```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

```
df_d2z = pd.read_csv("D2z.txt",sep=" ",names=["x1","x2","y"])
```

```
df_test = pd.DataFrame(columns=["x1","x2"])
```

```
for i in np.arange(-2,2.1,0.1):
  for j in np.arange(-2,2.1,0.1):
    temp_dict={"x1":i,"x2":j}
    df_test = df_test.append(temp_dict,ignore_index=True)
```

```
X_train = df_d2z[["x1","x2"]]
y_train = df_d2z["y"]
X_test = df_test
```

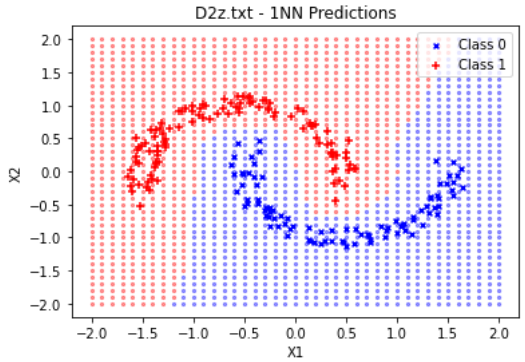
```
neigh = KNeighborsClassifier(n_neighbors=1)
neigh.fit(X_train, y_train)
predictions = neigh.predict(df_test)
```

```
df_test["y"] = predictions
```

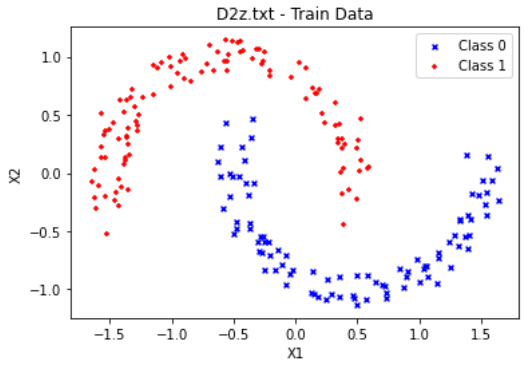
```
df_test["color"] = df_test['y'].apply(lambda x: "blue" if x == 0 else "red")
```

```
cols = df_test["color"].values
```

```
for i in np.arange(-2,2.1,0.1):
  for j in np.arange(-2,2.1,0.1):
    plt.scatter(i.round(1),j.round(1),alpha=0.4,s=5,c=df_test.loc[(df_test["x1"].round(1)==np.round(i,1)) & (df_test["x2"].round(1)==np.
plt.scatter(df_d2z[df_d2z["y"]==0]["x1"].values,df_d2z[df_d2z["y"]==0]["x2"].values,marker='x',c="blue",s=15,label='Class 0')
plt.scatter(df_d2z[df_d2z["y"]==1]["x1"].values,df_d2z[df_d2z["y"]==1]["x2"].values,marker='+',c="red",label='Class 1')
plt.title("D2z.txt - 1NN Predictions")
plt.legend(loc="upper right")
plt.xlabel("X1")
plt.ylabel("X2")
plt.savefig("d2z.pdf")
```




```
plt.scatter(df_d2z[df_d2z["y"]==0]["x1"].values,df_d2z[df_d2z["y"]==0]["x2"].values,marker='x',c="blue",s=15,label='Class 0')
plt.scatter(df_d2z[df_d2z["y"]==1]["x1"].values,df_d2z[df_d2z["y"]==1]["x2"].values,marker='+',c="red",s=15,label='Class 1')
plt.legend(loc="upper right")
plt.title("D2z.txt - Train Data")
plt.xlabel("X1")
plt.ylabel("X2")
plt.savefig("d2z_train.pdf")
```



Spam Filter Questions

