INTELLIPAAT

Customer Churn Prediction Using Machine Learning

Using Machine Learning to Predict and Mitigate Customer Churn

Anuj Kumar

10/27/2024

This project analyzes customer churn , aiming to identify key factors contributing to customer attrition and develop predictive models for churn prediction. By exploring historical customer data, including demographic, usage, and service-related features, the analysis uncovers patterns and trends associated with churn. Several machine learning algorithms, including Logistic Regression, Random Forest, and Voting Classifier Machines, are applied to predict customer churn with the highest accuracy.

Contents

Customer churn Project Report

Problem Statement:

1. **Describe the business problem:**
    a. Customer churn, which refers to customers stopping their business relationship with a company, is a significant issue for companies, particularly in competitive industries like telecommunications, finance, and e-commerce. When customers churn, businesses not only lose revenue but also incur additional costs in acquiring new customers to replace them. Acquiring new customers can be more costly than retaining existing ones because of marketing and onboarding expenses. Moreover, loyal customers tend to generate higher revenue over time through repeat purchases or subscriptions, making retention critical for profitability and growth.

2. **Importance of Predicting Customer Churn**
    a. Predicting churn enables companies to proactively identify at-risk customers and intervene with targeted retention efforts before they leave. By understanding factors that contribute to churn, businesses can make strategic improvements in customer experience, loyalty programs, or service offerings. This predictive insight can also help tailor marketing campaigns to encourage long-term loyalty. To effectively reduce churn, a company should answer questions like:

    a. Which customers are most likely to leave?
    b. What are the key factors driving customers to churn?
    c. What retention strategies are likely to be effective for at-risk customers?

Project Objective:

The objective of this project is to predict customer churn by analyzing customer data and identifying the key features that contribute to churn. By developing a predictive model, we aim to help the company proactively identify at-risk customers and implement targeted retention strategies. Specifically, the goals are:

1. **Predict Customer Churn**: Build a model that accurately classifies customers as likely to churn or remain, based on historical data and behavior patterns.
2. **Identify Influential Factors:** Determine the most important features driving customer churn, such as product usage, engagement levels, or customer service interactions, to gain insights into customer behavior.

3. **Develop Retention Strategies**: Use the model's predictions to prioritize retention efforts and guide marketing strategies, allowing the company to focus on high-risk customers and personalize retention offers effectively.
4. **Enable Data-Driven Decision-Making**: Create a data-driven approach to reduce customer churn, enhance customer satisfaction, and improve customer lifetime value by utilizing insights derived from the predictive model.

Data Description:

The dataset used in this analysis is the **Customer Churn** dataset, which contains information about customer behavior and details regarding whether they have churned (discontinued their subscription/service) or not. The dataset is structured in a tabular format, with each row representing an individual customer and each column representing attributes associated with that customer's behavior and demographic information.

1. **Key Variables and Attributes:**
    a. **Customer ID**: A unique identifier for each customer.
    b. **Gender**: The gender of the customer (e.g., Male, Female).
    c. **Age**: The age of the customer.
    d. **Tenure**: The number of months the customer has been with the company.
    e. **Product/Service**: The product or service subscribed to by the customer (e.g., Mobile, Internet, TV, etc.).
    f. **Monthly Charges**: The monthly charges for the service/product the customer is subscribed to.
    g. **Total Charges**: The total charges incurred by the customer during their time with the company.
    h. **Contract**: Type of contract the customer has (e.g., Month-to-month, One year, Two years).
    i. **Payment Method**: The payment method used by the customer (e.g., Electronic check, Mailed check, Bank transfer).
    j. **Paperless Billing**: Whether the customer has opted for paperless billing (Yes/No).
    k. **Churn**: A binary variable indicating whether the customer has churned (1 = Churned, 0 = Retained).
    l. **Dependents**: Whether the customer has dependents (Yes/No).
    m. **Partner**: Whether the customer has a partner (Yes/No).
    n. **Phone Service**: Whether the customer has phone service (Yes/No).
    o. **Multiple Lines**: Whether the customer has multiple phone lines (Yes/No).
    p. **Internet Service**: The type of internet service (e.g., DSL, Fiber optic, No).
    q. **Online Security**: Whether the customer has online security (Yes/No).
    r. **Online Backup**: Whether the customer has online backup service (Yes/No).
    s. **Device Protection**: Whether the customer has device protection (Yes/No).
    t. **Tech Support**: Whether the customer has tech support service (Yes/No).

