Supervised Machine Learning: Classification Churn Project

Paulina Kossowska
September 2021

“One of the main objectives of this course is to help you gain hands-on experience in communicating insightful and impactful findings to stakeholders. In this project you will use the tools and techniques you learned throughout this course to train a few machine learning algorithms on a data set that you feel passionate about, select the classification that best suits your needs, and communicate insights you found from your modeling exercise.”

Understanding the Data

Telco-Customer-Churn.csv

The dataset I want to analyze comes from Kaggle.

The Telco customer churn data contains information about a fictional telco company that provided home phone and Internet services to 7043 customers in California in Q3. It indicates which customers have left, stayed, or signed up for their service. Multiple important demographics are included for each customer, as well as a Satisfaction Score, Churn Score, and Customer Lifetime Value (CLTV) index.

Link

Understanding the Data:

- **Customers who left within the last month – the column is called Churn**
- **Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies**
- **Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges**
- **Demographic info about customers – gender, SeniorCitizen which indicates if the customer is 65 or older, and if they have partners and dependents**

Goals of this analysis

There are two main goals for this analysis:

1. Performed Exploratory Data Analysis which will help me understand with what type of data I work on.
2. Performed Classification Machine Learning algorithms together with Model Evaluation which will help me built a model to predict target - **CHURN** variable with higher accuracy.

1. EDA:
    - 1.1. Data Exploration
    - 1.2. Data Transformation
    - 1.3. Data Visualization
    - 1.4. Data Pre-processing

2. Classification Modeling:
    - 2.1. Logistic Regression
    - 2.2. K-Nearest Neighbor
    - 2.3. Support Vector Machines
    - 2.4. Decision Tree

Exploratory Data Analysis

Data Exploration

Using shape(), describe(), dtypes(), isnull() methods from pandas library I found out that in my dataset were 7043 rows and 21 columns.

One of column - ‘customerID’ was dropped at the beginning of my analysis because it will not bring any valuable informations.

17 of them was an object columns and the rest was numeric columns. I also learned that there were 11 missing values in column ‘TotalCharges’, so I decided to drop them all.

Data Transformation

In this section I wanted to take a closer look at data types I had and if needed prepared them for machine learning.

Doing some step by step analysis I identified variables which are binary, categorical and not ordinal, categorical and ordinal and finally numeric variables.

Then I took LabelBinarizer() and LabelEncoder() from sklearn.preprocessing together with pandas get_dummies and transformed all of this variables into format which sufficient my expectations.

Binary Variables:

- gender
- SeniorCitizen
- Partner
- Dependents
- PhoneServic
- PaperlessBilling

Categorical and ordinal:

- tenure
- Contract

Categorical and not ordinal:

- MultipleLines
- InternetService
- OnlineSecurity
- OnlineBackup
- DeviceProtection
- TechSupport
- StreamingTV
- StreamingMovies
- PaymentMethod

Numerical Variables:

- MonthlyCharges
- TotalCharges

Target Variables:

