**CUSTOMER CHURN PREDICTION**

The objective of a customer churn prediction project is to proactively identify and predict which customers are likely to stop using a product or service, also known as customer churn, and take appropriate actions to retain those customers. This is a common goal in industries such as telecommunications, subscription services, e-commerce, and more. The primary objectives of such a project include:

1. **Early Identification**: Identify customers who are at risk of churning before they actually leave, allowing for targeted interventions.

2. **Retention Strategy**: Develop and implement strategies to reduce churn rates, such as personalized offers, improved customer service, or product enhancements.

3**. Customer Segmentation**: Segment the customer base to better understand the characteristics and behaviors of different customer groups and tailor retention strategies accordingly.

4. **Predictive Modeling**: Build machine learning models that predict which customers are most likely to churn based on historical data, such as usage patterns, customer demographics, and interactions with the company.

5. **Data Analysis:** Analyze historical customer data to uncover patterns, trends, and factors that contribute to churn.

6. **Key Feature Identification**: Determine which factors have the most significant impact on customer churn and focus on addressing those issues.

7. **Performance Evaluation**: Establish key performance metrics to assess the effectiveness of the churn prediction models and retention strategies.

8. **Deployment**: Implement the churn prediction models into the company's operational systems for real-time monitoring and decision-making.

9. **Continuous Improvement**: Continuously monitor and refine the churn prediction models and retention strategies to adapt to changing customer behaviors and market conditions.

10. **Cost Reduction**: Reduce the costs associated with acquiring new customers by retaining existing ones.

11. **Revenue Growth**: Increase revenue by retaining customers who might otherwise have churned and potentially upselling or cross-selling to them.

Step 1: Pre-Requisites for Building a Churn Prediction Model

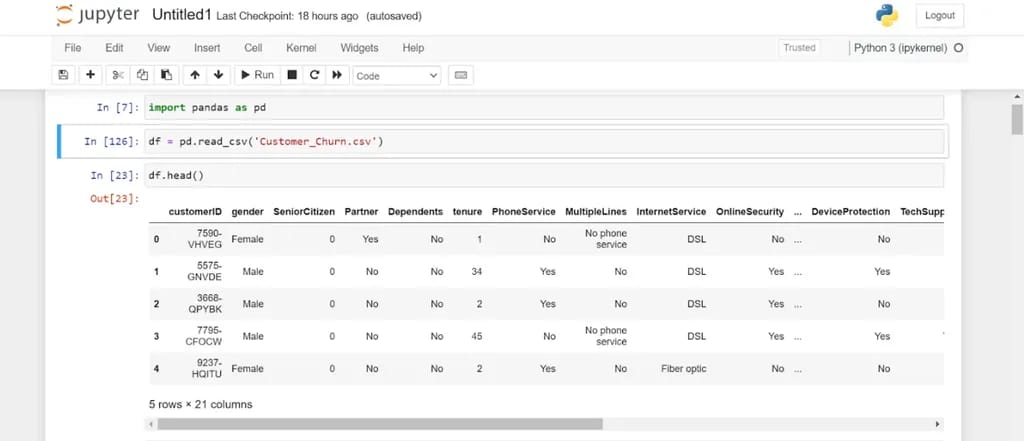
Step 2: Reviewing the Dataset

First, let’s load the dataframe into Python with the pandas library and take a look at its head. I’ve renamed the file to “customer\_churn.csv”, and it is the name I will be using below:

**import** pandas **as** pd

df = pd.read\_csv('Customer\_Churn.csv')

