052: UserWarning: Note that pos_label (set to
'positive') is ignored when average != 'binary' (got 'micr
o'). You may use labels=[pos_label] to specify a single posit
ive class.
  % (pos_label, average), UserWarning)
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\utils\deprec
```

```
ation.py:77: DeprecationWarning: Function plot_precision_reca
ll_curve is deprecated; This will be removed in v0.4.0. Pleas
e use scikitplot.metrics.plot_precision_recall_curve instead.
  warnings.warn(msg, category=DeprecationWarning)
```

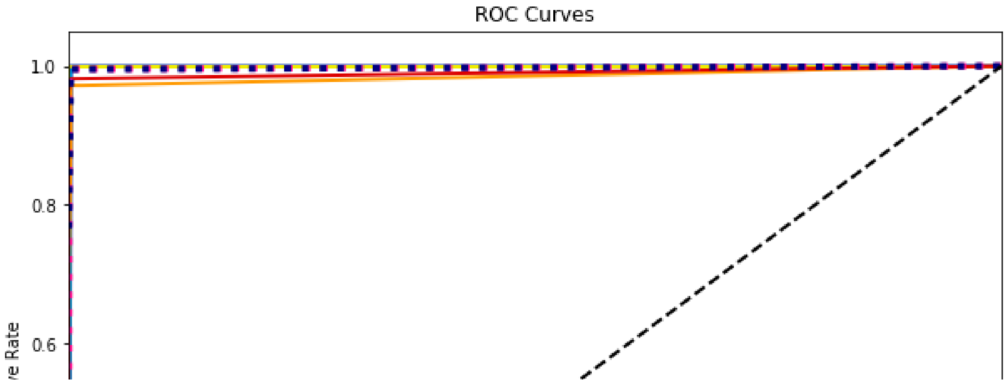


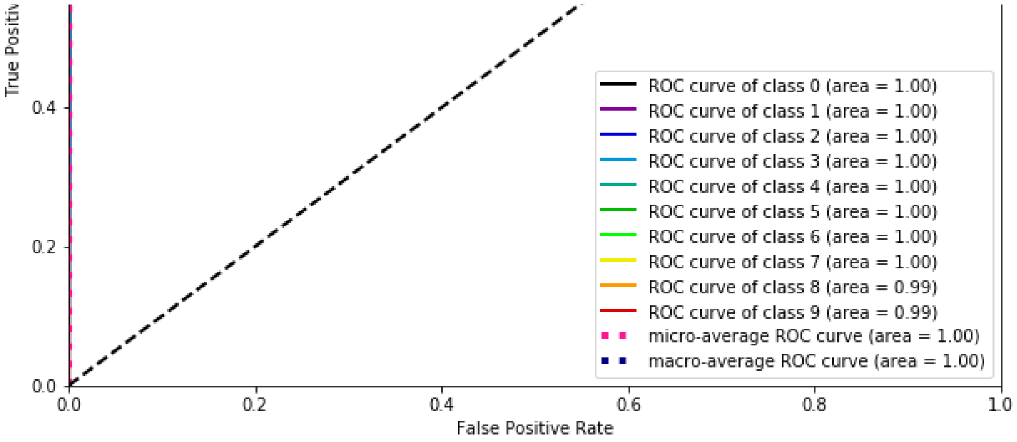


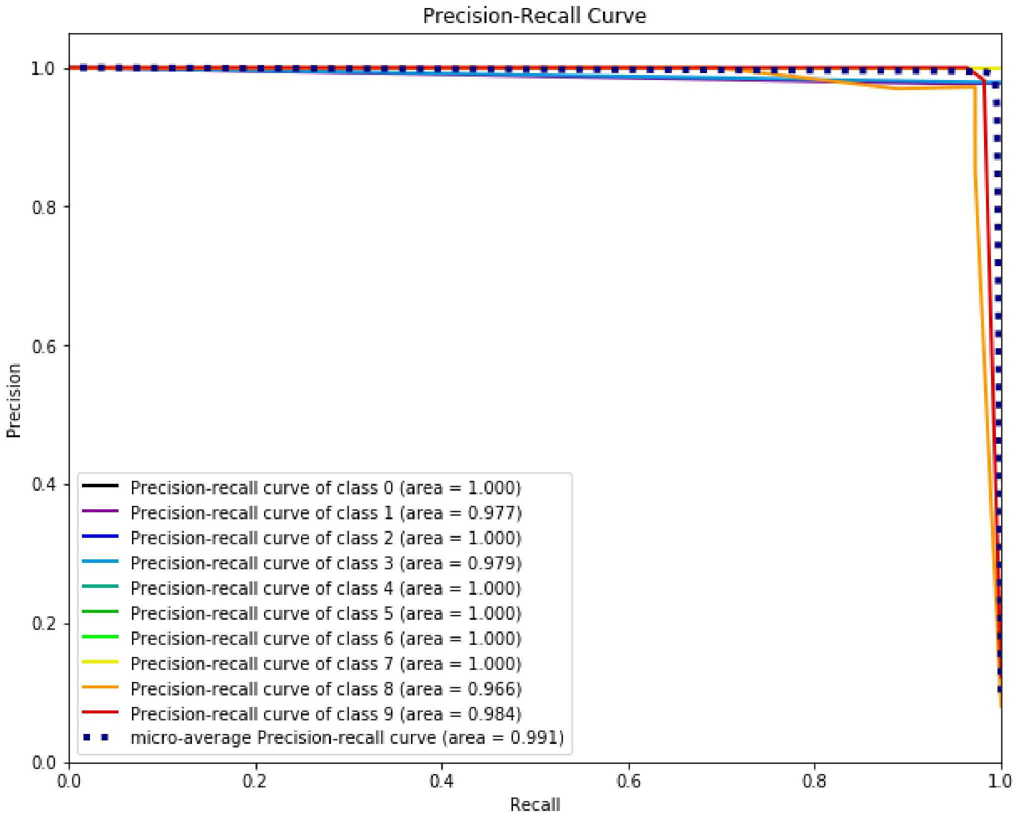


```
SVC Accuracy: 0.9866369710467706 Best Parametre {'clf__gamm
