ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ 1Η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

ΚΟΥΡΕΑΣ ΑΘΑΝΑΣΙΟΣ ΑΜ:4392

ΓΕΩΡΓΙΟΥ ΚΩΝΣΤΑΝΤΙΝΟΣ ΑΜ:4333

1η Μέθοδος Ταξινόμησης:

Α)Nearest Neighbor k-NN με Ευκλείδεια Απόσταση

Παρακάτω θα παραθέσω τον κώδικα που χρειάστηκα για να βγάλω τα αποτελέσματα των δοκίμων

Κώδικας σε python

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import load_model
import os
import pandas as pd
import time
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnis
t.load_data() #Edw fortwnoume ta arxeia apo to sunolo dedomenwn pou exo
ume to fashion mnist



train_images = train_images.reshape((60000, 28, 28, 1)) #Edw katalabain
oume oti oi eikones mas exoun 4 diastaseis opou i prwti diastasi einai
oi test kai oi train eikones
                            #Deuteri kai triti diastasi einai to ipsos
kai to platos tis eikonas kai i tetarti diastasi einai 1 giati
test_images = test_images.reshape((10000, 28, 28, 1))   #exoume asproma
vres eikones
```

```
train_images = train_images / 255.0
test_images = test_images / 255.0
training_size = 6000
test_size = 1000


train_images = train_images.reshape(training_size*10, 784) #28*28 #Gia
na ftiaxoume tis methodous kai na tis xrisimopoiisoume xreiazomaste oi
eikones na einai mia diastasi
test_images = test_images.reshape(test_size*10, 784)    #gia auto tis k
anoume monodiastates


x_train,  y_train, x_test, y_test =train_images, train_labels,test_imag
es,test_labels

x_train = x_train[:10000] #Edw xrisimopoioume ligotera dedomena giati
pairnei polu xrono gia na exetasei ola ta dedomena
y_train = y_train[:10000]

start=time.time()

classifier= KNeighborsClassifier(n_neighbors=5, metric='euclidean')
#Edw xrisimopoioume tin methodo K-
NN pou ypologizei tin euklideia apostasi
classifier.fit(x_train, y_train)

y_pred_knn = classifier.predict(x_test)
knn_accuracy = accuracy_score(y_test, y_pred_knn) #Ypologizoume accurac
y
knn_recall = recall_score(y_test, y_pred_knn,average= "weighted") #Ypol
ogizoume recall
knn_precision = precision_score(y_test, y_pred_knn,average= "weighted")
 #Ypologizoume precision
knn_f1 = f1_score(y_test, y_pred_knn, average= "weighted") #Ypologizoum
e f1_score
end=time.time()
knn_time=end-start

print("-----------------K-NN Report for Euclidean Distance------------
--")
print("F1 score: {}".format(knn_f1))
print("Accuracy score: {}".format(knn_accuracy))
print("Precision Score: {}".format(knn_precision))
print("Recall Score: {}".format(knn_recall))
print("Estimated Time {}".format(knn_time))
```

Αποτελέσματα 1ης Μεθόδου Ευκλείδεια Απόσταση για k=1):

```
-----------------K-NN Report for Euclidean Distance---------------
F1 score: 0.8050389193711215
Accuracy score: 0.8038
Precision Score: 0.810551225467716
Recall Score: 0.8038
Estimated Time 172.49013328552246
```

Αποτελέσματα 1ης Μεθόδου Ευκλείδεια Απόσταση για k=5):

```
-----------------K-NN Report for Euclidean Distance---------------
F1 score: 0.8176757982415854
Accuracy score: 0.8179
Precision Score: 0.8231371964518323
Recall Score: 0.8179
Estimated Time 177.45053791999817
```

Αποτελέσματα 1ης Μεθόδου Ευκλείδεια Απόσταση για k=10):