```
df_spam = pd.read_csv("emails.csv")
```

```
df_spam.head()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	mi
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	
4	Email	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	

```
kf = KFold(n_splits=5)
kf.get_n_splits(df_spam)
```

```
5
```

```
for train_index, test_index in kf.split(df_spam):
  print(train_index,test_index)
```

```
 4420 4421 4422 4423 4424 4425 4426 4427 4428 4429 4430 4431 4432 4433
 4434 4435 4436 4437 4438 4439 4440 4441 4442 4443 4444 4445 4446 4447
 4448 4449 4450 4451 4452 4453 4454 4455 4456 4457 4458 4459 4460 4461
 4462 4463 4464 4465 4466 4467 4468 4469 4470 4471 4472 4473 4474 4475
 4476 4477 4478 4479 4480 4481 4482 4483 4484 4485 4486 4487 4488 4489
 4490 4491 4492 4493 4494 4495 4496 4497 4498 4499 4500 4501 4502 4503
 4504 4505 4506 4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517
 4518 4519 4520 4521 4522 4523 4524 4525 4526 4527 4528 4529 4530 4531
 4532 4533 4534 4535 4536 4537 4538 4539 4540 4541 4542 4543 4544 4545
 4546 4547 4548 4549 4550 4551 4552 4553 4554 4555 4556 4557 4558 4559
 4560 4561 4562 4563 4564 4565 4566 4567 4568 4569 4570 4571 4572 4573
 4574 4575 4576 4577 4578 4579 4580 4581 4582 4583 4584 4585 4586 4587
 4588 4589 4590 4591 4592 4593 4594 4595 4596 4597 4598 4599 4600 4601
 4602 4603 4604 4605 4606 4607 4608 4609 4610 4611 4612 4613 4614 4615
 4616 4617 4618 4619 4620 4621 4622 4623 4624 4625 4626 4627 4628 4629
 4630 4631 4632 4633 4634 4635 4636 4637 4638 4639 4640 4641 4642 4643
 4644 4645 4646 4647 4648 4649 4650 4651 4652 4653 4654 4655 4656 4657
 4658 4659 4660 4661 4662 4663 4664 4665 4666 4667 4668 4669 4670 4671
 4672 4673 4674 4675 4676 4677 4678 4679 4680 4681 4682 4683 4684 4685
 4686 4687 4688 4689 4690 4691 4692 4693 4694 4695 4696 4697 4698 4699
```

KNN

```
count=1
for train_index, test_index in kf.split(df_spam):
    #  print("TRAIN:", train_index, "TEST:", test_index)
     print("Fold"+str(count))
     df_train, df_test = df_spam.iloc[train_index,:], df_spam.iloc[test_index,:]
     X_train, y_train = df_train.iloc[:,1:3001],df_train.iloc[:,-1]
     X_test, y_test = df_test.iloc[:,1:3001], df_test.iloc[:,-1]
     neighbors = KNeighborsClassifier(n_neighbors=1)
     neighbors.fit(X_train, y_train)
     predictions = neighbors.predict(X_test)
     print(classification_report(y_test, predictions, target_names=['Spam','Not Spam']))
     print("Accuracy")
     print(accuracy_score(y_test, predictions))
     print("Recall")
     print(recall_score(y_test, predictions))
     print("Precision")
     print(precision_score(y_test, predictions))
     count+=1
```

```
    accuracy                           0.85      1000
   macro avg       0.81      0.86      0.83      1000
weighted avg       0.87      0.85      0.86      1000

Accuracy
0.853
Recall
0.8664259927797834
Precision
0.6857142857142857
Fold3
              precision    recall  f1-score   support

        Spam       0.93      0.87      0.90       716
```

```
Accuracy
0.851
Recall
0.8163265306122449
Precision
0.7164179104477612
Fold5
              precision    recall  f1-score   support

        Spam       0.88      0.78      0.83       694
    Not Spam       0.61      0.76      0.67       306

    accuracy                           0.78      1000
   macro avg       0.74      0.77      0.75      1000
weighted avg       0.80      0.78      0.78      1000

Accuracy
0.775
Recall
0.7581699346405228
Precision
0.6057441253263708
```

$Fold$	$Accuracy$	$Precision$	$Recall$
1	0.825	0.6544	0.8175
2	0.853	0.6857	0.8644
3	0.862	0.7212	0.8380
4	0.851	0.7164	0.8163
5	0.775	0.6057	0.7581

