LEUMENI TCHOUAMENI LIONEL FLORENT_MASTER PRO_GL0 2 ENSPD_DOUALA CAMEROUN

TP : Prédiction du taux de désabonnement ou perte de clients

In [2]:
```
import pandas as pd
import os
import numpy as np
import matplotlib.ticker as mtick
import seaborn as sns
import matplotlib.pyplot as plt
import math
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import precision_recall_curve, auc, f1_score, ConfusionMatrixD
```

Chargement des données

Le jeu de données se présente sous le format (format .csv), nous utilisons donc pandas.read_csv pour charger l'ensemble de données.

In [4]:
```
# Import dataset and make a CustomerID column to index
os.chdir('C:/Users/user/Desktop/data science/')
df= pd.read_csv('Data_telco.csv',index_col='customerID')

df.head()
df.shape
```

Out[4]:
```
(7043, 20)
```

Le jeu de données contient 7 043 lignes et 20 colonnes. Ensuite, nous vérifierons les valeurs uniques de chaque colonne des données.

In [46]:
```
df_columns = df.columns.tolist()
for column in df_columns:
    print(f"{column} unique values : {df[column].unique()}")
```

```
gender unique values : ['Female' 'Male']
SeniorCitizen unique values : [0 1]
Partner unique values : ['Yes' 'No']
Dependents unique values : ['No' 'Yes']
tenure unique values : [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 3
0 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]
PhoneService unique values : ['No' 'Yes']
MultipleLines unique values : ['No phone service' 'No' 'Yes']
InternetService unique values : ['DSL' 'Fiber optic' 'No']
OnlineSecurity unique values : ['No' 'Yes' 'No internet service']
OnlineBackup unique values : ['Yes' 'No' 'No internet service']
DeviceProtection unique values : ['No' 'Yes' 'No internet service']
TechSupport unique values : ['No' 'Yes' 'No internet service']
StreamingTV unique values : ['No' 'Yes' 'No internet service']
StreamingMovies unique values : ['No' 'Yes' 'No internet service']
Contract unique values : ['Month-to-month' 'One year' 'Two year']
PaperlessBilling unique values : ['Yes' 'No']
PaymentMethod unique values : ['Electronic check' 'Mailed check' 'Bank transfer (a
utomatic)'
 'Credit card (automatic)']
MonthlyCharges unique values : [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges unique values : ['29.85' '1889.5' '108.15' ... '346.45' '306.6' '684
4.5']
Churn unique values : ['No' 'Yes']
```

In [47]:
```
# Statistic descriptive
df.describe()
```

Out[47]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7043 entries, 7590-VHVEG to 3186-AJIEK
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7043 non-null   object 
 1   SeniorCitizen     7043 non-null   int64  
 2   Partner           7043 non-null   object 
 3   Dependents        7043 non-null   object 
 4   tenure            7043 non-null   int64  
 5   PhoneService      7043 non-null   object 
 6   MultipleLines     7043 non-null   object 
 7   InternetService   7043 non-null   object 
 8   OnlineSecurity    7043 non-null   object 
 9   OnlineBackup      7043 non-null   object 
 10  DeviceProtection  7043 non-null   object 
 11  TechSupport       7043 non-null   object 
 12  StreamingTV       7043 non-null   object 
 13  StreamingMovies   7043 non-null   object 
 14  Contract          7043 non-null   object 
 15  PaperlessBilling  7043 non-null   object 
 16  PaymentMethod     7043 non-null   object 
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   object 
 19  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB
```

Il y a quelque chose qui ne va pas avec l'ensemble de données sur la colonne TotalCharges, il doit s'agir d'un float et non d'un objet, nous allons donc le changer en float.

In [49]:
```
# Change TotalCharges to float
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")
```

Vérification du mode de paiement

In [50]:
```
df["PaymentMethod"].unique()
```

Out[50]:
```
array(['Electronic check', 'Mailed check', 'Bank transfer (automatic)',
       'Credit card (automatic)'], dtype=object)
```

Dans le mode de paiement, il y a le mot automatique et nous le supprimerons car si nous visualisons, l'étiquette est longue.

In [51]:
```
# Delete "automatic" from PaymentMethod
df["PaymentMethod"] = df["PaymentMethod"].str.replace(" (automatic)", "", regex=Fal
```

Valeurs manquantes

In [52]:
```
features_na = [feature for feature in df.columns if
df[feature].isnull().sum() > 1]
for feature in features_na:
    print(f"{feature}, {round(df[feature].isnull().mean(), 4)} % Missing values")
```

