kaggleX

KaggleX- Showcase 2023

Customer Churn in Telecommunication

Lavinia Lau

Background

Professional / academic background	• Business Analytics Manager at Gateway Casinos, Canada • MSc in Data Science and Business Statistics, and degree in Marketing
Current line of work	• Datamine customer and marketing data to tackle business challenges, and deliver actionable insights for informed decision-making • Translate abstract data science concepts into practical solutions for non-technical internal clients
Aspirations	• Continuously learn and stay updated on various data science techniques to effectively address business challenges with the most suitable solutions • I am an experienced R user and R package developer, and would like to use python as smoothly as I do in R.

Project Definition

Project summary: Analyze customers' churn based on their demographics, subscribed services and account information, so as to adjust marketing offers / services to prevent the potential churn (i.e. leaving a company).

Topics covered:

Preparation & Exploration	• Project setup for reproducible results • Data cleansing • Exploratory data analysis (EDA) and feature engineering • Data visualization to study relationship • Mutual information
Machine Learning	• Handle imbalance classification data using SMOTE • Data preprocessing for categorical data (label encoding) and numerical data (scaling) • 3 ML models: Random forest, XGBoost and TensorFlow • Model evaluation for classification (accuracy score and classification report) • Permutation importance for feature selection
Model Explainability & Applications	• 1D partial dependence plots for discovering tipping points • 2D partial dependence plots for further insights into marketing applications • SHAP summary plot

Preparation and Exploration

This is a classification problem with an imbalanced target variable, which I will address the imbalance using SMOTE later.

1. Target variable: Churn

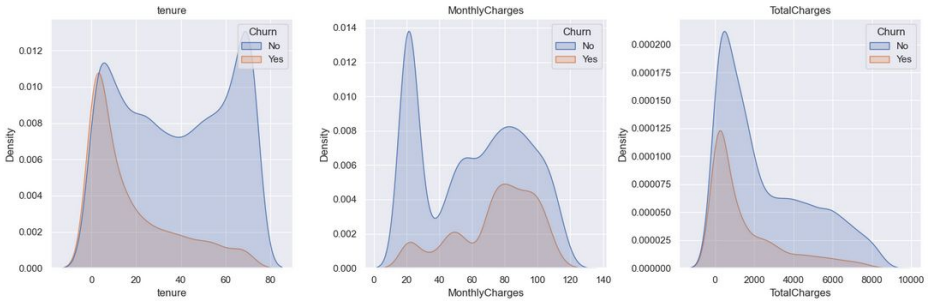
Let's explore the target variable first.

```
# churn count
cnt = org.Churn.value_counts()
print('Count of Churn \n', cnt, '\n')

# churn %
cnt_perent = cnt/org.shape[0] * 100
print('Percentage of Churn \n', cnt_perent, '\n')

# Churned customers' value
print('Monthly Charges from Customers by Churn Group')
org.groupby('Churn')['MonthlyCharges'].sum() / org['MonthlyCharges'].sum() * 100
```

```
Count of Churn
 Churn
No     5174
Yes    1869
Name: count, dtype: int64

Percentage of Churn
 Churn
No     73.463013
Yes    26.536987
Name: count, dtype: float64
```

