VELS

INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)
PALLAVARAM - CHENNAI
ACCREDITED BY NAAC WITH 'A' GRADE
Marching Beyond 25 Years Successfully

CUSTOMER CHURN ANALYSIS

A main project report
Submitted for the partial fulfillment for the award of the degree of

MASTER OF SCIENCE

IN

DATA SCIENCE AND BUSINESS ANALYTICS

NAME: GOWTHAM K

REGISTRATION NO.: 22235136

Under the guidance of

Dr. S. Sathya, M.Sc., M.Phil., Ph.D.,

Assistant Professor,

Department of Information Technology

School of Computing Sciences

VISTAS

VELS

INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)
PALLAVARAM - CHENNAI
ACCREDITED BY NAAC WITH 'A' GRADE
Marching Beyond 25 Years Successfully

CERTIFICATE

This is to certify that the project titled "**CUSTOMER CHURN ANALYSIS"** is the original project work done by **GOWTHAM K, (REGISTER NO.: 22235136),** under my guidance and supervision for the partial fulfillment for the award of the degree in Master of Science in data science and business analytics (M.Sc. DS&BA), as per the syllabus prescribed by the VELS INSTITUTE OF SCIENCE TECHNOLOGY & ADVANCED STUDIES (VISTAS).

GUIDE HEAD OF THE DEPARTMENT

Submitted for the Viva-Voice examination held on _____ at VELS INSTITUTE OF SCIENCE TECHNOLOGY & ADVANCED STUDIES (VISTAS).

Place: Chennai EXAMINERS

Date: 1.

VELS
INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)
PALLAVARAM - CHENNAI
ACCREDITED BY NAAC WITH 'A' GRADE
Marching Beyond 25 Years Successfully

DECLARATION

I, **GOWTHAM K (REGISTER NO.: 22235136)**, declares that the project entitled "**CUSTOMER CHURN ANALYSIS**" submitted by me during the period from **July, 2024 – Nov, 2024** under the guidance of **Dr. S. Sathya, Assistant Professor, Department of Information Technology, VIATAS** and has not formed the basis for the award of any degree diploma, associate-ship, fellowship, titles in this or any other University or other similar institutions of higher learning.

Place: Chennai Signature of the Student

Date:

VELS

INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)
PALLAVARAM - CHENNAI
ACCREDITED BY NAAC WITH 'A' GRADE
Marching Beyond 25 Years Successfully

BONAFIDE CERTIFICATE

I certify that the project titled "**CUSTOMER CHURN ANALYSIS"** for the Master of Science in data science and business analytics (M.Sc. DS&BA) done by **GOWTHAM K (REGISTER NO.: 22235136)** is the project work carried out by him during the period **2024** under my guidance and supervision and that this work has not formed the basis for the award of any degree, diploma, associate-ship, fellowship, titles in this or any other University or other similar institutions of higher learning. He fulfills the eligibility criteria for submission of this project as per rules and regulations of the University.

Place: Chennai

Date: Guide

ACKNOWLEDGEMENT

I thank our **Honorable Founder & Chancellor Dr Ishari K. Ganesh, Honorable Vice-Chancellor, Dr. S. Sriman Narayanan** and **Honorable Vice President Dr. Preethaa Ganesh** VISTAS, for allowing me to pursue my M.Sc. Programme.

I thank our **Registrar, Dr. P. Saravanan** and VISTAS for their support.

I thank our **The Controller of Examinations**, **Dr. A. Udhayakumar,** VISTAS for their support**.**

I sincerely **thank Dr. P. Sujatha, Professor, Associate Professor** and **Head of the Department of Information Technology, VIATAS** for her continued support and motivation.

I profoundly thank my research supervisor **Dr. S. Sathya, Assistant Professor, Department of Information Technology, VIATAS,** for her total encouragement, inspiration and advice from the beginning till the end of this project / dissertation.

I am grateful to the faculties of the department of Information Technology, VISTAS for their assistance for the successful completion of this dissertation.

I thank VISTAS librarians for providing me access to the useful textual materials available in VISTAS which supported my research genre.

I am grateful to my friends and family for rendering me their recommendations and moral support.

Name of the student / scholar

GOWTHAM K

ABSTRACT

PowerCo, a significant player in the gas and electricity utility sector. It outlines the challenges faced by PowerCo in retaining customers, particularly in the SME segment, due to the dynamics of the energy market in Europe, marked by liberalization and evolving consumer preferences.

To address these challenges, the project conducts a comprehensive analysis of various factors influencing customer churn, such as service quality, pricing structures, customer engagement, and technological advancements. It aims to formulate strategic interventions based on industry best practices, case studies, and data-driven insights.

The dataset provided, comprising client_data and price_data, offers valuable information on various aspects relevant to PowerCo's customer base and pricing dynamics. The client_data includes details like customer activity, consumption patterns, contract dates, and churn status, while price_data provides pricing information over time.

The analysis involves data preprocessing steps, such as handling missing values, converting data types, and encoding categorical variables for modeling purposes. Exploratory data analysis (EDA) and visualization are likely next steps to gain insights into the data distribution and relationships among variables.

Feature Importance: Visualize the importance of different features in predicting churn using a bar chart. This helps identify which factors have the most significant impact on churn prediction.

Customer Demographics: Use bar charts to visualize customer demographics such as age groups, geographic location, or subscription type. This provides insights into the characteristics of churned vs. retained customers.

Product Usage: Plot bar charts to show the distribution of product usage metrics such as energy consumption levels or frequency of service usage among churned and retained customers.

Distribution of Numeric Variables: Create histograms to visualize the distribution of numeric variables such as energy consumption, pricing data, or customer tenure. This helps identify patterns or outliers that may correlate with churn behavior.

Time Series Analysis: Use histograms to analyze trends over time, such as changes in consumption patterns or pricing fluctuations. This can reveal seasonal variations or anomalies that may impact churn.

Churn Rate by Segment: Create stacked bar charts to compare churn rates across different customer segments, such as by subscription type, geographic region, or customer tenure. This highlights which segments have the highest churn rates.

Effectiveness of Interventions: Visualize the impact of interventions, such as offering discounts or promotional campaigns, on churn rates over time. Stacked bar charts can show how churn rates change before and after implementing these interventions.

To analyze customer churn behavior and evaluate the effectiveness of offering discounts in reducing churn and increasing profit. The dataset comprised customer information, consumption data, and pricing details. Key findings suggested that customers were generally insensitive to price changes, with a significant portion switching providers. Various features, including consumption metrics and price differentials, were incorporated into a RandomForest model for predicting churn.

The model yielded insights into customer behavior and enabled the assessment of discount strategies. Despite the initial conclusion that price changes did not impact churn significantly, a novel approach was employed to evaluate the potential impact of discounts on profit.

By calculating the expected profit before and after offering discounts to predicted churning customers, it was observed that a discount could lead to increased profit for a subset of customers.