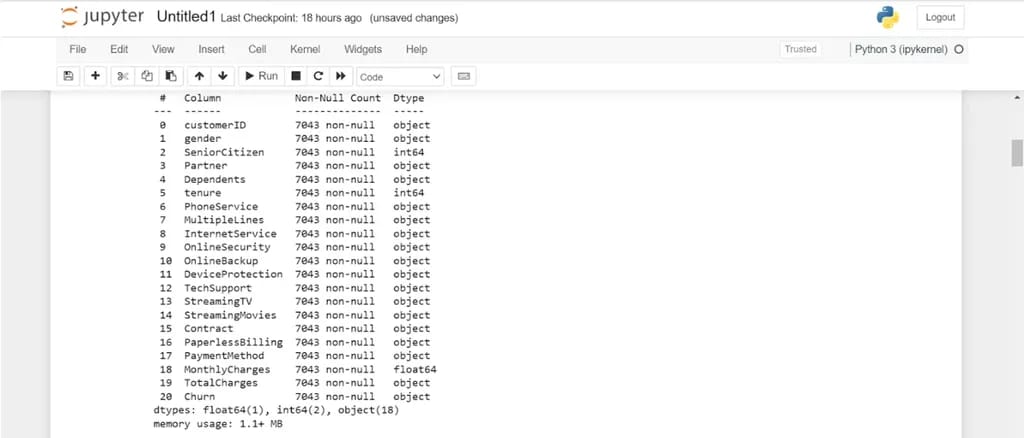
df.head()



Notice that the dataframe has 21 columns related to telecom user subscription behavior.

Let’s look into these variables further by listing them out:

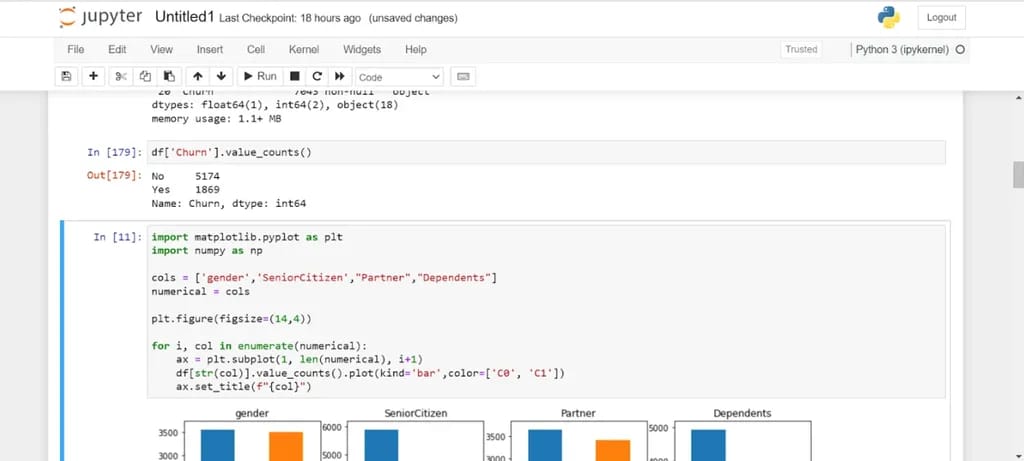
df.info()



Each user is identified through a unique customer ID. There are 19 independent variables used to predict the target feature – customer churn. In this dataset, customer churn is defined as users who have left within the last month.

Let’s count the number of customers in the dataset who have churned:

df["Churn"].value\_counts()



Only around 27% of the customers in the dataset have churned. This means that we are dealing with an [imbalanced classification problem](https://link.springer.com/chapter/10.1007/978-3-642-29958-2_3). We will need to perform some feature engineering to create a balanced training dataset before building the predictive model.

## Step 3: Exploratory Data Analysis for Customer Churn Prediction

Now, let’s perform some exploratory data analysis to gain a better understanding of the independent variables in the dataset and their relationship with customer churn.

We will start by analyzing the demographic data points:

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** numpy **as** np

cols = ['gender','SeniorCitizen',"Partner","Dependents"]

numerical = cols

plt.figure(figsize=(20,4))

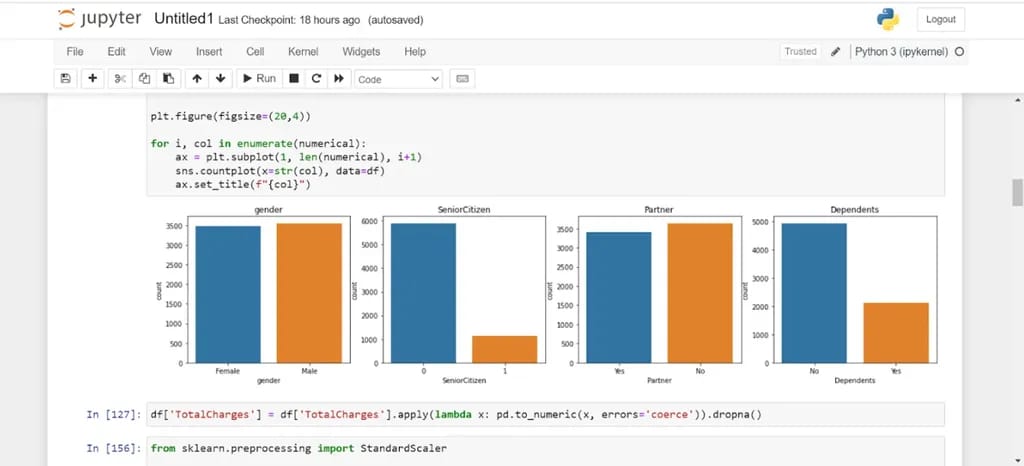
**for** i, col **in** enumerate(numerical):

