Διερεύνηση του Ρυθμού Αποχωρήσεων Πελατών Επιχείρησης Τηλεπικοινωνιών

Βελλής Εμμανουήλ
AM:2019019
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων, Διεθνές Πανεπιστήμιο της Ελλάδος,
Σίνδος, Θεσσαλονίκη, Ελλάδα
iee2019019@iee.teithe.gr

***Abstract*—Στην εργασία θα αναλυθεί ο τρόπος με τον οποίο η εταιρία μπορεί να (α) να ομαδοποιήσει τους πελάτες της με βάση συγκεκριμένα χαρακτηριστικά τους και (β)να κατασκευάσει ένα μοντέλο πρόβλεψης για την αποχώρηση ενός πελάτη από την εταιρεία με τη βοήθεια της γλώσσας προγραμματισμού python**

.

I. Εισαγωγη

Στα πλαίσια της παρούσας εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού python και συγκεκριμένα η βιβλιοθήκη scikit-learn (sklearn) για να ομαδοποιηθούν οι πελάτες από τα δεδομένα που έχει συλλέξει η εταιρία ανάλογα με τα χαρακτηριστικά τους με 3 αλγορίθμους ομαδοποίησης .Έπειτα να συγκριθούν οι κατηγοριοποιητές : (α) δέντρο αποφάσεων και (β) κ εγγύτεροι γείτονες να προταθεί ο κατάλληλος κατηγοριοποιητής ανάλογα με την πρόβλεψη απώλειας πελάτη.

Όνομα	**Περιγραφή**	**Τιμές**
region	Γεωγραφική Περιοχή	{1, 2, 3, 4, 5}
marital	Οικογενειακή Κατάσταση	1: Παντρεμένος 0: Ανύπαντρος
gender	Φύλο	0: Άντρας 1: Γυναίκα
longmon	Δείκτης χρήσης Υπεραστικών κλήσεων	Πραγματικές τιμές*
tollmon	Δείκτης χρήσης Υπεραστικών κλήσεων	Πραγματικές τιμές*
equipmon	Δείκτης χρήσης Εξοπλισμού	Πραγματικές τιμές*
cardmon	Δείκτης χρήσης Κάρτας Κλήσης	Πραγματικές τιμές*
wiremon	Δείκτης χρήσης Ασύρματου Δικτύου	Πραγματικές τιμές*
multline	Υπηρεσία Πολλαπλών Γραμμών	0: ΟΧΙ 1: ΝΑΙ
voice	Υπηρεσία Φωνητικής	0: ΟΧΙ 1: ΝΑΙ
pager	Υπηρεσία Paging	0: ΟΧΙ 1: ΝΑΙ
internet	Υπηρεσία Internet	0: ΟΧΙ 1: ΝΑΙ
callid	Υπηρεσία Αναγνώρισης Κλήσης	0: ΟΧΙ 1: ΝΑΙ
callwait	Υπηρεσία Αναμονής Κλήσεων	0: ΟΧΙ 1: ΝΑΙ
forward	Υπηρεσία Προώθησης Κλήσεων	0: ΟΧΙ 1: ΝΑΙ
confer	Υπηρεσία Διάσκεψης	0: ΟΧΙ 1: ΝΑΙ
ebill	Υπηρεσία Ηλεκτρονικής Πληρωμής	0: ΟΧΙ 1: ΝΑΙ
custcat	Κατηγορία Πελάτη	1: Βασικές Υπηρεσίες, 2: Ηλεκτρονικές Υπη 3: Επιπρόσθετες Υπηρεσίες, 4: Σύνολο Υπηρ
churn	Αποχώρηση από την Εταιρεία	0: ΟΧΙ 1: ΝΑΙ

Πίνακας 1 ΔΕΔΟΜΕΝΑ*μεγαλύτερη τιμή ενός δείκτη χρήσης υποδεικνύει μεγαλύτερη χρήση

Τα παραπάνω δεδομένα είναι αποθηκευμένα στο αρχείο telco_2023.csv με την ίδια σειρά.