a': 0.001}
```

```
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\metrics\clas
sification.py:1052: UserWarning: Note that pos_label (set to
'positive') is ignored when average != 'binary' (got 'micr
o'). You may use labels=[pos_label] to specify a single posit
ive class.
  % (pos_label, average), UserWarning)
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:77: DeprecationWarning: Function plot_precision_reca
ll_curve is deprecated; This will be removed in v0.4.0. Pleas
e use scikitplot.metrics.plot_precision_recall_curve instead.
  warnings.warn(msg, category=DeprecationWarning)
```

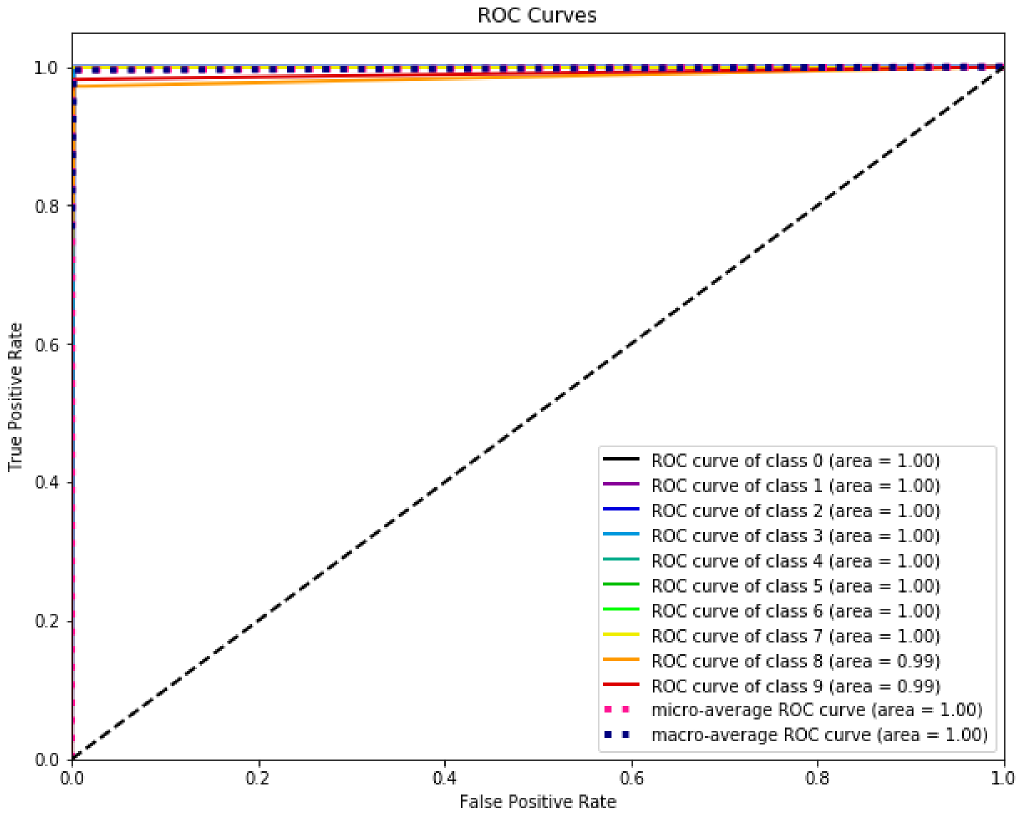


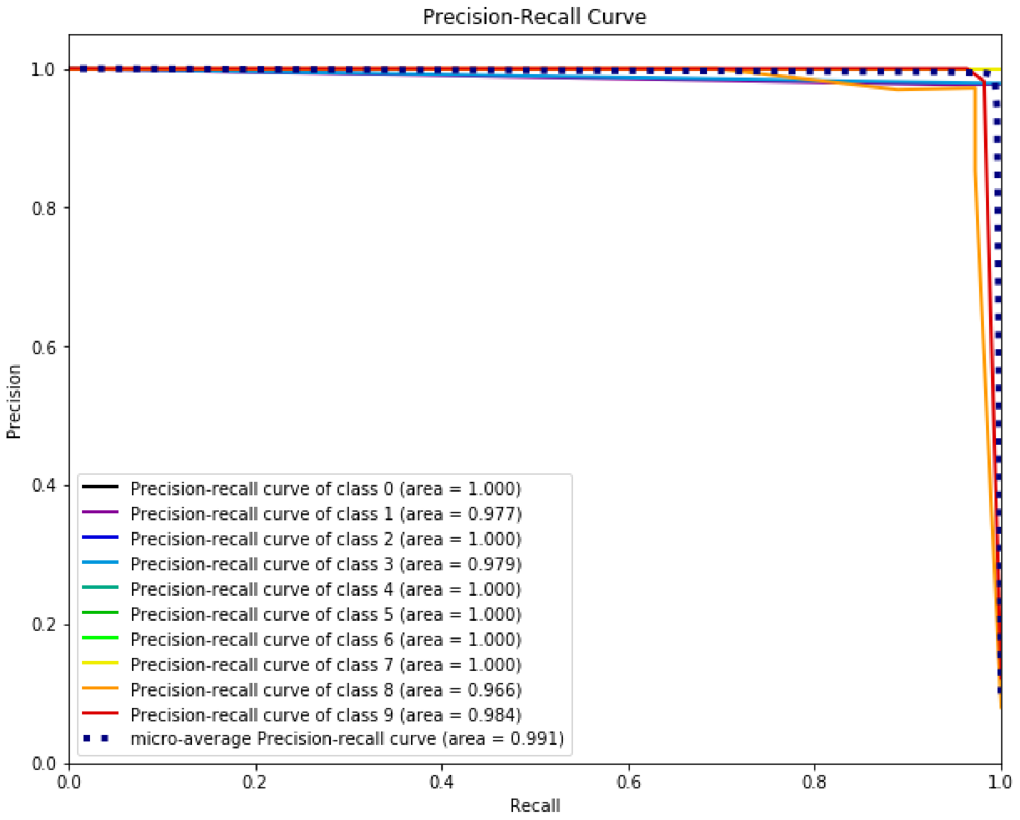

True Positiv