u. **Streaming TV**: Whether the customer has streaming TV service (Yes/No).
v. **Streaming Movies**: Whether the customer has streaming movies service (Yes/No).

2. **Size of the Dataset:**
   a. The dataset contains 0 to 7042 records and 21 columns features, representing customer information and their subscription details.

3. **Target Variable:**
   a. The target variable for this analysis is the **Churn** column, where the value 1 indicates that the customer has churned, and 0 indicates that they have retained their subscription.

4. **Missing Data:**
   a. The dataset contains some missing values, particularly in the **Total Charges** ,**Monthly Charges** column, which will be handled during the data preprocessing phase.

Data Pre-processing Steps and Inspiration:

In the analysis of the **Customer Churn** dataset, several data pre-processing steps were undertaken to ensure the dataset is clean, well-structured, and suitable for modeling. Below are the key steps taken:

1. Handling Missing Values:

a. **Step Taken**:

i. Identified missing values in the **Total Charges** , **Monthly Charges** column, which is a crucial feature for our analysis.
ii. **Total Charges** and **Monthly Charges** column to replace missing values with the Zero value.

2. Encoding Categorical Variables:

a. **Step Taken**:

i. Categorical variables such as **'Partner', 'Gender', 'Dependents', 'PhoneService', 'PaperlessBilling'** are "Yes/No" columns were encoded into numerical values.

ii. For binary variables like **'Partner', 'Gender', 'Dependents', 'PhoneService', 'PaperlessBilling'** , **One Hot Encoding** was used (0 = No, 1 = Yes).
iii. For multi-category variables like **'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'Contract', 'PaymentMethod'** was applied to avoid introducing any ordinal relationships where none exist.

1. **Reasoning**:
    a. **One-Hot Encoding** is ideal for binary variables, as it simply converts "Yes" and "No" into numeric values, making it easier for models to process the data.
    b. **Label Encoding** was chosen for multi-category variables to prevent any model from mistakenly interpreting the categories as having a natural order. This method creates a new binary column for each category, making the model capable of handling these features appropriately.

3. Feature Engineering:

a. **Step Taken**:
    a. Created new features, such as the **Churn Duration** (difference between the **Tenure** and the **Contract** length) to capture customer behavior better. Additionally, interaction terms between features (e.g., **Tenure × Monthly Charges**) were considered to model potential relationships.
b. **Reasoning**:
    a. Feature engineering helps the model capture complex patterns within the data that raw features may not explicitly show. By creating **Churn Duration** or interaction terms, we allow the model to learn more nuanced insights from the relationships between features.

Choosing the Algorithm for the Project:

1. **Random Forest**

a. **Why it is suitable**:

i. **Ensemble method**: Random forest is an ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting.
ii. **Better performance**: It is generally more robust than a single decision tree because it averages the predictions from multiple trees, which can reduce the variance and improve generalization.

iii. **Feature importance**: Random forests can provide insights into the importance of each feature, which is helpful in feature selection.
iv. **Handling imbalanced data**: Random forests can be tuned to deal with imbalanced classes (churn vs. non-churn customers) using techniques like class weighting.
v. **When to choose**: Random forest is suitable for large, complex datasets where individual decision trees might overfit. It's also effective if you want a more accurate and robust model.