```
-----------------K-NN Report for Euclidean Distance---------------
F1 score: 0.812096817392384
Accuracy score: 0.8128
Precision Score: 0.8176699674598117
Recall Score: 0.8128
Estimated Time 180.2935791015625
```

A)Nearest Neighbor k-NN με Συνημιτονοειδή Απόσταση

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import load_model
import os
import pandas as pd
import time
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
fashion_mnist = keras.datasets.fashion_mnist


(train_images, train_labels), (test_images, test_labels) = fashion_mnis
t.load_data()
```

```
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
train_images = train_images / 255.0
test_images = test_images / 255.0
training_size = 6000
test_size = 1000



train_images = train_images.reshape(training_size*10, 784) #28*28
test_images = test_images.reshape(test_size*10, 784)


x_train,  y_train, x_test, y_test = train_images, train_labels,test_ima
ges,test_labels

x_train = x_train[:10000]
y_train = y_train[:10000]

start=time.time()
classifier2= KNeighborsClassifier(n_neighbors=1, metric='cosine')
classifier2.fit(x_train, y_train)

y_pred_knn2 = classifier2.predict(x_test)
knn_accuracy2 = accuracy_score(y_test, y_pred_knn2)
knn_recall2 = recall_score(y_test, y_pred_knn2,average= "weighted")
knn_precision2 = precision_score(y_test, y_pred_knn2,average= "weighted
")
knn_f12 = f1_score(y_test, y_pred_knn2, average= "weighted")
end=time.time()
knn2_time=end-start

print("-----------------K-NN Report for Cosine Distance-------------")
print("F1 score: {}".format(knn_f12))
print("Accuracy score: {}".format(knn_accuracy2))
print("Precision Score: {}".format(knn_precision2))
print("Recall Score: {}".format(knn_recall2))
print("Estimated Time {}".format(knn2_time))
```

Αποτελέσματα 1ης Μεθόδου Συνημιτονοειδή Απόσταση για k=1 :

```
-----------------K-NN Report for Cosine Distance---------------
F1 score: 0.8135041691925162
Accuracy score: 0.814
Precision Score: 0.8202003296970454
Recall Score: 0.814
Estimated Time 7.01285529136657
```

Αποτελέσματα 1ης Μεθόδου Συνημιτονοειδή Απόσταση για k=5 :

```
-----------------K-NN Report for Cosine Distance---------------
F1 score: 0.8128422152699226
Accuracy score: 0.8168
Precision Score: 0.824690657867799
Recall Score: 0.8168
Estimated Time 6.864148855209351
```

Αποτελέσματα 1ης Μεθόδου Συνημιτονοειδή Απόσταση για k=10:

```
-----------------K-NN Report for Cosine Distance---------------
F1 score: 0.805449478862958
Accuracy score: 0.8106
Precision Score: 0.8219570381992511
Recall Score: 0.8106
Estimated Time 6.810957670211792
```

2η Μέθοδος Ταξινόμησης:

A)Neural Networks με 1 κρυμμένο επίπεδο και 500 κρυμμένους νευρώνες

Παρακάτω θα παραθέσω τον κώδικα που χρειάστηκα για να βγάλω τα αποτελέσματα των δοκίμων

Κώδικας σε python

```
import tensorflow as tf
```

```
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import load_model
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten, Dropout, BatchNorm
alization
from keras.optimizers import SGD, Adagrad, RMSprop, Adam
import time
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnis
t.load_data()



train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

input_shape = (28, 28, 1)
train_images = train_images / 255.0
test_images  = test_images / 255.0



x_train,  y_train, x_test, y_test =train_images, train_labels,test_imag
es,test_labels

x_train = x_train[:10000]
y_train = y_train[:10000]

model = Sequential()
model.add(Flatten(input_shape=input_shape))
model.add(Dense(500, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
model.compile(optimizer=SGD(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

start-time.time()
fit = model.fit(train_images, train_labels, epochs=30, verbose=0)
y_pred_NN = model.predict_classes(test_images, verbose=0)
```

```
NN_accuracy = accuracy_score(test_labels, y_pred_NN)
NN_precision = precision_score(test_labels, y_pred_NN,average= "weighte
d")
NN_recall = recall_score(test_labels, y_pred_NN,average= "weighted")
NN_f1 = f1_score(test_labels, y_pred_NN,average= "weighted")
end=time.time()
NN_time=end-start

print("-----------------
Neural Networks Report for 1 Hidden Layer with K=500---------------")
print("Accuracy:{}".format(NN_accuracy))
# precision tp / (tp + fp)
print("Precision:{}".format(NN_precision))
# recall: tp / (tp + fn)
print("Recall:{}".format(NN_recall))
# f1: 2 tp / (2 tp + fp + fn)
print("F1 Score:{}".format(NN_f1))
print("Estimated Time:{}".format(NN_time))
```

Αποτελέσματα 2ης Μεθόδου Α):

```
---------Neural Networks Report for 1 Hidden Layer with K=500----------
Accuracy:0.881
Precision:0.8815372075418735
Recall:0.881
F1 Score:0.8809905653878153
Estimated Time:1408.963793039322
```

Β)Neural Networks με 2 κρυμμένα επίπεδα και 500 κρυμμένους νευρώνες και 200 κρυμμένους νευρώνες