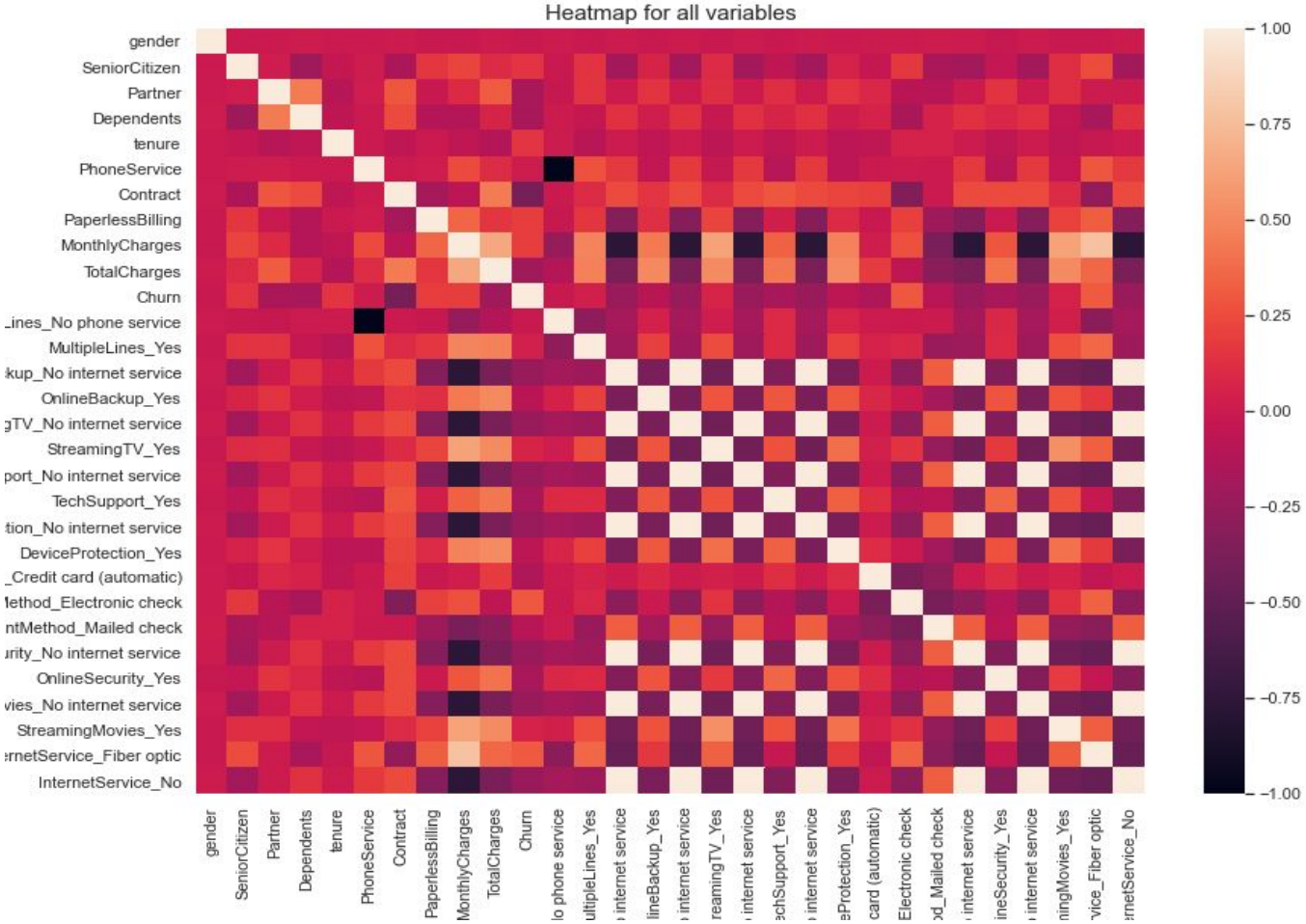
- Churn

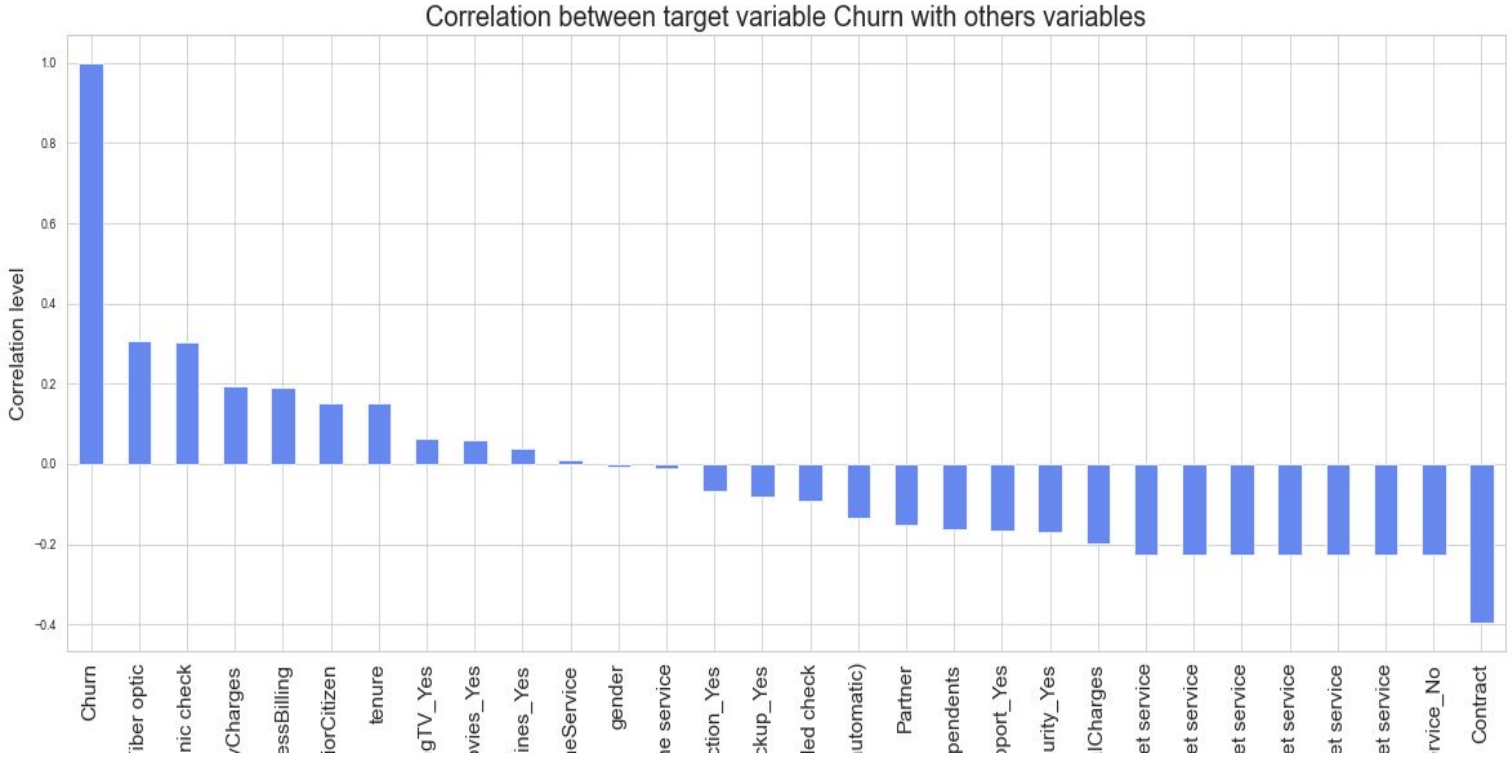
Data Visualization

In this section first I will focus on showing correlations between target variable - **CHURN** and other variables. I created heatmap and bar chart which presents this metric.

Key Insights:

- three variables with which Churn variable is positive correlated the most:
  - InternetService_Fiber optic
  - PaymentMethod_Electronic check
  - MonthlyCharges
- three variables with which Churn variable is negative correlated the most:
  - Contract
  - StreamingMovies_No internet service
  - DeviceProtection_No internet service
- a barely noticeable correlation exists between the target and the variables:
  - PhoneService
  - gender
  - MultipleLines_No phone service


Heatmap for all variables
gender
SeniorCitizen
Partner
Dependents
tenure
PhoneService
Contract
PaperlessBilling
MonthlyCharges
TotalCharges
Churn
_ines_No phone service
MultipleLines_Yes
kup_No internet service
OnlineBackup_Yes
gTV_No internet service
StreamingTV_Yes
port_No internet service
TechSupport_Yes
tion_No internet service
DeviceProtection_Yes
_Credit card (automatic)
lethod_Electronic check
ntMethod_Mailed check
urity_No internet service
OnlineSecurity_Yes
vies_No internet service
StreamingMovies_Yes
rnetService_Fiber optic
InternetService_No
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

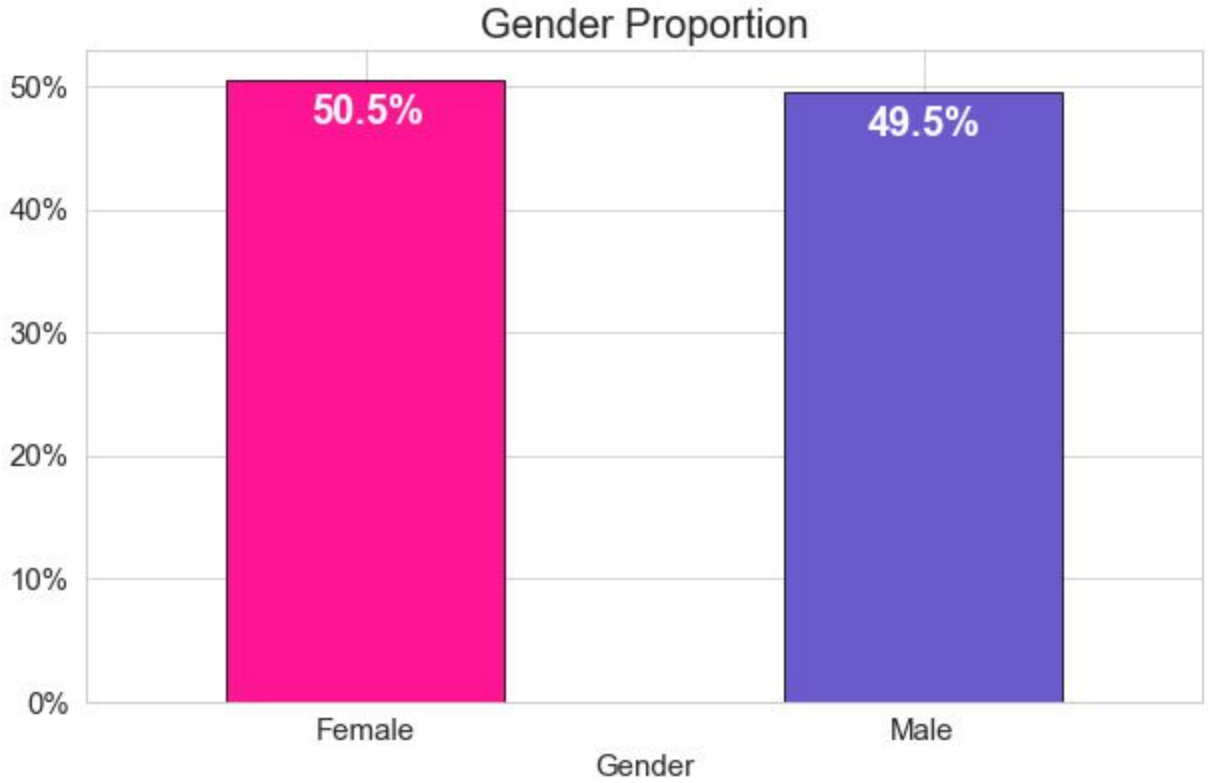

Correlation between target variable Churn with others variables
Correlation level
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
Churn
iber optic
nic check
rCharges
assBilling
iorCitizen
tenure
gTV_Yes
wies_Yes
ines_Yes
eService
gender
e service
ction_Yes
ckup_Yes
led check
utomatic)
Partner
pendents
port_Yes
urity_Yes
lCharges
et service
et service
et service
et service
et service
et service
rvice_No
Contract

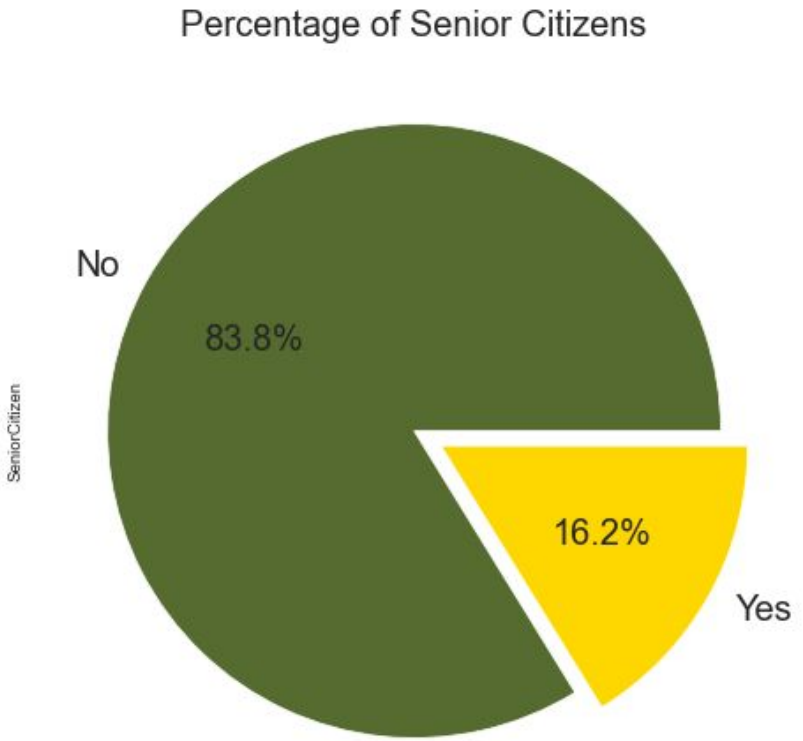
Data Visualization

In the next couple of slides I will presents some visualization about demographics part of data I have.

