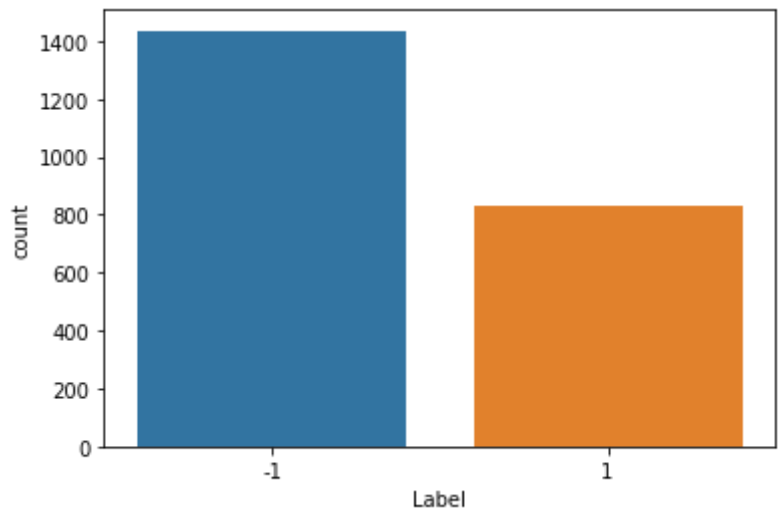
```
from sklearn.metrics import roc_auc_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import ShuffleSplit
import os
from sklearn import metrics
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
```

Load Dataset

```
xdata=pd.read_csv("./trainset.data")
ydata=pd.read_csv("./testset.dat")
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(xdata['Label'],label="Sum")
plt.show()
xlab=xdata['Label']
xid=xdata['Sequence']
yid=ydata['ID']

pdA=xdata.drop(['Label'], axis = 1)
pdB=ydata.drop(['ID'], axis = 1)
pdB=pdB.rename(columns = {" Sequence": "Sequence"})
```



Composition of k-spaced amino acid pairs (CKSAAP)

```
def returnCKSAAP(pdA):
    AA = 'ACDEFGHIKLMNPQRSTVWY'
    gap=1
    encodings = []
    aaPairs = []
    for aa1 in AA:
        for aa2 in AA:
            aaPairs.append(aa1 + aa2)
    for i in pdA['Sequence']:
        name, sequence = i,i
        code = []
        for g in range(gap+1):
            myDict = {}
            for pair in aaPairs:
                myDict[pair] = 0
            sum = 0
            for index1 in range(len(sequence)):
                index2 = index1 + g + 1
                if index1 < len(sequence) and index2 < len(sequence) and sequence[inde›
                    myDict[sequence[index1] + sequence[index2]] = myDict[sequence[inde›
                    sum = sum + 1
            for pair in aaPairs:
                code.append(myDict[pair] / sum)
        encodings.append(code)
    return encodings
```

Dipeptide composition (DPC)

```
def returnDPC(pdA):
    AA = 'ACDEFGHIKLMNPQRSTVWY'
    encodings = []
    diPeptides = [aa1 + aa2 for aa1 in AA for aa2 in AA]


    AADict = {}
    for i in range(len(AA)):
        AADict[AA[i]] = i

    for i in pdA['Sequence']:
        name, sequence = i,i
        code = []
        tmpCode = [0] * 400
        for j in range(len(sequence) - 2 + 1):
            tmpCode[AADict[sequence[j]] * 20 + AADict[sequence[j+1]]] = tmpCode[AADict[
        if sum(tmpCode) != 0:
            tmpCode = [i/sum(tmpCode) for i in tmpCode]
        code = code + tmpCode
        encodings.append(code)
    return encodings
```

800 features from CKSAAP and 400 features from DPC

```
```

```
train_features1=np.array(returnCKSAAP(pdA))
print(train_features1.shape)
unique_d1 = [list(x) for x in set(tuple(x) for x in train_features1)]
print(len(unique_d1))

test_features1=np.array(returnCKSAAP(pdB))
print(test_features1.shape)
unique_d1 = [list(x) for x in set(tuple(x) for x in test_features1)]
print(len(unique_d1))

train_features2=np.array(returnDPC(pdA))
print(train_features2.shape)
unique_d1 = [list(x) for x in set(tuple(x) for x in train_features2)]
print(len(unique_d1))

test_features2=np.array(returnDPC(pdB))
print(test_features2.shape)
unique_d1 = [list(x) for x in set(tuple(x) for x in test_features2)]
print(len(unique_d1))


train_features=np.concatenate((train_features1,train_features2), axis=1)
print(train_features.shape)
unique_d1 = [list(x) for x in set(tuple(x) for x in train_features)]
print(len(unique_d1))


test_features=np.concatenate((test_features1,test_features2), axis=1)
print(test_features.shape)
unique_d1 = [list(x) for x in set(tuple(x) for x in test_features)]
print(len(unique_d1))
```

```
(2270, 800)
2270
(567, 800)
567
(2270, 400)
2267
(567, 400)
567
(2270, 1200)
2270
(567, 1200)
567
```

I. Best: K-Neighbour (CKSAAP+DPC)

```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=2)
ss=ShuffleSplit(n_splits=20, test_size=0.5, random_state=10)
scores = cross_val_score(neigh, train_features,xlab, cv=ss,n_jobs=-1, verbose=1)
print(np.mean(scores))
print(scores)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
0.8038766519823788
[0.79118943 0.78942731 0.81321586 0.80352423 0.82643172 0.80176211
 0.82555066 0.80528634 0.79471366 0.80352423 0.80264317 0.80528634
 0.79295154 0.81409692 0.80440529 0.8        0.79295154 0.80881057
 0.80528634 0.79647577]
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:   47.9s finished
```

```
neigh.fit(train_features, xlab)
pred= neigh.predict(test_features)
ids= np.arange(1001, 1568)
df= pd.DataFrame(ids, columns= ["ID"])
df['Label']= pred
df.to_csv('./submnk.csv', index= False)
```

II Best:Keras Classifier(CKSAAP Features)

```
def baseline_model():
    # create model
    model = Sequential()
    model.add(Dense(1200, input_dim=800, activation='relu'))
    model.add(Dense(2, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy
    return model
estimator = KerasClassifier(build_fn=baseline_model, epochs=200, batch_size=5,verbose=0
kfold = KFold(n_splits=5, shuffle=True)
results = cross_val_score(estimator,train_features1,xlab, cv=kfold)
print(np.mean(results))
print(results)
```

```
estimator.fit(train_features1, xlab)
pred= estimator.predict(test_features1)
ids= np.arange(1001, 1568)
df= pd.DataFrame(ids, columns= ["ID"])
df['Label']= pred
df.to_csv('./submkc.csv', index= False)
```

```
1
```