```
TotalCharges, 0.0016 % Missing values
```

Le jeu de données comporte des valeurs manquantes. TotalCharges est de 0,0016 %, ce qui représente donc un petit pourcentage.

In [53]:
```
# Check observation of missing values
df[df[features_na[0]].isnull()]
```

Out[53]:

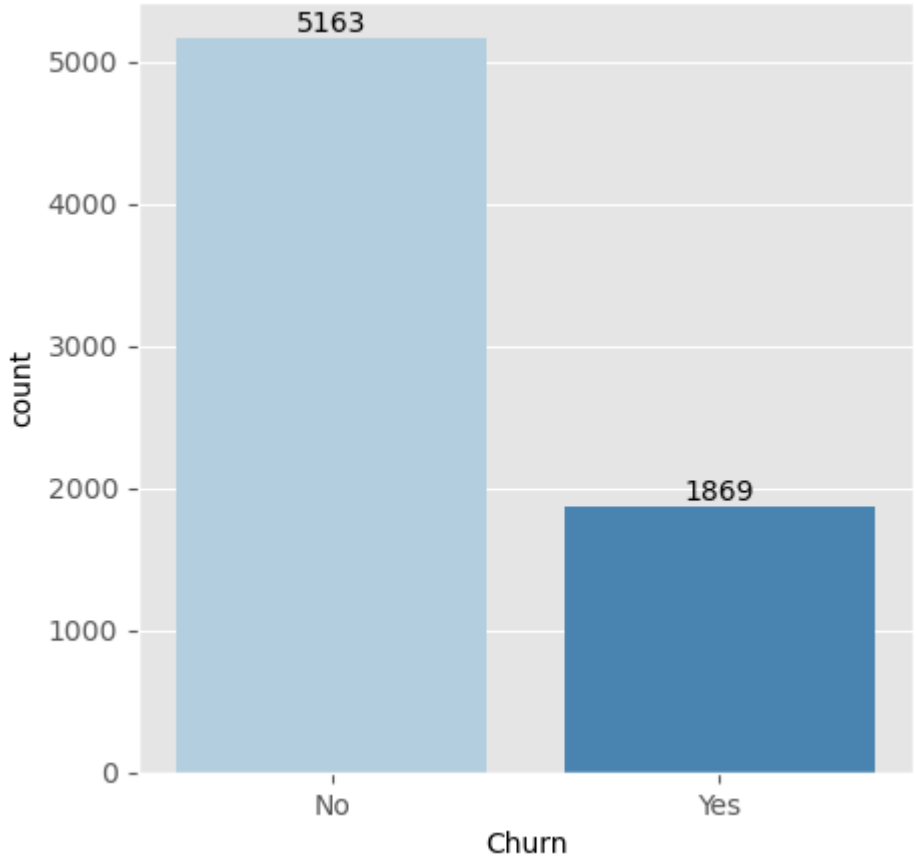
customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Int
4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	
3115-CZMZD	Male	0	No	Yes	0	Yes	No	
5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	
4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	
1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	
7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	
3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	
2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	
2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	
4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	
2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	

Comme indiqué ci-dessus, nous voyons que toutes les durées sont nulles, nous supprimerons donc les valeurs manquantes.

In [54]:
```
# Drop missing values
df.dropna(inplace=True)
```

Visualisation des variables cibles

In [55]:
```
#Apply the ggplot style
plt.style.use("ggplot")
plt.figure(figsize=(5,5))
ax = sns.countplot(x = df["Churn"],palette="Blues")
ax.bar_label(ax.containers[0])
plt.show()
```



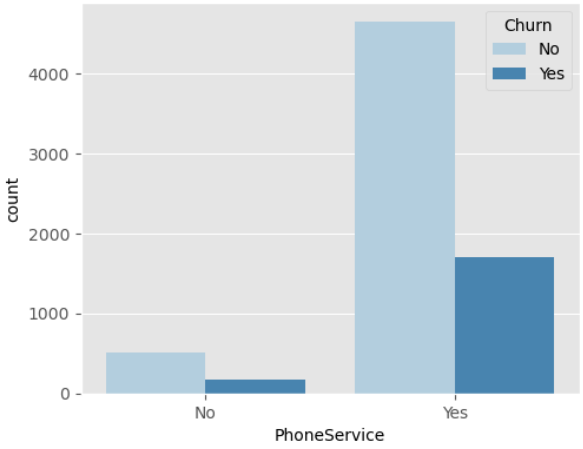
Le graphique à barres suivant montre la variable cible de désabonnement oui et non. La proportion de désabonnement est un ensemble de données déséquilibré car les deux classes ne sont pas également réparties. Pour y faire face, le rééchantillonnage serait une approche appropriée. Pour simplifier les choses, nous conserverons l'ensemble de données déséquilibré et utiliserons de nombreuses matrices d'évaluation pour évaluer les modèles.

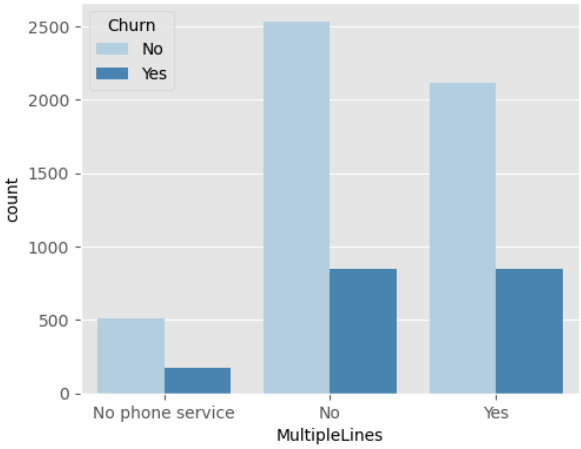
Services d'analyse pour chaque client

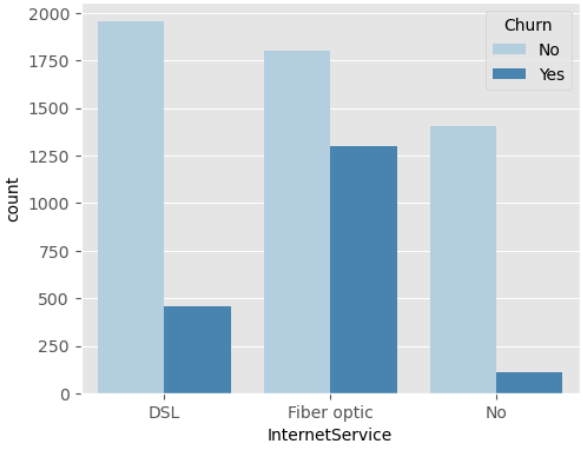
In [56]:
```
#Make a function to plot categorical data according to target
def plot_categorical_to_target(df,categorical_values, target):
        number_of_columns = 2
        number_of_rows = math.ceil(len(categorical_values)/2)
        fig = plt.figure(figsize = (12, 5*number_of_rows))
        for index, column in enumerate(categorical_values, 1):
            ax = fig.add_subplot(number_of_rows,number_of_columns,index)
            ax = sns.countplot(x = column, data = df, hue = target, palette="Blues'
        ax.set_title(column)
        return plt.show()
```

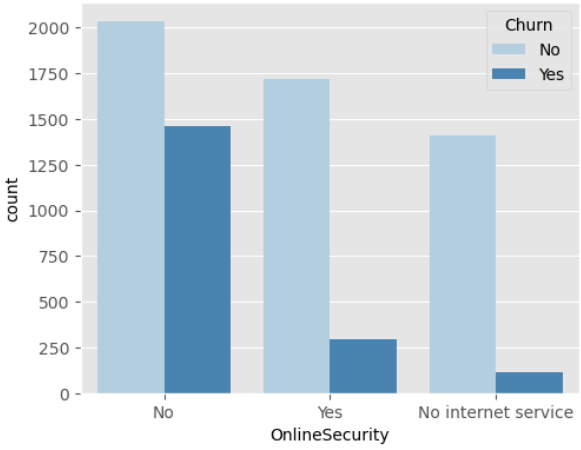
Nous évaluerons un objectif de pourcentage pour chaque colonne de services client (PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies).

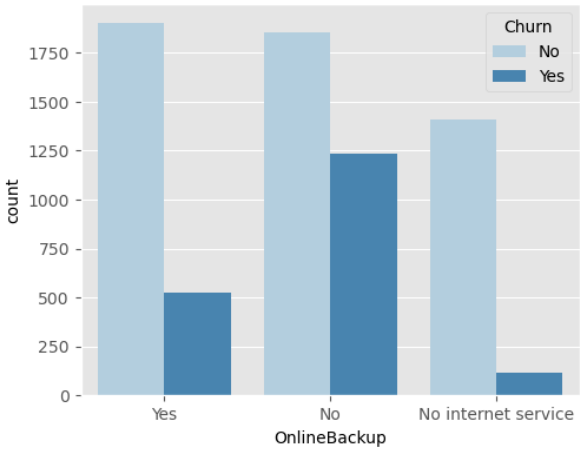
In [57]:
```
customer_services = ["PhoneService","MultipleLines","InternetService","OnlineSecuri
plot_categorical_to_target(df,customer_services, "Churn")
```

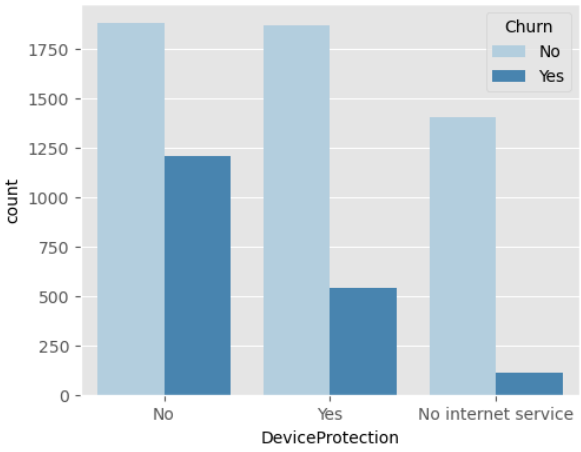

Churn
No
Yes
count
PhoneService
No
Yes

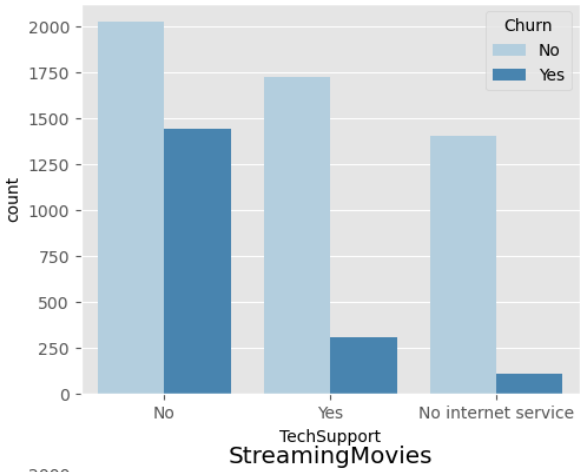

Churn
No
Yes
count
MultipleLines
No phone service
No
Yes

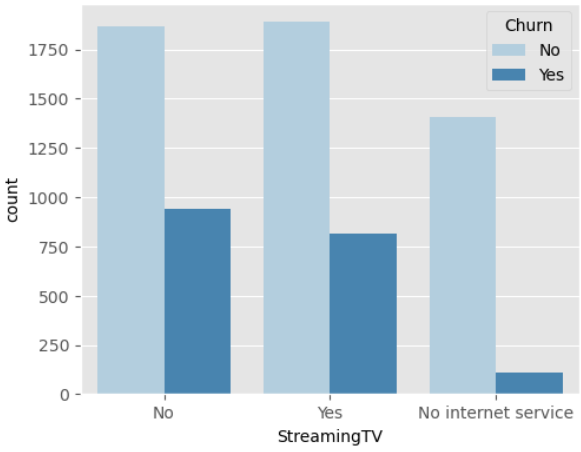

Churn
No
Yes
count
InternetService
DSL
Fiber optic
No

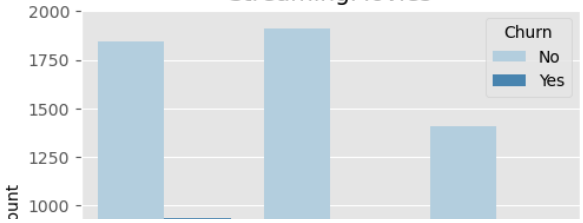

Churn
No
Yes
count
OnlineSecurity
No
Yes
No internet service

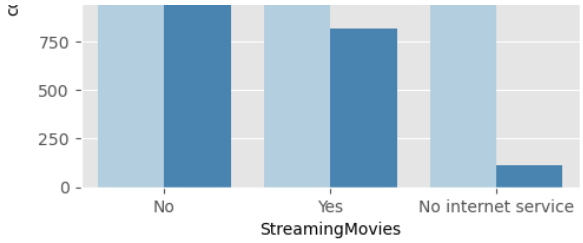

Churn
No
Yes
count
OnlineBackup
Yes
No
No internet service


Churn
No
Yes
count
DeviceProtection
No
Yes
No internet service


Churn
No
Yes
count
TechSupport
No
Yes
No internet service


Churn
No
Yes
count
StreamingTV
No
Yes
No internet service


StreamingMovies
Churn
No
Yes

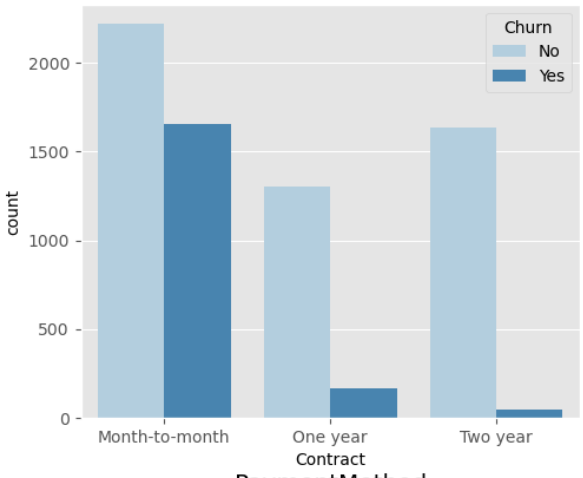


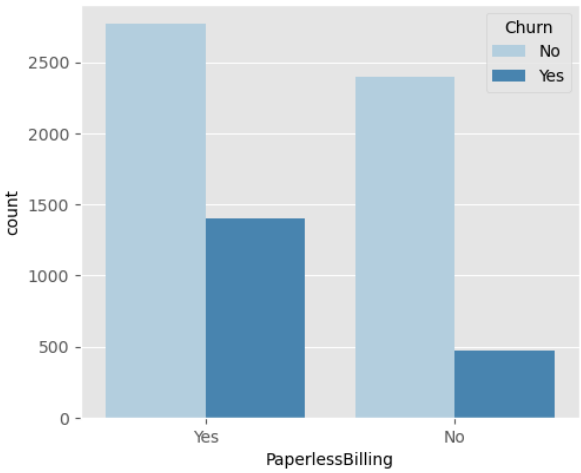
Nous pouvons extraire les conclusions suivantes en évaluant les attributs du service • Le taux de désabonnement modérément plus élevé pour les clients disposant du service téléphonique. • Les clients disposant d'un service Internet par fibre optique ont un taux de désabonnement plus élevé que celui du DSL et du No. • Le taux de désabonnement beaucoup plus élevé pour les clients sans sécurité en ligne. • Les clients qui n'ont pas accès au support technique ont tendance à partir plus fréquemment que ceux qui y ont accès. • Les clients sans sauvegarde en ligne ni protection des appareils ont un taux de désabonnement plus élevé.

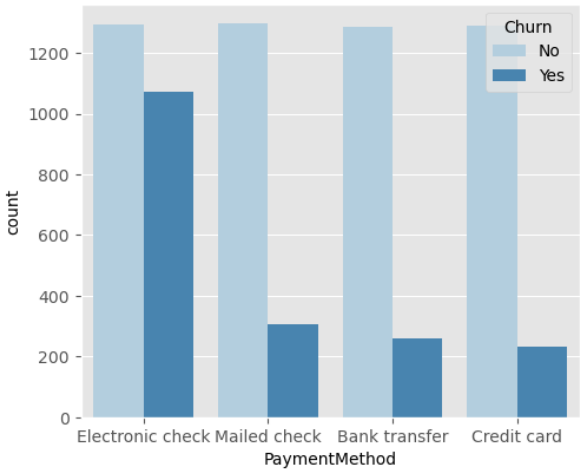
Analyse des informations sur le compte client – Variables catégorielles

Nous évaluerons le pourcentage de désabonnement pour chaque information de compte client (Contrat, PaperlessBilling, PaymentMethod)

In [58]:
```
customer_account_cat = ["Contract","PaperlessBilling","PaymentMethod"]
plot_categorical_to_target(df,customer_account_cat,"Churn")
```





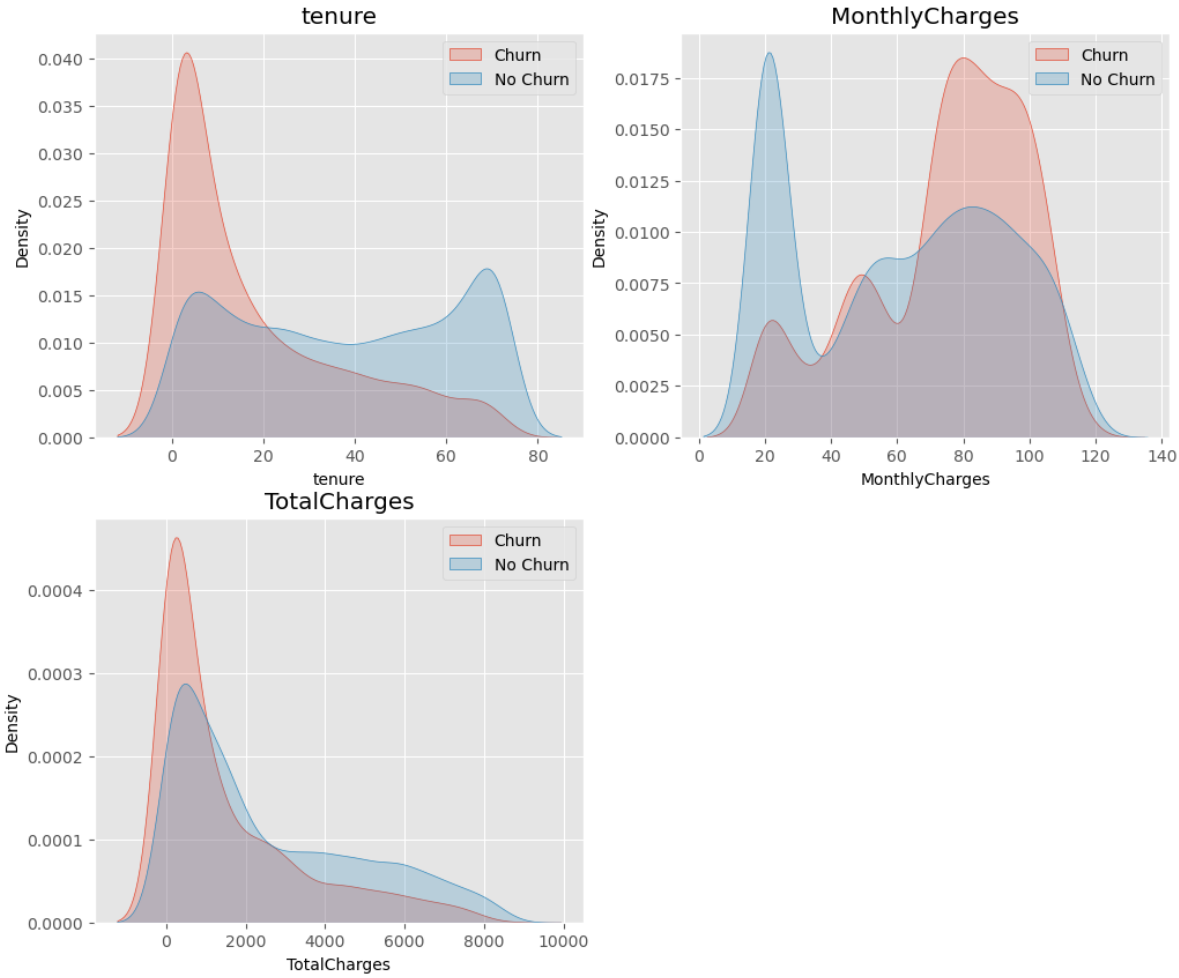


Le graphique à barres suivant présenté ci-dessus peut en tirer des conclusions : • Les clients sont plus susceptibles d'opter pour des contrats mensuels. • Taux de désabonnement modérément plus élevé avec le mode de paiement par chèque électronique. • Les clients bénéficiant de la facturation sans papier ont des taux de désabonnement plus élevés.

Analyse des informations sur le compte client – Variables numérique

Nous évaluerons la distribution de chaque variable numérique à partir des informations du compte client (mandat, MonthlyCharges, TotalCharges).

In [59]:
```
def histogram_plots(df, numerical_values, target):
     number_of_columns = 2
     number_of_rows = math.ceil(len(numerical_values)/2)
     fig = plt.figure(figsize=(12,5*number_of_rows))
     for index, column in enumerate(numerical_values,1):
         ax = fig.add_subplot(number_of_rows, number_of_columns, index)
         ax = sns.kdeplot(df[column][df[target]=="Yes"] ,fill = True)
         ax = sns.kdeplot(df[column][df[target]=="No"], fill = True)
         ax.set_title(column)
         ax.legend(["Churn","No Churn"], loc='upper right')
     plt.savefig("numerical_variables.png", dpi=300)
     return plt.show()