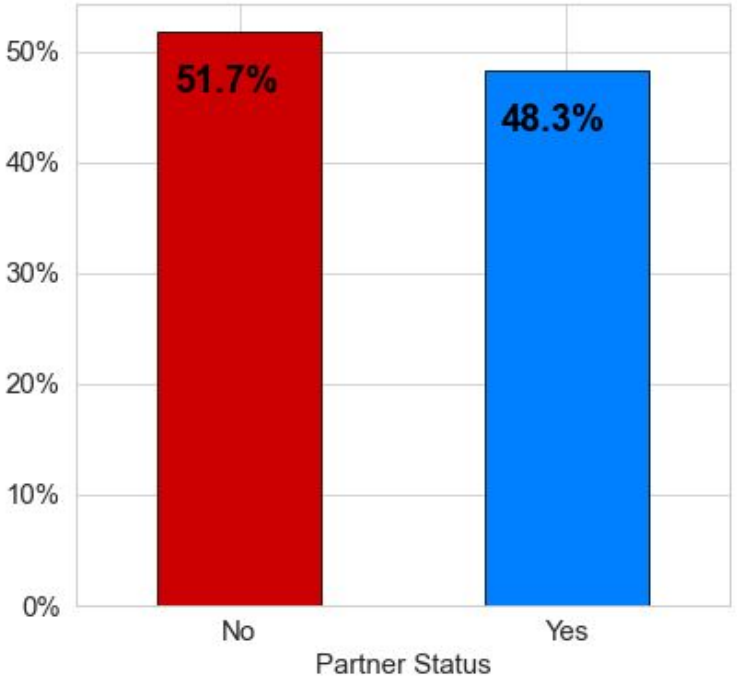
Key Insights:

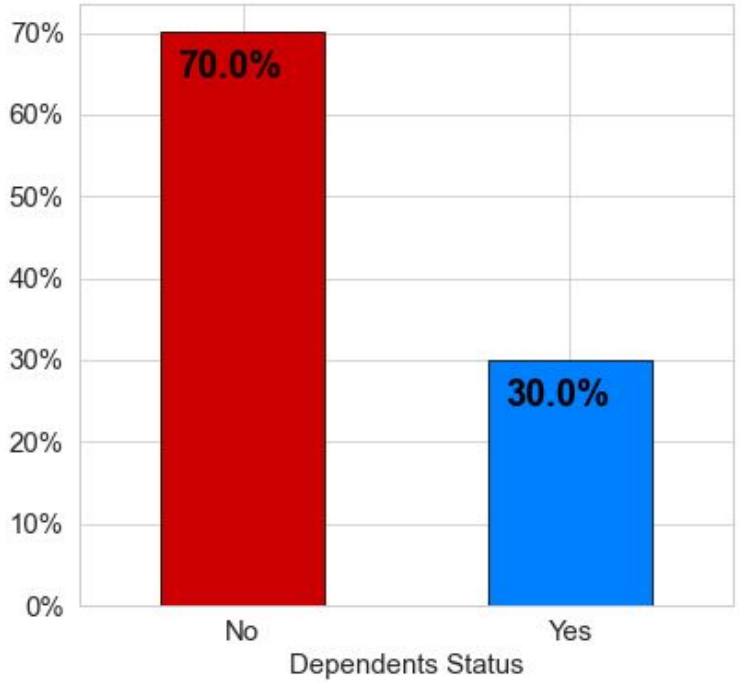
- here are equal spread of gender in our dataset with slightly better result for Female which pose 50.5% and Male which pose 49.5%
- SeniorCitizen column indicates if the customer is 65 or older, we now thanks to pie chart presented above that more than 80% of our customers are less than 65 year old, and those within this age or older pose 16.2%
- as we can see on a bar plot, the result for Partner Status are spread equal in dataset: 51.7% don't have partner and 48.3% declared that have one
- in other hand the result for Dependents Status are not spread so equal and we can see that 70% of our customer don't have any dependents and only 30% declared that have one