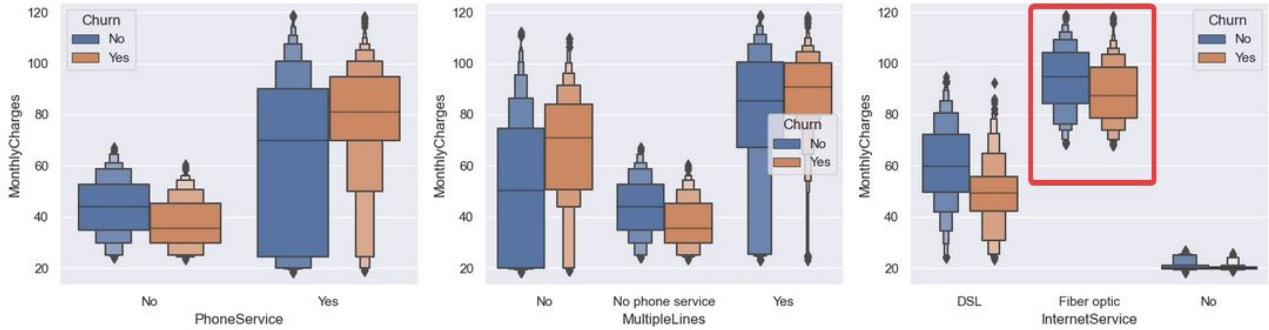
I have also studied the churn rate with all independent variables. Below is an example of plotting multiple density plots at once for EDA. Monthly charges and contract length are the two key factors influencing churn.

```
# numerical
def plot_numerical(variables, nrow, ncol, size):
    fig, axes = plt.subplots(nrow, ncol, figsize=(15, size))
    axes = axes.flatten()

    for i, variable in enumerate(variables):
        ax = axes[i]
        sns.kdeplot(data=org, x=variable, hue='Churn', fill=True, ax=ax)
        ax.set_title(variable)

    plt.tight_layout()
    plt.show()


# plot
var_num = ['tenure', 'MonthlyCharges', 'TotalCharges']
plot_numerical(var_num, 1, 3 ,5)
```

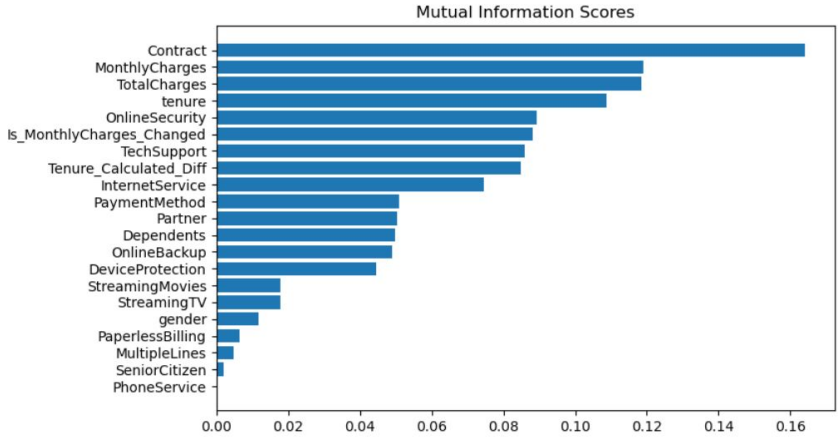


Preparation and Exploration

After locating price is a key factor in churn, I have plotted it against other categorical variables. Some services (e.g. fibre optics) are more expensive than others.



Before modelling, I have used mutual information to rank the variables by their scores.



Machine Learning (Model Parameters and Feature Selection)

After preprocessing the numerical data by scaling and categorical by label encoding, I proceed to predict churn by applying random forest, XGBoost and TensorFlow.

We also need to finetune the models' number of estimators to see when accuracy score peaks.

In [46]:
```
# test the best number of tree
def get_accuracy(no_of_estimators):
    rf = RandomForestClassifier(n_estimators=no_of_estimators, random_state=0)
    rf_model = rf.fit(X_train_rf, y_train)
    preds = rf_model.predict(X_test_rf)
    acc = accuracy_score(y_test, preds)
    return(acc)


# loop to check best no of estimators
for estimators in [25, 50, 100, 500, 750, 1000]:
    my_acc = get_accuracy(estimators)
    print('No of estimators ', estimators, "; accuracy score ", my_acc)
```

```
No of estimators  25 ; accuracy score  0.8419018167761886
No of estimators  50 ; accuracy score  0.843448009277155
No of estimators  100 ; accuracy score  0.8426749130266719
No of estimators  500 ; accuracy score  0.8484731349052957
No of estimators  750 ; accuracy score  0.8484731349052957
No of estimators  1000 ; accuracy score  0.8465403942790878
```