customer_account_num = ["tenure", "MonthlyCharges","TotalCharges"]
histogram_plots(df,customer_account_num, "Churn")
```

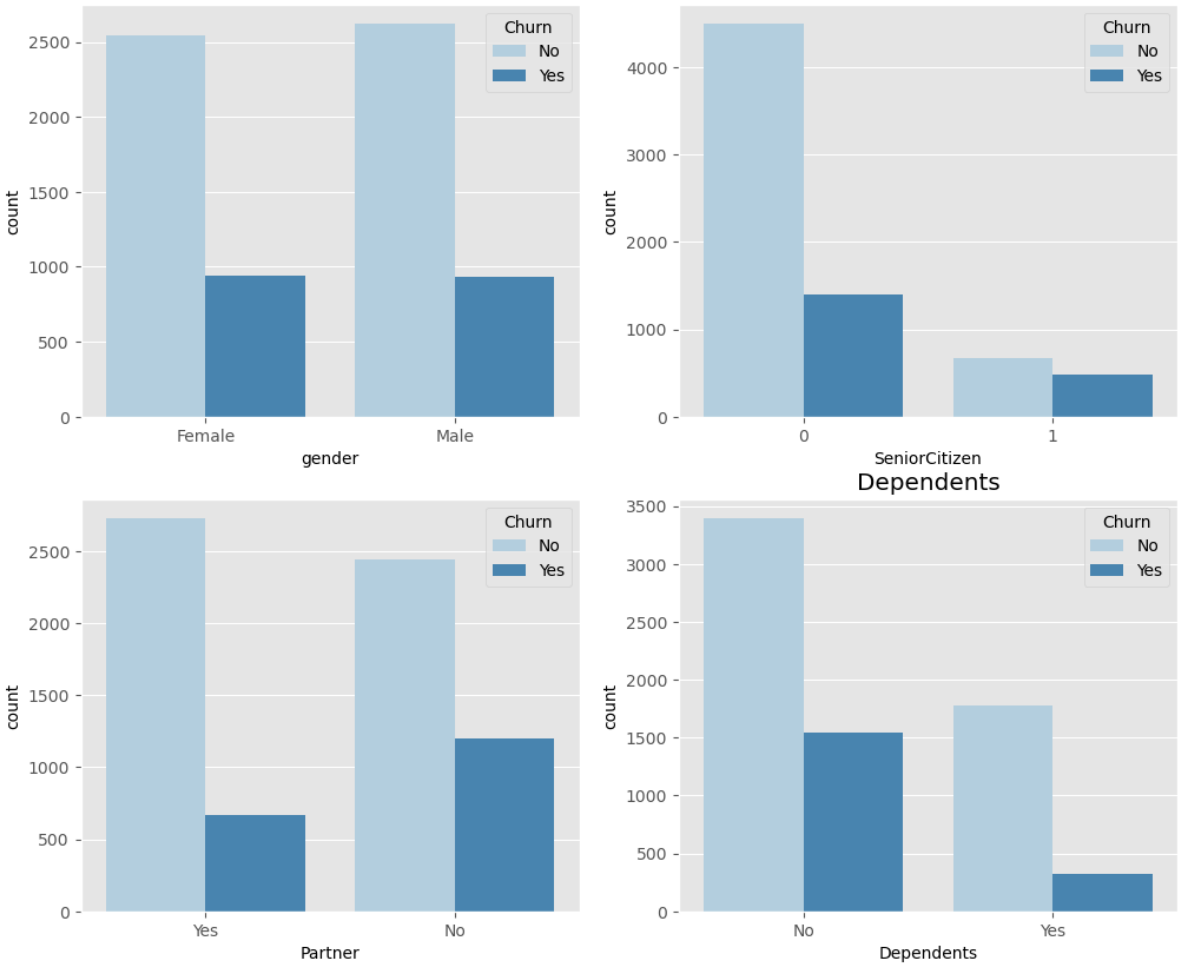


Les histogrammes suivants ci-dessus nous permettent de tirer les conclusions suivantes : • Les clients ayant une courte durée d'occupation sont plus susceptibles de se désinscrire. • Les clients qui paient davantage sur les frais mensuels ont des taux de désabonnement plus élevés. • Les clients dont les frais totaux sont élevés ont tendance à se désinscrire.

Analyse des informations démographiques du client

Nous évaluerons le pourcentage de taux de désabonnement du client en fonction des informations démographiques (sexe, SeniorCitizen, Partner, Dependents)

In [60]:
```
customer_account_cat = ["gender", "SeniorCitizen", "Partner", "Dependents"]
plot_categorical_to_target(df,customer_account_cat,"Churn")
```



Le graphique à barres suivant ci-dessus nous permet de tirer quelques conclusions : • Le taux de désabonnement et l'absence de désabonnement n'ont pas de différences pour chaque sexe. • Les jeunes clients sont plus susceptibles de se désinscrire que les anciens clients. • Les clients avec un partenaire sont inférieurs au taux de désabonnement par rapport à un partenaire.

Vérification des valeurs aberrantes avec Boxplot pour les variables numériques

Une valeur aberrante est un ensemble de données d'observation qui diffère considérablement des observations des autres sur l'ensemble de données. Les valeurs aberrantes peuvent créer des modèles d'apprentissage automatique avec une faible précision, il est donc important de les vérifier. Nous vérifierons les valeurs aberrantes de chaque variable numérique telle que la titularisation, les MonthlyCharges et les TotalCharges.

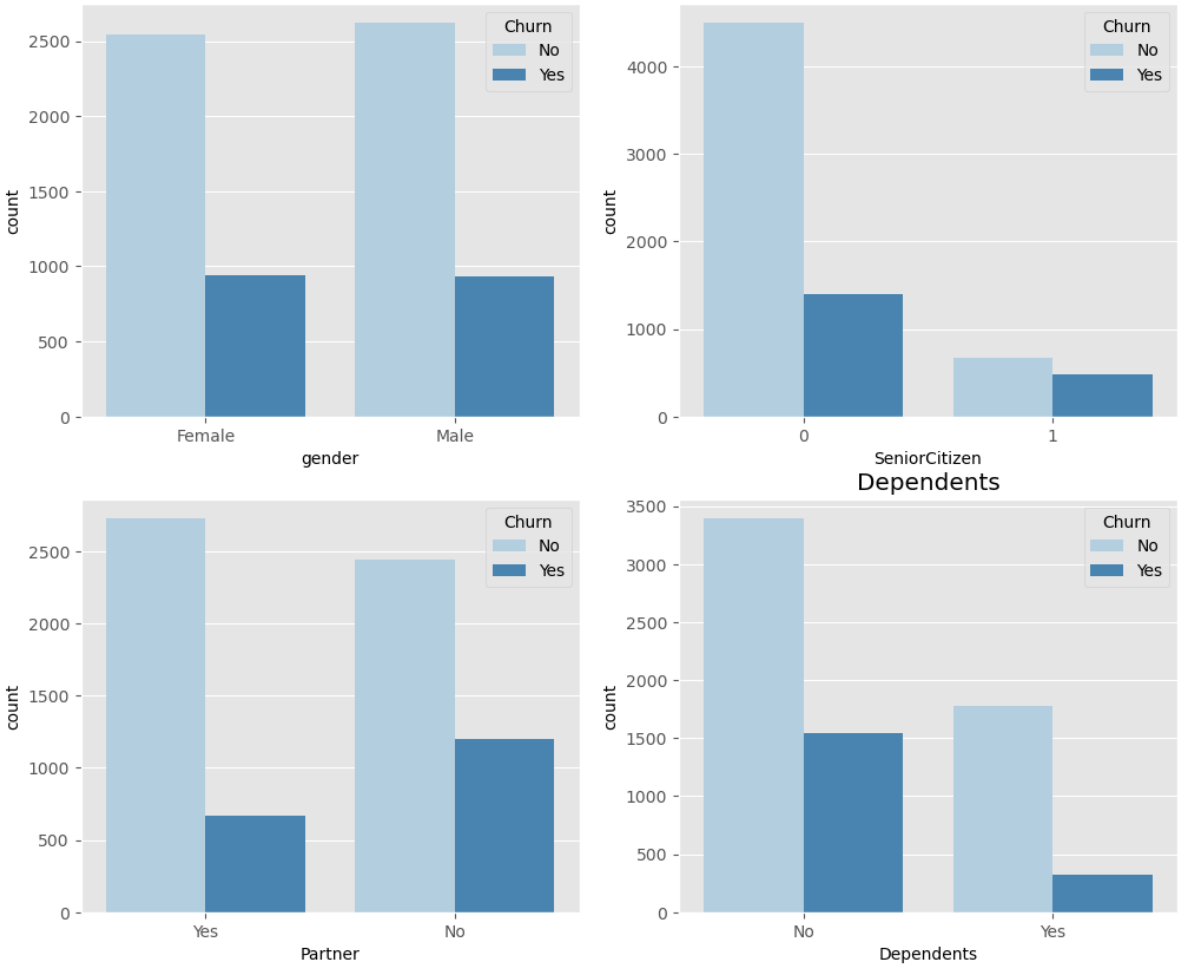
In [61]:
```
def outlier_check_boxplot(df,numerical_values):
 number_of_columns = 2
 number_of_rows = math.ceil(len(numerical_values)/2)