Further analysis revealed that among predicted churning customers, offering discounts resulted in a net increase in expected profit. Notably, while a majority of customers experienced a profit increase, a minority experienced a decrease, highlighting the complexity of discount strategies. Overall, the findings suggest that targeted discount offers could be a viable strategy to mitigate churn and enhance profitability in the energy sector

Table of Contents

CHAPTER 1…….4
1 INTRODUCTION…….5

CHAPTER 2……...12
2 SYSTEM SPECIFICATION……13

CHAPTER 3……17
3 MODEL ANALYSIS……18

CHAPTER 4……24
4 MODEL DESIGN……25

CHAPTER 5……37
5.1 Feature Engineering ……..……38

SOURCE CODE……51

CHAPTER 1

1 INTRODUCTION

1.1 OBJECTIVES

The primary objective of this project is to enhance customer satisfaction and reduce churn rates by implementing data-driven strategies and predictive modeling techniques. By leveraging advanced analytics on historical customer data, the project aims to identify key factors influencing customer behavior and churn propensity. Through this analysis, we seek to gain deeper insights into customer preferences, satisfaction levels, and pain points, enabling us to tailor products, services, and marketing initiatives to better meet their needs. Additionally, the project aims to develop predictive models capable of accurately forecasting customer churn, allowing the organization to proactively intervene and implement retention strategies.

Another key objective of the project is to optimize operational efficiency and resource allocation by streamlining processes and workflows. By analyzing customer data and behavior patterns, we aim to identify opportunities for automation, personalization, and targeted interventions. This includes automating routine tasks, such as customer communication and billing processes, as well as implementing personalized marketing campaigns and loyalty programs based on individual customer profiles. Furthermore, the project seeks to optimize resource allocation by reallocating resources to areas with the highest impact on customer satisfaction and retention. Through these efforts, we aim to improve operational efficiency, reduce costs, and ultimately enhance the organization's bottom line.

Significance of the Project

Business Impact: Highlight how achieving the project objectives will drive tangible business outcomes such as increased revenue, reduced churn, or improved customer satisfaction.

Strategic Alignment: Explain how the project objectives support the organization's mission, vision, and long-term strategic objectives.

Market Dynamics: Discuss how addressing the identified objectives will help the organization stay competitive in a rapidly evolving market landscape.

Stakeholder Engagement: Emphasize the importance of stakeholder buy-in and collaboration in achieving the project objectives.

User Satisfaction Assessment

Customer-Centric Approach: Highlight the shift towards a customer-centric approach in today's business landscape and the importance of continuously monitoring and improving customer satisfaction.

Feedback Mechanisms: Discuss the various methods used to assess user satisfaction, such as surveys, feedback forms, or customer support interactions.

Data Analysis: Explain how data collected from user satisfaction assessments is analyzed to identify trends, pain points, and areas for improvement.

Continuous Improvement: Emphasize the iterative nature of quality improvement efforts, where feedback is used to make incremental changes and enhancements to products or services.

Competitive Positioning

Market Analysis: Provide an overview of the competitive landscape, including key competitors, market trends, and customer preferences.

Points of Differentiation: Highlight how achieving the project objectives will help the organization differentiate itself from competitors by offering unique value propositions or superior customer experiences.

Market Share Growth: Discuss how successful implementation of the project objectives can lead to increased market share and a stronger foothold in the industry.

Future Outlook: Offer insights into how the organization's competitive positioning may evolve as a result of achieving the project objectives, considering factors such as emerging technologies, regulatory changes, or shifting consumer behaviors.

1.2 CHALLENGES IN THE DOMAIN

The domain of customer churn prediction and satisfaction enhancement presents several significant challenges, ranging from data quality issues to the complexities of human behavior and market dynamics. One of the primary challenges is the availability and quality of data. Customer data is often fragmented across multiple systems and channels, making it challenging to consolidate and analyze effectively. Moreover, data may be incomplete, inaccurate, or outdated, leading to biased or unreliable insights. Addressing these data quality issues requires robust data integration and cleansing processes, as well as advanced techniques such as data imputation and outlier detection.

Another challenge in this domain is the complexity of customer behavior and preferences. Customers exhibit diverse behaviors influenced by various factors such as demographics, psychographics, and past interactions with the brand. Understanding and predicting these behaviors require sophisticated analytical models capable of capturing nonlinear relationships and interactions among variables. Furthermore, customer preferences may evolve over time, making it challenging to maintain accurate models and predictions. Addressing these challenges requires continuous monitoring of customer behavior and preferences, as well as the use of advanced machine learning algorithms capable of adapting to changing patterns.

Additionally, market dynamics and competitive pressures pose significant challenges in customer churn prediction and satisfaction enhancement. Markets are dynamic and subject to various external factors such as economic conditions, regulatory changes, and competitive actions. Moreover, customers have access to a wide range of alternatives and are increasingly empowered to switch providers based on factors such as price, convenience, and service quality. Competing effectively in such a dynamic environment requires organizations to stay agile and responsive, continuously monitoring market trends and customer feedback to adapt their strategies and offerings accordingly.

Furthermore, ethical considerations and privacy concerns represent significant challenges in this domain. Analyzing customer data to predict churn and enhance satisfaction raises important ethical questions regarding data privacy, consent, and transparency. Customers expect organizations to handle their data responsibly and respect their privacy rights. Addressing these challenges requires organizations to implement robust data governance frameworks, ensure compliance with regulatory requirements, and establish transparent communication practices with customers regarding data usage and privacy policies.

1.3 The significance of a gas and electricity

Supporting Business Operations: Gas and electricity are foundational utilities for businesses, enabling them to power various operations, machinery, and equipment. For corporate and SME customers, uninterrupted access to reliable energy is crucial for maintaining productivity and efficiency. Without these utilities, businesses may face disruptions in their operations, leading to financial losses and decreased competitiveness. Therefore, a gas and electricity utility plays a vital role in supporting the day-to-day functioning of businesses across various sectors.

Driving Economic Growth: Corporate and SME customers are key drivers of economic growth and employment generation in any economy. These businesses contribute to job creation, innovation, and overall economic development. By providing reliable gas and electricity services to corporate and SME customers, a utility facilitates their growth and expansion, thereby contributing to the overall economic prosperity of the region. Access to affordable and sustainable energy enables businesses to scale up their operations, invest in new technologies, and explore market opportunities, ultimately fueling economic growth and development.

Enhancing Energy Efficiency: Gas and electricity utilities play a crucial role in promoting energy efficiency among corporate and SME customers. By offering energy audits, efficiency incentives, and tailored solutions, utilities help businesses optimize their energy consumption, reduce waste, and lower operational costs. Energy-efficient practices not only benefit individual businesses by improving their bottom line but also contribute to environmental sustainability by reducing carbon emissions and resource depletion. Therefore, gas and electricity utilities serve as catalysts for promoting a culture of energy conservation and sustainability within the business community.

Facilitating Innovation and Resilience: In today's rapidly evolving business landscape, innovation and resilience are essential for staying competitive and adaptable to changing market dynamics. Gas and electricity utilities can partner with corporate and SME customers to drive innovation in energy technologies, processes, and business models. By fostering collaboration and providing access to renewable energy sources, smart grids, and advanced energy management systems, utilities empower businesses to innovate, improve their resilience to disruptions, and future-proof their operations against uncertainties such as climate change and supply chain disruptions.

