# *Proiect  Știința Datelor în afaceri*

Studenți:Isac Georgiana

Carmen Florina Bianca

Chirila Andreea Raluca

Merandisse Meddy Steeve

AACPI, An I

Profesor: Bizovi Mihai

## I. *Descrierea problemei pe care incercam sa o rezolvam, prin proiect*

Pentru proiectul pe care l-am realizat, noi am ales sa studiem analiza Churn a angajatilor.

**Acesta este procesul de utilizare a analizei datelor și a algoritmilor de învățare automată pentru a prezice clienții cel mai probabil să întrerupă relația cu o companie.**

**Ajută companiile să identifice clienții care riscă să plece, astfel încât să poată lua măsuri proactive pentru a-i păstra.**

Customer Churn Prediction este o tehnică de analiză predictivă utilizată de companii pentru a identifica clienții potențiali care ar putea să își anuleze serviciile sau să înceteze să-și folosească produsele în viitorul apropiat. Este un aspect crucial al managementului relațiilor cu clienții (CRM) și ajută companiile să ia măsuri proactive pentru a-și păstra clienții.

Scopul proiectului nostru, privind retragerea clienților este de a minimiza pierderea de clienți și de a îmbunătăți ratele de retenție a clienților. Acest lucru poate duce la creșterea veniturilor și la o bază de clienți mai puternică pentru companie.

## II. *Domeniul de activitate*

Procesul de predicție a pierderii clienților implică colectarea și analizarea datelor despre comportamentul clienților, cum ar fi istoricul achizițiilor, informațiile demografice și alți factori relevanți. Aceste date sunt apoi folosite pentru a antrena modele de învățare automată, cum ar fi arbori de decizie, păduri aleatorii și rețele neuronale artificiale, pentru a identifica modele și pentru a face predicții cu privire la clienții care sunt cel mai probabil să se retragă.

## III. *Importanta analizei pentru o firma*

In urma analizarii acestui set de date, respectiv estimarea modelului, este vizata o serie de actiuni, care implica, in cea mai mare parte, predicția ratei clienților, care ajută companiile în mai multe moduri:

- Retenție îmbunătățită a clienților: prin identificarea clienților care ar putea pleca, companiile pot lua măsuri proactive pentru a-i păstra, ceea ce duce la rate îmbunătățite de retenție a clienților.
- Venituri crescute: prin păstrarea clienților, companiile pot genera mai multe venituri și pot construi o bază de clienți mai puternică.
- O mai bună alocare a resurselor: concentrându-se pe păstrarea clienților expuși riscului, companiile își pot aloca resursele mai eficient, în loc să dobândească în mod constant noi clienți pentru a-i compensa pe cei pierduți.
- Creșterea valorii de viață a clienților: prin păstrarea clienților pentru perioade mai lungi de timp, companiile pot crește valoarea totală pe care o primesc de la fiecare client de-a lungul vieții.

- Înțelegerea îmbunătățită a clienților: predicția privind retragerea clienților poate oferi informații valoroase asupra comportamentului și preferințelor clienților, ceea ce poate informa deciziile viitoare despre produse și marketing.
- Cost redus al achiziției de clienți: prin păstrarea clienților existenți, companiile pot reduce costul achiziționării de noi clienți, care este adesea mai mare decât păstrarea celor existenți.

În general, predicția privind retragerea clienților ajută companiile să rețină clienții valoroși, să îmbunătățească satisfacția clienților, să mărească veniturile și să ia decizii bazate pe date.

## IV. Lecture Review

Xiao și colab. a prezentat un model de selecție a caracteristicilor de predicție a abandonului folosind un algoritm de ansamblu cu un mediu dinamic. Modelul lor sugerat include două straturi principale ale rețelei neuronale de tip GMDH și stratul de clasificare.

Modelul rezultat are performanțe ridicate în ceea ce privește alte metode de învățare prin transfer. Idris și Khan au depășit setul de date dezechilibrat din predicția abandonului și au prezentat o metodă de clasificare. Metoda prezentată se bazează pe un filter wrapper și algoritm de ansamblu.

În primul pas, au folosit metode bazate pe PSO pentru extragerea caracteristicilor de guvernare. În al doilea pas, GA a fost folosit pentru reducerea dimensiunii.

Faza finală constă în proiectarea unui nou spațiu de caracteristici și clasificare folosind un algoritm de ansamblu. Rezultatele arată o reducere optimă a dimensiunii pentru două seturi de date de portocaliu și Cell2Cell. Saghir et al. a evaluat predicția de abandon folosind o metodă de cerșire a rețelelor neuronale.

Scopul principal este de a crește acuratețea evaluării.

Rezultatele lor arată că metoda prezentată atinge o acuratețe de 81% pentru mai multe seturi de date. Amin şi colab. a folosit caracteristica distanței pentru predicția churn bazată pe certitudinea clasificării. Setul lor de date clasificat constă din doi clasificatori cu certitudine ridicată și scăzută pentru predicția abandonului. Valorile de evaluare includ acuratețea, precizia, sensibilitatea și scorul f.

Rezultatele arată că există o relație directă între distanță și certitudinea factorilor de clasificare. Dacă factorii de distanță au fost mari, prin urmare, clasificatorul obține o precizie ridicată. Această constatare a fost corectă pentru distanța mică, cu o certitudine scăzută.

Mou și colab. a prezentat problema de programare inversă a fluxului de permutare distribuită eficient din punct de vedere energetic pentru a reduce ajustarea și consumul de energie în același timp. În munca lor, Zenggang et al. au folosit calitatea serviciului a nodurilor și propriile resurse pentru a proiecta funcții de stabilire a prețurilor mesajelor care limitează capacitatea nodurilor de a face oferte frauduloase prin dezvăluirea stării resurselor ambelor noduri. Höppner și colab.au prezentat modele de detectare a abandonului pentru realizarea unui model cu profit ridicat.

Modelul s-a bazat pe profitul optim așteptat pentru numărul de clienți. Aceștia au prezentat un model de ProfTree pentru clasificarea și proiectarea arborilor de decizie. Modelul a depășit performanța de ultimă generație pe baza rezultatelor articolului. Maldonado et al. oferă o nouă metodă de predicție a pierderii care generează profit. Metoda este o versiune dezvoltată a algoritmului mașinii de probabilitate minimax pentru clasificarea churn-urilor. Metoda prezentată constă din algoritmii LASSO și Tikhonov pentru regularizare. Rezultatele arată că metoda prezentată joacă un rol esențial în creșterea profitului companiei. Calzada-Infante et al. a prezentat o metodă de predicție a abandonului bazată pe analiza comportamentului clienților.

