



Διερεύνηση του ρυθμού αποχωρήσεων πελατών από μία επιχείρηση τηλεπικοινωνιών (Customers' churn rate investigation of a telco).

Data Organization and Data Mining

Οικονόμου Αλέξανδρος

AM: 2019119

Χαρακτηριστικά Διερεύνησης:

- Γλώσσα προγραμματισμού: [Python](#) με χρήση [scikit-learn](#).
- Λογισμικό/Πρόγραμμα: [Visual Studio Code](#).
- Ζήτημα Α: Κατάλληλη εφαρμογή των αλγορίθμων συσταδοποίησης [k-means](#), [Agglomerative clustering](#), [DBSCAN](#) και σχολιασμός αποτελεσμάτων/συστάδων.
- Ζήτημα Β: Σύγκριση των αλγορίθμων κατηγοριοποίησης [D-Tree](#), [kNN](#) και επιλογή του βέλτιστου/καλύτερου μοντέλου.

Ζήτημα Α:

1. (K-means)

Μετά από τις κατάλληλες αλλαγές στον κώδικα, την εκτέλεση του αλγορίθμου συσταδοποίησης k-means, την αποθήκευση του telco_clusterAssignments.csv αρχείου αλλά και την μελέτη των δεδομένων του, παρατηρώ τα εξής:

- Κατάλληλη Επιλογή του K:

Βάζω **K=5** λόγω του κατάλληλου σημείου στο SSE (Sum of Squares Error), όπως φαίνεται και στην παρακάτω εικόνα (Elbow method), στο οποίο παρατηρείτε μια άνοδος αγκώνα.