Gender Proportion
50%
40%
30%
20%
10%
0%
50.5%
49.5%
Female
Male
Gender


Percentage of Senior Citizens
SeniorCitizen
No
83.8%
16.2%
Yes

Partner and Dependents Status





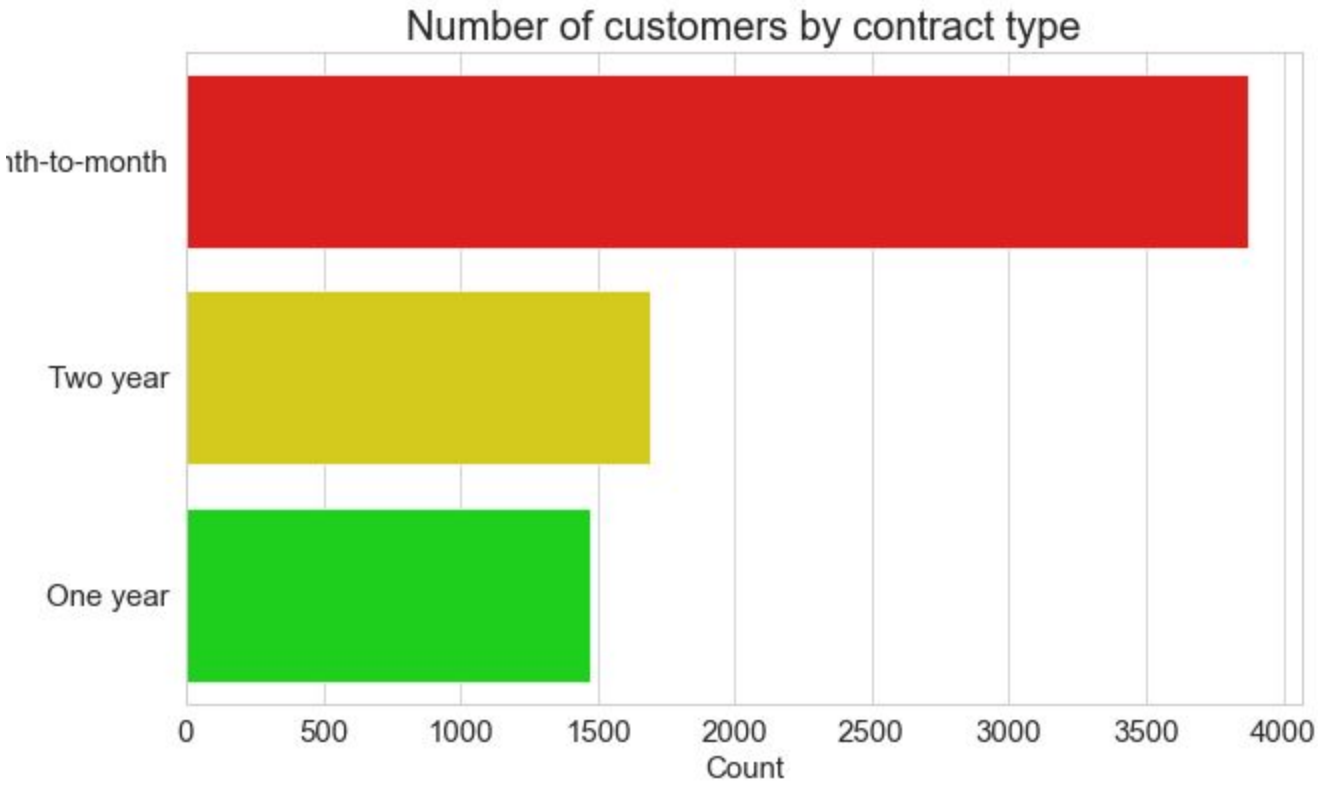
Data Visualization

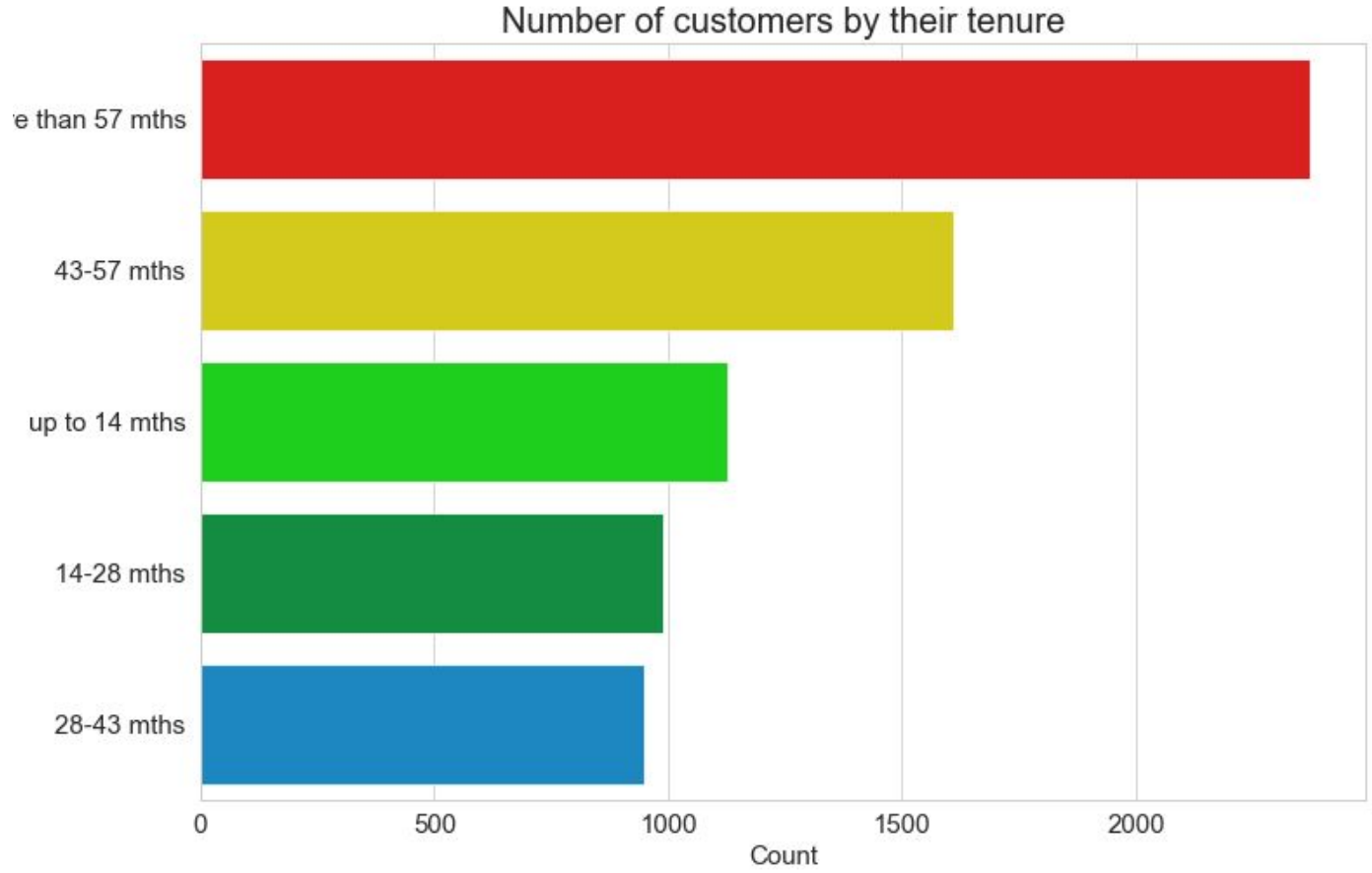
Next portion of my visualization will reflect to categorical and ordinal data I have.

What is important to notice is a fact that during transformation phase I used pandas cut options and transform tenure column into 5 bins which represent the among of time customers stay with us.

Key Insights:

- the main contract type is Month-to-month with more than 3800 customers had it, next there are Two year contract and One year contract with less than 1500 customers had it
- more than 55% (two main tenure category) of customers are with us more than 43 months which is more than 3 years
- values for three remaining category in tenure are spread similar and are about 15% in each one


Number of customers by contract type
ıth-to-month
Two year
One year
0
500
1000
1500
2000
2500
3000
3500
4000
Count


Number of customers by their tenure
e than 57 mths
43-57 mths
up to 14 mths
14-28 mths
28-43 mths
0
500
1000
1500
2000
Count

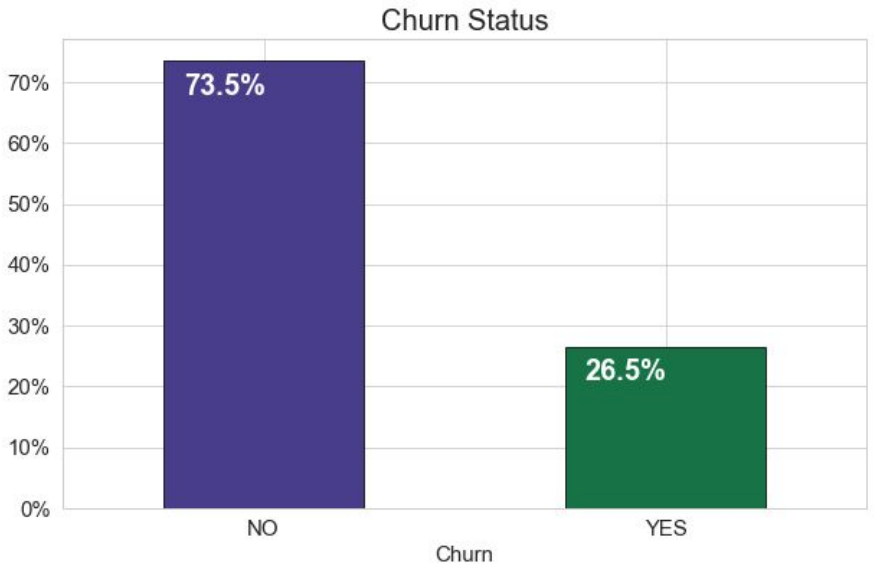
Data Visualization

In this part of visualization I focused on target variables - **CHURN**.

First thing important to notice is a fact that our target variable is highly imbalanced.

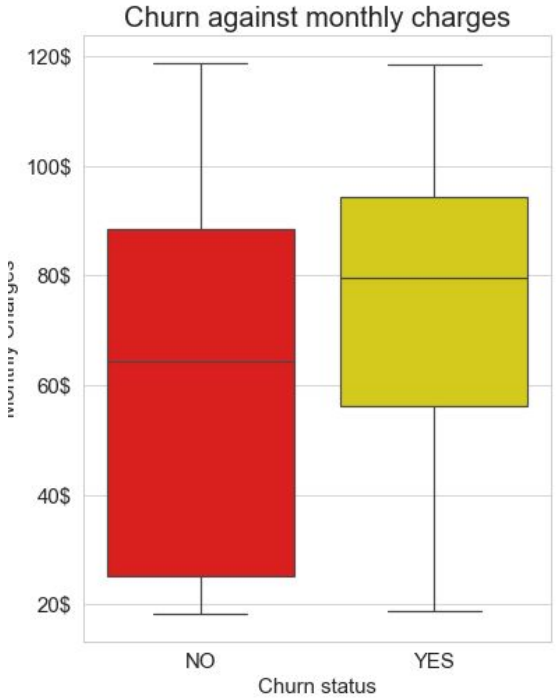
73.5% of customers will stay with us as a telecommunication provider and 26.5% will churn.