Logistic Reg

```
count=1
for train_index, test_index in kf.split(df_spam):
     print("Fold"+str(count))
    #  print("TRAIN:", train_index, "TEST:", test_index)
     df_train, df_test = df_spam.iloc[train_index,:], df_spam.iloc[test_index,:]
     X_train, y_train = df_train.iloc[:,1:3001],df_train.iloc[:,-1]
     X_test, y_test = df_test.iloc[:,1:3001], df_test.iloc[:,-1]
     clf = LogisticRegression(random_state=0,penalty='none',max_iter=1000)
     clf.fit(X_train, y_train)
     predictions = clf.predict(X_test)
     print(classification_report(y_test, predictions, target_names=['Spam','Not Spam']))
    #  print(accuracy_score(y_test, predictions))
     print("Accuracy")
     print(accuracy_score(y_test, predictions))
     print("Precision")
     print(precision_score(y_test, predictions))
     print("Recall")
     print(recall_score(y_test, predictions))
     count+=1
```

```
    accuracy                           0.97      1000
   macro avg       0.97      0.97      0.97      1000
weighted avg       0.98      0.97      0.98      1000

Accuracy
0.975
```

```
              Spam       0.96      0.97      0.97       706
          Not Spam       0.92      0.91      0.92       294

          accuracy                           0.95      1000
         macro avg       0.94      0.94      0.94      1000
      weighted avg       0.95      0.95      0.95      1000

      Accuracy
      0.951
      Precision
      0.9180887372013652
      Recall
      0.9149659863945578
      Fold5
                    precision    recall  f1-score   support

              Spam       0.95      0.94      0.95       694
          Not Spam       0.86      0.90      0.88       306

          accuracy                           0.93      1000
         macro avg       0.91      0.92      0.91      1000
      weighted avg       0.93      0.93      0.93      1000

      Accuracy
      0.925
      Precision
      0.8620689655172413
      Recall
      0.8986928104575164
```

$Fold$	$Accuracy$	$Precision$	$Recall$
1	0.967	0.92	0.9684
2	0.975	0.9436	0.9675
3	0.966	0.9251	0.9577
4	0.951	0.918	0.9149
5	0.925	0.862	0.8986

multiple k

```
for k in range(1,11,2): # k as asked in question
  mean_accuracy_score = 0
  print(k)
  for train_index, test_index in kf.split(df_spam):
      #  print("TRAIN:", train_index, "TEST:", test_index)
      df_train, df_test = df_spam.iloc[train_index,:], df_spam.iloc[test_index,:]
      X_train, y_train = df_train.iloc[:,1:3001],df_train.iloc[:,-1]
      X_test, y_test = df_test.iloc[:,1:3001], df_test.iloc[:,-1]
      neighbors = KNeighborsClassifier(n_neighbors=k)
      neighbors.fit(X_train, y_train)
      predictions = neighbors.predict(X_test)
      print(classification_report(y_test, predictions, target_names=['Spam','Not Spam']))
      mean_accuracy_score += accuracy_score(y_test,predictions)
      print(mean_accuracy_score)
```

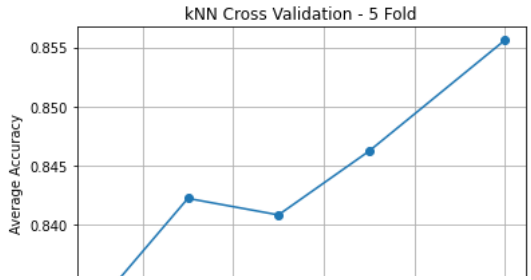
```
    1.6989999999999998
                  precision    recall  f1-score   support

           Spam       0.92      0.89      0.91       716
       Not Spam       0.75      0.82      0.78       284

       accuracy                           0.87      1000
      macro avg       0.84      0.85      0.85      1000
   weighted avg       0.88      0.87      0.87      1000

    2.57
                  precision    recall  f1-score   support

           Spam       0.92      0.91      0.91       706
       Not Spam       0.79      0.80      0.80       294

       accuracy                           0.88      1000
      macro avg       0.85      0.86      0.86      1000
   weighted avg       0.88      0.88      0.88      1000

    3.4499999999999997
                  precision    recall  f1-score   support

           Spam       0.87      0.80      0.83       694
       Not Spam       0.61      0.74      0.67       306

       accuracy                           0.78      1000
      macro avg       0.74      0.77      0.75      1000
   weighted avg       0.79      0.78      0.78      1000

    4.228
```

```
accuracy_dict = {}
for k in range(1,11,2):
  mean_accuracy_score = 0
  print("k:",k)
  for train_index, test_index in kf.split(df_spam):
      #  print("TRAIN:", train_index, "TEST:", test_index)
      df_train, df_test = df_spam.iloc[train_index,:], df_spam.iloc[test_index,:]
      X_train, y_train = df_train.iloc[:,1:3001],df_train.iloc[:,-1]
      X_test, y_test = df_test.iloc[:,1:3001], df_test.iloc[:,-1]
      neighbors = KNeighborsClassifier(n_neighbors=k,algorithm='brute')
      neighbors.fit(X_train, y_train)
      predictions = neighbors.predict(X_test)
      # print(classification_report(y_test, predictions, target_names=['Spam','Not Spam']))
      mean_accuracy_score += accuracy_score(y_test,predictions)
  print("Accuracy",mean_accuracy_score/5)
  accuracy_dict[k] = mean_accuracy_score/5
```

```
k: 1
Accuracy 0.8332
k: 3
Accuracy 0.8422000000000001
k: 5
Accuracy 0.8408
k: 7
Accuracy 0.8462
k: 9
Accuracy 0.8455999999999999
```