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 1.00)
ROC curve of class 2 (area = 1.00)
ROC curve of class 3 (area = 1.00)
ROC curve of class 4 (area = 1.00)
ROC curve of class 5 (area = 1.00)
ROC curve of class 6 (area = 1.00)
ROC curve of class 7 (area = 1.00)
ROC curve of class 8 (area = 0.99)
ROC curve of class 9 (area = 0.99)
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)


Precision-Recall Curve
Precision
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Recall
Precision-recall curve of class 0 (area = 1.000)
Precision-recall curve of class 1 (area = 0.977)
Precision-recall curve of class 2 (area = 1.000)
Precision-recall curve of class 3 (area = 0.979)
Precision-recall curve of class 4 (area = 1.000)
Precision-recall curve of class 5 (area = 1.000)
Precision-recall curve of class 6 (area = 1.000)
Precision-recall curve of class 7 (area = 1.000)
Precision-recall curve of class 8 (area = 0.966)
Precision-recall curve of class 9 (area = 0.984)
micro-average Precision-recall curve (area = 0.991)

```
RandomForestRegressor Accuracy: 0.9717891610987379 Best Param
etre {'clf__n_estimators': 200}
```

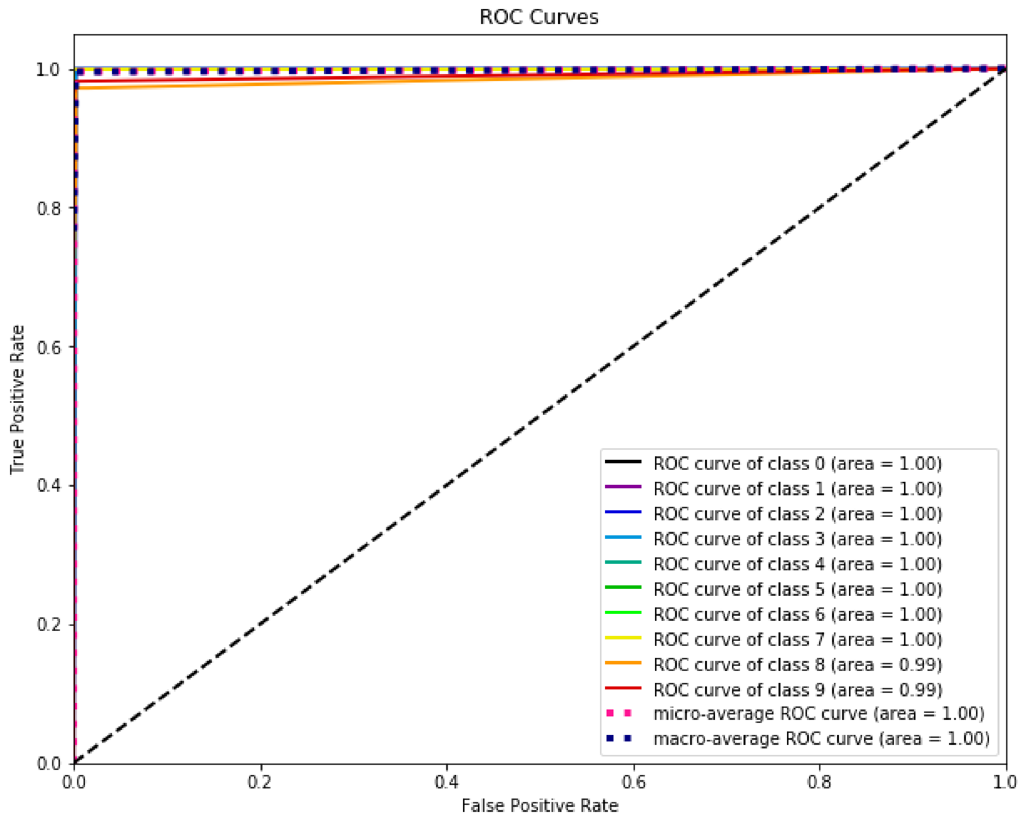
```
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\metrics\clas
sification.py:1052: UserWarning: Note that pos_label (set to
'positive') is ignored when average != 'binary' (got 'micr
o'). You may use labels=[pos_label] to specify a single posit
ive class.