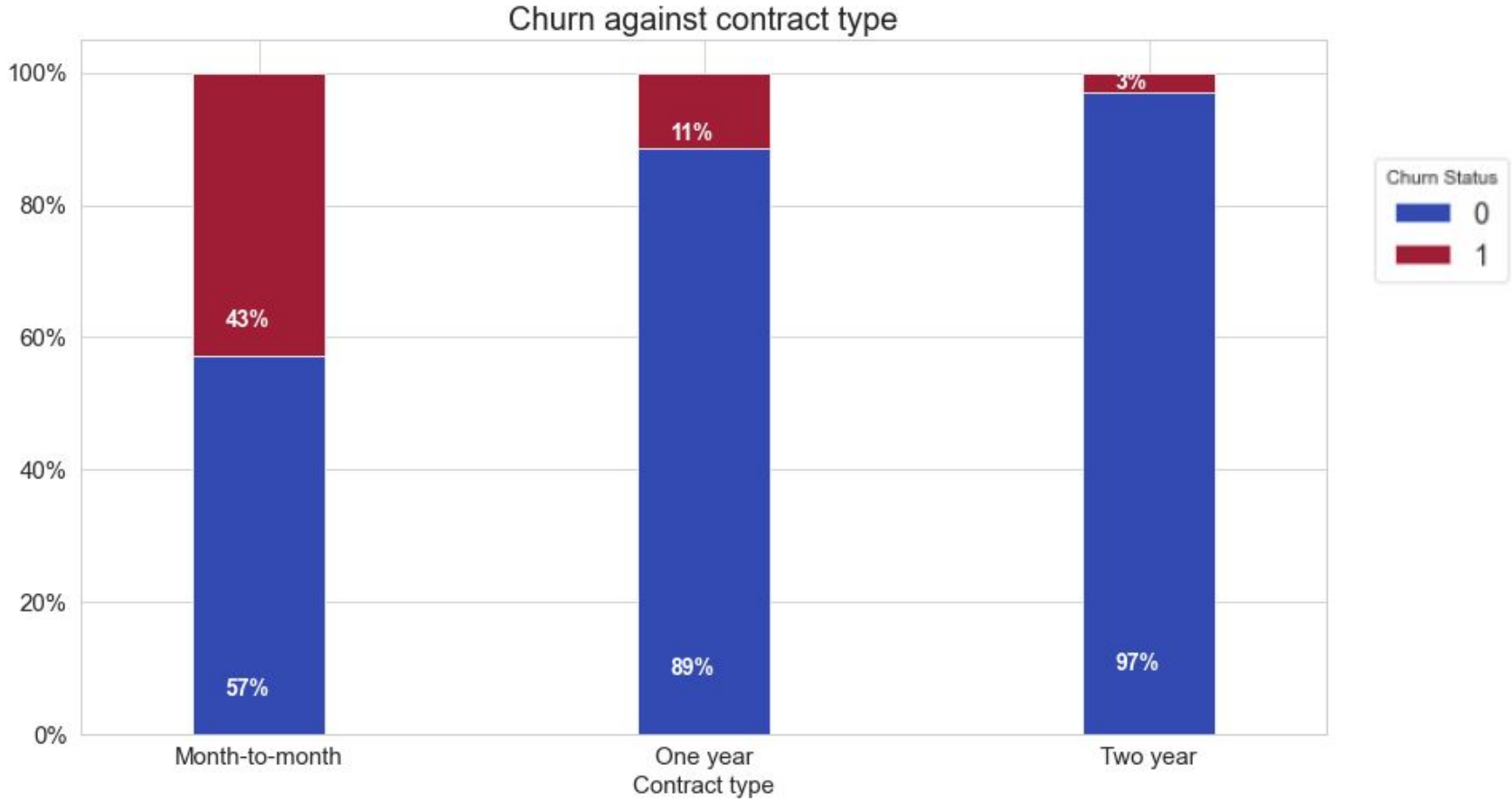
Clearly the data is skewed as we would expect a large majority of the customers to not churn.