ax = plt.subplot(1, len(numerical), i+1)

sns.countplot(x=str(col), data=df)

ax.set\_title(f"{col}")

The code above will render the following charts:



Most customers in the dataset are younger individuals without a dependent. There is an equal distribution of user gender and marital status.

Now, let’s look into the relationship between cost and customer churn. In the real world, users tend to unsubscribe to their mobile service provider and switch to a different brand if they find the monthly subscription cost too high. Let’s check if that behavior is reflected in our dataset:

sns.boxplot(x='Churn', y='MonthlyCharges', data=df)



The assumption above is true. Customers who churned have a higher median monthly charge than customers who renewed their subscription.

Finally, let’s analyze the relationship between customer churn and a few other categorical variables captured in the dataset:

cols = ['InternetService',"TechSupport","OnlineBackup","Contract"]

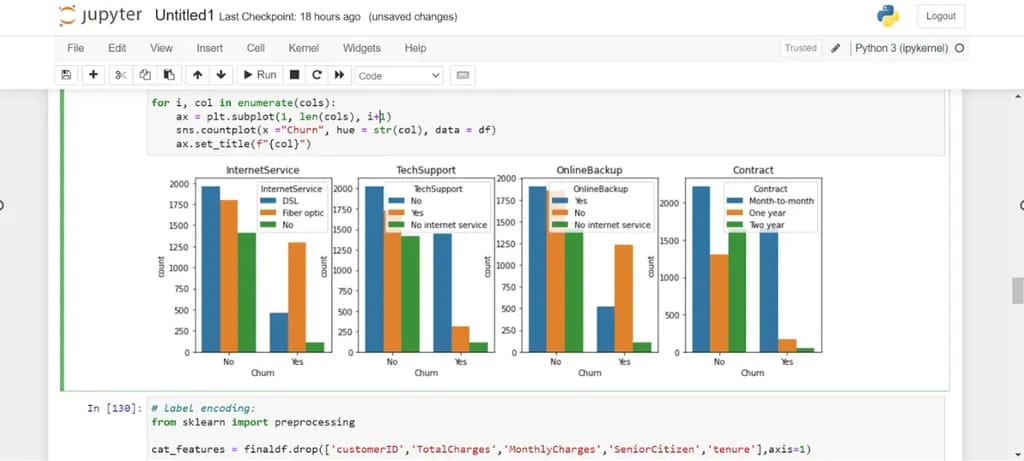
plt.figure(figsize=(14,4))

**for** i, col **in** enumerate(cols):

ax = plt.subplot(1, len(cols), i+1)

sns.countplot(x ="Churn", hue = str(col), data = df)

ax.set\_title(f"{col}")



Let’s look into each attribute:

1. **InternetService**: It is clear from the visual above that customers who use fiber optic Internet churn more often than other users. This might be because fiber Internet is a more expensive service, or this provider doesn’t have good coverage.
2. **TechSupport**: Many users who churned did not sign up for tech support. This might mean that these customers did not receive any guidance on fixing technical issues and decided to stop using the service.
3. **OnlineBackup**: Many customers who had churned did not sign up for an online backup service for data storage.
4. **Contract**: Users who churned were almost always on a monthly contract. This makes sense, since these customers pay for the service on a monthly basis and can easily cancel their subscription before the next payment cycle.

Even without building a fancy machine learning model, a simple data-driven analysis like this can help organizations understand why they are losing customers and what they can do about it.