  % (pos_label, average), UserWarning)
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:77: DeprecationWarning: Function plot_precision_reca
ll_curve is deprecated; This will be removed in v0.4.0. Pleas
e use scikitplot.metrics.plot_precision_recall_curve instead.
  warnings.warn(msg, category=DeprecationWarning)
```

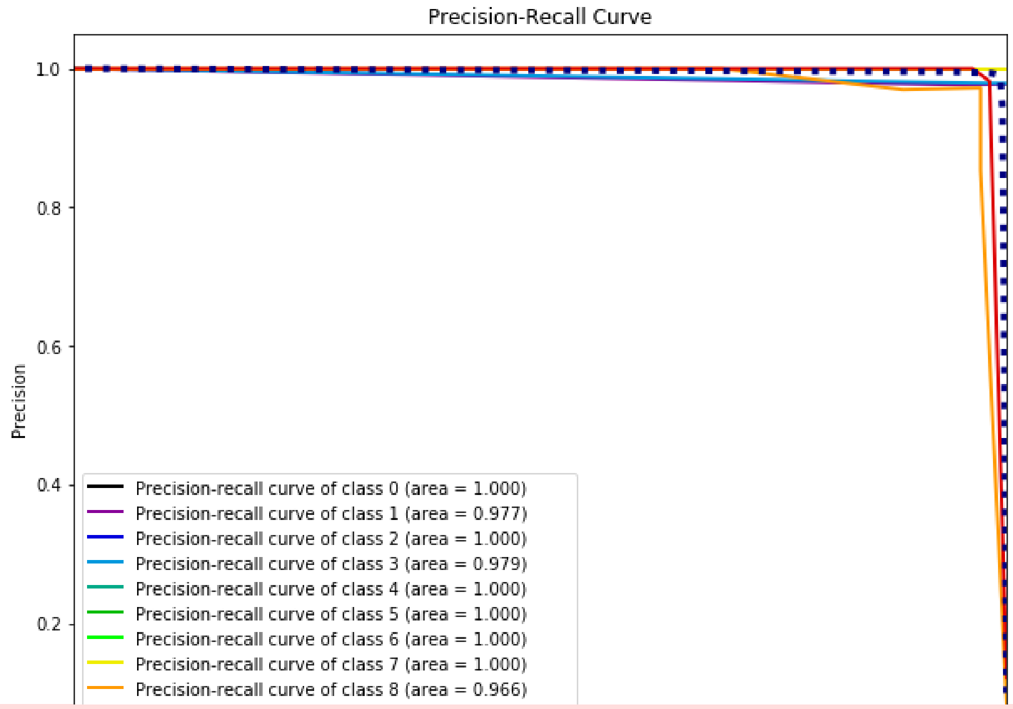



Precision-Recall Curve
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Precision-recall curve of class 0 (area = 1.000)
Precision-recall curve of class 1 (area = 0.977)
Precision-recall curve of class 2 (area = 1.000)
Precision-recall curve of class 3 (area = 0.979)
Precision-recall curve of class 4 (area = 1.000)
Precision-recall curve of class 5 (area = 1.000)
Precision-recall curve of class 6 (area = 1.000)
Precision-recall curve of class 7 (area = 1.000)
Precision-recall curve of class 8 (area = 0.966)
Precision-recall curve of class 9 (area = 0.984)
micro-average Precision-recall curve (area = 0.991)

```
DecisionTreeRegressor Accuracy: 0.835931700074239 Best Parame
tre {'clf__max_depth': 15.0}
```

```
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\metrics\clas
sification.py:1052: UserWarning: Note that pos_label (set to
'positive') is ignored when average != 'binary' (got 'micr
o'). You may use labels=[pos_label] to specify a single posit
ive class.