 fig = plt.figure(figsize=(12,5*number_of_rows))
 for index, column in enumerate(numerical_values, 1):
        ax = fig.add_subplot(number_of_rows, number_of_columns, index)
        ax = sns.boxplot(x = column, data = df, palette = "Blues")
        ax.set_title(column)
 plt.savefig("Outliers_check.png", dpi=300)
 return plt.show()
 numerical_values = ["tenure","MonthlyCharges","TotalCharges"]
 outlier_check_boxplot(df,numerical_values)
```

#l'image Outliers_check.png se trouve dans le repertoire courannt Les histogrammes suivants ci-dessus nous permettent de tirer les conclusions suivantes : • Les clients ayant une courte durée d'occupation sont plus susceptibles de se désinscrire. • Les clients qui paient davantage sur les frais mensuels ont des taux de désabonnement plus élevés. • Les clients dont les frais totaux sont élevés ont tendance à se désinscrire.

Analyse des informations démographiques du client

Nous évaluerons le pourcentage de taux de désabonnement du client en fonction des informations démographiques (sexe, SeniorCitizen, Partner, Dependents)

In [62]:
```
customer_account_cat = ["gender", "SeniorCitizen", "Partner", "Dependents"]
plot_categorical_to_target(df,customer_account_cat,"Churn")
```

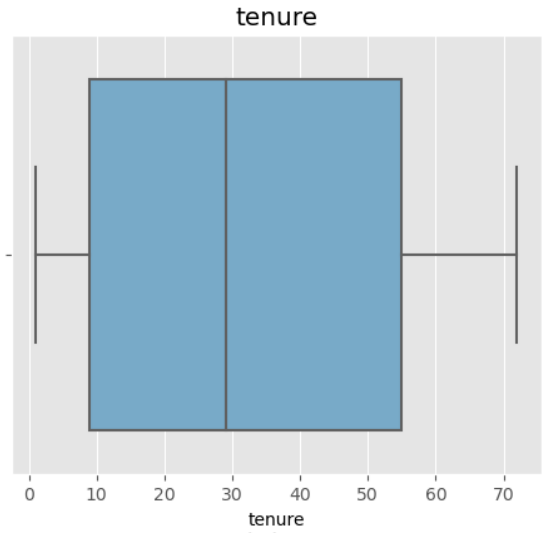


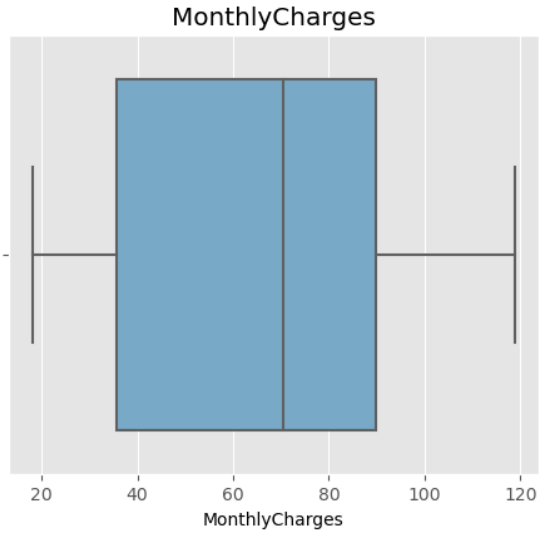
Le graphique à barres suivant ci-dessus nous permet de tirer quelques conclusions : • Le taux de désabonnement et l'absence de désabonnement n'ont pas de différences pour chaque sexe. • Les jeunes clients sont plus susceptibles de se désinscrire que les anciens clients. • Les clients avec un partenaire sont inférieurs au taux de désabonnement par rapport à un partenaire.

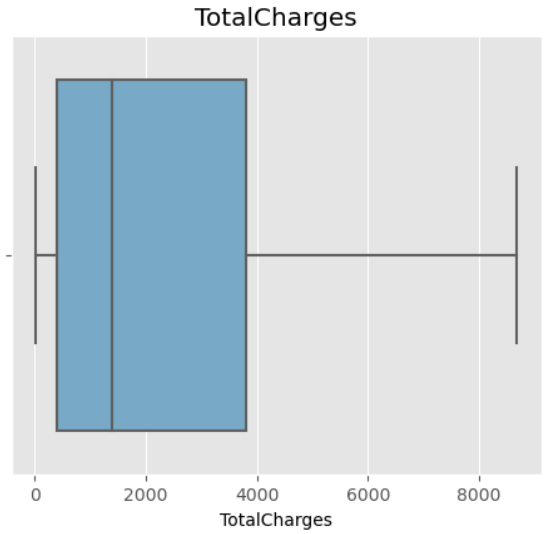
Vérification des valeurs aberrantes avec Boxplot pour les variables numériques

Une valeur aberrante est un ensemble de données d'observation qui diffère considérablement des observations des autres sur l'ensemble de données. Les valeurs aberrantes peuvent créer des modèles d'apprentissage automatique avec une faible précision, il est donc important de les vérifier. Nous vérifierons les valeurs aberrantes de chaque variable numérique telle que la titularisation, les MonthlyCharges et les TotalCharges

In [63]:
```
def outlier_check_boxplot(df,numerical_values):
 number_of_columns = 2
 number_of_rows = math.ceil(len(numerical_values)/2)