Image1: Elbow method

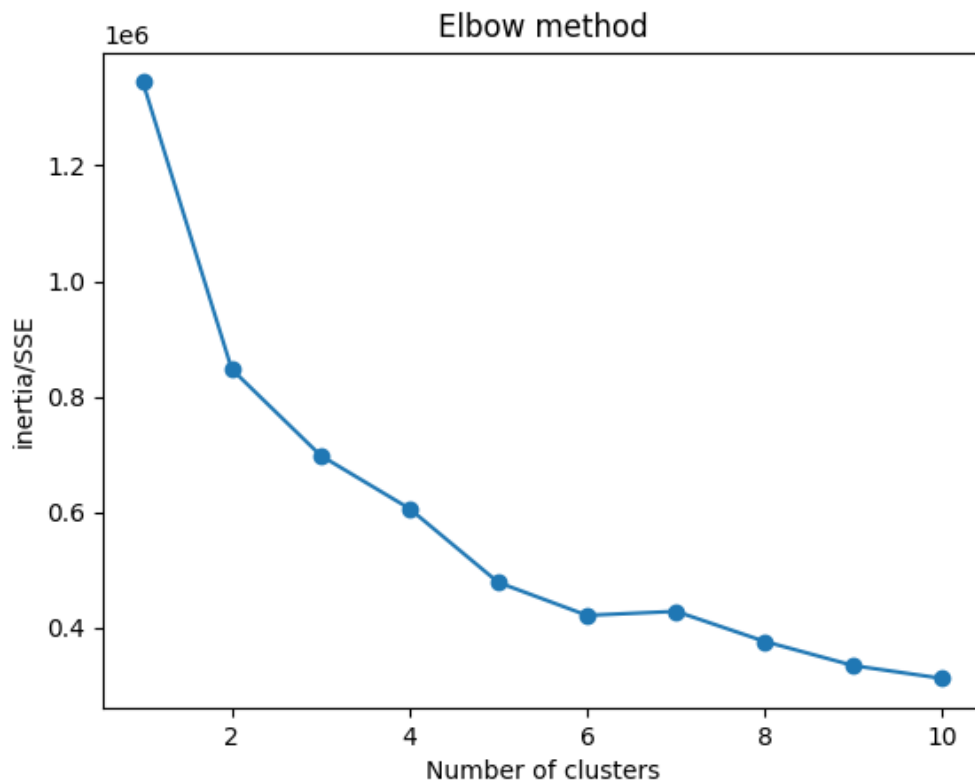


Image2: kmeans – telco_2023.py

```
kmeans - telco_2023.py X
Python > kmeans - telco_2023.py > ...
1  #https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
2
3  from sklearn.cluster import KMeans
4  import pandas as pd
5  import seaborn as sns
6  import matplotlib.pyplot as plt
7  from pandas.plotting import parallel_coordinates
8
9  #Loading data
10 telco = pd.read_csv('Datasets/telco_2023.csv')
11
12 print(telco.head())
13 print(telco.info())
14
15 #Selecting features for clustering
16 data = telco[['longmon', 'tollmon', 'equipmon', 'cardmon', 'wiremon',
17              'multiline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill']]
18
19 #Elbow method
20 sse = []
21 for i in range(1,11):
22     kmeans = KMeans(n_clusters=i, n_init='auto')
23     kmeans.fit(data)
24     sse.append(kmeans.inertia_)
25
26 plt.plot(range(1,11), sse, marker='o')
27 plt.title('Elbow method')
28 plt.xlabel('Number of clusters')
29 plt.ylabel('inertia/SSE')
30 plt.savefig('Elbow method - telco.png')
31 plt.show()
32
33 #Applying k-means clustering with the appropriate number of clusters
34 k = 5
35 kmeans = KMeans(n_clusters=k, n_init='auto')
36 kmeans.fit(data)
37
38 print('SSE:', kmeans.inertia_)
39 print('Final locations of the centroid:', kmeans.cluster_centers_)
40 print("The number of iterations required to converge", kmeans.n_iter_)
41
42 #Assigning cluster labels to each customer
43 telco['cluster'] = kmeans.labels_.tolist()
44
45 #Plotting some clusters in scatter plots
46 sns.scatterplot(x='longmon', y='tollmon', hue='cluster', data=telco)
47 plt.show()
48 sns.scatterplot(x='equipmon', y='cardmon', hue='cluster', data=telco)
49 plt.show()
50 sns.scatterplot(x='wiremon', y='ebill', hue='cluster', data=telco)
51 plt.show()
52
53 print(telco)
54
55 #Saving cluster assignments to a csv file
56 telco.to_csv('telco_clusterAssignments.csv')
57
58 df = pd.DataFrame(telco, columns = ['longmon', 'tollmon', 'equipmon', 'cardmon', 'wiremon',
59                                   'multiline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill'])
60 df['Clusters']=kmeans.labels_
61 print(df)
62 parallel_coordinates(df, 'Clusters', color=('#383c4a', '#0a3661', '#dcb536'))
63 plt.show()
```

- Ανάλυση των 5 Clusters στο Excel:

Cluster 0:

Συνολικά 90 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 5.

Cluster 1:

Συνολικά 192 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 82.

Cluster 2:

Συνολικά 199 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 31.

Cluster 3:

Συνολικά 447 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 141.

Cluster 4:

Συνολικά 72 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 15.

Άρα: Η 4η συστάδα (Cluster 3) θεωρείται η πιο προβληματική καθώς εκεί παρατηρείτε μεγαλύτερο ποσοστό πελατών που αποχώρησαν από την επιχείρηση τηλεπικοινωνιών.

2. (Agglomerative Clustering)

Μετά από τις κατάλληλες αλλαγές στον κώδικα, την εκτέλεση του αλγορίθμου Ιεραρχικής Συσταδοποίησης (Agglomerative Clustering), την αποθήκευση του telco_clusterAssignmentsHierarchical.csv αρχείου αλλά και την μελέτη των δεδομένων του, παρατηρώ τα εξής:

- Κατάλληλη Επιλογή Αριθμού των Clusters:

Επιλέγω συνολικά **3 Clusters** λόγω του παρακάτω Δενδρογράμματος (Dendrogram), στο οποίο χρησιμοποίησα **ward** μέθοδο και **euclidean** απόσταση. Επίσης, το έκοψα στο ύψος **450** για τον καλύτερο διαμοιρασμό των συστάδων (Clusters).