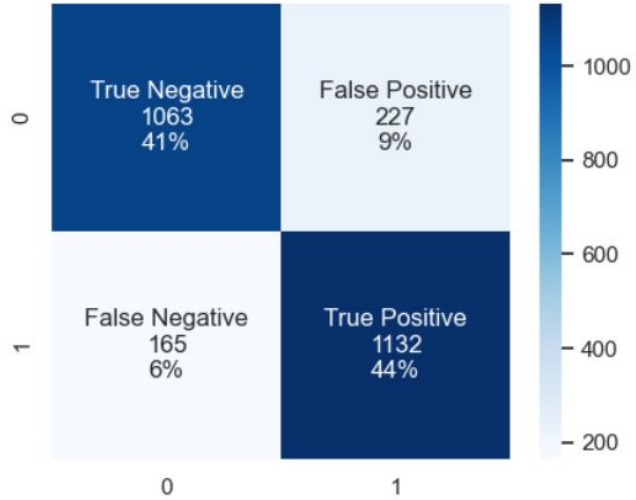
In addition to adjusting parameters, I employed permutation importance for feature selection. If a feature has a low impact, I remove it and rerun the model to assess if performance improves.

```
perm = PermutationImportance(rf_model, random_state=1).fit(X_test_rf, y_test)
eli5.show_weights(perm, top=no_of_features, feature_names = X_test_rf.columns.tolist())
```

Weight	Feature
0.1089 ± 0.0161	Contract
0.0478 ± 0.0124	MonthlyCharges_Scaled
0.0435 ± 0.0051	TechSupport
0.0405 ± 0.0025	OnlineSecurity
0.0207 ± 0.0032	OnlineBackup
0.0196 ± 0.0061	tenure_Scaled
0.0169 ± 0.0022	TotalCharges_Scaled
0.0167 ± 0.0084	Is_MonthlyCharges_Changed
0.0152 ± 0.0032	PaymentMethod
0.0118 ± 0.0033	Tenure_Calculated_Diff
0.0109 ± 0.0058	Dependents
0.0081 ± 0.0033	Partner
0.0080 ± 0.0036	DeviceProtection
0.0057 ± 0.0049	InternetService
0.0048 ± 0.0017	PaperlessBilling
0.0036 ± 0.0016	PhoneService

Machine Learning (Model Evaluation and Comparison)

```
Accuracy score:
 0.8484731349052957

Classification matrix:
              precision    recall  f1-score   support

           0       0.87      0.82      0.84      1290
           1       0.83      0.87      0.85      1297

    accuracy                           0.85      2587
   macro avg       0.85      0.85      0.85      2587
weighted avg       0.85      0.85      0.85      2587
```



The picture on the left shows the accuracy score and classification matrix of the random forest model. We will mainly focus on churned group in the classification matrix as we will want to retain the potential churned customers.

Below is the comparison of different models:

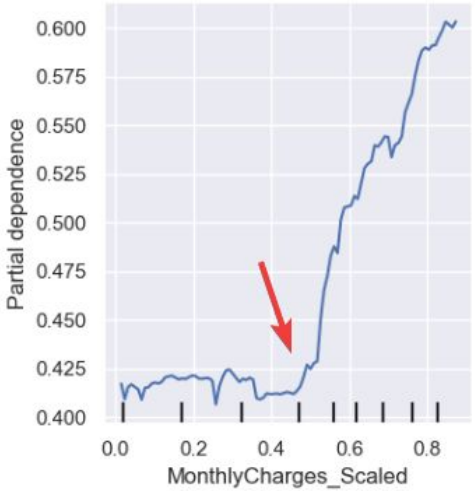
Models	Random Forest	XGBoost	TensorFlow
Accuracy score (overall)	84.8%	84.7%	82.6%
Precision of churned customers	83%	83%	82%
Recall of churned customers	87%	87%	83%

Precision for churned customers measures how accurately we identify customers who will churn. Recall for churned customers gauges our ability to catch all customers who genuinely churn.

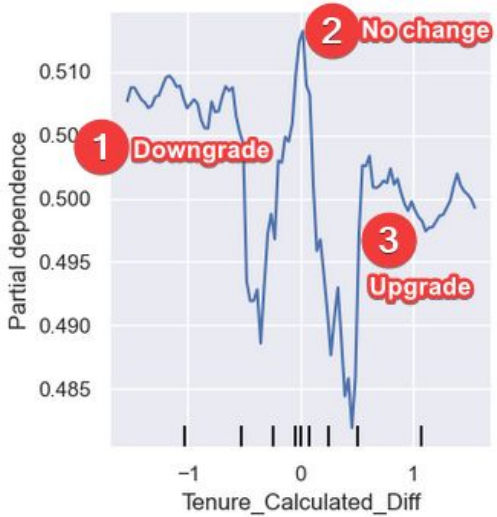
Random Forest got the best result.