Ei au folosit rețeaua socială pentru a evalua comportamentul clienților folosind metode de clasificare. Ei împart clienții în churners și nonchurner.

Rezultatele au fost ilustrate în ordine la timp și diagramă temporală statică cu metrici proximale. Rezultatele arată o performanță mai mare decât metoda prezentată în comparație cu alte abordări.

Rezultatele lor compară algoritmii după diverse metrici prin exploatarea tehnicilor de supraeșantionare.

Cu toate acestea, rezultatele lor arată că GBT este mai bun în comparație cu modelul profund. Saghir a evaluat predicția de pierdere folosind o metodă de cerșire a rețelelor neuronale. Scopul principal este de a crește acuratețea evaluării.

Rezultatele lor arată că metoda prezentată atinge o acuratețe de 81% pentru mai multe seturi de date.

| References | Prediction method | Industry | Evaluation metrics |
|---|---|---|---|
| [35] | Feedforward network, deep belief network, and Autoencoder | Telecom | AUC |
| [39] | SMOTE, decision trees, random forest, AdaBoost, multilayer perceptron, K-means and Bayesian logistic regression | Telecom | *P*-value, accuracy, AUC, precision, recall, F1-Score |
| [40] | CCPBI-TAMO, CPIO-FS | Telecom | Precision, recall, accuracy, F-Score, ROC |
| [41] | Xgboost, AdaBoost, catboost, decision trees, SVM, KNN | Telecom | Accuracy, AUC, precision, recall, F-Measures |
| [37] | Deep feed-forward networks | Subscription companies | Accuracy |
| [38] | Deep ANN, machine learning algorithms | Telecom | Accuracy, precision, recall, F1-score, and AUC |
| [12] | Neural network with bagging | Telecom | Accuracy, precision, recall, F-score, kappa, absolute error, relative error, and classification error |
| [10] | Transfer learning of ensemble | Telecom | Area under curve of ROC (AUC) and complexity |
| [11] | Ensemble algorithm | Telecom | Area under curve of ROC (AUC) |
| [12] | Begging and neural network | Telecom | Accuracy and precision of classification |
| [42] | Artificial neural network (ANN) and self-organized map (SOM) | Telecom | Accuracy, recall, F-score, and precision |
| [15] | Profit tree | Telecom | Accuracy, cost, and profit |
| [16] | Minimax probability machines | Telecom | AUC and EMPC |
| [17] | similarity forests | Telecom | AUC, and tenlift AUPR |
| [21] | Temporal point processes (TPP) and recurrent neural networks (RNN) | Telecom | MAE and MRE |
| [22] | Cross-company just-in-time approach | Telecom | Accuracy, Kappa, and Recall |
| [25] | Multiobjective and colony optimization | Telecom | AUC |
| [27] | graph theory | Telecom | Top decile lift |
| [31] | Boosted-stacked learners and bagged-stacked learners | Telecom | G means and AUC |
| [34] | Logistic regression and Logit Boost | Telecom | Accuracy, Kappa, MAE, coverage case, and RMSE |
| [43] | Decision tree, artificial neural, networks, K-Nearest neighbors, and support vector machine | Telecom | Confusion matrix, precision, and recall |
| [44] | Logistic regression and random forest | Telecom, bank, and retail | AUC and cross-validation |
| [45] | Genetic algorithm, classification, and covering & LEM2 (rules generation algorithms) | Telecom | Accuracy, recall, mutual information, coverage, and F measure |
| [46] | Data transformations, DT, KNN, DNN, GBT, and SRI | Telecom | Probability of detection, probability of false alarm, area under the curve, and g-mean |
| [47] | Exhaustive, genetic, covering and LEM2 | Telecom | TP, FP, FN, TN, COV, PRE REC, ER, ACC, and SPEFM |

Tabelul 1 prezintă rezumatul lucrărilor de cercetare.

## V. *Datele-Importarea setului de date*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
def datapreparation(filepath):

    df = pd.read_csv(filepath)
    df.drop(["customerID"], inplace = True, axis = 1)

    df.TotalCharges = df.TotalCharges.replace(" ",np.nan)
    df.TotalCharges.fillna(0, inplace = True)
    df.TotalCharges = df.TotalCharges.astype(float)

    cols1 = ['Partner', 'Dependents', 'PaperlessBilling', 'Churn',
'PhoneService']
    for col in cols1:
        df[col] = df[col].apply(lambda x: 0 if x == "No" else 1)

    df.gender = df.gender.apply(lambda x: 0 if x == "Male" else 1)
    df.MultipleLines = df.MultipleLines.map({'No phone service': 0, 'No':
0, 'Yes': 1})

    cols2 = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport', 'StreamingTV', 'StreamingMovies']
    for col in cols2:
        df[col] = df[col].map({'No internet service': 0, 'No': 0, 'Yes':
1})

    df = pd.get_dummies(df, columns=['InternetService', 'Contract',
'PaymentMethod'], drop_first=True)