Image3: Dendrogram

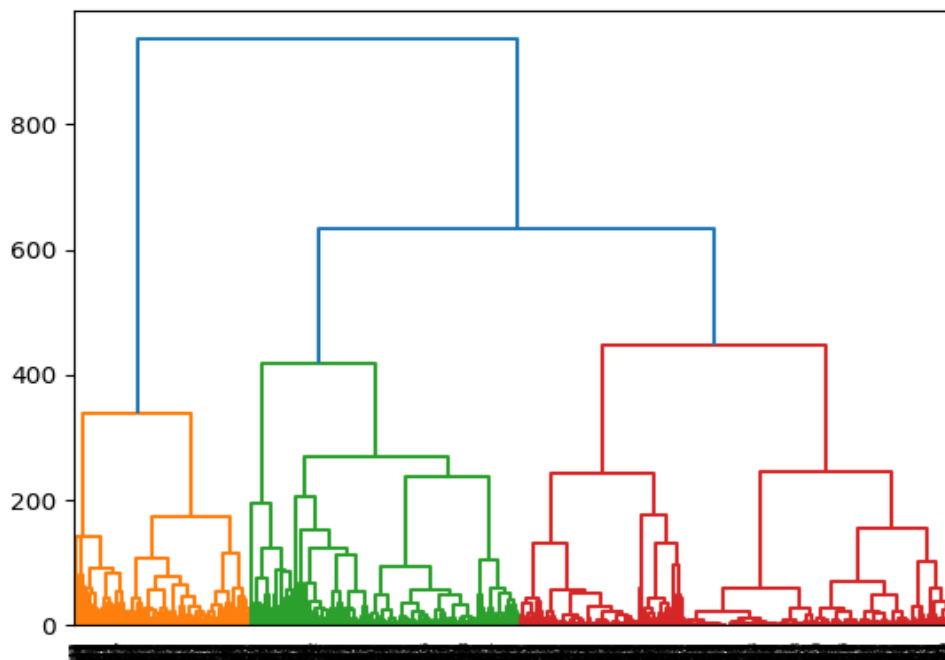


Image4: agglomerative – telco_2023.py

```
agglomerative - telco_2023.py X
Python > agglomerative - telco_2023.py > ...
1 #https://scipy.github.io/devdocs/reference/generated/scipy.cluster.hierarchy.dendrogram.html
2 #https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering
3
4 from scipy.cluster.hierarchy import dendrogram, linkage
5 from sklearn.cluster import AgglomerativeClustering
6 import pandas as pd
7 import matplotlib.pyplot as plt
8 import seaborn as sns
9 from pandas.plotting import parallel_coordinates
10
11 telco = pd.read_csv('Datasets/telco_2023.csv')
12
13 data = telco[['longmon', 'tollmon', 'equipmon', 'cardmon', 'wiremon',
14             'multiline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill']]
15
16 #Using 'ward' method and 'euclidean' metric.
17 linkage_data = linkage(data, method='ward', metric='euclidean')
18 dendrogram(linkage_data, color_threshold=450) #Cutting the dendrogram at 450
19 plt.savefig('Dendrogram - telco.png')
20 plt.show()
21
22 hierarchical_cluster = AgglomerativeClustering(n_clusters=3, metric='euclidean', linkage='ward')
23 labels = hierarchical_cluster.fit_predict(data)
24
25 telco['cluster'] = hierarchical_cluster.labels_.tolist()
26 sns.scatterplot(x='longmon', y='equipmon', hue='cluster', data=telco, color='blue')
27 plt.show();
28
29 print(telco)
30 telco.to_csv('telco_clusterAssignmentsHierarchical.csv')
31
32 df = pd.DataFrame(telco, columns = ["longmon", "tollmon", "equipmon", "cardmon", "wiremon",
33                                   "multiline", "voice", "pager", "internet", "forward", "confer", "ebill"])
34 df['Clusters']=hierarchical_cluster.labels_
35 print(df)
36 parallel_coordinates(df, 'Clusters', color=('#383c4a', '#0a3661', '#dcb536'))
37 plt.show()
```

○ Ανάλυση των 3 Clusters στο Excel:

Cluster 0:

Συνολικά 493 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 139.

Cluster 1:

Συνολικά 202 δεδομένο/γραμμή (πελάτης).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι 87.

Cluster 2:

Συνολικά 305 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 48.

Άρα: Η 1η συστάδα (Cluster 0) θεωρείται η πιο προβληματική καθώς εκεί παρατηρείτε μεγαλύτερο ποσοστό πελατών που αποχώρησαν από την επιχείρηση τηλεπικοινωνιών.