Explainability & Applications - What Are the Tipping Points?

Tipping point means the threshold where customers' churn rate changes significantly. We can create 1D partial dependence plots for individual variables to understand how they affect customer churn. Below are some examples of applications.



Churn rate increases significantly when monthly charges hit **$68.5** (i.e. Monthly charges scaled > 0.5).



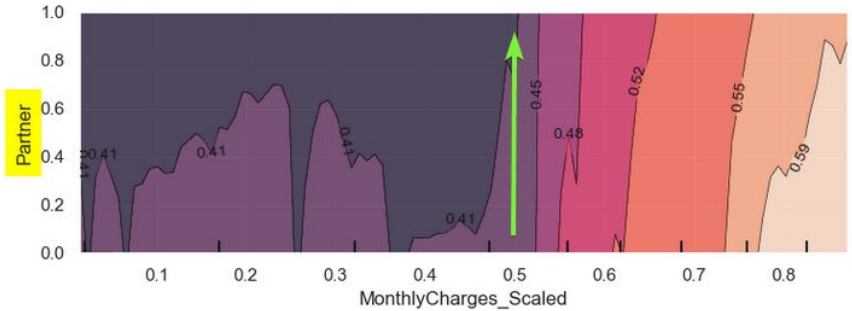
During feature engineering, I have created a new variable to see if customers have downgrade / upgrade their contract.

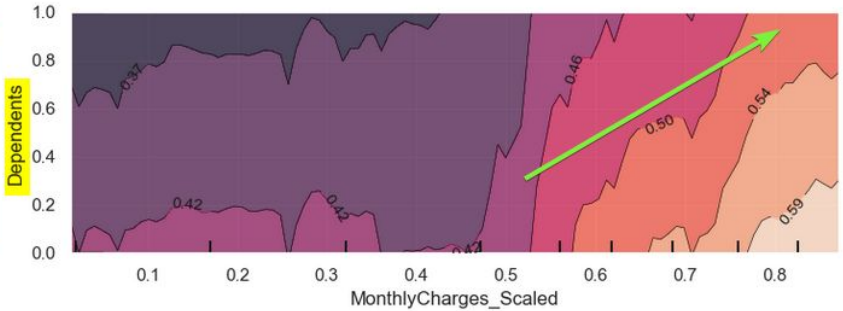
Observations:
- No change: Highest churn
- Downgrade: Some degree of churn
- Upgrade: Lowest churn

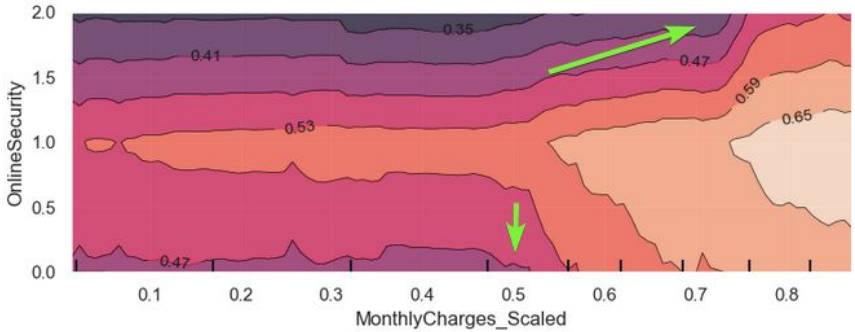
Implications:
Allow customers to change their contract if they reach out to terminate it.

Explainability & Applications - What Other Factors Can We Manipulate?

We want to determine if customers are more willing to accept higher charges under specific circumstances so we can adapt our strategy accordingly. Below are some examples of 2D partial dependence plots for this purpose.







Observations:
The top two graphs show customers will leave if charges go higher than $68.5, with or without a partner. But Customers with dependents are more accepting of higher charges.

The bottom graph shows customers with online security service leave when charges reach $88.6, $20 dollar more than the baseline tipping point $68.5.

Project Links

- Project details with python code and commentary: https://github.com/Lavinialau/Telecom-churn/blob/master/01_code/0_summary.ipynb
- Youtube: https://youtu.be/GEbF1Ujwn3k