Ensuring Social Responsibility: Gas and electricity utilities have a responsibility to serve the broader interests of society, including corporate and SME customers. This entails ensuring equitable access to energy services, promoting social inclusivity, and addressing the needs of vulnerable communities. Utilities can engage in corporate social responsibility initiatives, such as energy assistance programs, community outreach, and workforce development initiatives, to support the well-being and prosperity of their customers and communities. By fulfilling their social responsibility mandates, utilities enhance their reputation, build trust with customers, and contribute to the overall social welfare.

In summary, the significance of a gas and electricity utility that serves corporate and SME customers extends beyond the provision of essential services. It encompasses supporting business operations, driving economic growth, enhancing energy efficiency, fostering innovation and resilience, and fulfilling social responsibility obligations. By effectively meeting the diverse needs of corporate and SME customers, utilities play a pivotal role in advancing economic development, environmental sustainability, and social well-being.

1.4 SIGNIFICANCE OF MACHINE LEARING (Predictive Analytics) IN UNDERSTANDING USER FEEDBACK

Improved Decision Making: Predictive analytics enables businesses to leverage historical feedback data to forecast future trends, identify patterns, and anticipate customer behavior. By using machine learning algorithms to analyze past feedback and performance metrics, businesses can make data-driven decisions regarding product development, marketing strategies, customer service enhancements, and resource allocation. This proactive approach helps businesses stay ahead of market changes, mitigate risks, and capitalize on opportunities, leading to more informed and effective decision-making processes.

Personalized Customer Experiences: Machine learning predictive analytics enables businesses to deliver personalized customer experiences by segmenting customers based on their feedback, preferences, and behaviors. By clustering customers into distinct groups and creating predictive models, businesses can tailor their offerings, messaging, and interactions to resonate with each segment's unique characteristics and preferences. This personalized approach enhances customer satisfaction, loyalty, and engagement, ultimately driving revenue growth and customer lifetime value.

Early Detection of Issues: Predictive analytics can help businesses identify potential issues and trends in customer feedback before they escalate into significant problems or impact brand reputation. By monitoring sentiment trends, sentiment shifts, and key metrics in real-time, businesses can proactively address emerging issues, resolve customer concerns, and prevent negative feedback from spreading across social media or review platforms. This proactive approach to issue management helps businesses maintain positive brand perception, customer trust, and brand loyalty.

Continuous Improvement: Machine learning predictive analytics facilitates a continuous feedback loop wherein businesses can collect, analyze, and act upon customer feedback iteratively. By integrating predictive analytics into their feedback management processes, businesses can monitor the effectiveness of their actions, measure customer sentiment changes over time, and track the impact of improvements on key performance indicators (KPIs). This iterative approach enables businesses to refine their products, services, and processes continuously, driving incremental improvements in customer satisfaction and business performance.

Competitive Advantage: Leveraging machine learning predictive analytics to understand user feedback provides businesses with a significant competitive advantage in today's data-driven marketplace. Organizations that can harness the power of predictive analytics to anticipate customer needs, personalize experiences, and deliver proactive solutions gain a competitive edge over their peers. By staying attuned to customer feedback and adapting swiftly to evolving preferences and expectations, businesses can differentiate themselves, attract and retain customers, and drive sustainable growth in a highly competitive landscape.

In summary, machine learning predictive analytics plays a vital role in understanding user feedback by providing enhanced customer understanding, improving decision-making processes, enabling personalized experiences, facilitating early issue detection, fostering continuous improvement, and delivering a competitive advantage. By harnessing the insights derived from predictive analytics, businesses can cultivate stronger customer relationships, drive innovation, and position themselves for long-term success in today's dynamic business environment.

CHAPTER 2

2 SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION (GOOGLE COLABORATORY)

The Project is developed in the system having following configuration

Processor	Intel(R) Xeon(R) CPU @ 2.20GHz
Ram	12.7 GB
Monitor	15” COLOR
Hard Disk	107.7 GB

2.2 SYSTEM REQUIREMENTS

Operation System	Ubuntu (Google Colaboratory)
Programming Language	Python
Environment	IPYNB (Python Notebook)

Python Environment

You should have a Python environment installed on your system. The code is written in Python, so you'll need Python to execute it. You can download Python from the official Python website (https://www.python.org/downloads/) and choose a version compatible with your operating system.

Python Libraries

The code relies on several Python libraries. You can install these libraries using `pip`. You mentioned specific libraries in your original code, but here's a complete list for reference:

- pandas
- matplotlib
- seaborn
- numpy

sklearn Machine Learning

- LogisticRegression
- RandomForestClassifier

You can install these libraries with the following commands:

pip install pandas numpy contractions seaborn matplotlib scikit-learn

scikit-learn: often abbreviated as sklearn, is a popular open-source machine learning library for Python. It provides simple and efficient tools for data mining and data analysis, scikit-learn features various algorithms for classification, regression, clustering, dimensionality reduction, and more, as well as utilities for preprocessing data and evaluating model performance.

Logistic Regression: is a linear model used for binary classification tasks. It's particularly useful when the relationship between the features and the target variable is logistic, meaning it follows a sigmoid curve.

Despite its name, logistic regression is a classification algorithm, not a regression algorithm, It estimates the probability that a given instance belongs to a particular class.

Random Forest: is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

It's a powerful algorithm known for its robustness and ability to handle high-dimensional datasets with a mixture of numerical and categorical features.

Random Forest reduces overfitting by averaging multiple decision trees.

Operating System

The code should work on most major operating systems, including Windows, macOS, and Linux.

Hardware

The code should run on standard desktop or laptop computers. There are no specific hardware requirements, but if you're working with very large datasets, you may benefit from a computer with more memory and processing power.

IDE or Text Editor

You can run the code in a Python IDE (e.g., PyCharm, VSCode, or Jupyter Notebook) or a simple text editor, depending on your preference. In this research, Google Colaboratory

.

Google Colaboratory

Google Colaboratory, or Colab, is an as-a-service version of Jupyter Notebook that enables you to write and execute Python code through your browser.

Jupyter Notebook is a free, open source creation from the Jupyter Project. A Jupyter notebook is like an interactive laboratory notebook that includes not just notes and data, but also code that can manipulate the data. The code can be executed within the notebook, which, in turn, can capture the code output. Applications such as Matlab and Mathematica pioneered this model, but unlike those applications, Jupyter is a browser-based web application.

Google Colab is built around Project Jupyter code and hosts Jupyter notebooks without requiring any local software installation. But while Jupyter notebooks support multiple languages, including Python, Julia and R, Colab currently only supports Python.

Colab notebooks are stored in a Google Drive account and can be shared with other users, similar to other Google Drive files. The notebooks also include an autosave feature, but they do not support simultaneous editing, so collaboration must be serial rather than parallel.

Colab is free, but has limitations. There are some code types that are forbidden, such as media serving and crypto mining. Available resources are also limited and vary depending on demand, though Google Colab offers a pro version with more reliable resourcing. There are other cloud services based on Jupyter Notebook, including Azure Notebooks from Microsoft and SageMaker Notebooks from Amazon.

The Benefits of Google Colab

Enterprise data analysts and analytics developers can use Colab to work through data analytics and manipulation problems in collaboration. They can write, execute and revise core code in a tight loop, developing the documentation in Markdown format, LaTeX or HTML as they go.

Notebooks can include embedded images as part of the documentation or as generated output. In addition, you can copy finished analytics code, with documentation, into other platforms for production use once sufficiently tested and debugged.