Για την υλοποίηση της εργασίας χρησιμοποιήθηκαν :

-το λογισμικό spyder αφού σκοπός είναι η ανάλυση των δεδομένων και την παραγωγή γραφημάτων και όχι δημιουργία εκτελέσιμου αρχείου

-το διαδραστικό περιβάλλον τερματικού Ipython για απευθείας εκτέλεση κώδικα και προσωρινή αποθήκευση μεταβλητών

-Βιβλιοθήκες python:

- scikit-learn
- seaborn
- yellowbrick

- kneed
- scipy
- numpy
- pandas

II. ΟΜΑΔΟΠΟΙΗΣΗ (CLUSTERING)

Αρχικά δημιουργήθηκε ένα python script το Ergasia.py στο οποίο θα γίνει η ομαδοποίηση και διαβάστηκε το αρχείο με τα δεδομένα ως dataframe της βιβλιοθήκης pandas και αφαιρούνται οι στήλες που δεν θεωρούνται σημαντικές από την εταιρία:

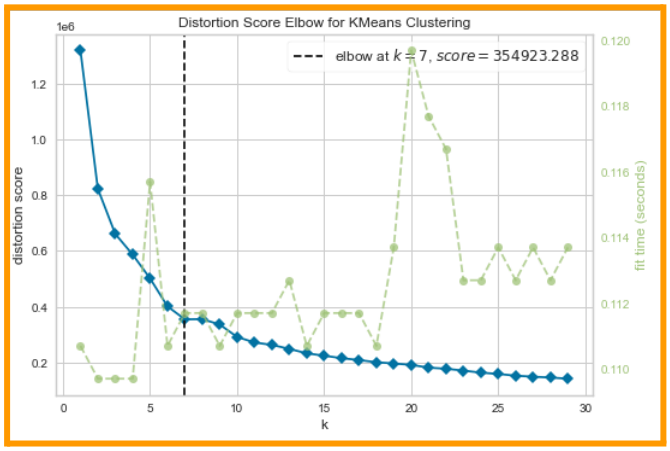
```
data = pd.read_csv('./telco_2023.csv')
df = pd.DataFrame(data, columns=data.columns)
X = df.drop(['region', 'marital', 'gender', 'callid', 'callwait', 'custcat', 'churn'], axis=1)
```

Έπειτα με ένα απλό μενού ζητείται από το χρήστη να επιλέξει έναν από τους παρακάτω αλγορίθμους ομαδοποίησης

A. k-means

a. Υλοποίηση

με την επιλογή kmeans δημιουργείται ένα μοντέλο ομαδοποιητή kmeans και με τη χρήση της μεθόδου kElbowVisualizer() οπτικοποιείται ο βαθμός παραμόρφωσης



Εικόνα 1 *ellbow*

και επιλέγεται ο αγκώνας του διαγράμματος με τις μεθόδους της yellowbrick.Στη συνέχεια με την επιλογή ως k συστάδες

```
if choice==1 :
    #elbow
    model = KMeans(random_state=42,n_init='auto')
    visualizer = KElbowVisualizer(model, k=(1,30))
    visualizer.fit(X)
    visualizer.show()



    # επιλογή του elbow
    k = visualizer.elbow_value_
    final_model = KMeans(n_clusters=k, random_state=42)
    final_model.fit(X)

    # Obtain the cluster labels for the data points
    cluster_labels = final_model.labels_
    kmeansdata=df
    kmeansdata['cluster']=cluster_labels+1
    num_unique_values = len(kmeansdata['cluster'])


    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(kmeansdata)
```

την τιμή στον ‘αγκώνα γίνεται η ομαδοποίηση και οι ταμπέλες (labels) των cluster εισέρχονται ως καινούργια