    return df
df = datapreparation(filepath = "C:/Data/isacg-Customer-Churn.csv")
df.head()
```

| | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | OnlineSecurity | OnlineBackup | DeviceProtection | ... | MonthlyCharges | TotalCharges | Churn | In |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | ... | 29.85 | 29.85 | 0 | |
| **1** | 0 | 0 | 0 | 0 | 34 | 1 | 0 | 1 | 0 | 1 | ... | 56.95 | 1889.50 | 0 | |
| **2** | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 0 | ... | 53.85 | 108.15 | 1 | |
| **3** | 0 | 0 | 0 | 0 | 45 | 0 | 0 | 1 | 0 | 1 | ... | 42.30 | 1840.75 | 0 | |
| **4** | 1 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | ... | 70.70 | 151.65 | 1 | |

*Figure 1-Datele folosite*

## VI. *Analiza exploratorie*

Datele sunt preluate de pe Tempo Online, fiind 4073 de inregistrari referitoare la numarul clienților potențiali care ar putea să își anuleze serviciile sau să înceteze să-și folosească produsele în viitorul apropiat.Toate datele sunt preluate pentru anul 2022.

## *VII. Construirea modelului*

Voi construi și voi regla un model de pădure aleatoriu, deoarece în acest caz metoda bazată pe copaci ar funcționa mai bine. De asemenea, sunt interesat de probabilitatea de agitare a clientului individual și de înțelegerea modului în care modelul o calculează folosind valorile Shap.

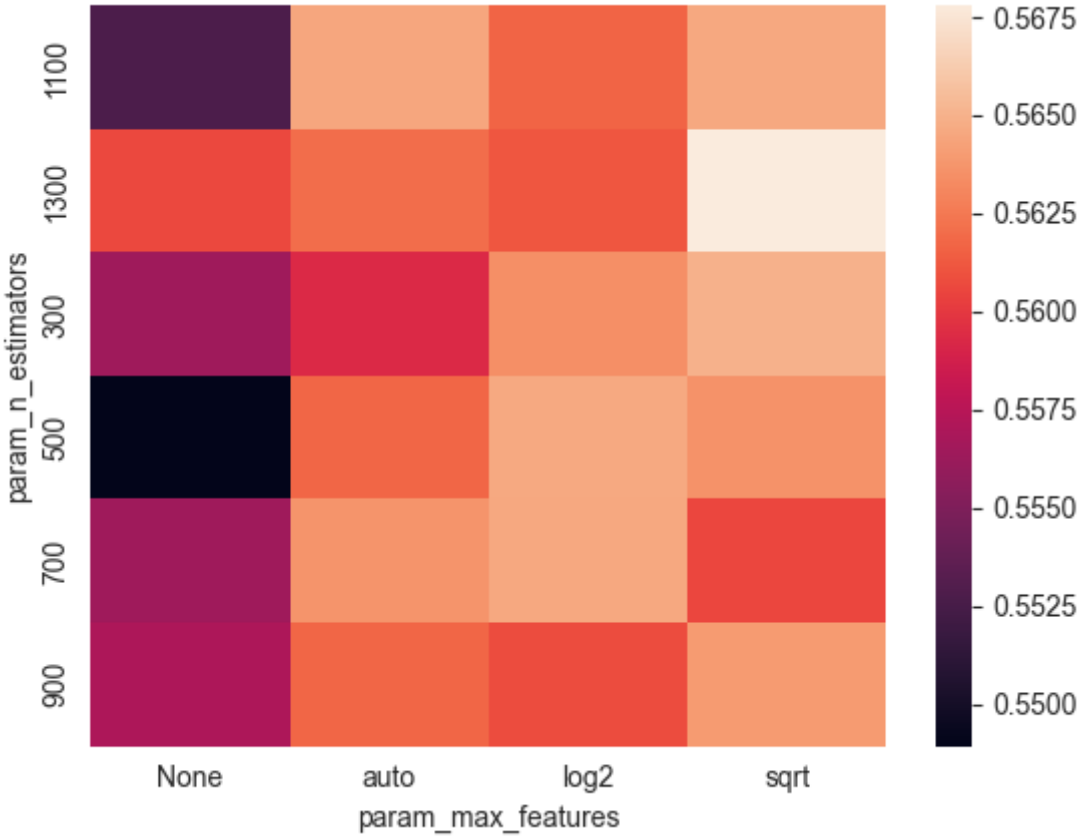
```
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report
from sklearn.metrics import roc_auc_score, roc_curve, precision_score,
recall_score, f1_score
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier


train, test = train_test_split(df, test_size=0.2, random_state=111,
stratify = df.Churn)


x = df.columns[df.columns!="Churn"]
y = "Churn"
train_x = train[x]
train_y = train[y]
test_x = test[x]
test_y = test[y]


#function for model fitting
def churn_prediction(algo, training_x, training_y, testing_x, testing_y,
cols, cf = 'coefficients'):
    algo.fit(training_x,training_y)
    predictions = algo.predict(testing_x)
    probabilities = algo.predict_proba(testing_x)[:,1]

    #coeffs
    if cf == "coefficients":
        coefficients = pd.DataFrame(algo.coef_.ravel())
    elif cf == "features":
        coefficients = pd.DataFrame(algo.feature_importances_)

    column_df = pd.DataFrame(cols)
```

```
    coef_sumry = (pd.merge(coefficients,column_df,left_index= True,
                                right_index= True, how = "left"))
    coef_sumry.columns = ["coefficients","features"]
    coef_sumry = coef_sumry.sort_values(by = "coefficients",ascending =
False)

    print (algo)
    print ("\n Classification report :
\n",classification_report(testing_y,predictions))
    print ("Accuracy   Score : ",accuracy_score(testing_y,predictions))

    #confusion matrix
    conf_matrix = confusion_matrix(testing_y,predictions)
    plt.figure(figsize=(12,12))
    plt.subplot(221)
    sns.heatmap(conf_matrix, fmt = "d",annot=True, cmap='Blues')
    plt.title('Confuion Matrix')
    plt.ylabel('True Values')
    plt.xlabel('Predicted Values')

    #roc_auc_score
    model_roc_auc = roc_auc_score(testing_y,probabilities)
    print ("Area under curve : ",model_roc_auc,"\n")
    fpr,tpr,thresholds = roc_curve(testing_y,probabilities)

    plt.subplot(222)
    plt.plot(fpr, tpr, color='darkorange', lw=1, label = "Auc : %.3f"
%model_roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")

    plt.subplot(212)
    sns.barplot(x = coef_sumry["features"] ,y = coef_sumry["coefficients"])
    plt.title('Feature Importances')
    plt.xticks(rotation="vertical")

    plt.show()
```

- **Selectarea ponderii clasei și a estimatorilor**

```
param_grid1 = {'max_features':['auto', 'sqrt', 'log2', None],
          'n_estimators':[300, 500, 700, 900, 1100, 1300]
         }

rf_model = RandomForestClassifier()
grid1 = GridSearchCV(estimator=rf_model, param_grid=param_grid1, n_jobs=-1,
cv=3, verbose=1, scoring = 'f1')
grid1.fit(train_x, train_y)
grid1.best_estimator_
dt = pd.DataFrame(grid1.cv_results_)
dt.param_max_features = dt.param_max_features.astype(str)
dt.param_n_estimators = dt.param_n_estimators.astype(str)
```