Google Colab eliminates the need for complex configuration setup and installation, as it runs right in the browser. It also includes pre-installed Python libraries that require no setup to use.

CHAPTER 3

3 MODEL ANALYSIS

3.1 PROBLEM STATEMENT

PowerCo are concerned about their customers leaving for better offers from other energy providers. When a customer leaves to use another service provider, this is called **churn**, This is becoming a big issue for PowerCo and they have engaged BCG to help diagnose the reason why their customers are churning.

Machine Learning Accuracy

Accuracy is a commonly used metric in machine learning to evaluate the performance of a classification model. It measures the proportion of correctly predicted instances among all instances in the dataset. The accuracy score is calculated using the formula**:**

Number of Correct Predictions: This refers to the total count of instances for which the model's prediction matches the actual class labels in the dataset .Total Number of Predictions: This is the total number of instances in the dataset that the model has made predictions for . By dividing the number of correct predictions by the total number of predictions and multiplying by 100%, we obtain the accuracy score as a percentage.

Accuracy is a straightforward metric and provides a general indication of how well the model is performing across all classes. However, it may not always be the most appropriate metric, especially in cases of imbalanced datasets where one class dominates the others. In such scenarios, a high accuracy score can be misleading if the model is simply predicting the majority class for all instances**.**

Therefore, it's essential to consider other evaluation metrics such as precision, recall, F1-score, and the confusion matrix, especially when dealing with imbalanced datasets or when different types of errors have different consequences**.**

Data Visualization for Insightful Communication

Data visualization is a powerful tool for insightful communication, enabling the transformation of complex datasets into easily understandable visuals. Here are some key ways data visualization enhances communication:

Clarity and Understanding: Visual representations such as charts, graphs, and maps help distill complex information into clear and easily digestible formats. They allow viewers to quickly grasp patterns, trends, and relationships within the data.

Identification of Patterns and Trends: Through visualization, patterns and trends that might not be apparent in raw data can become visible. This helps in identifying correlations, outliers, and other insights that drive decision-making.

Storytelling: Visualizations facilitate storytelling by organizing data into a narrative structure. They guide the audience through the data, highlighting key points and supporting arguments with visual evidence.

Engagement: Visualizations engage the audience more effectively than raw data or text alone. Colorful charts, interactive elements, and compelling designs capture attention and encourage exploration of the data.

Comparisons and Contrasts: Visualizations allow for easy comparison of different datasets or variables. Side-by-side comparisons, bar charts, and scatter plots enable viewers to understand differences and similarities at a glance.

Decision Support: Visualizations aid decision-making by presenting data in a format that supports analysis and interpretation. Decision-makers can use visualizations to evaluate options, assess risks, and identify opportunities.

Communication Across Audiences: Visualizations transcend language and technical barriers, making them accessible to diverse audiences. Whether presenting to stakeholders, clients, or the general public, visualizations facilitate communication across different backgrounds and levels of expertise.

Exploratory Analysis: Interactive visualizations allow users to explore data dynamically, drilling down into specific subsets or adjusting parameters on the fly. This enables deeper understanding and discovery of insights that may not be immediately obvious.

Data-Driven Decision-Making: Ultimately, data visualization empowers data-driven decision-making by making information more accessible, understandable, and actionable. By leveraging visual insights, organizations can make more informed choices and drive better outcomes.

3.2 CHALLENGES AND LIMITATIONS IN EXISTING SYSTEMS FOR MACHINE LEARNING

Data Quality and Quantity:

One of the fundamental challenges in machine learning is the availability of high-quality and sufficient data. Data may be incomplete, noisy, or biased, leading to suboptimal model performance. Additionally, obtaining labeled data for supervised learning tasks can be time-consuming and expensive**.**

Feature Engineering:

Feature engineering plays a crucial role in the performance of machine learning models. However, identifying and selecting relevant features from raw data can be a labor-intensive and heuristic-driven process. Automated feature engineering techniques are still evolving and may not always capture complex patterns in the data.

Overfitting and Generalization:

Overfitting occurs when a model learns to memorize the training data rather than generalize to unseen data. Balancing model complexity and generalization performance is a significant challenge, especially in deep learning models with millions of parameters. Techniques such as regularization and cross-validation help mitigate overfitting but may not always suffice.

Interpretability and Explainability:

Many machine learning models, particularly deep neural networks, are black boxes, making it challenging to interpret their decisions and understand their internal mechanisms. Explainable AI (XAI) techniques aim to address this limitation by providing insights into model predictions, but achieving both high accuracy and interpretability remains a challenge**.**

Computational Resources and Scalability:

Training complex machine learning models, especially deep learning models, requires substantial computational resources, including GPUs and TPUs. Scaling models to large datasets or deploying them in real-time production environments can be computationally intensive and costly.

Model Evaluation and Validation:

Assessing the performance of machine learning models involves various metrics and techniques, such as cross-validation and hyperparameter tuning. However, selecting appropriate evaluation metrics and avoiding data leakage or selection bias can be challenging, leading to misleading results and unreliable models.

Ethical and Social Implications:

Machine learning systems can inadvertently perpetuate biases present in the training data, leading to unfair or discriminatory outcomes. Addressing ethical and social implications, such as bias mitigation, fairness, and privacy preservation, requires careful consideration and specialized techniques.

Robustness and Security:

Machine learning models are susceptible to adversarial attacks, where subtle perturbations to input data can cause the model to make incorrect predictions. Ensuring robustness and security against such attacks is crucial, especially in safety-critical applications like autonomous vehicles and healthcare.

Domain-Specific Challenges:

Different domains present unique challenges for machine learning, such as unstructured data in natural language processing (NLP), temporal dependencies in time series forecasting, and spatial relationships in computer vision. Tailoring machine learning algorithms and techniques to specific domains requires domain expertise and specialized knowledge.

Lifecycle Management and Maintenance:

Machine learning models are not static; they require continuous monitoring, updating, and maintenance throughout their lifecycle. Managing model versions, retraining on new data, and deploying updates in production environments can be complex and resource-intensive tasks.

3.3 PROPOSED MODEL

The proposed model aims to address the challenge of predicting customer churn in the power utility sector, specifically targeting PowerCo, a fictional power utility company. By leveraging machine learning techniques, the model will analyze historical customer data to identify patterns and indicators associated with churn behavior. The ultimate goal is to develop a predictive analytics system that enables PowerCo to proactively intervene and retain at-risk customers, thereby reducing churn rates and increasing customer satisfaction and profitability.

Key Components of the Proposed Model:

Data Collection and Preprocessing:

The model will aggregate and preprocess various sources of data, including customer demographics, usage patterns, billing history, customer service interactions, and feedback. Data cleaning, normalization, and feature engineering techniques will be applied to ensure the quality and relevance of input data for predictive modeling.

Feature Selection and Engineering:

The model will employ advanced feature selection and engineering techniques to identify the most predictive variables associated with churn. This may include analyzing temporal trends, customer segmentation, seasonality effects, and customer lifetime value (CLV) metrics. Domain knowledge and insights from exploratory data analysis (EDA) will guide the selection of relevant features.