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import load_model
from keras.models import Sequential
```

```
from keras.layers import Dense, Activation, Flatten, Dropout, BatchNorm
alization
from keras.optimizers import SGD, Adagrad, RMSprop, Adam
import time
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
fashion_mnist = keras.datasets.fashion_mnist


(train_images, train_labels), (test_images, test_labels) = fashion_mnis
t.load_data()



train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

input_shape = (28, 28, 1)
train_images = train_images / 255.0
test_images  = test_images / 255.0



x_train,  y_train, x_test, y_test =train_images, train_labels,test_imag
es,test_labels

x_train = x_train[:10000]
y_train = y_train[:10000]

model = Sequential()
model.add(Flatten(input_shape=input_shape))
model.add(Dense(500, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
model.compile(optimizer=SGD(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


start=time.time()

fit = model.fit(train_images, train_labels, epochs=30, verbose=0)
y_pred_NN2 = model.predict_classes(test_images, verbose=0)
```

```
NN2_accuracy = accuracy_score(y_test, y_pred_NN2)
NN2_precision = precision_score(y_test, y_pred_NN2,average= "weighted")
NN2_recall = recall_score(y_test, y_pred_NN2,average= "weighted")
NN2_f1 = f1_score(y_test, y_pred_NN2,average= "weighted")
end=time.time()
NN2_time=end-start

print("-----------------
Neural Networks Report for 2 Hidden Layers with K1=500 and K2=200------
---------")
print("Accuracy:{}".format(NN2_accuracy))
# precision tp / (tp + fp)
print("Precision:{}".format(NN2_precision))
# recall: tp / (tp + fn)
print("Recall:{}".format(NN2_recall))
# f1: 2 tp / (2 tp + fp + fn)
print("F1 Score:{}".format(NN2_f1))
print("Estimated Time:{}".format(NN2_time))
```

Αποτελέσματα 2ης Μεθόδου Β):

```
---Neural Networks Report for 2 Hidden Layers with K1=500 and K2=200---
Accuracy:0.8806
Precision:0.8835198883243569
Recall:0.8806
F1 Score:0.8804790305037774
Estimated Time:202.74010276794434
```

3η Μέθοδος Ταξινόμησης:

Α)Support Vector Machines (SVM): Μηχανές διανυσματικής στήριξης, χρησιμοποιώντας (α) γραμμική συνάρτηση πυρήνα (linear kernel)

Παρακάτω θα παραθέσω τον κώδικα που χρειάστηκα για να βγάλω τα αποτελέσματα των δοκίμων

Κώδικας σε python

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
```

```
from tensorflow.keras.models import load_model
import os
import pandas as pd
import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
fashion_mnist = keras.datasets.fashion_mnist
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from keras.datasets import fashion_mnist
from sklearn.multiclass import OneVsRestClassifier



(train_images, train_labels), (test_images, test_labels) = fashion_mnis
t.load_data()


train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
train_images = train_images / 255.0
test_images = test_images / 255.0

training_size = 6000
test_size = 1000

train_images = train_images.reshape(training_size*10, 784) #28*28
test_images = test_images.reshape(test_size*10, 784)


x_train,  y_train, x_test, y_test =train_images, train_labels,test_imag
es,test_labels

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train.astype(np.float64))

x_train = x_train[:10000]

train = y_train[:10000]

start=time.time()
clf = OneVsRestClassifier(SVC(C=1, kernel='linear'))
clf.fit(x_train, y_train)


y_pred_svm = clf.predict(x_test)
```

```
svm_accuracy = accuracy_score(y_test, y_pred_svm)
svm_recall = recall_score(y_test, y_pred_svm,average= "weighted")
svm_precision = precision_score(y_test, y_pred_svm,average= "weighted")
svm_f1 = f1_score(y_test, y_pred_svm, average= "weighted")
end=time.time()
svm_time=end-start

print("-----------------SVM Report for Linear---------------")
print("F1 score: {}".format(svm_f1))
print("Accuracy score: {}".format(svm_accuracy))
print("Precision Score: {}".format(svm_precision))
print("Recall Score: {}".format(svm_recall))
print("Estimated Time:{}".format(svm_time))
```

Αποτελέσματα μεθόδου Support Vector Machines χρησιμοποιώντας γραμμική συνάρτηση πυρήνα (linear kernel)

```
-----------------SVM Report for Linear---------------
F1 score: 0.7212719781138753
Accuracy score: 0.7212
Precision Score: 0.7652041142264403
Recall Score: 0.7212
Estimated Time:335.7148184776306
```