K	$Accuracy$
1	0.8332
2	0.8422
3	0.8408
4	0.8462
5	0.8455

```
#accuracy_dict.pop(9) # pop out 9
accuracy_dict[10]=0.8556 # ran this seperately
plt.plot(list(accuracy_dict.keys()),list(accuracy_dict.values()),marker='o')
plt.grid(True)
plt.xlabel("k")
plt.ylabel("Average Accuracy")
plt.title("kNN Cross Validation - 5 Fold")
plt.savefig("2-4.pdf")
```




```
list(accuracy_dict.values())
```

```
[0.8332, 0.8422000000000001, 0.8408, 0.8462, 0.8556]
```

```
k=10
mean_accuracy_score = 0
print("k",k)
for train_index, test_index in kf.split(df_spam):
    #  print("TRAIN:", train_index, "TEST:", test_index)
    df_train, df_test = df_spam.iloc[train_index,:], df_spam.iloc[test_index,:]
    X_train, y_train = df_train.iloc[:,1:3001],df_train.iloc[:,-1]
    X_test, y_test = df_test.iloc[:,1:3001], df_test.iloc[:,-1]
    neighbors= KNeighborsClassifier(n_neighbors=k)
    neighbors.fit(X_train, y_train)
    predictions = neighbors.predict(X_test)
    # print(classification_report(y_test, predictions, target_names=['Spam','Not Spam']))
    mean_accuracy_score += accuracy_score(y_test,predictions)
print("Accuracy mean KNN",mean_accuracy_score/5)
```

```
k 10
Accuracy mean KNN 0.8556000000000001
```

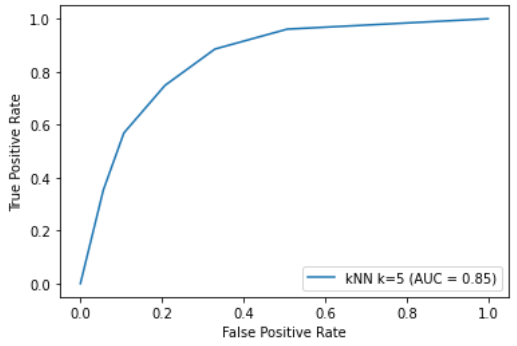
```
#Single Split
df_train = df_spam[:4000]
df_test = df_spam[4000:]
X_train, y_train = df_train.iloc[:,1:3001],df_train.iloc[:,-1]
X_test, y_test = df_test.iloc[:,1:3001], df_test.iloc[:,-1]
```

```
neighbors = KNeighborsClassifier(n_neighbors=5)
neighbors.fit(X_train, y_train)
predictions_knn = neighbors.predict(X_test)
print(classification_report(y_test, predictions, target_names=['Spam','Not Spam']))
```

```
              precision    recall  f1-score   support

        Spam       0.86      0.82      0.84       694
    Not Spam       0.63      0.69      0.66       306

    accuracy                           0.78      1000
   macro avg       0.74      0.76      0.75      1000
weighted avg       0.79      0.78      0.78      1000
```

```
y_pred_knn = neighbors.predict_proba(X_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_knn[:, 1])
roc_auc = auc(fpr, tpr)
fig = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,estimator_name='kNN k=5')
fig.plot()
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7f024b0d1490>
```