Predictive Modeling:

The heart of the proposed model lies in its predictive modeling capabilities. Various machine learning algorithms, including logistic regression, random forest, gradient boosting machines (GBM), and neural networks, will be explored to develop accurate churn prediction models. Ensemble methods and model stacking techniques may be employed to combine the strengths of multiple algorithms and improve predictive performance.

Model Evaluation and Validation:

Rigorous evaluation and validation procedures will be conducted to assess the performance of the churn prediction models. This will involve splitting the dataset into training and testing sets, cross-validation techniques, and the use of appropriate evaluation metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic (ROC)curve.

Deployment and Integration:

Once developed and validated, the churn prediction models will be deployed into PowerCo's operational systems for real-time usage. Integration with existing customer relationship management (CRM) platforms and decision support systems will enable seamless access to predictive insights and facilitate proactive customer retention strategies.

Continuous Monitoring and Improvement:

The proposed model will incorporate mechanisms for continuous monitoring and improvement. This includes tracking model performance over time, monitoring prediction drift, updating models with new data, and retraining models periodically to adapt to evolving customer behavior and market dynamics.

Expected Outcomes and Benefits:

Improved Customer Retention:

By accurately identifying customers at risk of churn, PowerCo can implement targeted retention strategies such as personalized offers, loyalty programs, proactive communication, and service enhancements to prevent churn and increase customer loyalty.

Enhanced Operational Efficiency:

The predictive analytics system will enable PowerCo to allocate resources more efficiently by focusing retention efforts on high-risk customers. This can lead to cost savings, reduced churn-related revenue losses, and optimized resource allocation across marketing, sales, and customer service departments**.**

Increased Customer Satisfaction:

Proactive intervention and personalized engagement with at-risk customers can enhance overall customer satisfaction levels. By addressing customer concerns and needs in a timely manner, PowerCo can foster stronger customer relationships and improve the overall customer experience.

Competitive Advantage:

By leveraging advanced analytics and predictive modeling techniques, PowerCo can gain a competitive advantage in the power utility market. The ability to anticipate and mitigate churn effectively can differentiate PowerCo from competitors, attract new customers, and strengthen its market position.

CHAPTER 4

4 MODEL DESIGN

4.1 MODULES

- DATA COLLECTION
- DATA PREPROCESSING
- EXPLORATORY DATA ANALYSIS
- FEATURE ENGINEERING
- MODEL TRAINING
- MODEL TESTING
- MODEL EVALUATION
- MODEL DEPLOYMENT

DATA COLLECTION

↓

DATA PREPROCESSING

↓

EXPLORATORY DATA ANALYSIS

↓

FEATURE ENGINEERING

↓

MODEL TRAINING

↓

MODEL TESTING

↓

MODEL EVALUATION

↓

MODEL DEPLOYMENT

4.2 DATA COLLECTION

Data Sources:

BCG X: Description of BCG X's role in data collection, such as being a consulting firm or data analytics provider.

PowerCo: Description of PowerCo's role as the energy company providing the dataset.

Data Set-Description

Data Set Name: client_data, price_data

Source: BCG X, From PowerCo

Format: CSV (Comma-Separated Values)

Size:

client_data :14606 rows × 26 columns (3,79,756 datas)

price_data :193002 rows × 8 columns (15,44,016 datas)

Attributes:

client_data.csv

- id = client company identifier
- activity_new = category of the company's activity
- channel_sales = code of the sales channel
- cons_12m = electricity consumption of the past 12 months
- cons_gas_12m = gas consumption of the past 12 months
- cons_last_month = electricity consumption of the last month
- date_activ = date of activation of the contract
- date_end = registered date of the end of the contract
- date_modif_prod = date of the last modification of the product
- date_renewal = date of the next contract renewal

- forecast_cons_12m = forecasted electricity consumption for next 12 months
- forecast_cons_year = forecasted electricity consumption for the next calendar year
- forecast_discount_energy = forecasted value of current discount
- forecast_meter_rent_12m = forecasted bill of meter rental for the next 2 months
- forecast_price_energy_off_peak = forecasted energy price for 1st period (off peak)
- forecast_price_energy_peak = forecasted energy price for 2nd period (peak)
- forecast_price_pow_off_peak = forecasted power price for 1st period (off peak)
- has_gas = indicated if client is also a gas client
- imp_cons = current paid consumption
- margin_gross_pow_ele = gross margin on power subscription
- margin_net_pow_ele = net margin on power subscription
- nb_prod_act = number of active products and services
- net_margin = total net margin
- num_years_antig = antiquity of the client (in number of years)
- origin_up = code of the electricity campaign the customer first subscribed to
- pow_max = subscribed power
- churn = has the client churned over the next 3 months

price_data.csv

- id = client company identifier
- price_date = reference date
- price_off_peak_var = price of energy for the 1st period (off peak)
- price_peak_var = price of energy for the 2nd period (peak)
- price_mid_peak_var = price of energy for the 3rd period (mid peak)
- price_off_peak_fix = price of power for the 1st period (off peak)

- price_peak_fix = price of power for the 2nd period (peak
- price_mid_peak_fix = price of power for the 3rd period (mid peak)

Datasets attributes are shown an below:

```
client_df.head()
```

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ	date_end	date_modif_prod	date_renewal	forecast_cons_12m	forecast_cons_year	forecast_discount_energy	foreca
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016-06-15	2015-11-01	2015-06-23	0.00	0	0.0	
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21	2016-08-30	2009-08-21	2015-08-31	189.95	0	0.0	
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16	2016-04-16	2010-04-16	2015-04-17	47.96	0	0.0	
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30	2016-03-30	2010-03-30	2015-03-31	240.04	0	0.0	
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13	2016-03-07	2010-01-13	2015-03-09	445.75	526	0.0	

```
price_df.head()
```

	id	price_date	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price_peak_fix	price_mid_peak_fix
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.266931	0.0	0.0
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.266931	0.0	0.0
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.266931	0.0	0.0
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44.266931	0.0	0.0
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44.266931	0.0	0.0

Collection Parameters:

Frequency of Collection:

Client Data: Specify how often new client data is collected, e.g., monthly, quarterly

Price Data: Specify how often new price data is collected, e.g., daily, weekly

Method of Collection:

Client Data: Describe the method used to collect client data, e.g., direct from customer databases, surveys

Price Data: Describe the method used to collect price data, e.g., automated data feeds, manual entry

Data Storage

Location:

Data is stored in secure servers hosted by BCG X and PowerCo.

Backup copies are stored in geographically distributed locations to ensure redundancy and disaster recovery.

Security Measures:

Access to data is restricted to authorized personnel only.

Data is encrypted during transmission and at rest to prevent unauthorized access.

Firewalls, intrusion detection systems, and other security measures are implemented to protect against cyber threats.

Handling User Agreements and Permissions

User Agreements: Clients provide consent for the collection and use of their data through explicit user agreements.

Agreements outline the purpose of data collection, how the data will be used, and the rights of the users.

Permissions: Users have the ability to control their data through consent management platforms.

Users can specify their preferences regarding data sharing, access by third parties, and marketing communications.

Data Preprocessing

Cleaning and Transformation:Raw data is cleaned to remove errors, inconsistencies, and missing values.

Data is transformed and standardized to ensure compatibility across different sources and formats.