```
table = pd.pivot_table(dt, values='mean_test_score',
index='param_n_estimators',
                       columns='param_max_features')

sns.heatmap(table)
grid1.best_score_
```

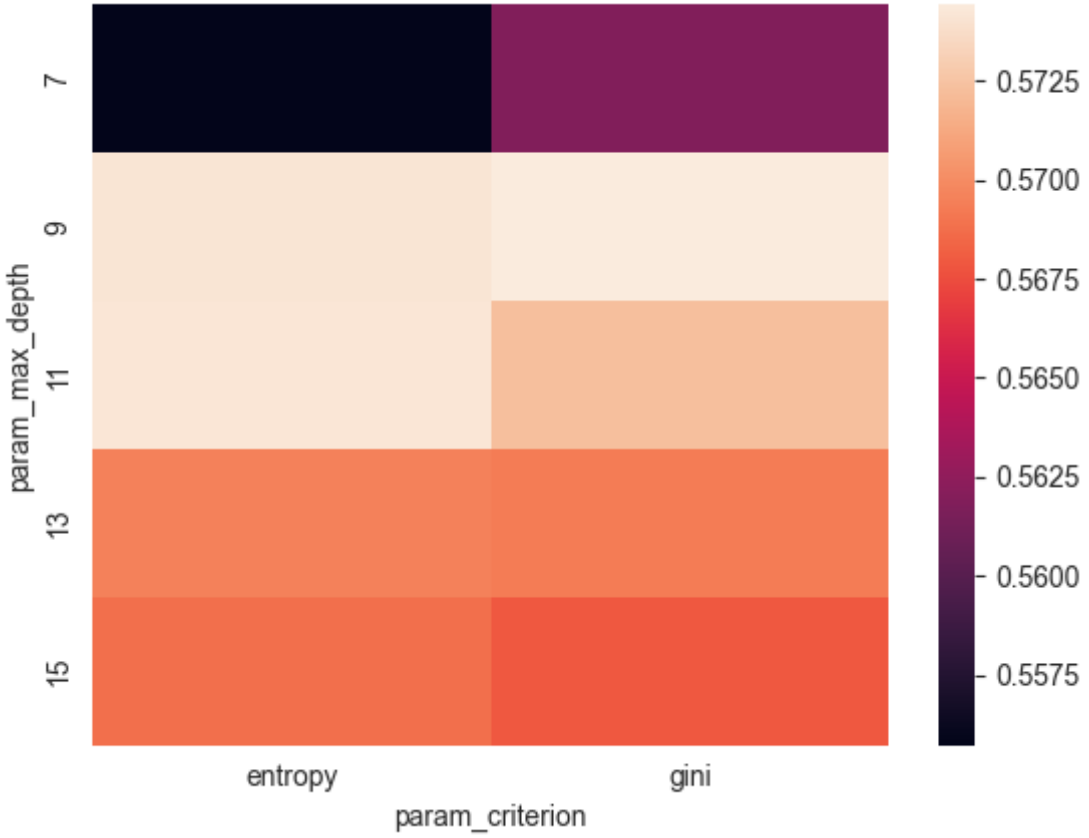



- **Selectarea adâncimii maxime și criteriul de împărțire**

```
param_grid2 = {'max_features':['auto'],
          'n_estimators':[1000, 1100, 1200],
           'criterion': ['entropy', 'gini'],
          'max_depth': [7, 9, 11, 13, 15, None],
         }

rf_model = RandomForestClassifier()
grid2 = GridSearchCV(estimator=rf_model, param_grid=param_grid2, n_jobs=-1,
cv=3, verbose=1, scoring = 'f1')
grid2.fit(train_x, train_y)
grid2.best_estimator_
dt = pd.DataFrame(grid2.cv_results_)
```

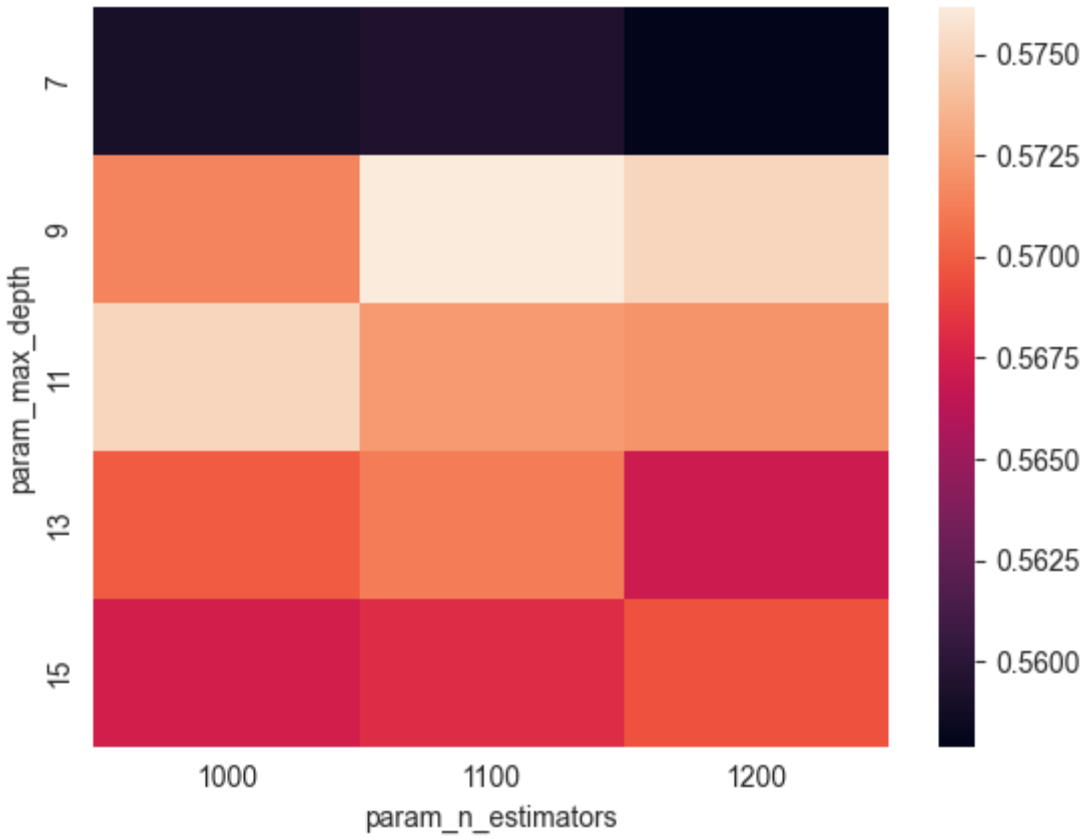
```
table = pd.pivot_table(dt, values='mean_test_score',
index='param_max_depth',
                       columns='param_criterion')
```

```
sns.heatmap(table)
```




```
table = pd.pivot_table(dt, values='mean_test_score',
index='param_max_depth',
                       columns='param_n_estimators')
```