Εικόνα 2

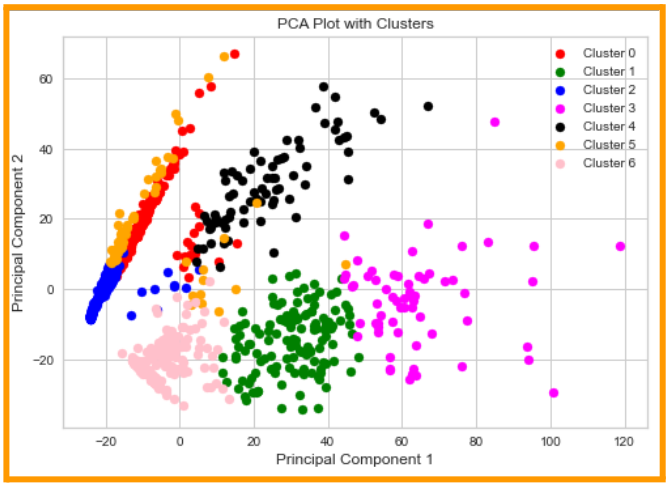
στήλη στο dataframe για κάθε στοιχείο και αποθηκεύονται σε ένα νέο dataframe ονόματι kmeans data.Τέλος γίνεται ανάλυση 2 κύριων συνιστωσών (Principal Component Analysis, PCA) για τη μείωση της διάστασης των δεδομένων, διατηρώντας παράλληλα τη μέγιστη ποσοστιαία

διακύμανση των δεδομένων στις νέες κύριες συνιστώσες. για την οπτικοποίηση των ομάδων με την matplotlib .

```
#οπτικοποίηση αποτελεσμάτων
unique_labels = np.unique(cluster_labels)
for label in unique_labels:
    mask = (cluster_labels == label)
    plt.scatter(principal_components[mask, 0], principal_

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Plot with Clusters')
plt.legend()
plt.show()
```

Εικόνα 3



Εικόνα 4 *Απεικόνιση PCA*

b. Συμπεράσματα

```
churn_rate = kmeansdata.groupby('cluster')['churn'].mean()
problematic_cluster = churn_rate.idxmax()
print("Problematic cluster is ", problematic_cluster)
print(churn_rate)
```

Εικόνα 5

Υπολογίζεται το churn rate κάνοντας groupby με το το cluster τα δεδομένα και έπειτα βρίσκεται ο μέσος όρος πελατών που έφυγαν για κάθε cluster για το κάθε ένα.

Αποτέλεσμα:

```
Problematic claster is  7
cluster
1    0.139785
2    0.469697
3    0.189542
4    0.306452
5    0.219178
6    0.062500
7    0.502825
Name: churn, dtype: float64
```

Εικόνα 6

Άρα το προβληματικό cluster είναι το 7ο με 50,2% των πελατών του να μην είναι πια συνδρομητές

B. Ιεραρχική συσταδοποίηση (agglomerative clustering)

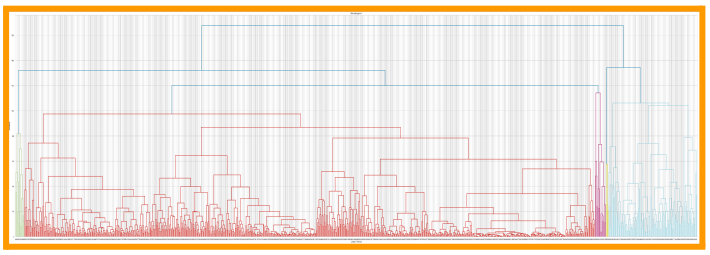
a. Υλοποίηση

Για τη δεύτερη επιλογή δημιουργούμε ένα αντικείμενο δενδρογράμματος με την βιβλιοθήκη scipy με σύνδεση βάση της ευκλείδειας απόστασης μεταξύ των σημείων.

```
Z = linkage(X, method='complete', metric='euclidean')

Plot the dendrogram
plt.figure(figsize=(70,20))
dn = dendrogram(Z)
plt.title('Dendrogram')
plt.xlabel('Data Points')
plt.ylabel('Distance')
plt.show()
```