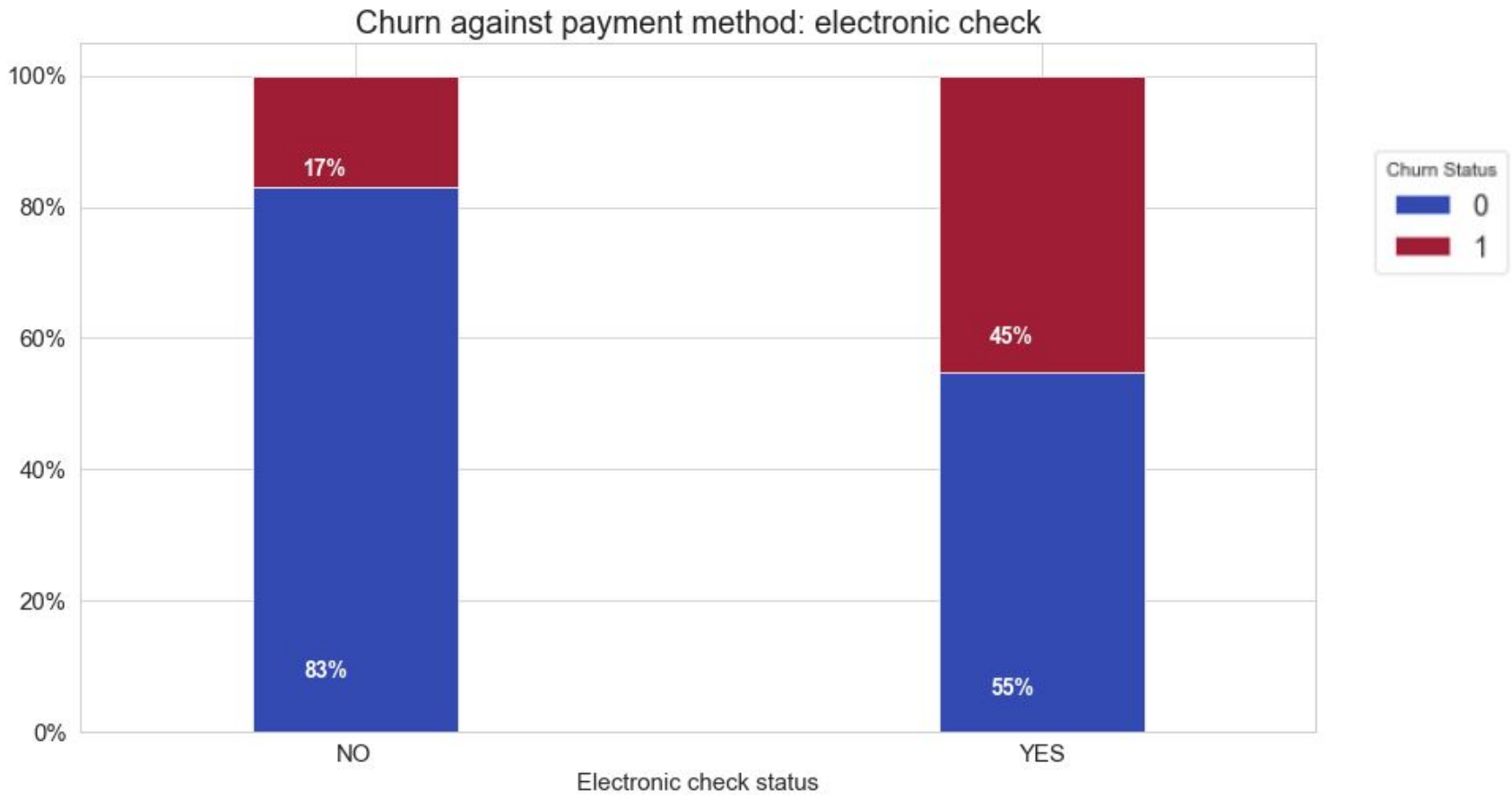


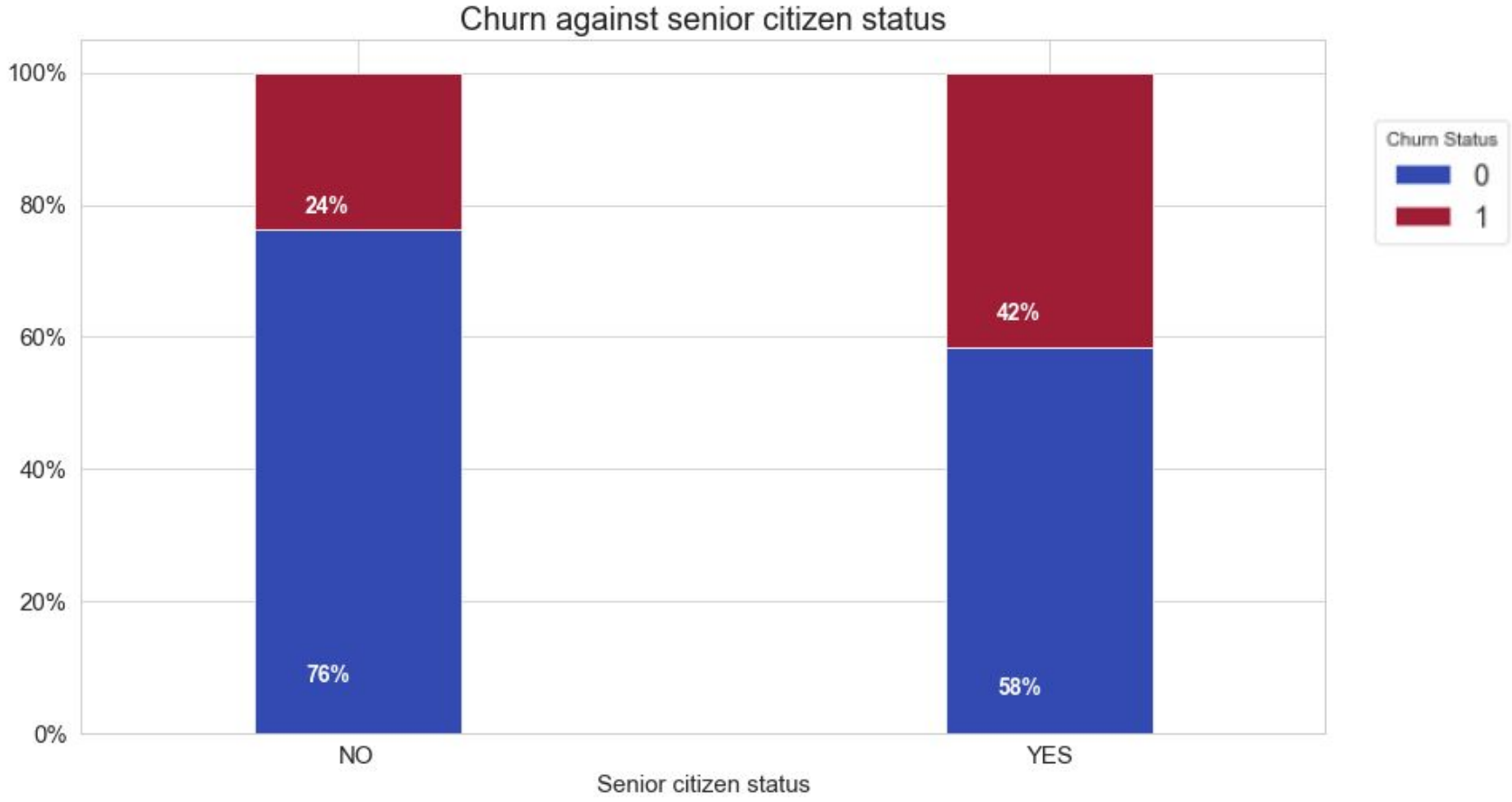
Data Visualization

Based on heatmap and bar graph I plotted before I decided to go further and check how churn looks in comparing to variables: **MonthlyCharges, Contract, PaymentMethod and SeniorCitizen status.**




Churn against contract type
100%
80%
60%
40%
20%
0%
43%
57%
11%
89%
3%
97%
Month-to-month
One year
Two year
Contract type
Churn Status
0
1


Churn against payment method: electronic check
100%
80%
60%
40%
20%
0%
17%
83%
45%
55%
NO
YES
Electronic check status
Churn Status
0
1


Churn against senior citizen status
100%
80%
60%
40%
20%
0%
24%
76%
42%
58%
NO
YES
Senior citizen status
Churn Status
0
1

Key Insights:

- customers who churn pay on average more than customers who not churn
- the highest percentage customers who churn is equal to 43% and it is among customers with month-to-month contract type
- 45% of clients with electronic check status are churn
- as we already know more than 16% of our clients are over 65 years and among them 42% are churn which is high indicator

Data Pre-processing

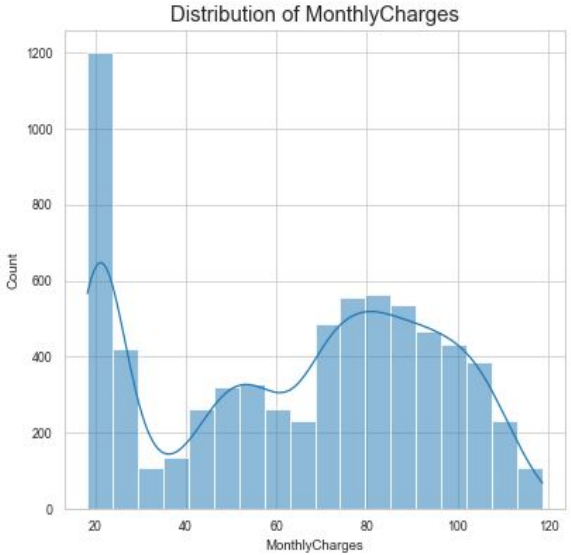
In this subsection of my EDA I proceed:

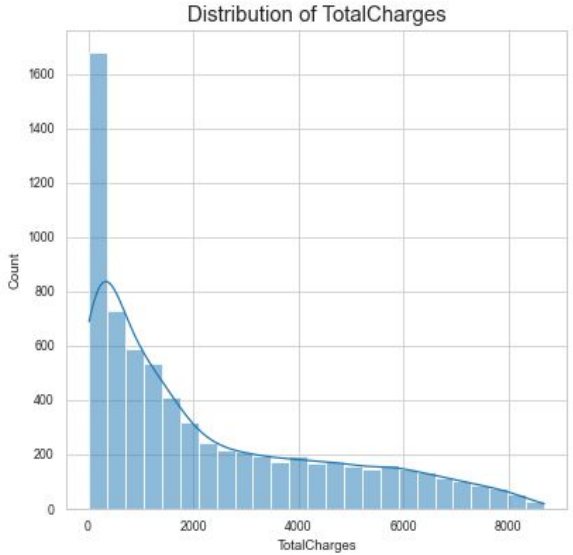
1. Data standardization using MinMaxScaler() method
2. Log Transformation on skewed variables
3. Used SMOTE method to deal with imbalanced dataset

Data Pre-processing

MinMaxScaler() method was used on four columns which were above 0-1 scale: tenure, Contract, MonthlyCharges and TotalCharges.

Then I checked my numerical variables for skewness and I found out that TotalCharges is right skewed, so I used np.log1p to deal with this problem.





Data Pre-processing

As a final step I performed SMOTE method.

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

```
counter = Counter(y_train)
print('Before sampling method', counter)
```

```
Before sampling method Counter({0: 3614, 1: 1308})
```

```
# Minority Over Sampling Technique

smt = SMOTE()
x_train_over, y_train_over = smt.fit_resample(x_train, y_train)

counter = Counter(y_train_over)
print('After SMOTE method', counter)
```

```
After SMOTE method Counter({1: 3614, 0: 3614})
```

Classification Models

Logistic Regression

Logistic Regression Model

I run Logistic Regression algorithms on both sampled data and those without SMOTE technique. In this scenario the overall accuracy for oversampling data got worse result than training data without sampling method.

The accuracy for first set - without sampling method is 80%, and for oversampling data 75%. But it doesn't meant that model with oversampling data is so bad. What is worth to notice is a fact that recall for predicting class 1 is much more better than in first case and result was improve from level of 52% to 76%.

Logistic Regression Model with Regularization

Then, I decided to tune hyperparameters both on sampled data and those without sampling. I was wondering what were the best possible parameters for those two models and how they will improve the accuracy of models.

I found that for my normal training set the best estimator were: C = 1.29 and penalty = l2. This gave my accuracy at level 81% which was slightly improvement compared to vanilla Logistic Regression.

For sampled data C = 6.0 and penalty = l2 too. Average accuracy increased to 80%.