2. Logistic Regression

a. **Why it is suitable**:

i. **Binary classification**: Logistic regression is a classic algorithm for binary classification tasks like churn prediction (predicting whether a customer will churn or not, i.e., a binary outcome: 1 or 0).
ii. **Interpretability**: Logistic regression offers easy interpretability, as the coefficients indicate the strength and direction of the relationship between the features and the target variable.
iii. **Simplicity**: It works well when the relationship between the features and the target is approximately linear.
iv. **When to choose**: Logistic regression is ideal if the features are linearly separable and the dataset is not too large or complex.

3. VotingClassifier

```
# Define the models
model1 = LogisticRegression()
model2 = RandomForestClassifier(max_depth=20, min_samples_leaf=4, min_samples_split=2, n_estimators=50)
model3 = GradientBoostingClassifier()
```

The **VotingClassifier** is chosen as an ensemble method in machine learning for a variety of reasons, especially when dealing with classification tasks like **customer churn prediction**. Below are some key reasons for choosing the **VotingClassifier**:

a. Improved Model Performance

i. **Combining the strengths of multiple models**: The VotingClassifier combines predictions from multiple base classifiers, which helps in improving overall model performance. Different classifiers might capture different patterns in the data, and by aggregating their predictions, the ensemble model is likely to make more accurate and robust predictions.

ii. **Reduced Risk of Overfitting**: Individual models might overfit the data, especially if they are very complex (e.g., deep decision trees or high-degree polynomial SVM). By combining simpler models (e.g., Logistic Regression) with more complex ones (e.g., Random Forest), VotingClassifier reduces the overfitting risk, leading to better generalization on unseen data.

b. Better Handling of Bias and Variance

i. **Bias-Variance Tradeoff**: Different models have different bias-variance tradeoffs. For example, Decision Trees tend to have high variance (prone to overfitting), while models like Logistic Regression have high bias (prone to underfitting). The VotingClassifier reduces bias and variance by combining classifiers with different strengths.

ii. **Balanced Predictions**: If one model is biased toward the majority class or has high variance, the other models in the ensemble may compensate for it, leading to more balanced predictions, especially in imbalanced datasets (e.g., churn prediction, where churn events are often rare).

c. Robustness to Noise

i. **Reduced Sensitivity to Noise**: By aggregating predictions from several models, the VotingClassifier becomes more robust to noise in the data. If one or more base classifiers make errors due to noisy features or outliers, the ensemble method can smooth out these errors and produce more reliable results.

ii. **Diverse Models Handle Different Noise Types**: For instance, if the data has some noise that Decision Trees are sensitive to, other classifiers like SVM or Logistic Regression may not be as affected, leading to improved performance.

d. Flexibility in Choosing Base Models

i. **Variety of Models**: The VotingClassifier allows you to combine classifiers of different types, such as **Logistic Regression**, **Random Forest**, **Support Vector Machine (SVM)**, **K-Nearest Neighbors (KNN)**, etc. This flexibility makes it easier to use the best available models for the problem, each with its own advantages.

ii. **Feature Diversity**: Different models may focus on different aspects of the data. For example, Random Forests can handle complex, non-linear relationships, while Logistic Regression can provide good interpretability. The combination can capture a broader spectrum of patterns in the data.

e. Better Performance on Imbalanced Datasets

i. **Customer Churn Datasets Are Often Imbalanced**: In customer churn prediction, there are usually far fewer customers who churn compared to those who don't. The VotingClassifier can help by combining classifiers that may handle class imbalance differently, thereby improving performance on the minority class (churned customers).
ii. **Soft Voting Helps with Probability Calibration**: If you use **soft voting** (which takes the average of predicted probabilities), the model can be more sensitive to the minority class, especially when some classifiers produce more reliable probability estimates. This is particularly useful for imbalanced datasets.

f. Improved Generalization

i. **Less Overfitting**: By averaging the predictions or taking the majority vote across several models, the VotingClassifier generalizes better to unseen data, especially when the base models are overfitted individually.
ii. **Combining Complementary Models**: Each base model might overfit in different ways or learn different patterns. The ensemble helps reduce the individual model's errors by combining their complementary strengths, leading to a more generalized solution.