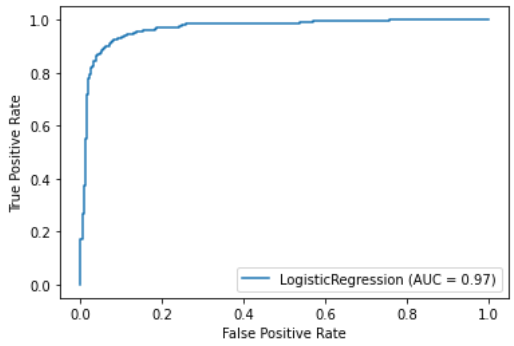

```
clf = LogisticRegression(random_state=0,penalty='none',max_iter=1000)
clf.fit(X_train, y_train)
```

```
predictions_lr = clf.predict(X_test)
print(classification_report(y_test, predictions, target_names=['Spam','Not Spam']))
```

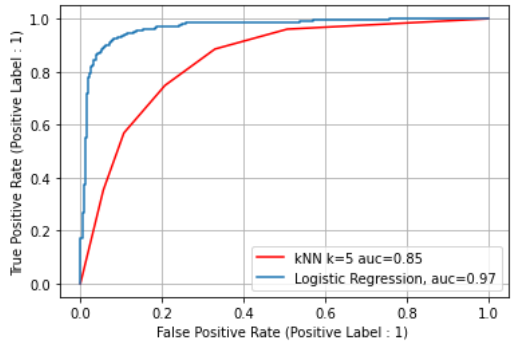
```
                  precision    recall  f1-score   support

            Spam       0.86      0.82      0.84       694
        Not Spam       0.63      0.69      0.66       306

        accuracy                           0.78      1000
       macro avg       0.74      0.76      0.75      1000
    weighted avg       0.79      0.78      0.78      1000
```

```
y_pred = clf.decision_function(X_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,estimator_name='LogisticRegression')
display.plot()
```

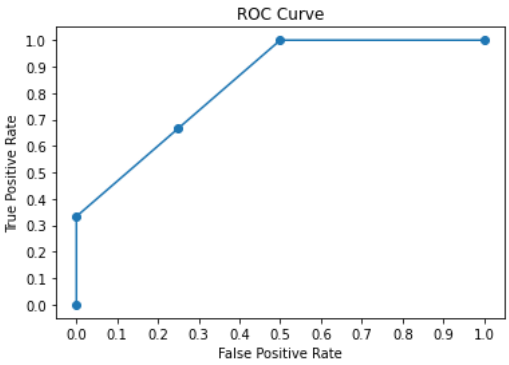
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7f024af3edf0>
```




```
from sklearn import metrics
pred = y_pred_knn
label = y_test
fpr, tpr, thresh = metrics.roc_curve(label, pred[:,1])
auc = metrics.roc_auc_score(label, pred[:,1])
plt.plot(fpr,tpr,label="kNN k=5 auc="+str(np.round(auc,2)),color='red')
pred = y_pred
label = y_test
fpr, tpr, thresh = metrics.roc_curve(label, pred)
auc = metrics.roc_auc_score(label, pred)
plt.plot(fpr,tpr,label="Logistic Regression, auc="+str(np.round(auc,2)))
plt.grid(True)
plt.xlabel('False Positive Rate (Positive Label : 1) ')
plt.ylabel('True Positive Rate (Positive Label : 1)')
plt.legend(loc=0)
plt.savefig('roc_auc.pdf')
```




```
x = [0,2/6,4/6,6/6,6/6,6/6]
y= [0,0,1/4,2/4,4/4]
plt.plot(y,x,marker='o')
worst_x = [0,0.4,0.8,1]
worst_y = [0,0.4,0.8,1]
#plt.plot(worst_x,worst_y,linestyle='dotted')
plt.xticks(np.arange(0,1.1,0.1))
plt.yticks(np.arange(0,1.1,0.1))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.savefig('ROC_5.pdf')
```



✓ 0s completed at 12:34 AM