3. (DBSCAN)

Μετά από τις κατάλληλες αλλαγές στον κώδικα, την εκτέλεση του αλγορίθμου Βάση Πυκνότητας (DBSCAN), την αποθήκευση του telco_clusterAssignmentsDBScan.csv αρχείου αλλά και την μελέτη των δεδομένων του, παρατηρώ τα εξής:

- ο Κατάλληλη Επιλογή epsilon και min_samples:

Αρχικά, επιλέγω **min_samples=5** επειδή το σύνολο δεδομένων είναι δυσδιάστατο (γενικά το MinPts/min_samples θα πρέπει να είναι μεγαλύτερο ή ίσο με τη διάσταση του συνόλου δεδομένων). Επέλεξα το 5 διότι παρατηρείτε μικρότερος θόρυβος στα δεδομένα σε σχέση με το 4. Στην συνέχεια και σύμφωνα με το παρακάτω k-dist Graph, κατέληξα ότι **epsilon=18** είναι η καλύτερη τιμή, επειδή στο συγκεκριμένο σημείο παρατηρείται η άνοδος της καμπύλης. Επομένως, θα έχουμε **K=2** (2 Clusters).

Image5: k-dist Graph

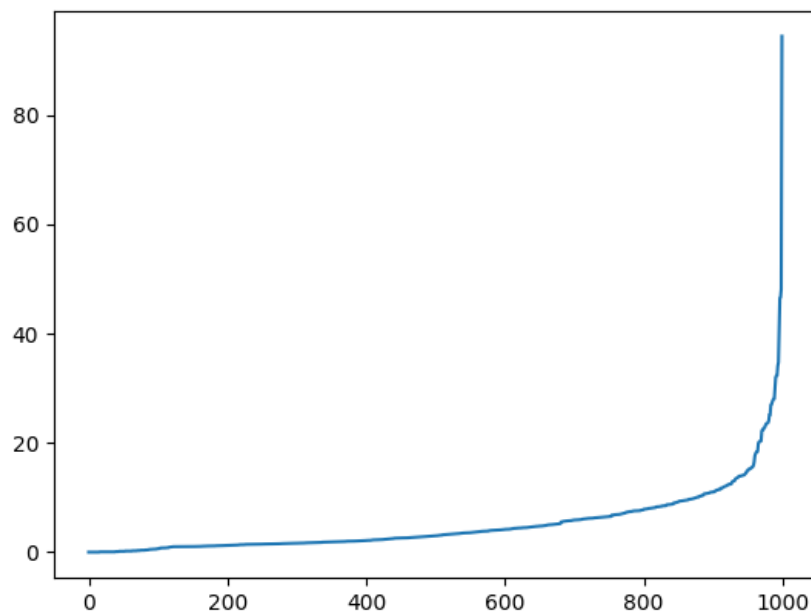


Image6: dbscan – telco_2023.py

```
dbscan - telco_2023.py X
Python > dbscan - telco_2023.py > ...
1  #https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
2
3  from sklearn.cluster import DBSCAN
4  import pandas as pd
5  import numpy as np
6  import seaborn as sns
7  import matplotlib.pyplot as plt
8  from pandas.plotting import parallel_coordinates
9
10 telco = pd.read_csv('Datasets/telco_2023.csv')
11
12 data = telco[['longmon', 'tollmon', 'equipment', 'cardmon', 'wiremon',
13 |             'multiline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill']]
14
15 #search for the eps parameter value
16 from sklearn.neighbors import NearestNeighbors # importing the library
17 neighb = NearestNeighbors(n_neighbors=2) # creating an object of the NearestNeighbors class
18 nbrs=neighb.fit(data) # fitting the data to the object
19 distances,indices=nbrs.kneighbors(data) # finding the nearest neighbours
20
21 # Sort and plot the distances results
22 distances = np.sort(distances, axis = 0) # sorting the distances
23 distances = distances[:, 1] # taking the second column of the sorted distances
24 plt.plot(distances) # plotting the distances
25 plt.show() # showing the plot
26
27 neighb = NearestNeighbors(n_neighbors=4) # creating an object of the NearestNeighbors class
28 nbrs=neighb.fit(data) # fitting the data to the object
29 distances,indices=nbrs.kneighbors(data) # finding the nearest neighbours
30
31 # Sort and plot the distances results
32 distances = np.sort(distances, axis = 0) # sorting the distances
33 distances = distances[:, 1] # taking the second column of the sorted distances