B)Support Vector Machines (SVM): Μηχανές διανυσματικής στήριξης, χρησιμοποιώντας Gaussian συνάρτηση πυρήνα (kernel) δοκιμάζοντας διάφορες τιμές της παραμέτρου της

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import load_model
import os
import pandas as pd
import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
fashion_mnist = keras.datasets.fashion_mnist
from sklearn.svm import SVC
from sklearn.multiclass import OneVsRestClassifier
```

```
from sklearn.preprocessing import StandardScaler

from keras.datasets import fashion_mnist



(train_images, train_labels), (test_images, test_labels) = fashion_mnis
t.load_data()


train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
train_images = train_images / 255.0
test_images = test_images / 255.0

training_size = 6000
test_size = 1000

train_images = train_images.reshape(training_size*10, 784) #28*28
test_images = test_images.reshape(test_size*10, 784)


x_train,  y_train, x_test, y_test =train_images, train_labels,test_imag
es,test_labels



scaler = StandardScaler()
x_train = scaler.fit_transform(x_train_s.astype(np.float64))
x_train = x_train[:10000]
y_train = y_train[:10000]
start=time.time()
svm2 = OneVsRestClassifier(SVC(C=1, kernel='rbf'))
svm2.fit(x_train, y_train)


y_pred_svm2 = svm2.predict(x_test)
svm2_accuracy = accuracy_score(y_test, y_pred_svm2)
svm2_recall = recall_score(y_test, y_pred_svm2,average= "weighted")
svm2_precision = precision_score(y_test, y_pred_svm2,average= "weighted
")
svm2_f1 = f1_score(y_test, y_pred_svm2, average= "weighted")
end=time.time()
svm2_time = end-start

print("-----------------SVM Report for Gaussian---------------")
print("F1 score: {}".format(svm2_f1))
print("Accuracy score: {}".format(svm2_accuracy))
print("Precision Score: {}".format(svm2_precision))
```

```
print("Recall Score: {}".format(svm2_recall))
print("Estimated Time:{}".format(svm2_time))
```

Αποτελέσματα μεθόδου Support Vector χρησιμοποιώντας Gaussian συνάρτηση πυρήνα (kernel)

```
-----------------SVM Report for Gaussian---------------
F1 score: 0.5646297698405152
Accuracy score: 0.5688
Precision Score: 0.705610697983974
Recall Score: 0.5688
Estimated Time:239.29149317741394
```

Γ)Support Vector Machines (SVM): Μηχανές διανυσματικής στήριξης, χρησιμοποιώντας συνημιτονοειδή συνάρτηση πυρήνα (cosine kernel)

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import load_model
import os
import pandas as pd
import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
fashion_mnist = keras.datasets.fashion_mnist
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from keras.datasets import fashion_mnist
from sklearn.multiclass import OneVsRestClassifier
import math
from numpy import linalg as LA

def my_kernel(X, Y):
#Συναρτηση για να υπολογίζει την συνημιτονοειδή συνάρτηση

    norm = LA.norm(X) * LA.norm(Y)
    return np.dot(X, Y.T)/norm

(train_images, train_labels), (test_images, test_labels) = fashion_mnis
t.load_data()
```

```
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
train_images = train_images / 255.0
test_images = test_images / 255.0

training_size = 6000
test_size = 1000

train_images = train_images.reshape(training_size*10, 784) #28*28
test_images = test_images.reshape(test_size*10, 784)


x_train,  y_train, x_test, y_test =train_images, train_labels,test_imag
es,test_labels

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train.astype(np.float64))

x_train = x_train[:10000]
y_train = y_train[:10000]


start=time.time()
svm3 = OneVsRestClassifier(SVC(kernel = my_kernel,C=1))
svm3.fit(x_train, y_train)


y_pred_svm = svm3.predict(x_test)
svm3_accuracy = accuracy_score(y_test, y_pred_svm)
svm3_recall = recall_score(y_test, y_pred_svm,average= "weighted")
svm3_precision = precision_score(y_test, y_pred_svm,average= "weighted"
)
svm3_f1 = f1_score(y_test, y_pred_svm, average= "weighted")
end=time.time()
svm3_time=end-start



print("-----------------SVM Report for Cosine---------------")
print("F1 score: {}".format(svm3_f1))
print("Accuracy score: {}".format(svm3_accuracy))
print("Precision Score: {}".format(svm3_precision))
print("Recall Score: {}".format(svm3_recall))
print("Estimated Time:{}".format(svm3_time))
```