```
sns.heatmap(table)
```




```
grid2.best_score_
```

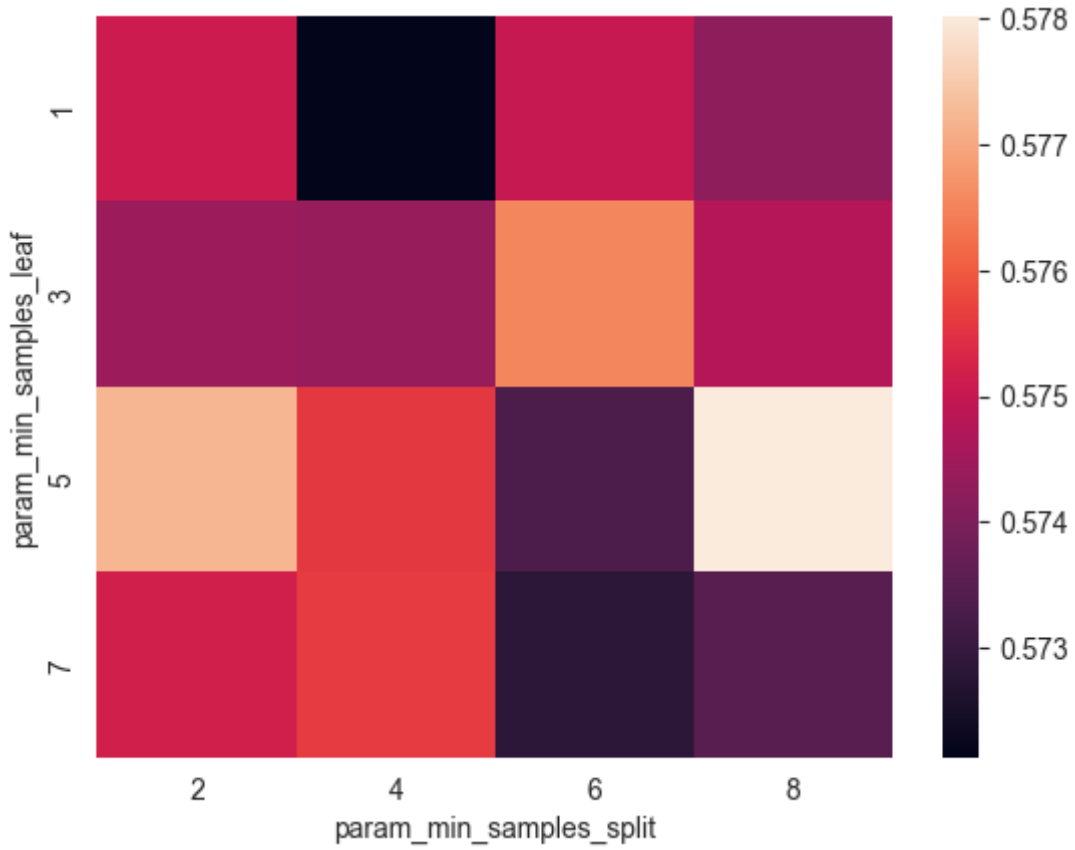
- **Verificarea dacă alte valori de adâncime și de estimare au rezultate mai bune**

```
param_grid2_2 = {'max_features':['auto'],
          'n_estimators':[950, 1000, 1050],
           'criterion': ['entropy'],
          'max_depth': [10, 11, 12],
         }

rf_model = RandomForestClassifier()
grid2_2 = GridSearchCV(estimator=rf_model, param_grid=param_grid2_2,
n_jobs=-1, cv=3, verbose=1, scoring = 'f1')
grid2_2.fit(train_x, train_y)
grid2_2.best_estimator_
grid2_2.best_score_
```

- **Selectarea probelor**

```
param_grid3 = {'max_features':['auto'],
          'n_estimators':[1000],
           'criterion': ['entropy'],
          'max_depth': [10],
```

```
            'min_samples_leaf': [1, 3, 5, 7],
            'min_samples_split': [2, 4, 6, 8]
          }

rf_model = RandomForestClassifier()
grid3 = GridSearchCV(estimator=rf_model, param_grid=param_grid3, n_jobs=-1,
cv=3, verbose=1, scoring = 'f1')
grid3.fit(train_x, train_y)
grid3.best_estimator_
dt = pd.DataFrame(grid3.cv_results_)

table = pd.pivot_table(dt, values='mean_test_score',
index='param_min_samples_leaf',
                       columns='param_min_samples_split')

sns.heatmap(table)
```




- **Selectarea ponderii clasei**

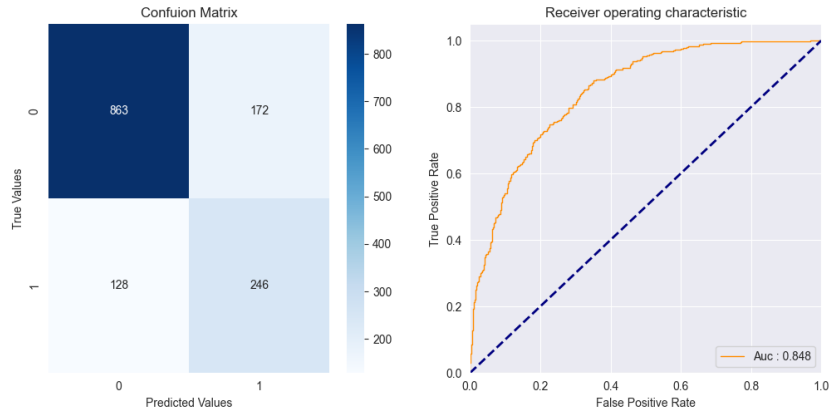
```
param_grid4 = {'class_weight':[{0:1, 1:1}, {0:1, 1:2}, {0:1, 1:3}],
              'max_features':['auto'],
            'n_estimators':[1000],
             'criterion': ['entropy'],
            'max_depth': [10],
            'min_samples_leaf': [1],
            'min_samples_split': [8]
          }
```

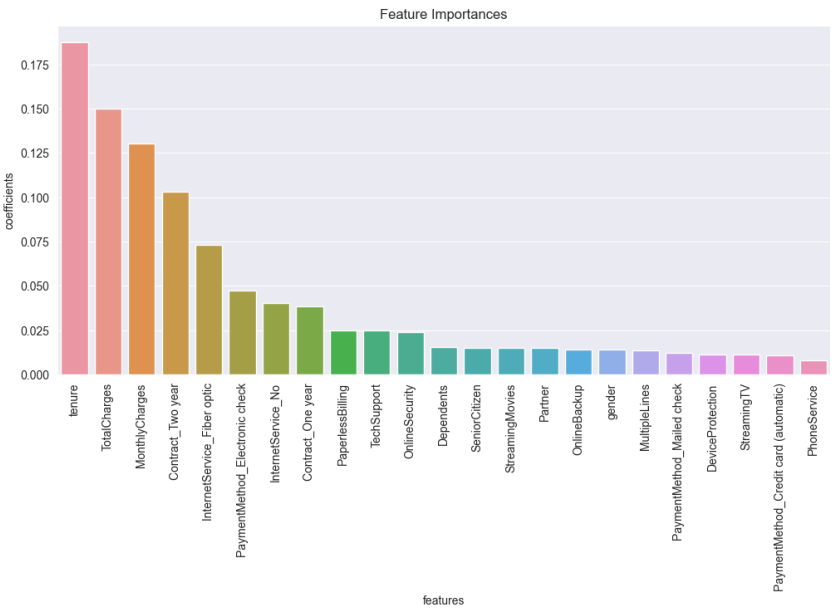
```
rf_model = RandomForestClassifier()
grid4 = GridSearchCV(estimator=rf_model, param_grid=param_grid4, n_jobs=-1,
cv=3, verbose=1, scoring = 'f1')
grid4.fit(train_x, train_y)
grid4.best_estimator_
dt = pd.DataFrame(grid4.cv_results_)
dt.param_class_weight = dt.param_class_weight.astype(str)
table = pd.pivot_table(dt, values='mean_test_score',
index='param_class_weight')