Like it was before the recall for tuned Logistic Regression on sampled data did better job for predicting class 1 - which are customer who are going to churn. The score of recall were improve from 52% to 73%.

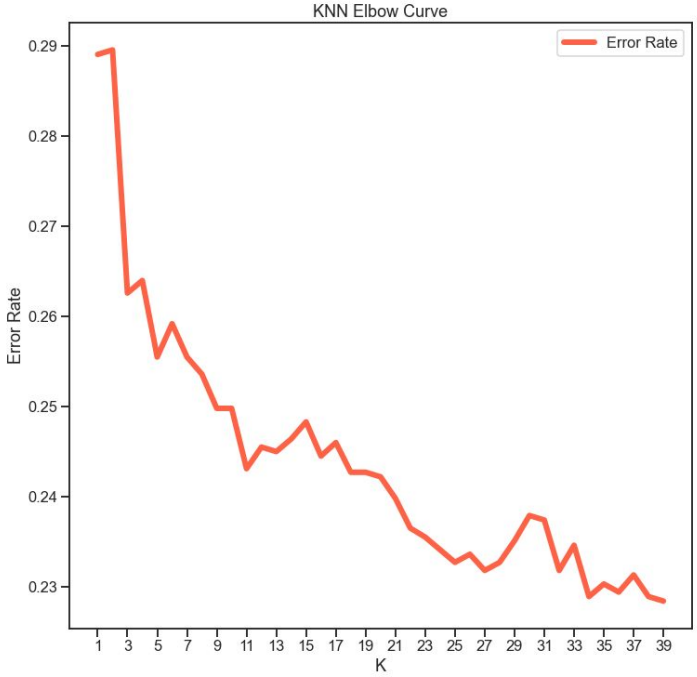
K-Nearest Neighbor

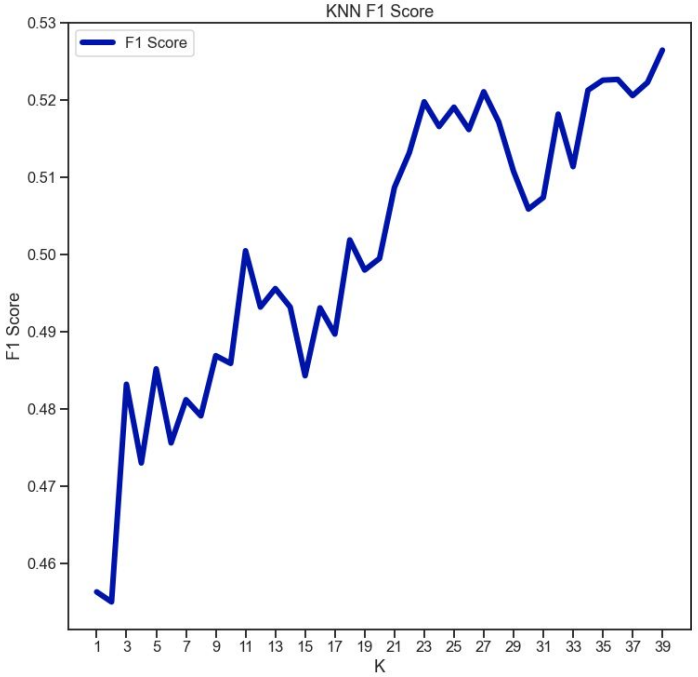
KNN Model

Like I did with Logistic Regression, in this case I also started for plain KNN model. I trained data and received average accuracy at level 75% but the model doesn't work well in predicting class 1. The recall is this case was 50%. In other hand on oversampled data the accuracy was 71% which was worse score but recall for predicting class 1 was increased to 67%.

Then, to figure out which K value is the best for model I looped through different values and based on elbow curve visualization found optimal point.

KNN Model - Elbow Curve and F1 Score

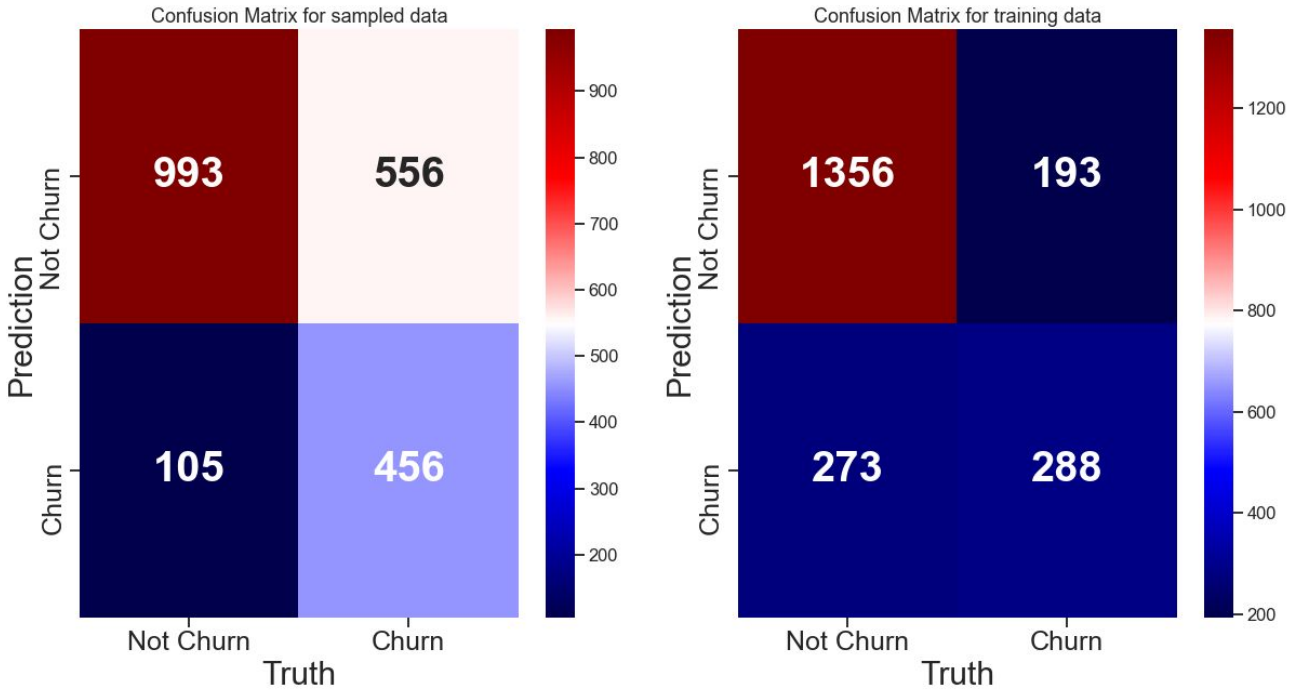




KNN Model

Having the optimal K value let me train one more time the data and compute new results for KNN models. Average accuracy thanks to elbow method which found optimal value of K = 27 were increased from 75% to 78% based on our training data. For sampled data the average accuracy slightly decreased from 71% to 69% but again I can noticed that thanks to sampled data the precision in predicting class 1 increased from 65% to 81% in this scenario.

KNN Model - Confusion Matrix



KNN Model - Confusion Matrix Interpretation

- On the left side I plotted information based on sampled data and on the right are normal training data. Just in quick remind the overall accuracy for first one was 69% and for second one was 78%.
- One important thing to notice is a fact that in our sampled confusion matrix I better predict class 1, there are 456 good predictions compared to 288 on the right confusion matrix
- Unfortunately precision (how many of my positive are actual positive class) for this class 1 predictions is not so sufficient as I wish to be, it is only 45% for left side matrix and 60% for right side matrix
- Type I error for sampled data are lower compared to right side matrix
- Type II error on the other hand is higher in our model with sampled data, this score is should be lower because it actually say that I predicted class 0 (Not Churn) when it is actual class 1 (Churn)

Support Vector Machines

SVM Model

I computed the same steps for SVM algorithms. I used rbf kernel in both cases. Classification report on the top of slide represented data without sampling and this below - sampled data.