Feature Engineering:

New features are derived from existing data to enhance predictive modeling and analysis.

Feature selection techniques are applied to identify the most relevant variables for modeling.

Anonymization and Privacy

Anonymization Techniques:Personally identifiable information (PII) is anonymized or pseudonymized to protect user privacy.

Methods such as data masking, tokenization, and generalization are employed to de-identify sensitive information.

Privacy Policies:

Privacy policies are communicated to users to inform them about data handling practices.

Policies detail how personal data is collected, used, stored, and shared, as well as users' rights regarding their data.

Ethical Considerations

Consent and Transparency: Users are provided with clear and transparent information about how their data will be used.

Consent is obtained prior to collecting and processing personal data, and users have the right to withdraw consent at any time.

Fairness and Bias:

Measures are taken to mitigate biases in data collection, preprocessing, and modeling.

Fair and transparent algorithms are developed to ensure equitable treatment of all individuals.

Data stewardship practices are implemented to ensure responsible and ethical handling of data throughout its lifecycle.

4.3 DATA PRE-PROCESSING

Handling missing values:

Deletion:

Listwise deletion: Delete entire rows with missing values.

Pairwise deletion: Analyze only the available data for each pair of variables in the analysis, discarding any missing data for specific pairs.

Imputation:

Mean, Median, or Mode imputation: Replace missing values with the mean, median, or mode of the respective feature.

Predictive imputation: Use machine learning algorithms to predict missing values based on other variables.

Forward fill or backward fill: Use the last known value to fill missing values in a time series or sequential data.

Hot-deck imputation: Replace missing values with a randomly selected value from a similar observation.

Interpolation: Estimate missing values by interpolating between known values.

Nearest neighbors imputation: Replace missing values with the values of nearest neighbors based on distance metrics.

Multiple imputation: Generate multiple imputations for missing values and analyze each dataset separately.

Flagging and Modeling:

Create an additional binary column to indicate whether a value is missing or not (flagging).

Use machine learning models that can handle missing values directly, such as decision trees or random forests.

Data type conversion

Converting date columns to the datetime data type is a common preprocessing step in data analysis and manipulation. By converting the date column to datetime, you can easily perform various operations like extracting specific components of the date (year, month, day), calculating time differences, filtering by date ranges, and more. This makes data manipulation and analysis much more convenient and efficient.

Handling Categorical Variables:

One-Hot Encoding:

One-hot encoding is used when a categorical feature does not have ordinality (i.e., there is no inherent order among its categories).

It creates binary columns for each category in the original feature, where each column indicates the presence or absence of that category.

This technique is suitable for algorithms that cannot work directly with categorical data, such as linear regression, support vector machines, and neural networks.

Label Encoding:

Label encoding is used when a categorical feature has ordinality (i.e., there is a meaningful order among its categories).

It assigns a unique integer value to each category, converting the categorical feature into numerical format.

This technique is appropriate for ordinal categorical features or when working with tree-based models like decision trees and random forests.

DATA SET ET BEFORE PRE-PROCESSING

client_df

```
# Assuming client_df is your DataFrame
specific_columns = ['id', 'channel_sales', 'date_activ', 'date_end', 'date_modif_prod', 'date_renewal', 'origin_up']
client_df_specific = client_df[specific_columns].head()
print(client_df_specific)
```

```
                                id                    channel_sales  \
0  24011ae4ebbe3035111d65fa7c15bc57  foosdfpfkusacimwkcsosbicdxkicaua
1  d29c2c54acc38ff3c0614d0a653813dd                           MISSING
2  764c75f661154dac3a6c254cd082ea7d  foosdfpfkusacimwkcsosbicdxkicaua
3  bba03439a292a1e166f80264c16191cb  lmkebamcaaclubfxadlmueccxoimlema
4  149d57cf92fc41cf94415803a877cb4b                           MISSING

   date_activ    date_end date_modif_prod date_renewal  \
0  2013-06-15  2016-06-15      2015-11-01   2015-06-23
1  2009-08-21  2016-08-30      2009-08-21   2015-08-31
2  2010-04-16  2016-04-16      2010-04-16   2015-04-17
3  2010-03-30  2016-03-30      2010-03-30   2015-03-31
4  2010-01-13  2016-03-07      2010-01-13   2015-03-09

                          origin_up
0  lxidpiddsbxsbosboudacockeimpuepw
1  kamkkxfxxuwbdslkwifmmcsiusiuosws
2  kamkkxfxxuwbdslkwifmmcsiusiuosws
3  kamkkxfxxuwbdslkwifmmcsiusiuosws
4  kamkkxfxxuwbdslkwifmmcsiusiuosws
```

```
167] price_df.head()
```

	id	price_date	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price_peak_fix	price_mid_peak_fix
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.266931	0.0	0.0
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.266931	0.0	0.0
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.266931	0.0	0.0
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44.266931	0.0	0.0
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44.266931	0.0	0.0

DATA SET AFTER PRE-PROCESSING

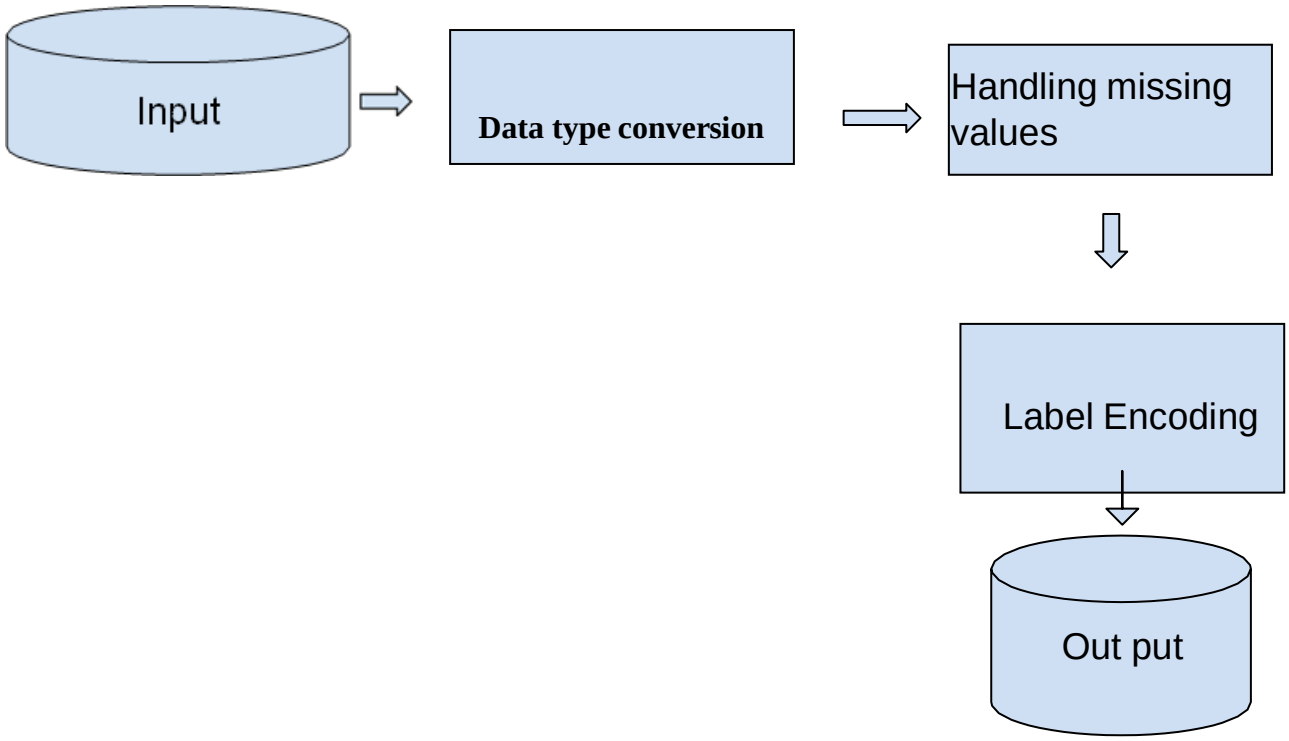
client_df

```
# Assuming client_df is your DataFrame
specific_columns = ['id', 'channel_sales', 'date_activ', 'date_end', 'date_modif_prod', 'date_renewal', 'origin_up','contract_start_year','contract_end_year']
client_df_specific = client_df[specific_columns].head()
print(client_df_specific)
```

```
                                id  channel_sales date_activ   date_end  \
0  24011ae4ebbe3035111d65fa7c15bc57              4 2013-06-15 2016-06-15
1  d29c2c54acc38ff3c0614d0a653813dd              0 2009-08-21 2016-08-30
2  764c75f661154dac3a6c254cd082ea7d              4 2010-04-16 2016-04-16
3  bba03439a292a1e166f80264c16191cb              5 2010-03-30 2016-03-30
4  149d57cf92fc41cf94415803a877cb4b              0 2010-01-13 2016-03-07

  date_modif_prod date_renewal  origin_up  contract_start_year  \
0      2015-11-01   2015-06-23          4                 2013
1      2009-08-21   2015-08-31          2                 2009
2      2010-04-16   2015-04-17          2                 2010
3      2010-03-30   2015-03-31          2                 2010
4      2010-01-13   2015-03-09          2                 2010

   contract_end_year
0               2016
1               2016
2               2016
3               2016
4               2016
```

```
[530] price_df.head()
```