sns.heatmap(table)
```


- **Modelul final**

```
model = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
class_weight={0: 1, 1: 2},
                       criterion='entropy', max_depth=10,
max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=8,
                       min_weight_fraction_leaf=0.0, n_estimators=1000,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
churn_prediction(model, train_x, train_y, test_x, test_y, x,"features")
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight={0: 1, 1
: 2},
```

```
                       criterion='entropy', max_depth=10, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=8,
                       min_weight_fraction_leaf=0.0, n_estimators=1000,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)

 Classification report :
               precision    recall  f1-score   support

           0       0.87      0.83      0.85      1035
           1       0.59      0.66      0.62       374

    accuracy                           0.79      1409
   macro avg       0.73      0.75      0.74      1409
weighted avg       0.80      0.79      0.79      1409

Accuracy   Score :  0.7877927608232789
Area under curve :  0.8479681727763569
```



- **Verificarea performanței modelului pe datele în sine**

```
train_scores = cross_val_score(model, train_x, train_y, cv = 5,
scoring='f1')
train_scores
```

```
array([0.60522273, 0.66346154, 0.62243286, 0.58139535, 0.63836478])
```

```
np.mean(train_scores)
```

**După cum putem vedea că performanța modelului pe datele de testare este aceeași cu da tele de antrenament. Așadar, putem concluziona că nu există supraajustare și subadapta re.**

- **Salvarea modelului**

```
import pickle
pickle.dump(model, open('model.pkl','wb'))
```

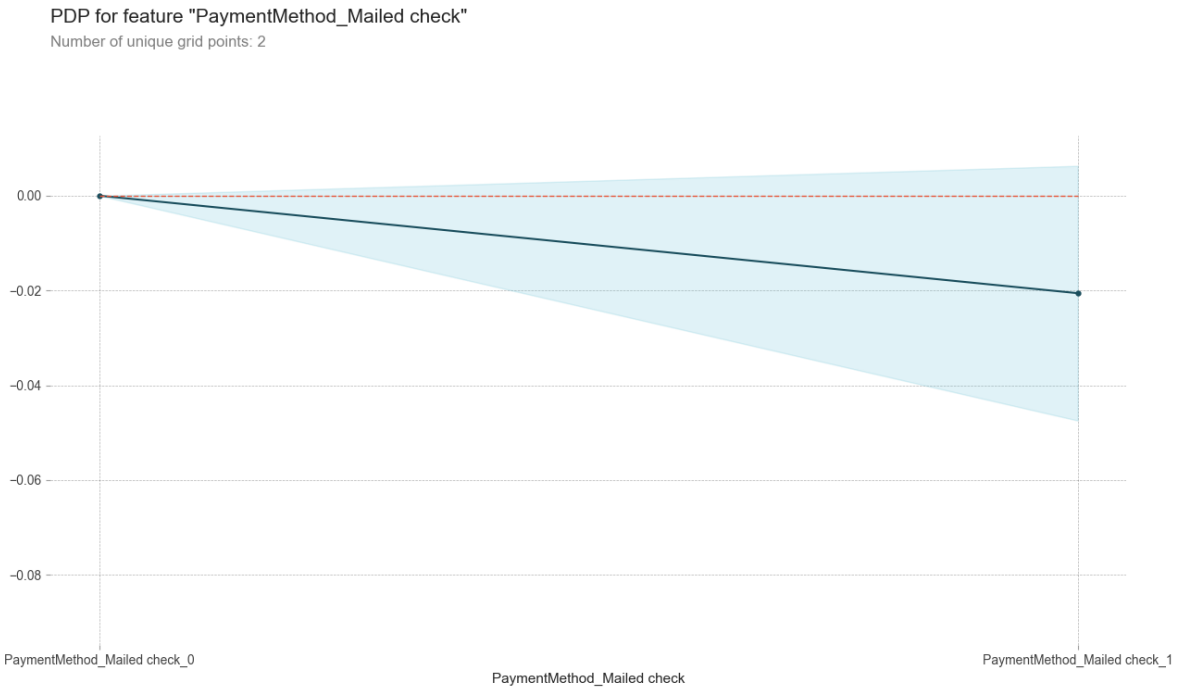
- **Explicarea modelului**