34 plt.plot(distances) # plotting the distances
35 plt.savefig('K-dist Graph - telco.png')
36 plt.show() # showing the plot
37
38 #parameter eps=18, min_samples=5
39 dbscan = DBSCAN(eps = 18, min_samples = 5)
40 dbscan.fit(data)
41
42 telco['cluster'] = dbscan.labels_.tolist()
43
44 sns.scatterplot(x='longmon', y='equipment', hue='cluster', data=telco, color='blue')
45 plt.show();
46
47 print(telco)
48 telco.to_csv('telco_clusterAssignmentsDBScan.csv')
49
50 df = pd.DataFrame(telco ,columns = ['longmon', 'tollmon', 'equipment', 'cardmon', 'wiremon',
51 |                                   'multiline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill'])
52 df['Clusters']=dbscan.labels_
53 print(df)
54 parallel_coordinates(df, 'Clusters',color=('red','blue','green', "yellow", "black"))
55 plt.show()
```


- Ανάλυση των 2 Clusters στο Excel:

Cluster -1:

Συνολικά 54 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 9.

Cluster 0:

Συνολικά 946 δεδομένα/γραμμές (πελάτες).

Τα δεδομένα με churn = 1 (αποχώρηση από την εταιρία) είναι συνολικά 265.

Άρα: Η 2η συστάδα (Cluster 0) θεωρείται ξεκάθαρα η πιο προβληματική καθώς εκεί παρατηρείτε μεγαλύτερο ποσοστό πελατών που αποχώρησαν από την επιχείρηση τηλεπικοινωνιών.

Ζήτημα B:

Μετά από τις κατάλληλες αλλαγές στους κώδικες και την εκτέλεση των αλγορίθμων κατηγοριοποίησης του Δέντρου Αποφάσεων (Decision Tree) και K-εγγύτερων Γειτόνων (kNN) αντίστοιχα, διαπίστωσα τα εξής:

- Ανάλυση Αποτελεσμάτων των x-fold cross validation:

Αρχικά, εκτελώ 10-fold cross validation για τον κάθε αλγόριθμο ξεχωριστά, επειδή θεωρείται ως μία από τις πιο αξιόπιστες μεθόδους για την αποτίμηση της **ακρίβειας (accuracy)**. Τα αποτελέσματα της εκτέλεσης είναι τα εξής:

dt_crossvalidation – telco_2023.py (Έξοδος/Output):

Accuracy of each fold - [0.7, 0.71, 0.71, 0.76, 0.65, 0.68, 0.63, 0.67, 0.58, 0.66]

Avg accuracy : 0.675

kNN_crossvalidation – telco.py (Έξοδος/Output):

Accuracy of each fold - [0.71, 0.67, 0.75, 0.81, 0.74, 0.8, 0.73, 0.71, 0.72, 0.68]

Avg accuracy : 0.732

- Ανάλυση Αποτελεσμάτων των αλγορίθμων kNN και D-Tree:

Έπειτα, εκτελώ τους αλγορίθμους **kNN** και **D-Tree** προκειμένου να βρω τις κατάλληλες τιμές της **ορθότητας (precision)**, της **ευαισθησίας (recall)** και του **F-score** αντίστοιχα. Τα αποτελέσματα της εκτέλεσης είναι τα εξής:

dt – telco_2023.py (Έξοδος/Output):

Precision: 0.345

Recall: 0.392

F-score: 0.367

kNN– telco.py (Έξοδος/Output):