	id	price_date	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price_peak_fix	price_mid_peak_fix
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.266931	0.0	0.0
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.266931	0.0	0.0
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.266931	0.0	0.0
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44.266931	0.0	0.0
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44.266931	0.0	0.0



4.4 Exploratory Data Analysis

Data Visualization:

Bar charts: Used to visualize the distribution of categorical variables such as sales channels and electricity campaign types.

Stacked bar charts: Show the distribution of categorical variables with respect to churn, allowing for comparison between churned and non-churned customers.

Histograms: Used to display the distribution of continuous numerical variables such as energy consumption and prices.

Descriptive Statistics:

Summary statistics: Calculated statistics like mean, median, maximum, and minimum to summarize the numerical data.

Counting: Counting the occurrence of values in certain columns to understand the distribution of categorical variables.

CHAPTER 5

5.1 Feature Engineering:

Feature engineering is the process of creating new features or transforming existing features in a dataset to improve the performance of machine learning models. In the context of the EDA you provided, here are some potential feature engineering techniques that could be applied:

Date Features: Extracting additional information from date-related columns such as date_activ, date_end, date_modif_prod, and date_renewal. This could include features such as month, year, day of the week, season, duration of the contract, etc.

Aggregation: Creating new features by aggregating existing ones. For example, you could calculate the total energy consumption over a certain period, average consumption, minimum and maximum consumption, etc.

Ratio Features: Creating features by dividing one numerical feature by another. For example, you could calculate the ratio of forecast_cons_12m to cons_12m, which could represent the accuracy of the forecast.

Interaction Features: Creating new features by combining existing ones. For example, multiplying two numerical features together or creating interaction terms between categorical variables.

Binning or Bucketing: Grouping numerical values into bins or categories. This can help simplify complex relationships and reduce the impact of outliers.

Encoding Categorical Variables: Transforming categorical variables into numerical representations using techniques such as one-hot encoding, label encoding, or target encoding.

Feature Scaling: Scaling numerical features to a similar range to prevent certain features from dominating others during model training. Common techniques include min-max scaling or standardization.

Feature Selection: Selecting the most relevant features for model training to reduce dimensionality and improve model performance. This can be done using techniques such as correlation analysis, feature importance from tree-based models, or model-based feature selection.

.

5.2 MODEL

Selecting the right model for Customer churn Analysis is crucial, and the choice between Random Forest classifier should be based on the characteristics of the data and the specific requirements of your project.

Machine learning is important because it gives enterprises a view of trends in customer behaviour and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google, and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

Machine learning Types

Supervised Learning

In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

Unsupervised Learning

This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through datasets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.

Semi-supervised Learning

This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.

Reinforcement Learning

Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

Supervised machine learning requires the data scientist to train the algorithm with both labeled inputs and desired outputs. Supervised learning algorithms are good for the following tasks:

Binary Classification: Dividing data into two categories

Multi-class Classification: Choosing between more than two types of answers.

Regression Modelling: Predicting continuous values.

Ensembling: Combining the predictions of multiple machine learning models to produce an accurate prediction.

5.3 Random Forest classifier.

The Random Forest classifier is a popular machine learning algorithm that belongs to the ensemble learning family.

Ensemble Learning:

Random Forest is an ensemble method that combines multiple decision trees to make predictions. Each tree in the forest independently predicts the target variable, and the final prediction is made by averaging (for regression) or voting (for classification) the predictions of all trees.

Decision Trees:

Each tree in the Random Forest is a decision tree. Decision trees recursively partition the feature space into regions, where each region corresponds to a specific prediction.

Random Feature Subsets:

Random Forest introduces randomness in the training process by considering only a subset of features at each split of the decision tree. This helps to decorrelate the trees and reduces overfitting.

Bootstrap Aggregation (Bagging):

Random Forest also uses bootstrap aggregation, commonly known as bagging, to train each decision tree on a random subset of the training data with replacement. This further enhances the diversity among the trees.

Voting or Averaging:

For classification tasks, Random Forest combines the predictions of individual trees by majority voting. For regression tasks, it combines predictions by averaging.

Hyperparameters:

Random Forest has several hyperparameters that can be tuned to optimize its performance, such as the number of trees in the forest, the maximum depth of the trees, and the number of features considered at each split.

Advantages:

Random Forest is known for its robustness to overfitting, versatility in handling various types of data, and capability to provide estimates of feature importance.

How Random Forest classifier works

Random Forest classifier is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees

Random Feature Selection: For each split in each tree, a random subset of features is selected. This helps to decorrelate the trees and ensures that each tree is trained on different subsets of features.

Growing Trees: Each tree is grown to its maximum depth without pruning. Typically, decision trees are grown using algorithms like CART (Classification and Regression Trees), and they split the data at each node based on the feature that provides the best split according to a certain criterion (e.g., Gini impurity for classification, mean squared error for regression).

Voting: During prediction, each tree in the forest independently predicts the class for the input sample. For classification tasks, the mode (most frequent class) of the predictions from individual trees is taken as the final prediction. For regression tasks, the mean of the predictions is taken.