 fig = plt.figure(figsize=(12,5*number_of_rows))
 for index, column in enumerate(numerical_values, 1):
     ax = fig.add_subplot(number_of_rows, number_of_columns, index)
     ax = sns.boxplot(x = column, data = df, palette = "Blues")
     ax.set_title(column)
 plt.savefig("Outliers_check.png", dpi=300)
 return plt.show()
numerical_values = ["tenure","MonthlyCharges","TotalCharges"]
outlier_check_boxplot(df,numerical_values)
```







À partir des boxplots, nous pouvons conclure que chaque variable numérique n'a pas de valeur aberrante.

Ingénierie des fonctionnalités

Les données brutes peuvent être transformées en fonctionnalités adaptées aux modèles d'apprentissage automatique qui traitent ce que l'on appelle l'ingénierie des fonctionnalités. L'objectif de l'ingénierie des fonctionnalités est de créer un modèle d'apprentissage automatique pour un meilleur apprentissage et obtenir plus de précision. Dans ce TP, nous étiqueterons l'encodage pour les variables catégorielles qui ont deux valeurs, un encodage à chaud pour les variables catégorielles qui ont plus de deux valeurs et une fonctionnalité appelant des variables numériques.

Encodage des étiquettes

L'encodage des étiquettes vise à convertir les variables catégorielles au format numérique. Dans cet article, nous modifierons les variables catégorielles qui ont deux valeurs telles que (Partenaire, Personnes à charge, PhoneService, Churn, PaperlessBilling et sexe). Leurs valeurs ont simplement Oui ou Non et nous passerons à 1 et 0, sauf que le sexe est Femme à 1 et Homme à 0.

In [64]:
```
feature_le = ["Partner","Dependents","PhoneService","Churn","PaperlessBilling"]
def label_encoding(df,features):
     for i in features:
         df[i] = df[i].map({"Yes":1, "No":0})
     return df
df = label_encoding(df,feature_le)
df["gender"] = df["gender"].map({"Female":1, "Male":0})
```

Encodage à chaud

De même avec l'encodage d'étiquettes, un encodage à chaud change également les variables catégorielles en variables numériques. Mais un encodage à chaud prend plus de deux valeurs. Un codage à chaud crée une nouvelle colonne entière binaire (1 ou 0) pour chaque niveau de la variable catégorielle. Les variables catégorielles qui ont plus de deux valeurs sont (MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaymentMethod).

In [65]:
```
features_ohe = ["MultipleLines","InternetService","OnlineSecurity","OnlineBackup",'
df_ohe = pd.get_dummies(df, columns=features_ohe)
```

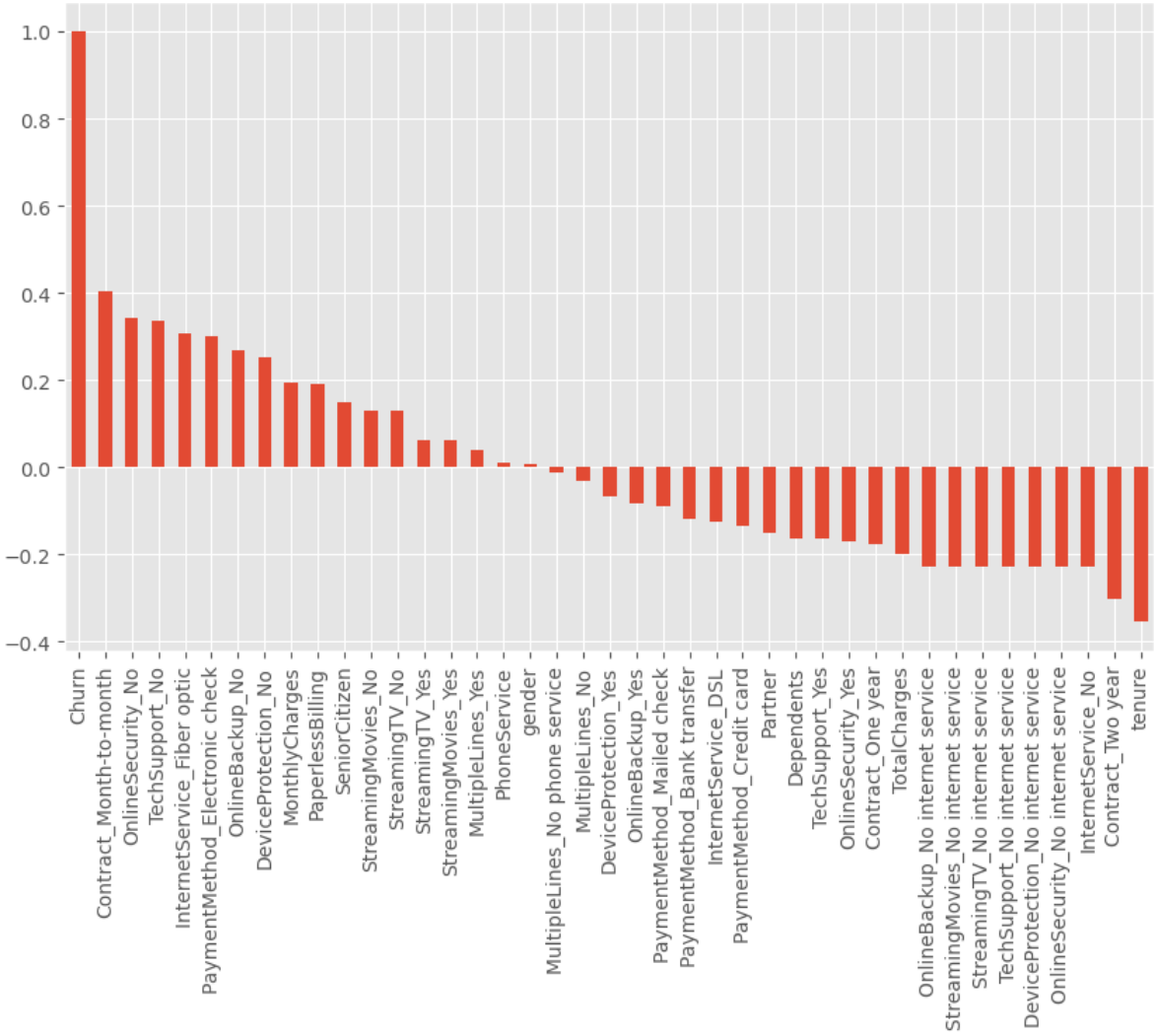
Mise à l'échelle des fonctionnalités (features)

Dans ce TP, nous utilisons MinMaxScaler pour la mise à l'échelle des fonctionnalités. MinMaxScaler peut créer des variables numériques mises à l'échelle de 0 à 1. La mise à l'échelle des fonctionnalités est importante pour interpréter les modèles d'apprentissage automatique afin d'avoir des fonctionnalités à la même échelle. Nous mettrons à l'échelle les variables numériques telles que MonthlyCharges, TotalCharges et tenure (la durée d'occupation).

In [66]:
```
features_mms = ["tenure","MonthlyCharges","TotalCharges"]
df_mms = pd.DataFrame(df_ohe, columns=features_mms)
df_remaining = df_ohe.drop(columns=features_mms)
mms = MinMaxScaler(feature_range=(0,1))
rescaled_feature = mms.fit_transform(df_mms)
rescaled_feature_df = pd.DataFrame(rescaled_feature, columns=features_mms,
index=df_remaining.index)
df = pd.concat([rescaled_feature_df,df_remaining],axis=1)
```

Analyse de correlation

In [67]:
```
plt.figure(figsize=(10,6))
df.corr()["Churn"].sort_values(ascending=False).plot(kind="bar")
plt.savefig("correlation.png", dpi=300)
plt.show()
```



La corrélation est une méthode statistique permettant de tester la relation entre des variables numériques ou catégorielles. Le graphique suivant ci-dessus vise à connaître la corrélation entre certaines variables pour cibler le taux de désabonnement des variables. La corrélation la plus élevée est la variable Contract_Month-to-month, de sorte que cette variable est ainsi liée au taux de désabonnement. L'ancienneté a une corrélation négative avec le taux de désabonnement, de sorte que des valeurs plus élevées liées à l'ancienneté entraînent un taux de désabonnement plus faible.

Répartition des donnée de tests et d'entrainement (test and train)

La répartition des donnée de tests et de train est un processus permettant de valider l'ensemble de données et de simuler les performances du modèle avec de nouvelles données. Dans ce TP, nous créons 80 % des données de train et 20 % des données de test. La colonne « Churn » que nous renommons (le « y ») et la colonne des fonctionnalités que nous renommons (le « X »).

In [68]:
```
X = df.drop(columns = "Churn")
y = df.Churn
X_train,X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=4
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[68]: ((5625, 40), (1407, 40), (5625,), (1407,))

Modèle de sélection et d'évaluation

Sélection du modèle Nous utilisons trois modèles pour la classification du taux de désabonnement, le modèle que nous utilisons est : K-Nearest Neighbours – rapide, simple et basé sur des instances Régression logistique — modèle linéaire Réseau neuronal – Plus compliqué comme les neurones du cerveau humain. Évaluation métrique Pour évaluer le modèle, nous utilisons certaines métriques d'évaluation telles que : • Pondérations des fonctionnalités (Feature Weights): répertorie la pondération supérieure des fonctionnalités pour effectuer une prédiction. Les pondérations des caractéristiques sont utilisées pour le modèle de régression logistique ; • Matrice de confusion — Affiche le vrai et le faux, puis les compare aux valeurs prédites et réelles. • Score de précision — Affiche le modèle de précision issu de la formation et des tests pour mesurer qu'un modèle est adapté à la prédiction. • Courbe ROC — Affiche la courbe du taux de vrais positifs (TPR) par rapport au taux de faux positifs (FPR) pour différents seuils et la courbe ROC est utilisée pour le modèle d'apprentissage automatique de diagnostic. • AUC (pour ROC) — L'aire sous la courbe (Area Under Curve, AUC) est utilisée pour calculer l'aire de la courbe ROC et l'augmentation de l'aire est meilleure pour le modèle. • Courbe de rappel de précision — Affiche la courbe pour comparer le taux de faux positifs (FPR) et le taux de faux négatifs (FNR) pour différents seuils et cette métrique est utilisée pour démontrer la capacité du modèle de diagnostic. Cette métrique convient aux ensembles de données cibles déséquilibrés en raison de l'accent mis sur la précision et le rappel et ne dépend pas du nombre de vrais négatifs. • Score F1 — Cette métrique est généralement meilleure en termes de précision, surtout si elle a une distribution cible d'ensemble de données déséquilibrée. • AUC (pour PRC) — Mesure l'aire globale sous la courbe de PRC.