  % (pos_label, average), UserWarning)
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:77: DeprecationWarning: Function plot_precision_reca
ll_curve is deprecated; This will be removed in v0.4.0. Pleas
e use scikitplot.metrics.plot_precision_recall_curve instead.
  warnings.warn(msg, category=DeprecationWarning)
```



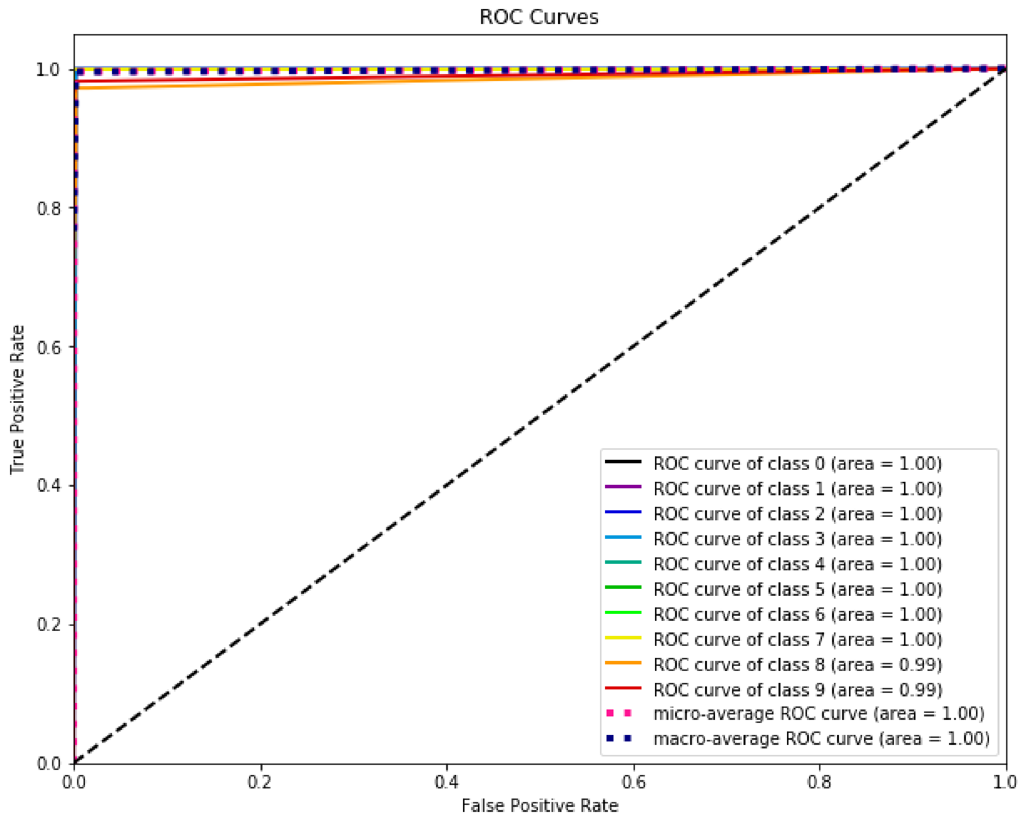


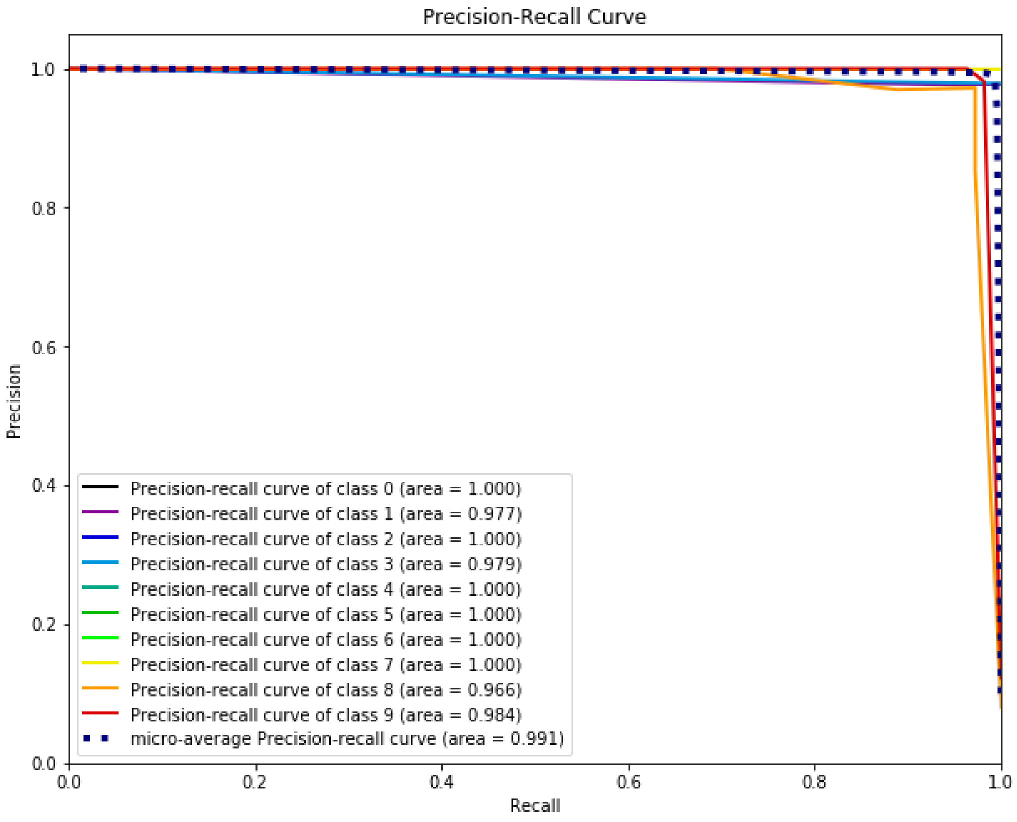
```
C:\Users\hed\Anaconda3\lib\site-packages\xgboost\sklearn.py:8
88: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_enco
der=False when constructing XGBClassifier object; and 2) Enco
de your labels (y) as integers starting with 0, i.e. 0, 1, 2,
..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[21:16:19] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old beha
vior.
```

```
XGBClassifier Accuracy: 0.9561989606533037 Best Parametre {'c
lf__random_state': 42}
[21:16:26] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective
'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old beha
vior.
```

```
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\metrics\clas
sification.py:1052: UserWarning: Note that pos_label (set to
'positive') is ignored when average != 'binary' (got 'micr
o'). You may use labels=[pos_label] to specify a single posit
ive class.
  % (pos_label, average), UserWarning)
C:\Users\hed\Anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:77: DeprecationWarning: Function plot_precision_reca
ll_curve is deprecated; This will be removed in v0.4.0. Pleas
e use scikitplot.metrics.plot_precision_recall_curve instead.
  warnings.warn(msg, category=DeprecationWarning)
```





Deploy the best model

In [29]:

```
import joblib
joblib.dump(clf,'KNeighborsClassifier.pkl')
```

```
# OPTIONAL : develop a web image for selecting an image from a local path
# display the image and its number (returned by the backend)
```

Out[29]:

```
['KNeighborsClassifier.pkl']
```

In [ ]:

```
```