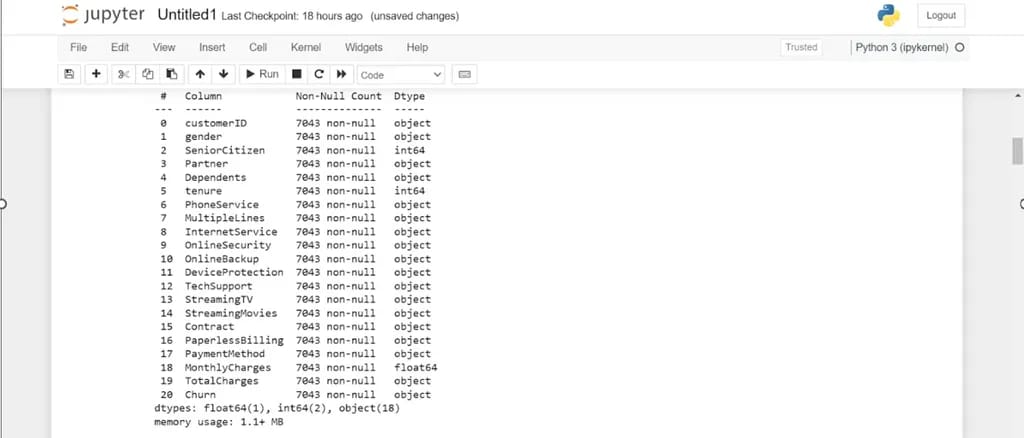
For instance, if the company realizes that most of their users who churn have not signed up for tech support, they can include this as a complimentary service in some of their future product offerings to prevent other customers from leaving.

## Step 4: Preprocessing Data for Customer Churn

Now that we have a better understanding of our dataset, let’s perform some data preparation before creating the machine learning model. There are three steps to this process:

### Cleaning the datasetd

Let’s look at the dataset summary again:



Notice that the variable “TotalCharges” has the data type “object,” when it should be a numeric column. Let’s convert this column into a numeric one:

df['TotalCharges'] = df['TotalCharges'].apply(**lambda** x: pd.to\_numeric(x, errors='coerce')).dropna()

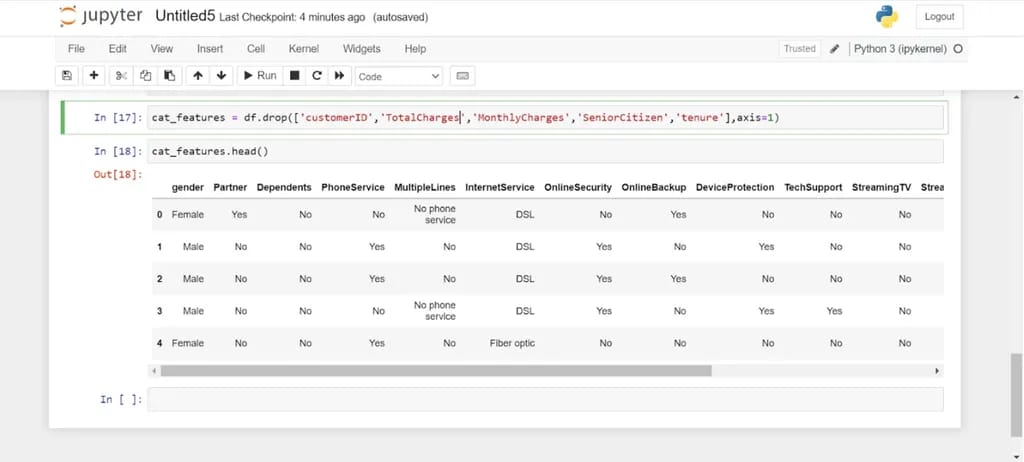
### Encoding Categorical Variables

The categorical variables in the dataset need to be converted into a numeric format before we can feed them into the machine learning model. We will perform the encoding using Scikit-Learn’s label encoder.