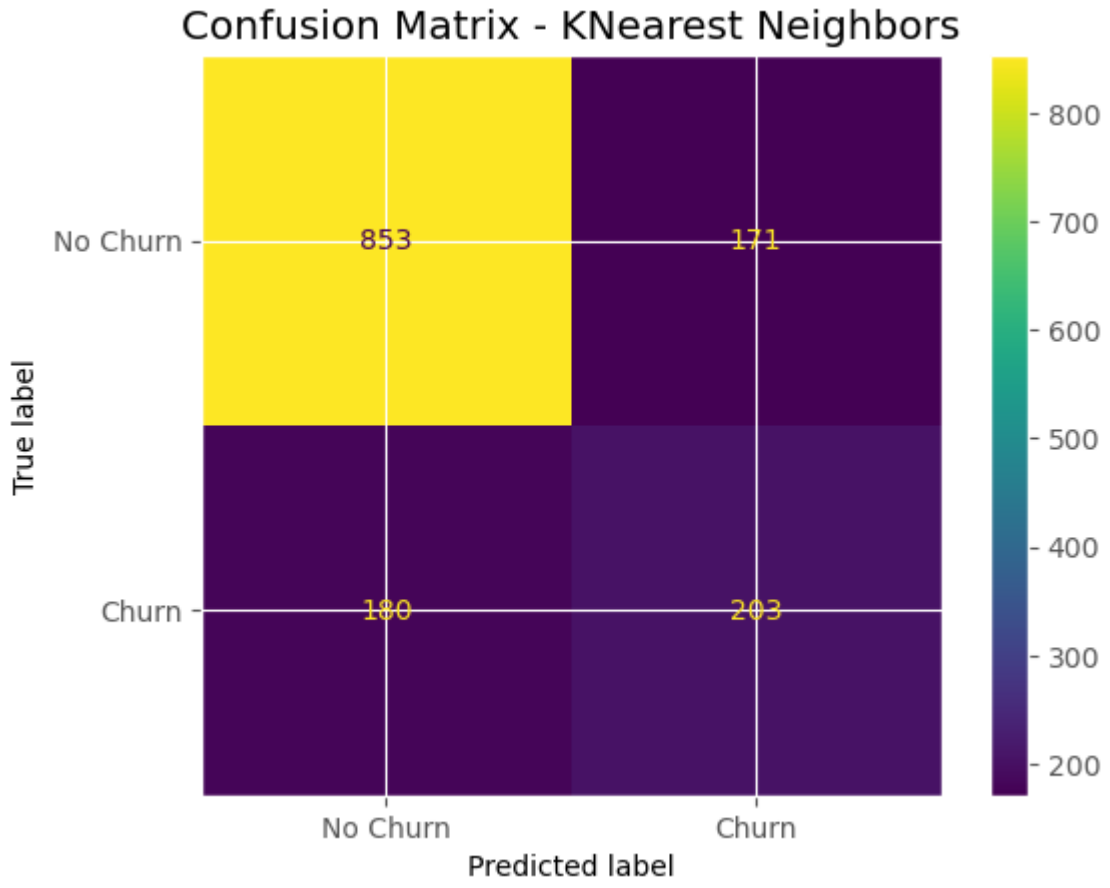
Création d'une fonction pour afficher les mesures d'évaluation

In [69]:
```
# For Logistic Regression
def feature_weights(X_df, classifier, classifier_name):
    weights = pd.Series(classifier.coef_[0], index = X_df.columns.values).sort_val
    top_10_weights = weights[:10]
    plt.figure(figsize=(7,6))
    plt.title(f"{classifier_name} - Top 10 Features")
    top_10_weights.plot(kind="bar")

    bottom_10_weights = weights[len(weights)-10:]
    plt.figure(figsize=(7,6))
    plt.title(f"{classifier_name} - Bottom 10 Features")
    bottom_10_weights.plot(kind="bar")
    print("")

def confusion_matrix_plot(X_train, y_train, X_test, y_test, y_pred, classifier, cla
    cm = confusion_matrix(y_pred,y_test)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["No Churn",
    disp.plot()
    plt.title(f"Confusion Matrix - {classifier_name}")
    plt.show()

    print(f"Accuracy Score Test = {accuracy_score(y_pred,y_test)}")
    print(f"Accuracy Score Train = {classifier.score(X_train,y_train)}")
    return print("\n")

def roc_curve_auc_score(X_test, y_test, y_pred_probabilities,classifier_name):
    y_pred_prob = y_pred_probabilities[:,1]
    fpr,tpr,thresholds = roc_curve(y_test, y_pred_prob)

    plt.plot([0,1],[0,1],"k--")
    plt.plot(fpr,tpr,label=f"{classifier_name}")
```

```
    plt.title(f"{classifier_name} - ROC Curve")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.show()
    return print(f"AUC Score (ROC):{roc_auc_score(y_test,y_pred_prob)}")

def precision_recall_curve_and_scores(X_test, y_test, y_pred, y_pred_probabilities,
    y_pred_prob = y_pred_probabilities[:,1]
    precision, recall, thresholds = precision_recall_curve(y_test, y_pred_prob)
    plt.plot(recall,precision, label=f"{classifier_name}")
    plt.title(f"{classifier_name}-ROC Curve")
    plt.xlabel("Recall")
    plt.ylabel("Precision")
    plt.show()
    f1_score_result, auc_score = f1_score(y_test,y_pred), auc(recall,precision)
    return print(f"f1 Score : {f1_score_result} \n AUC Score (PR) :{auc_score}")
```

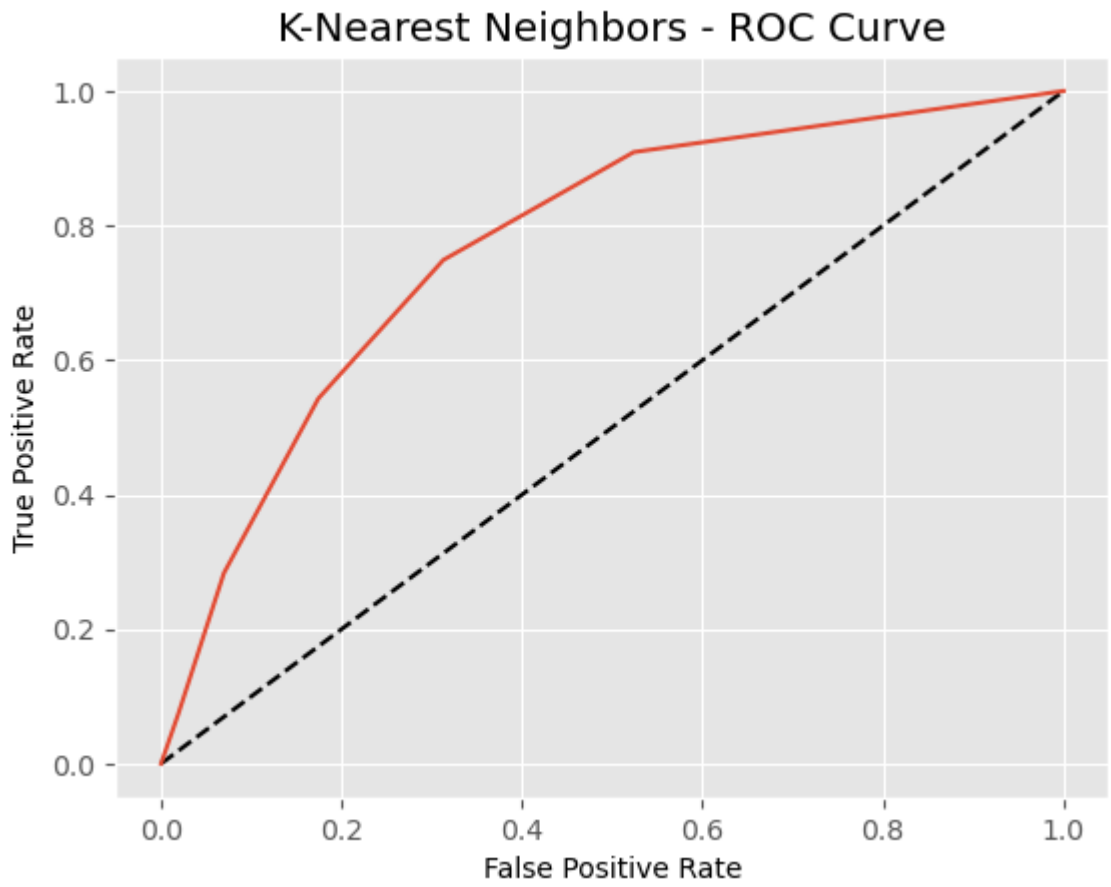
K-Nearest Neighbor

In [70]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
y_pred_knn_proba = knn.predict_proba(X_test)