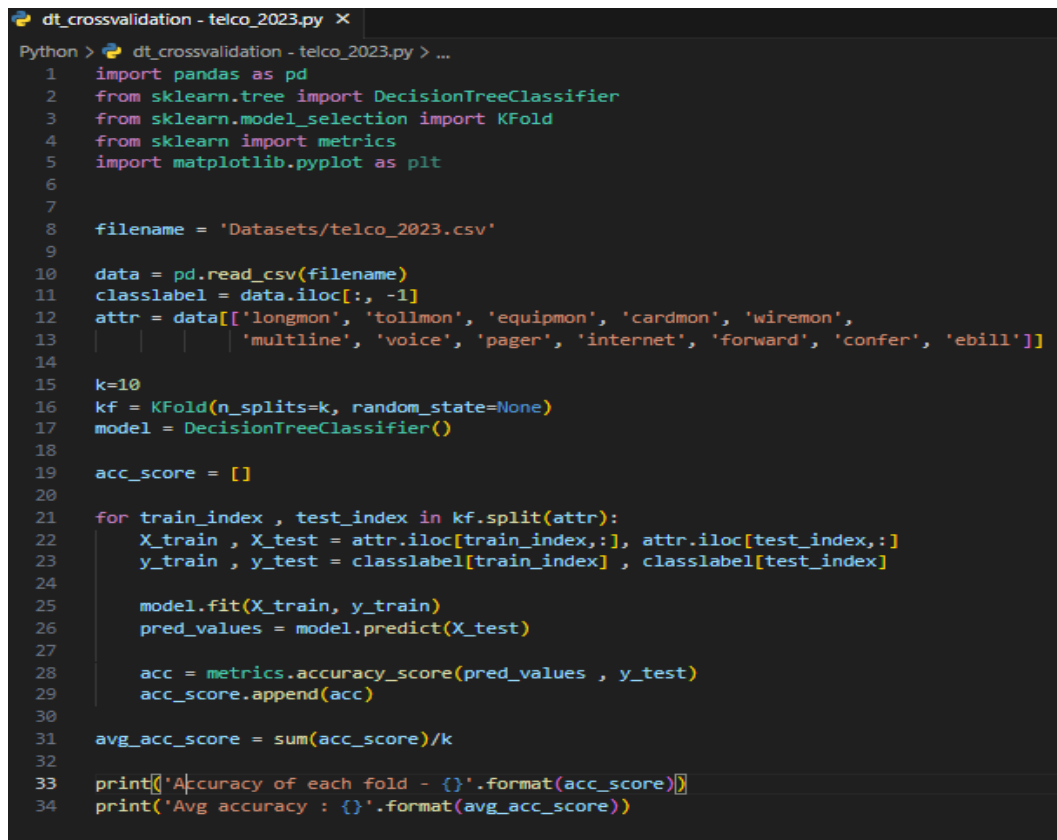
Precision: 0.514

Recall: 0.446

F-score: 0.477

Άρα: Εξαιτίας της υψηλότερης ακρίβειας του κάθε fold (Accuracy of each fold), της υψηλότερης μέσης ακρίβειας (Avg accuracy) αλλά και της καλύτερης ορθότητας (Precision), ευαισθησίας (Recall) και του F-score, διαπιστώνω ότι καλύτερος αλγόριθμος για τα συγκεκριμένα δεδομένα είναι ο **k-Nearest Neighbors (kNN)**.

Image7: dt_crossvalidation - telco_2023.py



```
dt_crossvalidation - telco_2023.py X
Python > dt_crossvalidation - telco_2023.py > ...
1  import pandas as pd
2  from sklearn.tree import DecisionTreeClassifier
3  from sklearn.model_selection import KFold
4  from sklearn import metrics
5  import matplotlib.pyplot as plt
6
7
8  filename = 'Datasets/telco_2023.csv'
9
10 data = pd.read_csv(filename)
11 classlabel = data.iloc[:, -1]
12 attr = data[['longmon', 'tollmon', 'equipmon', 'cardmon', 'wiremon',
13             'multline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill']]
14
15 k=10
16 kf = KFold(n_splits=k, random_state=None)
17 model = DecisionTreeClassifier()
18
19 acc_score = []
20
21 for train_index , test_index in kf.split(attr):
22     X_train , X_test = attr.iloc[train_index,:], attr.iloc[test_index,:]
23     y_train , y_test = classlabel[train_index] , classlabel[test_index]
24
25     model.fit(X_train, y_train)
26     pred_values = model.predict(X_test)
27
28     acc = metrics.accuracy_score(pred_values , y_test)
29     acc_score.append(acc)
30
31 avg_acc_score = sum(acc_score)/k
32
33 print('Accuracy of each fold - {}'.format(acc_score))
34 print('Avg accuracy : {}'.format(avg_acc_score))
```