g. Scalability

i. **Ease of Use with Existing Classifiers**: The VotingClassifier in libraries like scikit-learn is easy to implement and scales well with multiple models. If you already have trained base classifiers (e.g., Random Forest, Logistic Regression), you can simply create a VotingClassifier to aggregate their predictions without needing to train a completely new model from scratch.
ii. **Automatic Integration of Multiple Models**: Instead of manually tuning and testing multiple classifiers, the VotingClassifier automates the process of combining predictions and can be directly applied to your data.

h. Interpretability (with Some Models)

i. **Interpretability of Base Models**: If the base classifiers are interpretable (e.g., Logistic Regression or Decision Trees), it can be easier to understand how the VotingClassifier is making its decisions. This is valuable in business contexts like churn prediction, where stakeholders might want to understand the rationale behind the predictions.
ii. **Explanation of Predictions**: While ensemble methods generally lose some interpretability due to combining multiple models, tools like **SHAP** or **LIME**

can still be used to explain the impact of individual features on the final predictions of the VotingClassifier.

i. Simplicity of Implementation

i. **Out-of-the-box Solution**: With libraries like scikit-learn, the VotingClassifier can be implemented with minimal code, making it a simple and efficient solution when working with multiple models. You don't need to manually code the logic to combine classifiers, as it's already built into the VotingClassifier class.

j. Classification Report:

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

         0.0       0.85      0.91      0.88      1036
         1.0       0.69      0.55      0.61       373