confusion_matrix_plot(X_train,y_train,X_test, y_test, y_pred_knn, knn, "KNearest Ne
```



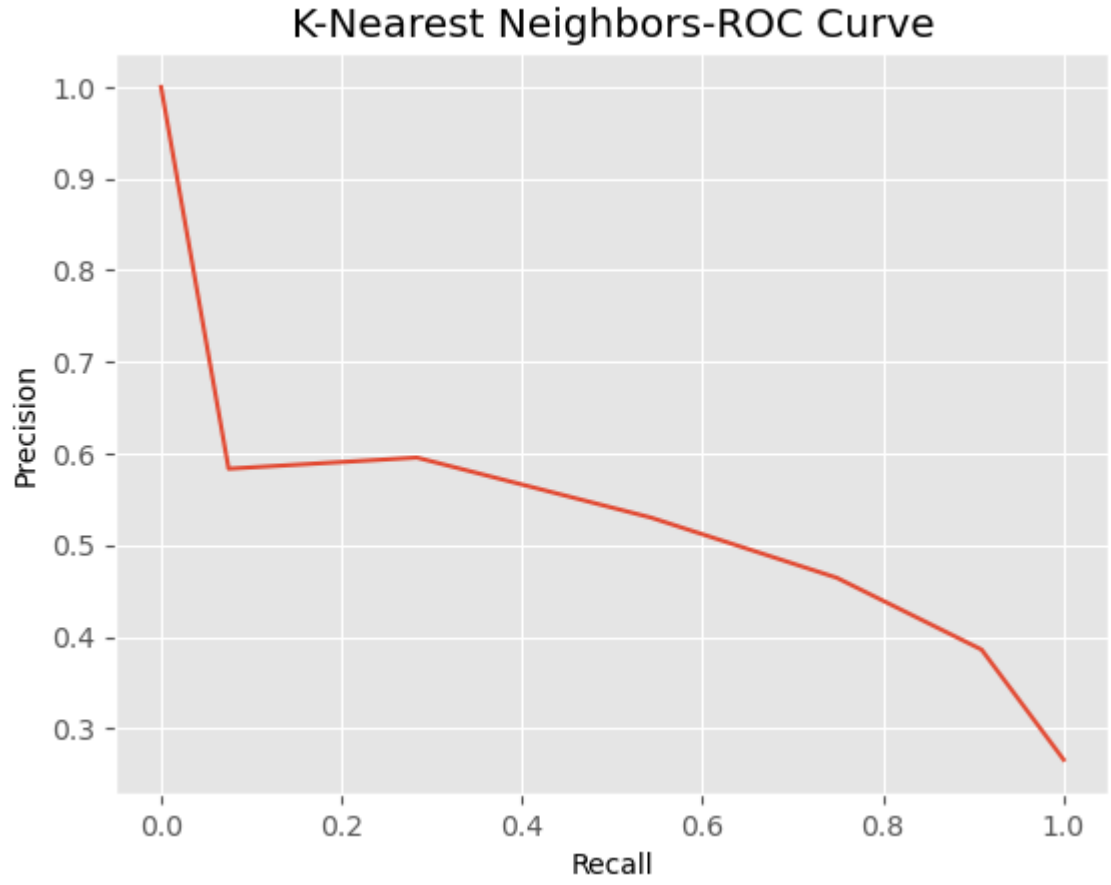

```
Accuracy Score Test = 0.7505330490405118
Accuracy Score Train = 0.8359111111111112
```

In [71]:
```
roc_curve_auc_score(X_test,y_test,y_pred_knn_proba, "K-Nearest Neighbors")
```




```
AUC Score (ROC):0.7718769898173121
```

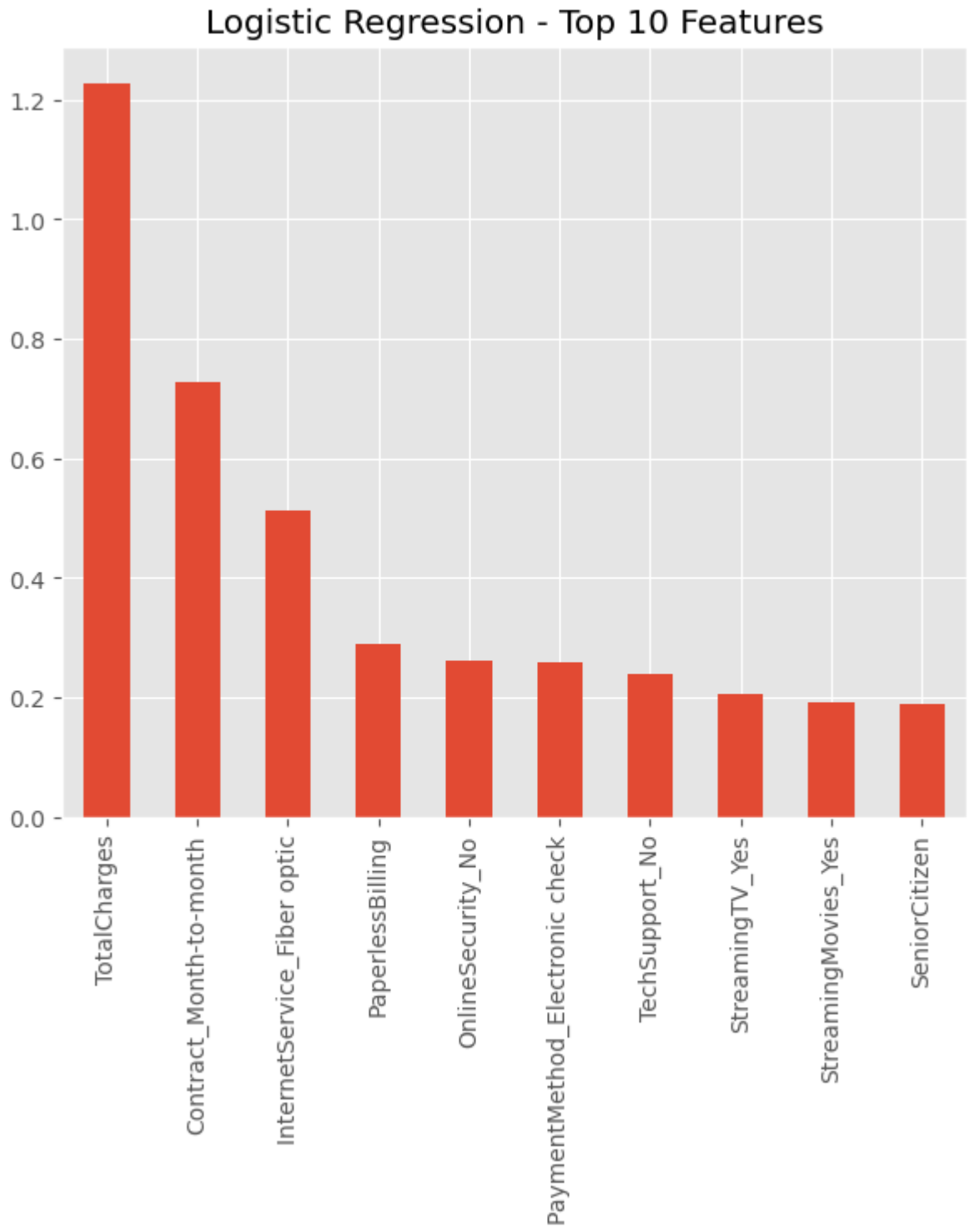
In [72]:
```
precision_recall_curve_and_scores(X_test,y_test,y_pred_knn,y_pred_knn_proba,"K-Near
```

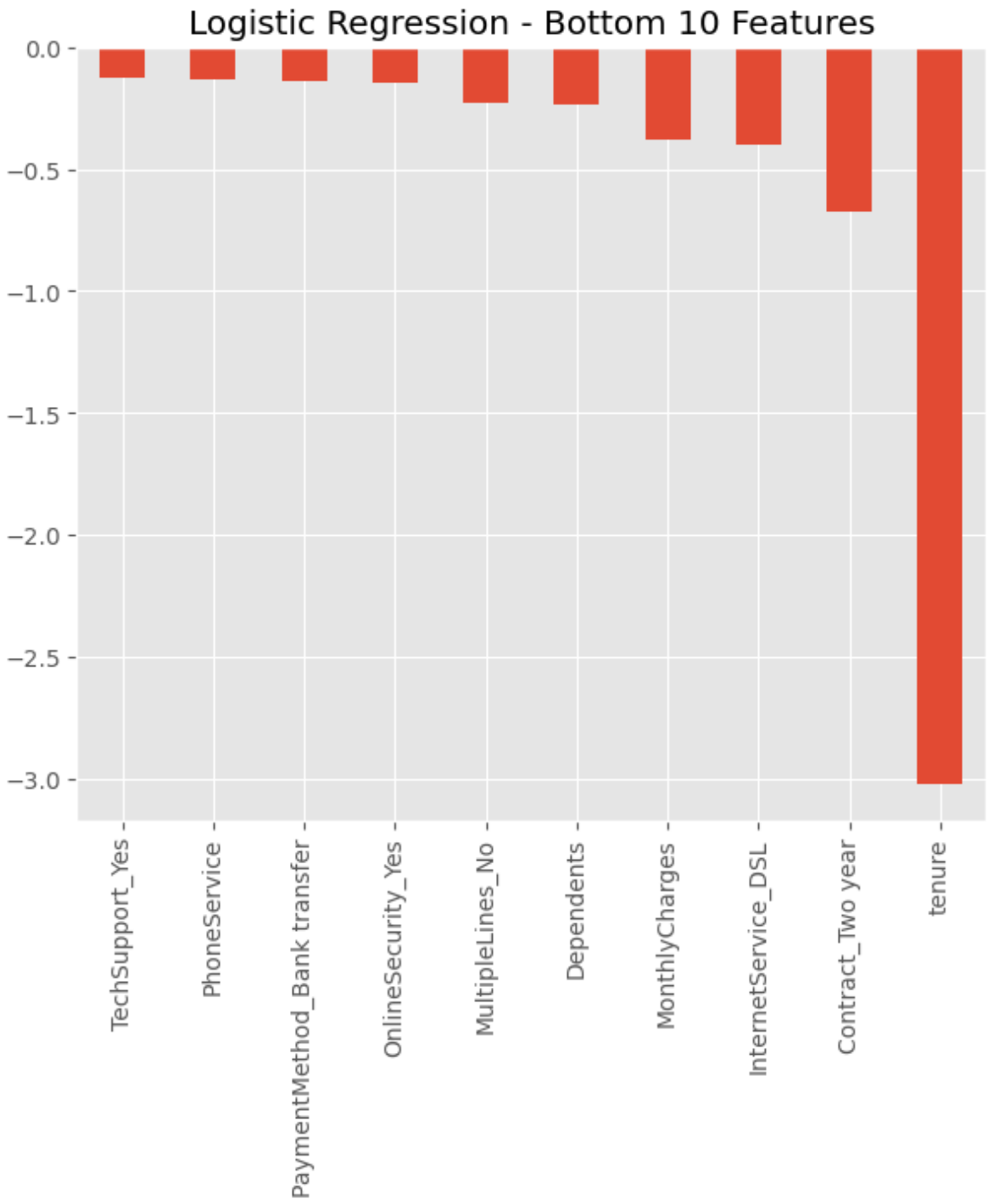