Image8: dt - telco_2023.py

```
dt - telco_2023.py X
Python > dt - telco_2023.py > ...
1  import pandas as pd
2  from sklearn.tree import DecisionTreeClassifier
3  from sklearn import metrics
4  import matplotlib.pyplot as plt
5  from sklearn.model_selection import train_test_split
6
7  filename = 'Datasets/telco_2023.csv'
8
9  data = pd.read_csv(filename)
10 classlabel = data.iloc[:, -1]
11 attr = data[['longmon', 'tollmon', 'equipmon', 'cardmon', 'wiremon',
12             'multiline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill']]
13
14 #Splitting the data into training and testing sets.
15 attr_train, attr_test, class_train, class_test = train_test_split(attr, classlabel, test_size=0.2, random_state=42)
16
17 model = DecisionTreeClassifier()
18 model.fit(attr_train, class_train)
19
20 predictions = model.predict(attr_test)
21
22 #Precision, Recall, F-score.
23 precision = metrics.precision_score(class_test, predictions)
24 recall = metrics.recall_score(class_test, predictions)
25 fscore = metrics.f1_score(class_test, predictions)
26 print('Precision:', precision)
27 print('Recall:', recall)
28 print('F-score:', fscore)
```

Image9: kNN_crossvalidation - telco_2023.py

```
kNN_crossvalidation - telco_2023.py X
Python > kNN_crossvalidation - telco_2023.py > ...
1  import pandas as pd
2  from sklearn.neighbors import KNeighborsClassifier
3  from sklearn.model_selection import KFold
4  from sklearn import metrics
5  import matplotlib.pyplot as plt
6
7  filename = 'Datasets/telco_2023.csv'
8
9  data = pd.read_csv(filename)
10 classlabel = data.iloc[:, -1]
11 attr = data[['longmon', 'tollmon', 'equipmon', 'cardmon', 'wiremon',
12             'multiline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill']]
13
14 k=10
15 kf = KFold(n_splits=k, random_state=None)
16 model = KNeighborsClassifier(n_neighbors=5)
17
18 acc_score = []
19
20 for train_index, test_index in kf.split(attr):
21     X_train, X_test = attr.iloc[train_index, :], attr.iloc[test_index, :]
22     y_train, y_test = classlabel[train_index], classlabel[test_index]
23
24     model.fit(X_train, y_train)
25     pred_values = model.predict(X_test)
26
27     acc = metrics.accuracy_score(pred_values, y_test)
28     acc_score.append(acc)
29
30 avg_acc_score = sum(acc_score)/k
31
32 print('Accuracy of each fold - {}'.format(acc_score))
33 print('Avg accuracy : {}'.format(avg_acc_score))
```

Image10: kNN - telco_2023.py

```
kNN - telco_2023.py X
Python > kNN - telco_2023.py > ...
1  import pandas as pd
2  from sklearn.neighbors import KNeighborsClassifier
3  from sklearn.model_selection import train_test_split
4  from sklearn import metrics
5
6  filename = 'Datasets/telco_2023.csv'
7
8  data = pd.read_csv(filename)
9  classlabel = data.iloc[:, -1]
10 attr = data[['longmon', 'tollmon', 'equipmon', 'cardmon', 'wiremon',
11             'multline', 'voice', 'pager', 'internet', 'forward', 'confer', 'ebill']]
12
13 #Splitting the data into training and testing sets.
14 attr_train, attr_test, class_train, class_test = train_test_split(attr, classlabel, test_size=0.33, random_state=42)
15
16 model = KNeighborsClassifier(n_neighbors=5)
17 model.fit(attr_train, class_train)
18
19 predictions = model.predict(attr_test)
20
21 #Precision, Recall, F-score.
22 precision = metrics.precision_score(class_test, predictions)
23 recall = metrics.recall_score(class_test, predictions)
24 fscore = metrics.f1_score(class_test, predictions)
25 print('Precision:', precision)
26 print('Recall:', recall)
27 print(['F-score:', fscore])
```