Αποτελέσματα μεθόδου Support Vector χρησιμοποιώντας συνημιτονοειδή συνάρτηση πυρήνα (kernel)

```
-----------------SVM Report for Cosine---------------

F1 score: 0.4976564483010756
Accuracy score: 0.541
Precision Score: 0.577944951511452
Recall Score: 0.541
Estimated Time:56.812220096588135
```

Σύγκριση Μεθόδων

Για τα K-NN παρατηρούμε ότι για την Ευκλείδεια Απόσταση για k=5 δηλαδή 5 κοντινότερους γείτονες έχει καλυτέρα αποτελέσματα(F1 Score,Accuracy,Recall,Precision) από την ίδια μέθοδο για k=1 η k=10.Επισης για τα K-NN παρατηρούμε ότι για την συνημιτονοειδή απόσταση πάλι για k=5 έχουμε τα καλυτέρα αποτελέσματα(F1 Score,Accuracy,Recall,Precision)) από την ίδια μέθοδο για k=1 η k=10.Αλλα αναμεσά στην μέθοδο με την ευκλείδεια απόσταση και την μέθοδο με την συνημιτονοειδή απόστασή ευκλείδεια απόσταση έχει καλυτέρα αποτελέσματα αλλά παίρνει πολύ περισσότερο χρόνο από την συνημιτονοειδή απόσταση δηλαδή Ευκλείδεια Απόσταση :

```
Estimated Time 177.45053791999817

Συνημιτονοειδής Απόσταση : Estimated Time 6.864148855209351
```

Αλλά τα αποτελέσματα των δυο μεθόδων είναι αρκετά κοντά οπότε καλύτερη μέθοδος είναι η K-NN με συνημιτονοειδή απόσταση γιατί κάνει πολύ λιγότερο χρόνο

Για τα Νευρωνικά Δίκτυα παρατηρούμε ότι όταν έχουμε δυο κρυμμένα επίπεδα η ταξινόμηση γίνεται πιο γρηγορά αλλά έχει μικρότερο accuracy(0.8806) από όταν έχουμε ένα κρυμμένο επίπεδο Επομένως παρατηρούμε ότι στα νευρωνικά δίκτυα χρειαζόμαστε περισσότερο χρόνο για να ταξινομήσουμε τα δεδομένα μας με ένα κρυμμένο επίπεδο αλλά έχουμε καλύτερο accuracy(0.881),όμως δεν είναι τόσο μεγάλη η διαφορά .Επίσης παρατηρούμε ότι από τα αλλά αποτελέσματα (F1 score,Precision,Recall) ,οι δυο μέθοδοι των νευρωνικών δικτυών είναι πολύ κοντά στα αποτελέσματα αλλά η μια χρειάζεται πάρα πολύ χρόνο . Επομένως η καλύτερη μέθοδος είναι όταν έχουμε δυο κρυμμένα επίπεδα στα νευρωνικά δίκτυα γιατί η ταξινόμηση γίνεται σε χρόνο (202.74010276794434 δευτερόλεπτα) που είναι περίπου το 1/7 του χρόνου των νευρωνικών δικτυών με 1 κρυμμένο επίπεδο

Για τα SVM παρατηρούμε ότι καλύτερη ακρίβεια έχει η γραμμική μέθοδος με σχετικά μεγάλη διαφορά δηλαδή SVM Linear accuracy 0.7212,SVM Gaussian accuracy 0.5688,SVM Cosine accuracy 0.541
Αλλά το SVM Linear χρειάζεται 335.7148184776306 δευτερόλεπτα ενώ το SVM Cosine χρειάζεται 56.812220096588135 δευτερόλεπτα και το

SVM Gaussian χρειάζεται 239.29149317741394 δευτερόλεπτα

Αλλά εφόσον έχουν τόσο μεγάλη διαφορά στην ακρίβεια(accuracy) η καλύτερη μέθοδος είναι η SVM Linear
Επίσης αν κρίνουμε και από τα αλλά αποτελέσματα (F1 score,Precision,Recall) η SVM Linear είναι η καλύτερη από τις άλλες μεθόδους.

Τέλος παρατηρούμε ότι η καλύτερη μέθοδος από όλες είναι η K-NN με συνημιτονοειδή απόσταση γιατί έχει πολύ καλά αποτελέσματα και κάνει και τον λιγότερο χρόνο από τις υπόλοιπες μεθόδους
Αλλά τα καλυτέρα αποτελέσματα μας τα δίνουν τα νευρωνικά δίκτυα με δυο κρυμμένα επίπεδα