```
f1 Score : 0.5363276089828268
 AUC Score (PR) :0.5283439083015596
```

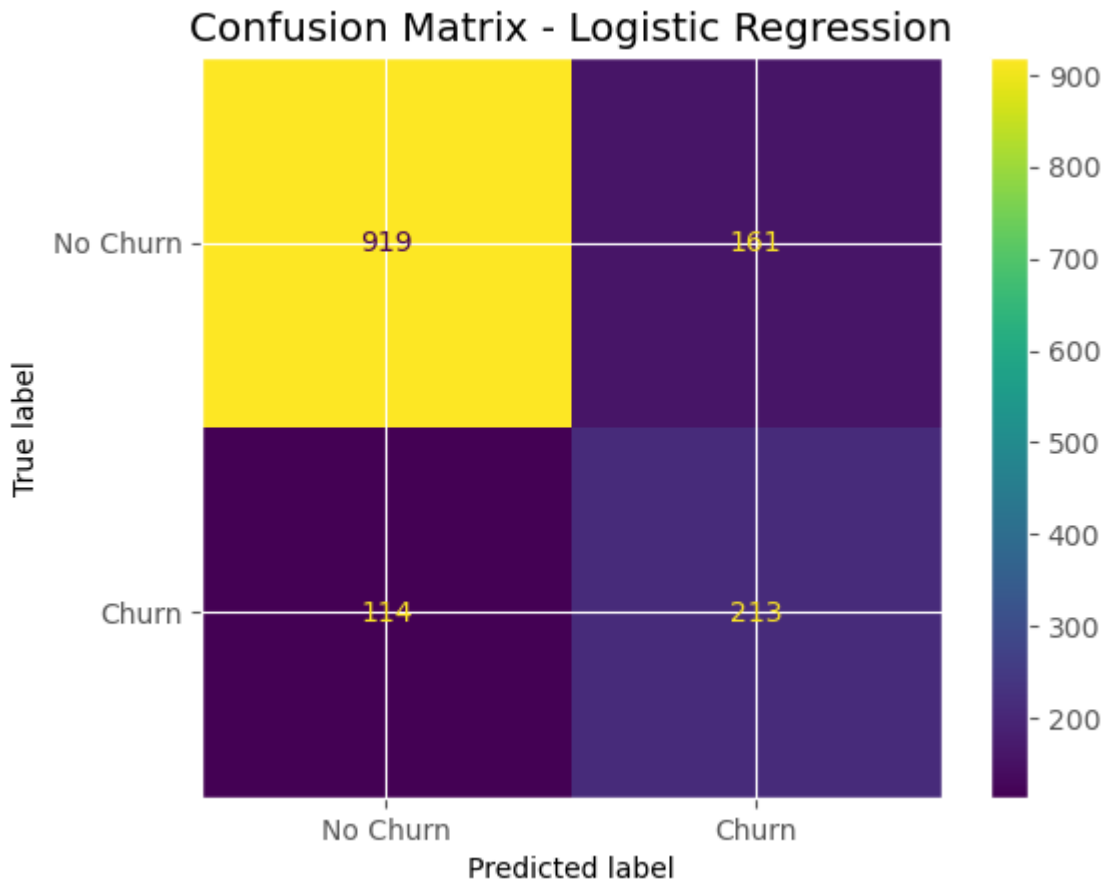
Logistic Regression

In [73]:
```
from sklearn.linear_model import LogisticRegression

# Créer et ajuster le modèle de régression logistique
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Appeler la fonction feature_weights() avec logreg
feature_weights(X_train, logreg, "Logistic Regression")
```



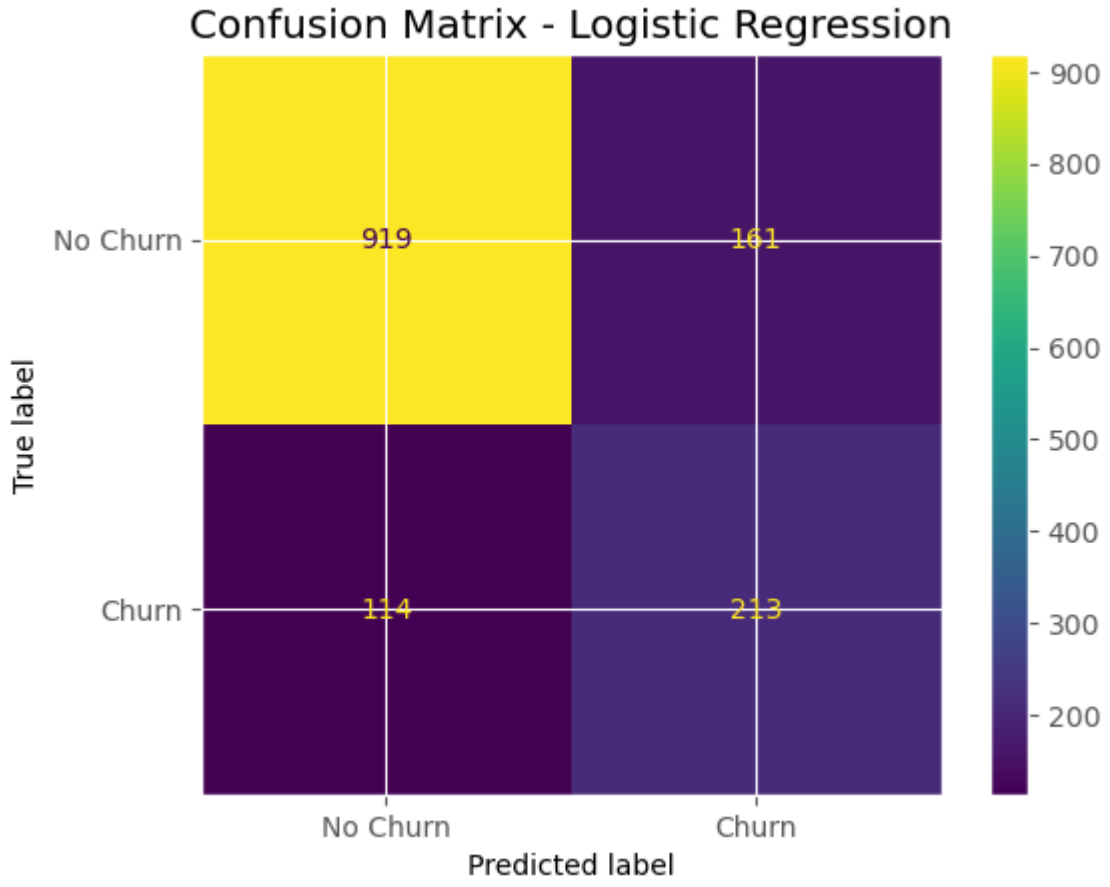


In [75]:
```
y_pred_train = logreg.predict(X_train)
y_pred_logreg = logreg.predict(X_test)

confusion_matrix_plot(X_train,y_train,X_test,y_test, y_pred_logreg,logreg,"Logistic
```



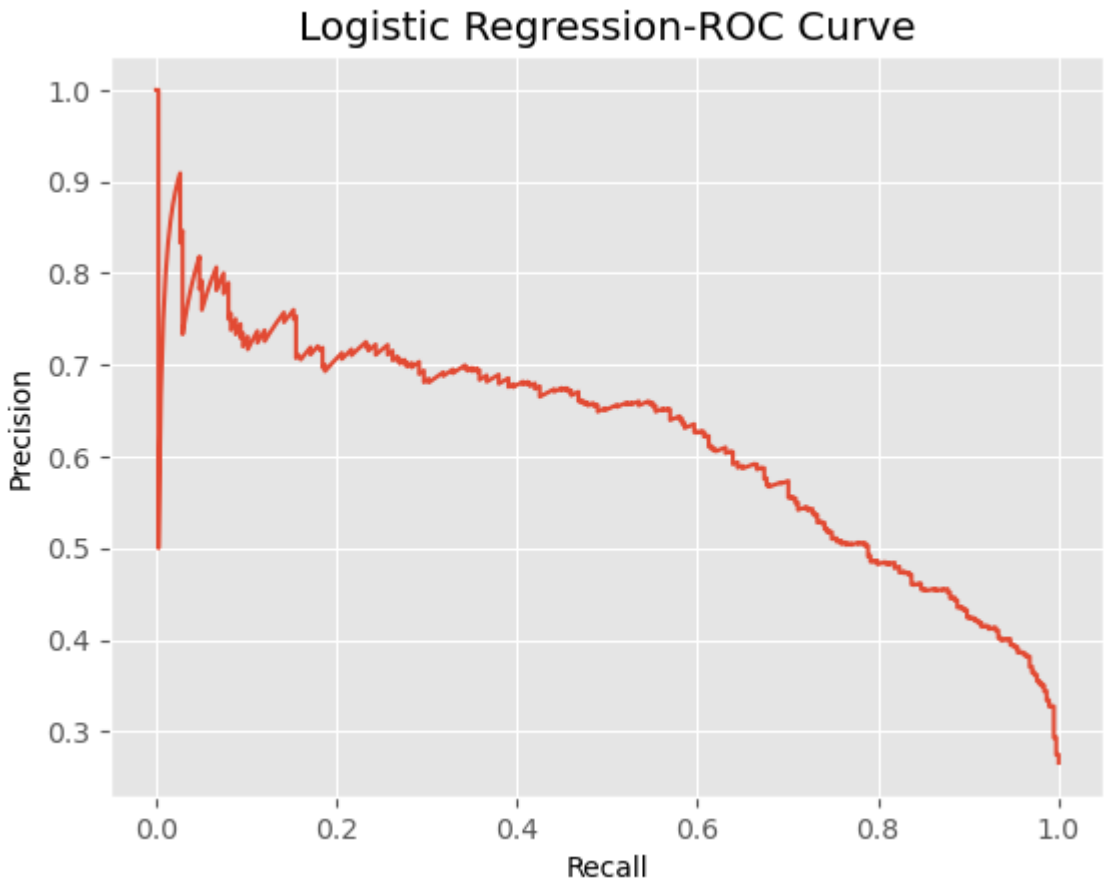

```
Accuracy Score Test = 0.8045486851457001
Accuracy Score Train = 0.8048
```

In [76]:
```
confusion_matrix_plot(X_train,y_train,X_test,y_test, y_pred_logreg,logreg,"Logistic
```



```
Accuracy Score Test = 0.8045486851457001
Accuracy Score Train = 0.8048
```

In [77]:
```
y_pred_logreg_proba = logreg.predict_proba(X_test)

precision_recall_curve_and_scores(X_test,y_test,y_pred_knn,y_pred_logreg_proba,"Log
```




```
f1 Score : 0.5363276089828268
 AUC Score (PR) :0.6185957002131842
```

MERCI!!!