```
import eli5
from eli5.sklearn import PermutationImportance

from pdpbox import pdp, info_plots
perm = PermutationImportance(model, random_state=1).fit(test_x, test_y)
eli5.show_weights(perm, feature_names = test_x.columns.tolist())
```

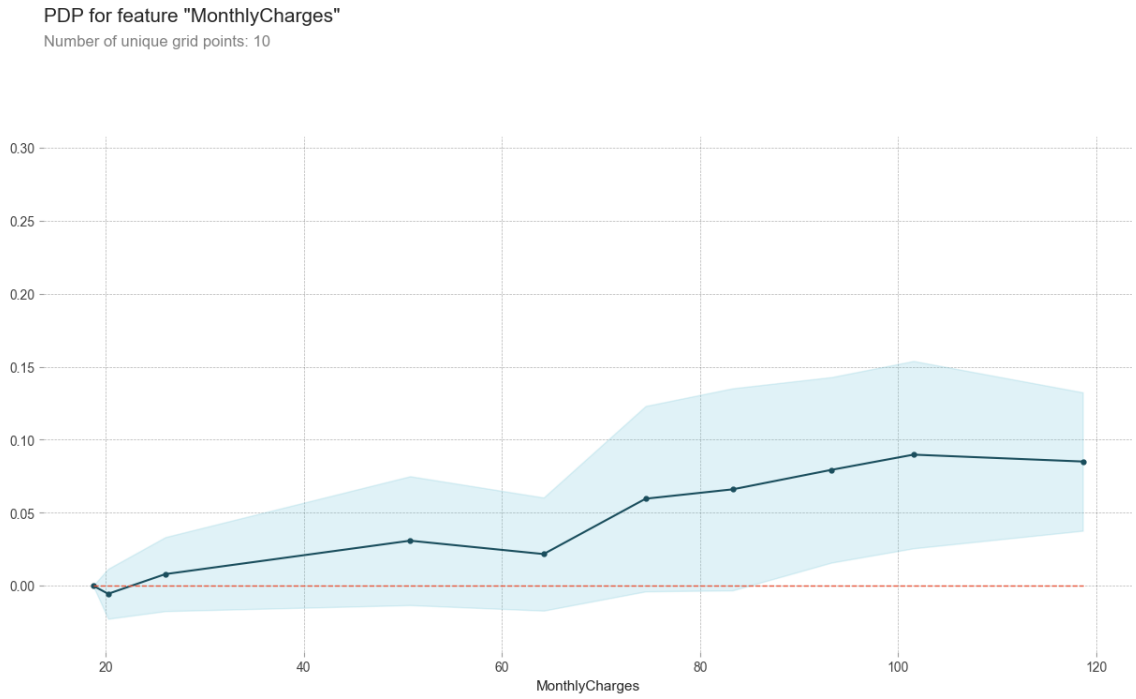
| Weight | Feature |
|---|---|
| 0.0158 ± 0.0053 | InternetService_Fiber optic |
| 0.0050 ± 0.0068 | InternetService_No |
| 0.0048 ± 0.0096 | tenure |
| 0.0041 ± 0.0064 | OnlineSecurity |
| 0.0040 ± 0.0086 | PaperlessBilling |
| 0.0037 ± 0.0060 | PaymentMethod_Electronic check |
| 0.0028 ± 0.0082 | Contract_Two year |
| 0.0027 ± 0.0120 | Contract_One year |
| 0.0010 ± 0.0026 | TechSupport |
| 0.0000 ± 0.0018 | gender |
| 0.0000 ± 0.0020 | OnlineBackup |
| 0.0000 ± 0.0063 | MonthlyCharges |
| -0.0007 ± 0.0020 | MultipleLines |
| -0.0010 ± 0.0041 | SeniorCitizen |
| -0.0011 ± 0.0019 | PhoneService |
| -0.0014 ± 0.0025 | StreamingMovies |
| -0.0016 ± 0.0023 | Dependents |
| -0.0017 ± 0.0044 | StreamingTV |
| -0.0020 ± 0.0011 | DeviceProtection |
| -0.0026 ± 0.0035 | PaymentMethod_Credit card (automatic) |
| | *... 3 more ...* |