First, let’s take a look at the categorical features in the dataset:

cat\_features = df.drop(['customerID','TotalCharges','MonthlyCharges','SeniorCitizen','tenure'],axis=1)

cat\_features.head()



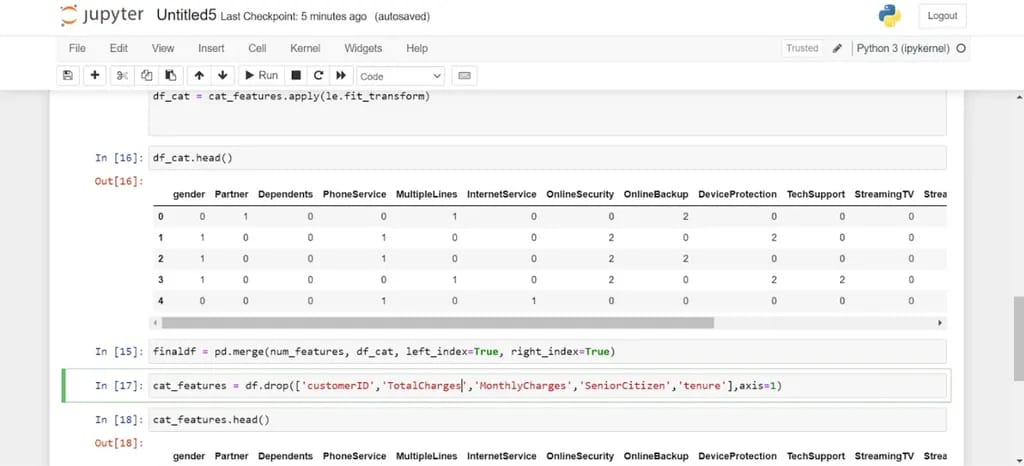
Now, let’s take a look at the dataset after encoding these categorical variables:

**from** sklearn **import** preprocessing

le = preprocessing.LabelEncoder()

df\_cat = cat\_features.apply(le.fit\_transform)

df\_cat.head()



Notice that all the categorical values in the dataset have now been replaced with numbers.

Finally, run the following lines of code to merge the dataframe we just created with the previous one:

num\_features = df[['customerID','TotalCharges','MonthlyCharges','SeniorCitizen','tenure']]

finaldf = pd.merge(num\_features, df\_cat, left\_index=True, right\_index=True)

### Oversampling

As mentioned above, the dataset is imbalanced, which means that a majority of values in the target variable belong to a single class. Most customers in the dataset did not churn - only 27% of them did.

This class imbalance problem can lead to an underperforming machine learning model. Some algorithms that train on an imbalanced dataset always end up predicting the majority class. In our case, for instance, the model may predict that none of the customers churned. While a model like this will be highly accurate (in this case it will be correct 73% of the time), it is of no value to us since it is always predicting a single outcome.

There are a variety of [techniques](https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/) that can be used to overcome the class imbalance problem in machine learning. In this tutorial, we will use a technique called oversampling. This is a process that involves randomly selecting samples from the minority class and adding it to the training dataset. We are going to oversample the minority class until the number of data points are equal to that of the majority class.

Before we oversample, let’s do a train-test split. We will oversample solely on the training dataset, as the test dataset must be representative of the true population:

**from** sklearn.model\_selection **import** train\_test\_split

finaldf = finaldf.dropna()

finaldf = finaldf.drop(['customerID'],axis=1)

X = finaldf.drop(['Churn'],axis=1)

y = finaldf['Churn']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

Now, let’s oversample the training dataset:

**from** imblearn.over\_sampling **import** SMOTE

oversample = SMOTE(k\_neighbors=5)

X\_smote, y\_smote = oversample.fit\_resample(X\_train, y\_train)

X\_train, y\_train = X\_smote, y\_smote

Let’s check the number of samples in each class to ensure that they are equal:

y\_train.value\_counts()

There should be 3,452 values in each class, which means that the training dataset is now balanced.

## Step 5: Building the Customer Churn Prediction Model

We will now build a random forest classifier to predict customer churn:

**from** sklearn.ensemble **import** RandomForestClassifier

rf = RandomForestClassifier(random\_state=46)

rf.fit(X\_train,y\_train)

## Step 6: Customer Churn Prediction Model Evaluation

Let’s evaluate the model predictions on the test dataset:

**from** sklearn.metrics **import** accuracy\_score

preds = rf.predict(X\_test)

print(accuracy\_score(preds,y\_test))

Our model is performing well, with an accuracy of approximately 0.78 on the test dataset.

--------------------------------------------