Εικόνα 7 *δημιουργία δενδρογράμματος*



Εικόνα 8 *δενδρόγραμμα*

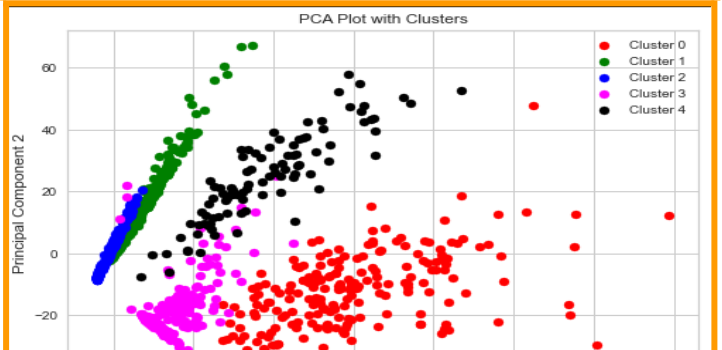
Με τα διαφορετικά χρώματα μπορούμε να διακρίνουμε τα clusters που δημιουργούνται.Σύμφωνα με τη βιβλιογραφία της βιβλιοθήκης η αλλαγή χρώματος γίνεται σε μεγάλη διαφορά απόστασης από διακλάδωση σε διακλάδωση ώστε να φαίνεται οπτικά ο αριθμός των cluster σαν βοήθημα.Οπότε αν μετρηθούν τα χρώματα και αφαιρεθεί το μπλέ της ρίζας ορίζεται το k σαν ξεχωριστά χρώματα -1 .Πίσω στον κώδικα:

```
unique_colors=set(dn['color_list'])
k=len(unique_colors)-1
```

```
unique_colors=set(dn['color_list'])
k=len(unique_colors)-1


agglomerative_model = AgglomerativeClustering(n_clusters=k)
cluster_labels = agglomerative_model.fit_predict(X)
agnesdata=df
agnesdata['cluster']=cluster_labels
pca = PCA(n_components=2)
principal_components = pca.fit_transform(agnesdata)

churn_rate = agnesdata.groupby('cluster')['churn'].mean()
problematic_cluster = churn_rate.idxmax()
```



Εικόνες 9,10 οπτικοποίηση και αποτελέσματα Ιεραρχικής ομαδοποίησης.

Παρόμοια με τον kmeans κάνουμε clustering με τη μέθοδο του agglomerative clustering αυτή τη φορά δημιουργούμε το agnesdata dataframe και αναλύουμε σε 2 συνιστώσες .

b. Συμπεράσματα

Ακριβώς όπως πρίν υπολογίζεται το προβληματικό cluster

```
Problematic cluster is  3
cluster
0    0.430693
1    0.127962
2    0.176471
3    0.454545
4    0.223404
Name: churn, dtype: float64
```

C. Dbscan

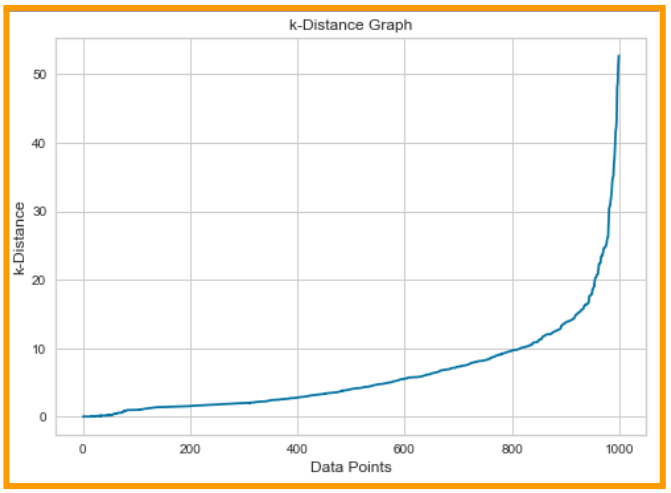
a. Υλοποίηση:

Δημιουργείται για αρχή ένα διάγραμμα k αποστάσεων για να βρεθεί η τιμή της απόστασης στο γόνατο ώστε να οριστεί το ε για το dbScan και γίνεται plot το γράφημα.Επιλέχθηκε το k=2 επειδή εμφανίζει πιο καθαρά το 'γόνατο' του γραφήματος.

```
k = 2 # Ορίζω την τιμή του k για το γράφημα της k-απόστασης
neigh = NearestNeighbors(n_neighbors=k+1) # Λαμβάνω υπόψη k+1 γείτονες για να συμπεριλάβει και τ
neigh.fit(X)
distances, indices = neigh.kneighbors(X)
distances = sorted(distances[:, k], reverse=False) # Ταξινομώ τις αποστάσεις σε αύξουσα σειρά
sorted_indices = np.argsort(distances) # Ταξινομώ τα indexes βάσει των ταξινομημένων αποστάσεων
sorted_distances = np.array(distances)[sorted_indices] # Ταξινομώ τις αποστάσεις βάσει των ταξιν

plt.plot(sorted_indices, sorted_distances)
plt.xlabel('Data Points')
plt.ylabel('k-Distance')
plt.title('k-Distance Graph')
plt.show()
```

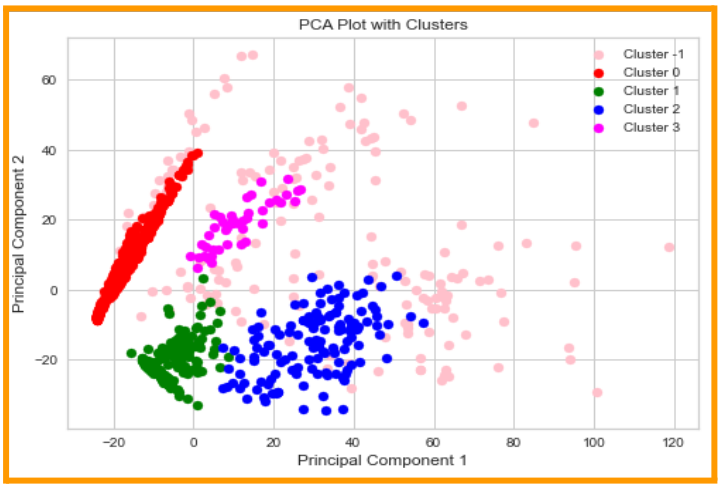
Εικόνα 11



Εικόνα 12 *-distance Graph*

Από το 'γόνατο' προκύπτει ότι η κατάλληλη τιμή για το ε είναι το 15.Και με minpts=12 (όσα είναι και τα

χαρακτηριστικά του dataframe που εξετάζουμε) πράγματι δημιουργούνται φαινομενικά σωστά clusters και outliers όσο γίνεται σε 2 διαστάσεις με την PCA ανάλυση.



Εικόνα 13 *αποτελέσματα dbscan ροζ σημεια= outliers*

C. Συμπεράσματα

```
Problematic cluster is  2
cluster
-1    0.195122
 0    0.162272
 1    0.491018
 2    0.503759
 3    0.302326
Name: churn, dtype: float64
```

Εικόνα 14

Παρατηρούμε πως με το dbscan έχουμε πιο περίπλοκα και ακριβή σχήματα clusters καθώς και μια πιο καθαρή εικόνα των outliers χωρις να αλλοιώνουν τους υπολογισμούς των ομάδων.Επίσης παρατηρούμε πως τα προβληματικά clusters σε κάθε μέθοδο ομαδοποίησης τείνουν να είναι κάτω και αριστερά στην PCA ανάλυση.Έτσι μπορούμε να δούμε για παράδειγμα χαρακτηριστικά που τείνουν να έχουν τα προβληματικά cluster.Για παράδειγμα τα προβληματικά cluster τείνουν να έχουν υψηλές τιμές equipmon.