**Vizualizarea modului în care parcelele de dependență parțială caută caracteristicile de t op.**

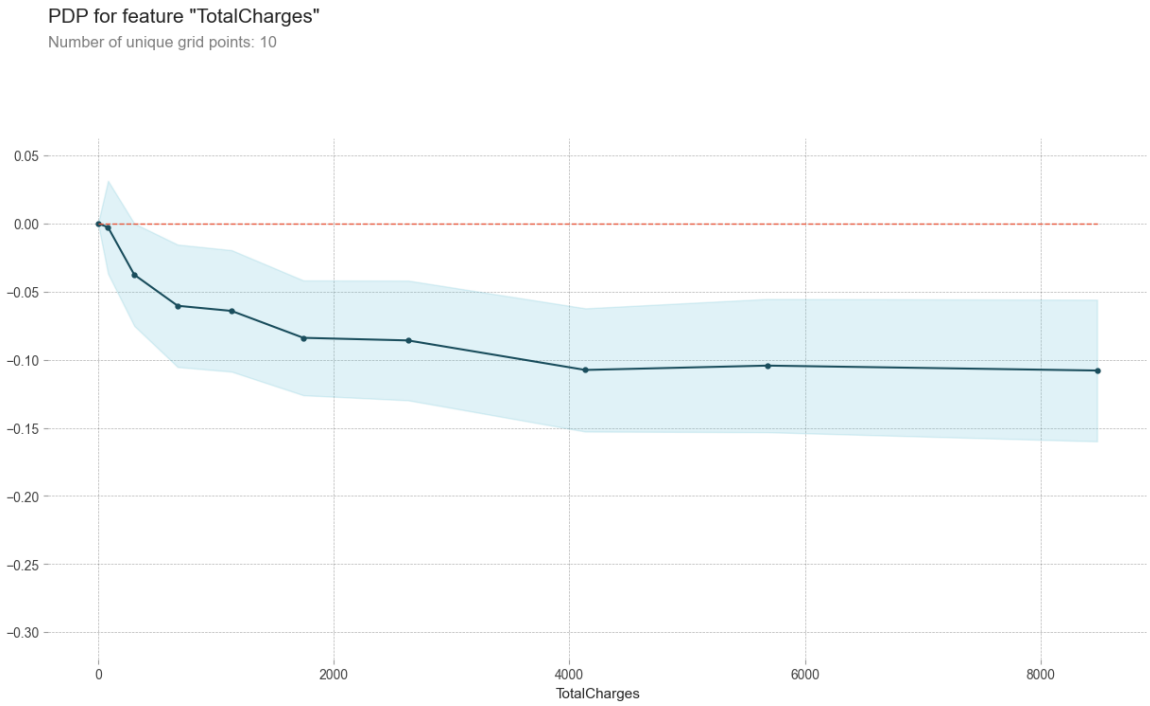
- **Metoda de plată: Cec trimis prin poștă**



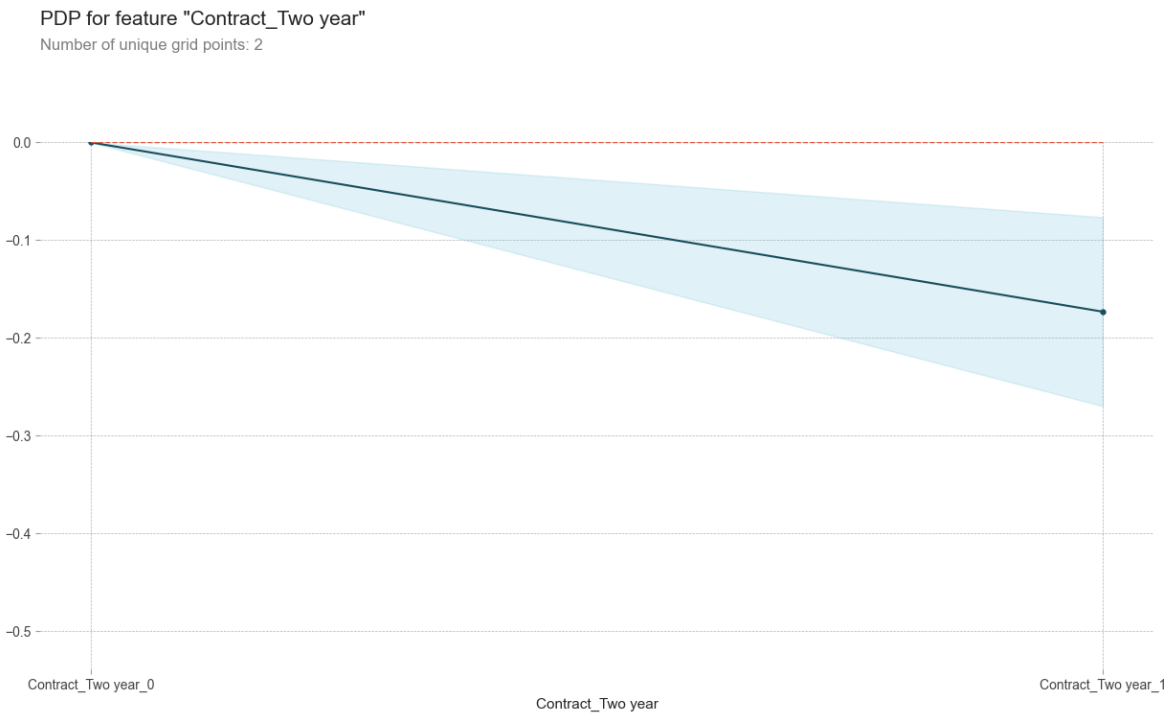
- **Metodă de plată: taxe lunare**



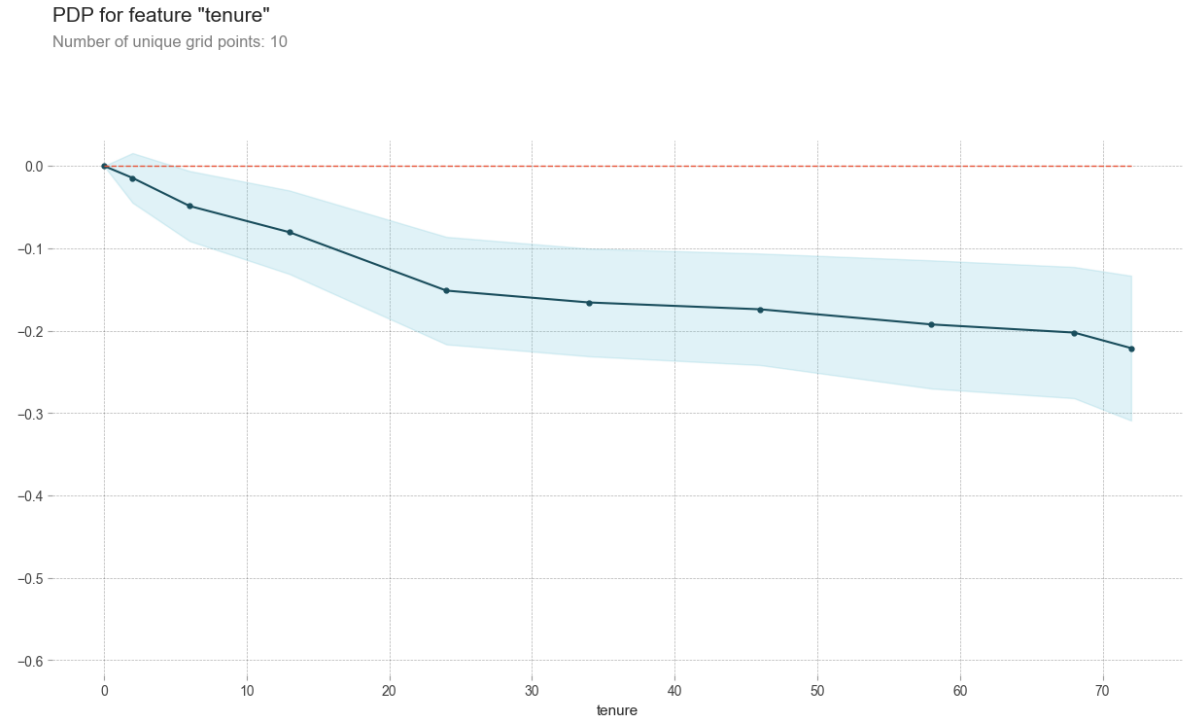

- **Metodă de plată: taxe totale**



- **Contract - Doi ani**




- **Posesiune**



- **Valori Shap**


# *VIII. Concluzii*

În concluzie, predicția ratei clienților este un instrument valoros pentru companiile care doresc să păstreze clienți valoroși, să îmbunătățească satisfacția clienților și să crească veniturile. Folosind algoritmi de analiză a datelor și de învățare automată pentru a prezice clienții care sunt cel mai probabil să își întrerupă relația cu o companie, companiile pot lua măsuri proactive pentru a-și păstra acești clienți și a-și aloca resursele mai eficient. Predicția privind retragerea clienților oferă, de asemenea, informații valoroase asupra comportamentului și preferințelor clienților, care pot informa deciziile viitoare despre produse și marketing. În ansamblu, predicția privind ratarea clienților este un aspect esențial al gestionării relațiilor cu clienții și un factor cheie al succesului în afaceri.