Both models achieve similar accuracy, but as we saw before in each cases, the model with sampled data got better recall score equal to 54%.

```
print('Classification Report:')
print(classification_report(y_test, y_pred_svc))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.87      0.84      1549
           1       0.55      0.43      0.48       561

    accuracy                           0.75      2110
   macro avg       0.68      0.65      0.66      2110
weighted avg       0.74      0.75      0.74      2110
```

```
print('Classification Report:')
print(classification_report(y_test, y_pred_svc_over))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.81      0.82      1549
           1       0.51      0.54      0.52       561

    accuracy                           0.74      2110
   macro avg       0.67      0.68      0.67      2110
weighted avg       0.74      0.74      0.74      2110
```

Decision Tree

Decision Tree Model

Classification report on the top of slide represented data without sampling and this below - sampled data.

Both decision tree models achieve similar accuracy, the recall for sampled data was barely improved.

```
print('Classification Report:')
print(classification_report(y_test, y_pred_tree))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.80      0.81      1549
           1       0.49      0.51      0.50       561

    accuracy                           0.73      2110
   macro avg       0.65      0.66      0.66      2110
weighted avg       0.73      0.73      0.73      2110
```

```
print('Classification Report:')
print(classification_report(y_test, y_pred_tree_over))
```

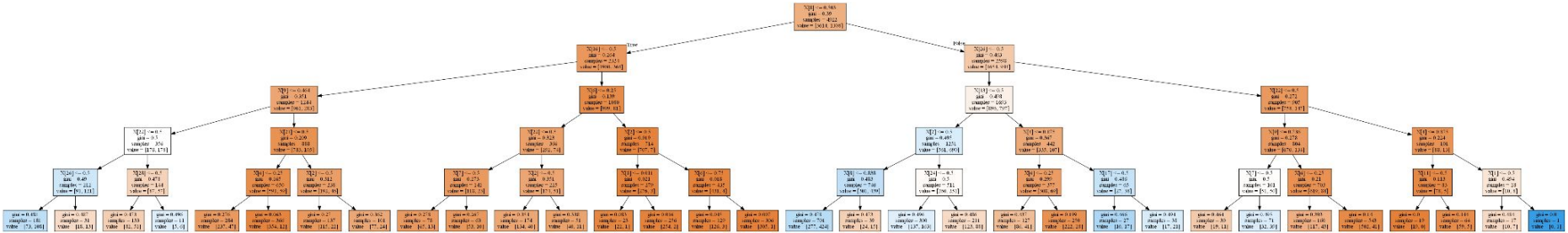
```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.79      0.82      1549
           1       0.50      0.59      0.54       561

    accuracy                           0.74      2110
   macro avg       0.67      0.69      0.68      2110
weighted avg       0.75      0.74      0.74      2110
```

Decision Tree

Having vanilla Decision Tree models for my data I decided to use **GridSearchCV** to find the best parameters for my models. I improved a little bit model for data without sampling from 73% to 75% and I used: gini as a criterion, max_depth = 5, max_features = auto. It allowed me to create a decision tree with 63 nodes.

Doing the same for oversampling data didn’t bring any important improvements, on the contrary it complicated model and created decision tree with max_depth = 18 and 3463 nodes, so I decided to do not use it.

Decision Tree after GridSearchCV on data without sampling



Key Findings - Metrics for Classification

In this section I will present findings and some basic metrics for each model I created earlier. I will concentrate on:

- **Accuracy** is the ratio of the total number of correct predictions and the total number of predictions.

- **Precision** is the ratio between the True Positives and all the Positives. Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.

- **Recall** is the measure of our model correctly identifying True Positives. Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

- **F1 Score** is the Harmonic mean of the Precision and Recall. F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

Metrics for data without sampling

Metrics among data without sampling in %

	LogisticRegression	Tuned LogisticRegression	KNN	Tuned KNN	SVM	DecisionTree	Tuned DecisionTree
accuracy	79.860000	79.810000	75.260000	77.910000	75.500000	72.610000	74.830000
precision	65.380000	65.240000	53.770000	59.880000	55.000000	48.570000	52.620000
recall	51.520000	51.520000	49.550000	51.340000	43.140000	51.340000	53.650000
f1	57.630000	57.570000	51.580000	55.280000	48.350000	49.910000	53.130000

Metrics for sampled data

Metrics among sampled data in %

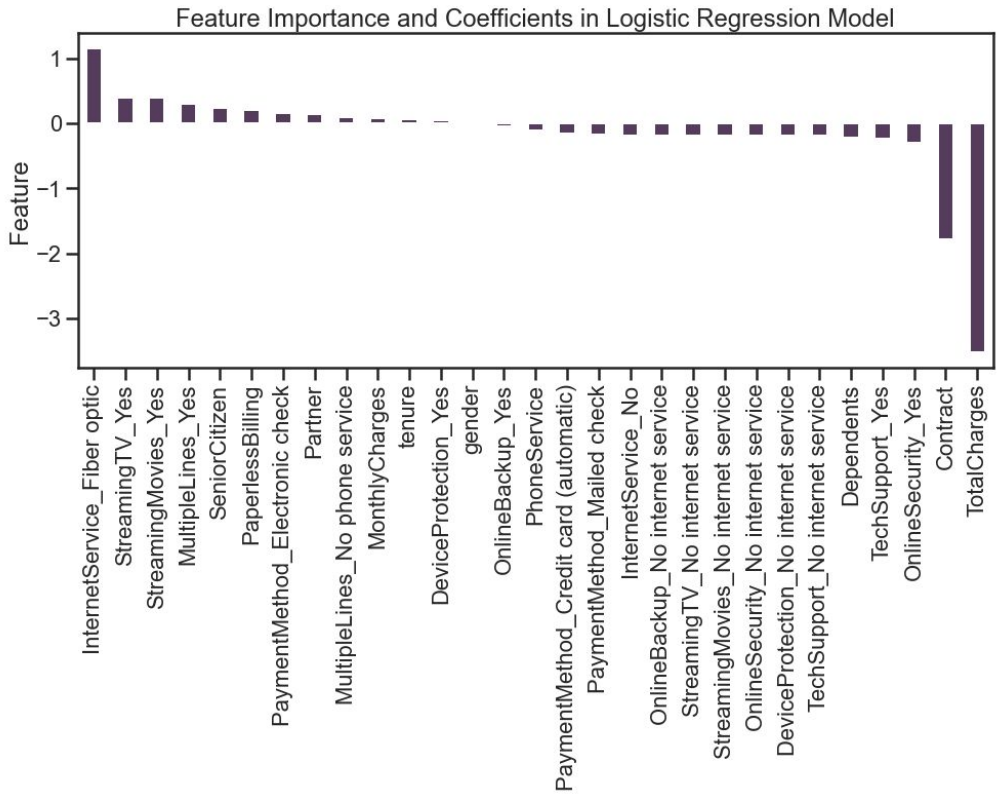
	LogisticRegression	Tuned LogisticRegression	KNN	Tuned KNN	SVM	DecisionTree
accuracy	75.070000	75.640000	70.190000	68.720000	74.080000	73.700000
precision	52.150000	53.030000	45.730000	45.150000	51.190000	50.460000
recall	75.580000	73.260000	64.880000	82.170000	53.650000	59.000000
f1	61.720000	61.530000	53.650000	58.280000	52.390000	54.400000

Key Insights

- the best model for overall average accuracy score was Logistic Regression both in sampled and data without sampling, the tuned hyperparameters for this models achieved better results than vanilla Logistic Regression
- performing SMOTE method allowed me to use data in predictions and by done this I improved recall in all cases compared to recall achieved by data without sampling
- on the other hand the precision is better used on data without sampling, in each model precision is higher than in models with sampled data, so my data without sampling predict better how many of them was actual positive
- f1 score is higher in sampled data models

Model and Metrics Evaluation

Model Evaluation - Feature Importance



In my dataset the most important features were those connected with services that customers use like: Internet services (Fiber optic), StreamingTV or StreamingMovies. In opposite side were TotalCharges and Contract Type as most important features with high coefficients magnitude.

Model and Metrics Evaluation

The models I created achieve good level of average accuracy but even good model could be further improve. Below I present some ideas which I could implement on my models to improve them:

- ★ doing my train and test split I used train_test_split method but instead this I could perform different type of splitting eg. StratifiedShuffleSplit
- ★ hyperparameters used in GridSearchCV could be further drop down to find even better models
- ★ when dealing with imbalanced target variable I used SMOTE method but in this step could be done something else eg. other variations of SMOTE like SMOTE-NC, Borderline SMOTE or ADASYN