III. Classification

A. Κλήση των μετρικών για συγκριση κατηγοριοποιητών.

Αρχικά σε ένα νέο python script Ergasia_metrics.py δημιουργήθηκε το dataframe απο το csv αρχείο όπως πριν και αφαιρέθηκαν τα χαρακτηριστικά που δεν θέλουμε για να πετύχουμε καλύτερες μετρικές.Έπειτα χωρίστηκε το dataframe σε χαρακτηριστικά X και στόχους Y,δηλαδή την στήλη 'churn' και με χρήση ενός standardScaler() της scikit-learn τυποποιούνται τα δεδομένα επειδή έπειτα από δοκιμές, σε δεύτερο χρόνο, επιτεύχθησαν καλύτερα αποτελέσματα.Όπως φαίνεται στην εικόνα 15

```
filename = './telco_2023.csv'
df = pd.read_csv(filename)  # Replace 'yofrom sklearn.preprocessing import StandardScaler


df=df.drop(['region','custcat','marital','gender','ebill','confer','pager','forward'],axis=1)
X = df.drop('churn', axis=1)  # Features
y = df['churn']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Εικόνα 15

```
y_pred_all=[]
y_test_all=[]
for j in range(100):

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    dt = DecisionTreeClassifier()  # Initialize the Decision Tree classifier
    dt.fit(X_train, y_train)  # Train the classifier using the training data
    y_pred=dt.predict(X_test)
    nrow=pd.DataFrame([{'accuracy':accuracy_score(y_test, y_pred),'recall':recall_score(y_test, y_pred, pos_label=1),'precision':pre
    dtmetrics=pd.concat([dtmetrics,nrow],ignore_index=True)
    y_pred_all.extend(y_pred)
    y_test_all.extend(y_test)
```

Εικόνα 16

Για να μετρηθούν τα αποτελέσματα των 2 κατηγοριοποιητών εγιναν παρόμοια και στους 2 μέσα σε μια for 100 φορές για το καθένα:

- διαχωρισμός train-test και τυποποίηση δεδομένων με τον standardscaler
- δημιουργία του κατηγοριοποιητή
- καταχώρηση των μετρικών accuracy,recall και precision για κάθε επανάληψη σε έναν πίνακα με positive class το 1(δηλαδή οταν ένας πελάτης έχει αποχωρήσει από την εταιρία)
- εκτύπωση των αποτελεσμάτων
- αποτύπωση του confusion matrix για τον καθένα μετά το τέλος ολων των επαναλήψεων

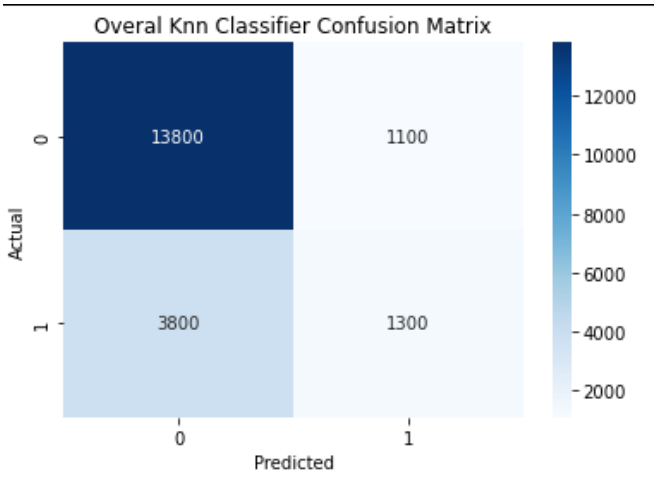
Αφαίρεσα επίσης τα χαρακτηριστικά που πίστευα ότι θα εκπαίδευαν καλύτερα και με περισσότερη ακρίβεια τους κατηγοριοποιητές.Πράγματι μόλις αφαιρέθηκαν αριθμητικά κωδικοποιημένα χαρακτηριστικά και ανούσια ως τη παραμονή των χρηστών επι το πλείστον όπως η οικογενειακή κατάσταση ή το φύλλο και πειραματίζοντας με άλλες λιγότερο εμφανείς χαρακτηριστικά κατέληξα στο καλύτερο αποτέλεσμα που μπορούσα.

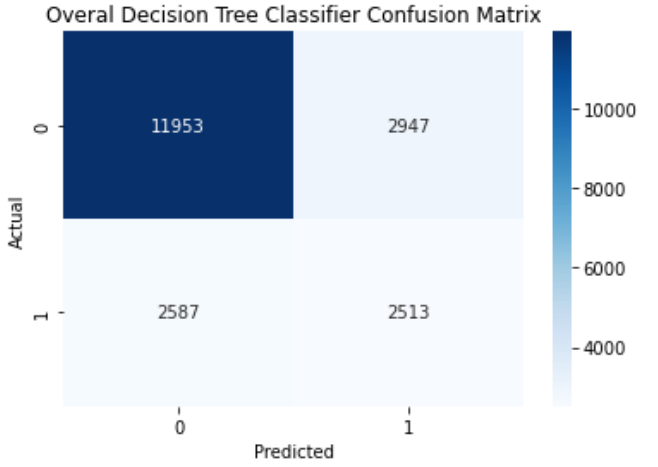
```
 metrics in 100 training cycles of decision tree")
y:", dtmetrics["accuracy"].mean())
", dtmetrics["recall"].mean())
on:", dtmetrics["precision"].mean())