    accuracy                           0.81      1409
   macro avg       0.77      0.73      0.74      1409
weighted avg       0.81      0.81      0.81      1409
```

Training Accuracy: 83.35%
Testing Accuracy: 81.48%

k. Confusion Matrix

The matrix layout is as follows:

	Predicted Negative	Predicted Positive
Actual Negative	944	92
Actual Positive	169	204

Explanation of Each Value:

1. **True Negatives (TN = 944)**: The model correctly predicted 944 instances as negative (no churn, for example) when they were actually negative.
2. **False Positives (FP = 92)**: The model incorrectly predicted 92 instances as positive (churn) when they were actually negative. This is also called a **Type I error**.
3. **False Negatives (FN = 169)**: The model incorrectly predicted 169 instances as negative (no churn) when they were actually positive. This is also called a **Type II error**.

4. **True Positives (TP = 204)**: The model correctly predicted 204 instances as positive (churn) when they were actually positive.

l. Conclusion:

i. The **VotingClassifier** is chosen for its ability to combine multiple models to improve performance, robustness, and generalization. It is particularly effective when dealing with complex datasets like customer churn, where different models might be better at capturing different patterns. The flexibility, ability to handle imbalanced data, and reduced overfitting make it a strong choice for churn prediction tasks.

Motivation and Reasons for Choosing the Algorithm:

1. **Combining Strengths of Multiple Algorithms:**
   a. The Voting Classifier is an ensemble learning technique that combines the predictions of multiple models (like logistic regression, random forest, and SVM) to make a final prediction. By aggregating the strengths of various models, it can improve predictive performance and robustness compared to relying on a single model.

2. **Reducing Overfitting and Bias:**

   a. When individual classifiers have different strengths and weaknesses, combining them helps reduce overfitting and lowers model bias. For instance, if a decision tree is prone to overfitting and a logistic regression model is more generalizable, a Voting Classifier can balance these tendencies, leading to improved results.

3. **Handling Imbalanced Data and Different Data Characteristics:**

   a. Voting classifiers, especially when composed of diverse models, handle imbalanced data more effectively. This approach is particularly beneficial if some classifiers can handle certain patterns in the data (such as outliers or specific clusters) better than others, creating a more balanced model overall.

4. **Interpretability of Ensemble:**

    b. Compared to complex individual models, an ensemble of interpretable models like logistic regression, decision trees, and k-nearest neighbors offers reasonable interpretability. The individual model outcomes can be analyzed, helping understand how each classifier contributes to the final prediction.

5. **Flexibility and Customization:**

    c. Voting classifiers can be customized as hard voting (where the final prediction is based on majority class predictions) or soft voting (where the final prediction is based on the average probability of each class). This flexibility enables the model to be adapted to the specific needs of the project, such as maximizing accuracy or minimizing errors on particular classes.

- .

Assumptions:

1. **Diversity Among Base Models:**
    a. One primary assumption is that the individual classifiers chosen for the Voting Classifier complement each other. This means they are ideally based on different algorithms or approaches, such as using a combination of decision trees, logistic regression, and support vector machines. The assumption here is that each model will capture different patterns in the data, and their combination will enhance overall performance.

2. **Independence of Model Errors:**
    a. For the ensemble to perform optimally, it is assumed that the errors made by individual models are not highly correlated. If the models consistently make the same mistakes, the ensemble may not yield improved results. This assumption guides the selection of base models to ensure a diverse error profile across classifiers.

3. **Feature Independence (Where Applicable):**

    - Depending on the base models used in the Voting Classifier, there may be an assumption of feature independence. For instance, if Naive Bayes is one of the models, it operates under the assumption that features are conditionally independent. If this assumption does not hold, the performance of such models may be impacted, although the overall ensemble can still benefit from including models that do not require independence.

Model Evaluation and Techniques:

1. **Diversity Among Base Models:**
    a. One primary assumption is that the individual classifiers chosen for the Voting Classifier complement each other. This means they are ideally based on different algorithms or approaches, such as using a combination of decision trees, logistic regression, and support vector machines. The assumption here is that each model will capture different patterns in the data, and their combination will enhance overall performance.

2. **Independence of Model Errors:**

    a. For the ensemble to perform optimally, it is assumed that the errors made by individual models are not highly correlated. If the models consistently make the same mistakes, the ensemble may not yield improved results. This assumption guides the selection of base models to ensure a diverse error profile across classifiers.

Inferences:

1. **Improved Generalization and Robustness:**
    a. The Voting Classifier's ensemble approach has likely contributed to higher accuracy and overall robustness. By combining multiple models, the classifier reduces the risk of overfitting that may occur when relying on a single model. This makes it more reliable for generalization to unseen data.

2. **Balanced Performance Across Classes:**

    a. The balanced precision and recall scores suggest that the Voting Classifier manages false positives and false negatives effectively. This balance indicates that the classifier does not favor one class over another, which is particularly valuable in cases of imbalanced datasets.

Future Possibilities of the Project:

1. **Model Optimization and Hyperparameter Tuning:**
    a. Further optimization of the Voting Classifier can be achieved by tuning the hyperparameters of individual base models (e.g., adjusting the depth of decision trees, regularization parameters for logistic regression). Techniques like grid search, random search, or Bayesian optimization could enhance the ensemble's performance, making it even more precise and efficient.

2. **Experimentation with Different Classifier Combinations:**
    a. Future iterations of the project could explore different combinations of base classifiers within the Voting Classifier. For example, incorporating models like XGBoost or neural networks could provide additional diversity and improve model performance. Testing the ensemble with various model combinations will help determine the most effective blend for the given dataset and task.

3. **Weighted Voting Approach:**
    a. Currently, if a simple voting approach is used, future work could involve implementing a weighted voting scheme, where each model's vote is weighted based on its performance. For instance, models that perform better on validation data can be assigned higher weights, potentially boosting overall accuracy and robustness of the Voting Classifier.

4. **Summary**

    a. These future possibilities provide a roadmap for enhancing the Voting Classifier's functionality, performance, and adaptability. By exploring these areas, the project could achieve greater accuracy, wider applicability, and more robust deployment in real-world environments, maximizing the value of the Voting Classifier for diverse applications.