_matrix(y_test_all, y_pred_all)
, annot=True, fmt="d", cmap="Blues")
ral Decision Tree Classifier Confusion Matrix")
edicted')
tual')
```

Εικόνα 17

B. Συμπεράσματα-Αποτελέσματα

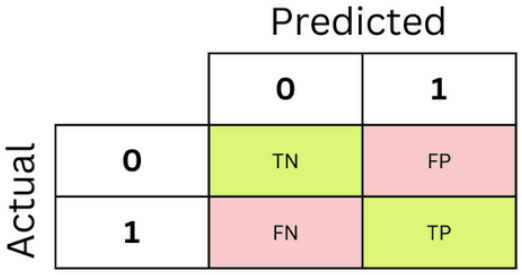





```
mean of metrics in 100 training cycles KNN
Accuracy: 0.7549999999999999
Recall: 0.25490196078431365
Precision: 0.5416666666666666
mean of metrics in 100 training cycles of decision tree
Accuracy: 0.7233000000000002
Recall: 0.4927450980392157
Precision: 0.4598828316006355
```

IV. Πορισμα

Αν και ο knn κατηγοριοποιητής είναι πιο ακριβής πρέπει η εταιρία να λάβει υπόψη και τις άλλες μετρικές το recall στον knn κατηγοριοποιητή είναι πολύ μικρό σε σχέση με αυτόν του δέντρου αποφάσεων.Σύμφωνα με τους τύπους των recall και precision:



$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

καταλαβαίνουμε πως το precision μετράει το ποσοστό των σωστών προβλέψεων αποχώρησης προς το σύνολο των θετικών προβλέψεων ενώ το recall μετράει το σύνολο των σωστών προβλέψεων αποχώρησης προς το σύνολο των πραγματικών πελατών που αποχώρησαν.Αφού το θέμα μας είναι να προβλεφθούν όσο πιο σωστά γίνεται οι πελάτες που πρόκειται να αποχωρήσουν θεωρώ πως πρέπει η εταιρία να επιλέξει τον κατηγοριοποιητή δέντρου αποφάσεων γιατί έχει σημαντικά μεγαλύτερη τιμή recall με ελάχιστα μικρότερες τιμές των άλλων δύο μετρικών από τον knn. Με απλά λόγια θα από όλους τους πελάτες που είναι να φύγουν με υψηλό recall θα προβλέψουμε περισσότερους από αυτούς πριν το κάνουν.

Βιβλιογραφια

[1] https://www.kdnuggets.com/2022/11/confusion-matrix-precision-recall-explained.html
[2] https://www.anaconda.com
[3] https://scikit-learn.org/stable/
[4] https://www.scikit-yb.org/en/latest/api/cluster/elbow.html
[5] https://seaborn.pydata.org/282a5e860d456fab2df24a16b
[6] https://www.spyder-ide.org/