Aggregation: The final prediction of the Random Forest is determined by aggregating the predictions of all the trees. This aggregation reduces the variance of the model and often leads to better generalization performance compared to individual decision trees.

Out-of-Bag Evaluation: In Random Forest, each tree is trained on a bootstrap sample, leaving out about one-third of the data on average. This out-of-bag (OOB) data is then used to evaluate the performance of the model without the need for a separate validation set.

Applications of Random Forest classifier

Classification:

Identifying spam emails from a large corpus of emails.

Predicting customer churn in telecommunications, banking, or subscription-based services.

Medical diagnosis, such as classifying diseases based on patient symptoms.

Regression:

Predicting house prices based on features like location, size, and amenities.

Forecasting stock prices or market trends based on historical data.

Estimating the demand for a product based on historical sales data and external factors**.**

Anomaly Detection:

Detecting fraudulent transactions in banking or credit card transactions.

Identifying network intrusions or cybersecurity threats.

Monitoring machinery or equipment for unusual behavior in predictive maintenance systems.

Feature Importance Analysis:

Identifying important features in large datasets for further analysis or feature engineering.

Understanding which factors contribute most to a particular outcome or prediction.

Recommendation Systems:

Generating personalized recommendations for products, movies, or content based on user preferences and behavior.

Matching users with similar interests or profiles in social networks or dating platforms.

Image and Video Analysis:

Classifying objects or scenes in images and videos for content moderation, surveillance, or autonomous vehicles.

Predicting facial expressions or emotions in facial recognition systems.

Natural Language Processing (NLP):

Sentiment analysis to determine the sentiment of text data, such as customer reviews or social media posts.

Text classification for document categorization, spam detection, or language identification.

Bioinformatics:

Predicting protein-protein interactions or protein functions in bioinformatics and computational biology.

Analyzing DNA sequences for gene prediction, disease diagnosis, or drug discovery**.**

5.4 Model Training

Implementation of the Random Forest classifier using Python's scikit-learn library.

Random Forest is a versatile and powerful ensemble learning algorithm used for classification and regression tasks. In this section, we'll discuss the implementation of the Random Forest classifier using Python's scikit-learn library. We'll cover the training process, parameter tuning, and the importance of hyperparameter optimization for enhancing model performance.

Loading Packages: You've imported necessary libraries for data preprocessing, model training, and evaluation, including StandardScaler, StratifiedKFold, LogisticRegression, RandomForestClassifier, and various metrics from sklearn.metrics.

from sklearn.ensemble import RandomForestClassifier

Data Preprocessing:

You've created a copy of the client data and extracted the year from date columns (date_modif_prod and date_renewal).

Use of LabelEncoder to encode categorical variables like 'has_gas'.

Calculation of price differences (diff_dec_jan_off_peak_var and diff_dec_jan_off_peak_fix) between maximum and minimum prices for each customer.

Similar calculations for other price attributes (price_peak_var, price_peak_fix, etc.).

Merging price changing trends into the training data.

Creating Training Data and Labels:

Drop unnecessary columns like 'id' and 'churn' from the feature matrix X.

Assign the target variable 'churn' to the label vector y.

Training Data Preparation Notes:

No need for data scaling as tree-based models like RandomForestClassifier are not sensitive to feature scaling.

Categorical variables are label-encoded instead of one-hot encoded.

Final Data Dimensions:

The feature matrix X has a shape of (14606, 36), indicating 14606 samples and 36 features.

The label vector y has a shape of (14606,), matching the number of samples.

5.5 MODEL TESTING

Initialize an array pred_train_labels to store predicted probabilities for each fold. It has a shape of (number of samples, 2) where the second dimension represents the probability of belonging to each class.

Create a DataFrame feature_importance_df to store feature importances**.**

Cross-validation Loop:

Split the data into training and test sets using 5-fold stratified cross-validation.

For each fold:

Train a RandomForestClassifier model (rf) on the training data (X_train, y_train).

Predict probabilities on the test data (X_test) using predict_proba.

Update pred_train_labels with predicted probabilities for the test samples in the current fold.

Update feature_importance_df with feature importances from the current model.

Print precision, recall, and accuracy scores for the current fold.

Aggregate Predictions:

Combine predicted probabilities from all folds into a single array pred_y by taking the class with the highest probability (argmax along the second axis).

Evaluation:

Print the overall precision, recall, and accuracy scores across all folds.

Display a barplot showing feature importances.

Print a classification report with precision, recall, and F1-score for each class (0 and 1) and the overall accuracy.

Display a confusion matrix.

Evaluation Metrics

Evaluation metrics are essential for assessing the performance of machine learning models, especially in the context of predicting customer churn for a power company.

Precision:

Precision measures the accuracy of positive predictions. In the context of customer churn prediction, precision represents the proportion of correctly identified churned customers among all customers predicted to churn. A high precision indicates that the model is making fewer false positive predictions, which is crucial for a power company to efficiently allocate resources for retaining customers.

Recall (Sensitivity):

Recall measures the proportion of actual positives that are correctly identified by the model. In the context of customer churn prediction, recall represents the proportion of churned customers that are correctly identified by the model among all actual churned customers. High recall indicates that the model is capturing a large portion of the churned customers, which is important for the power company to prevent customer loss effectively.

F1-Score:

The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is especially useful when there is an imbalance between the classes. A high F1-score indicates good performance in both precision and recall.

Accuracy:

Accuracy measures the overall correctness of the model's predictions. In the context of customer churn prediction, accuracy represents the proportion of correctly classified instances (both churned and non-churned) among all instances. While accuracy is important, it may not be the most suitable metric for imbalanced datasets like customer churn prediction, where the majority of customers may not churn.

Confusion Matrix:

The confusion matrix provides a tabular representation of the model's predictions against the actual labels. It includes metrics such as true positives, true negatives, false positives, and false negatives, which are useful for understanding the types of errors made by the model.

ROC Curve and AUC:

Receiver Operating Characteristic (ROC) curve plots the true positive rate against the false positive rate at various threshold settings. The Area Under the ROC Curve (AUC) provides an aggregate measure of the model's performance across all possible threshold settings. It is particularly useful for evaluating binary classification models like customer churn prediction.

Error Analysis

Error analysis is a crucial step in assessing the performance of a predictive model, especially in the context of customer churn prediction for a power company (PowerCo).

Identify Types of Errors: Analyze the errors made by the model to understand where it is failing. Errors can be categorized into:

False Positives: Instances where the model incorrectly predicts that a customer will churn when they actually don't.

False Negatives: Instances where the model fails to predict churn for customers who actually churn.

Investigate False Positives:

Examine the characteristics of false positive cases. Are there any common traits among these customers?

Look for patterns or similarities in false positive instances. Are there specific features or behaviors that are misleading the model.

Investigate False Negatives:

Examine the characteristics of false negative cases. What distinguishes these customers from others.

Look for patterns or similarities in false negative instances. Are there specific features or behaviors that the model is failing to capture?

Feature Importance Analysis:

Evaluate the importance of features in false positive and false negative predictions. Are there certain features that contribute more to these errors?

Determine if there are features that the model should consider more or less when making predictions.

Threshold Adjustment:

Assess the impact of adjusting the classification threshold on model performance. By changing the threshold for predicting churn, you can trade off between false positives and false negatives.

Plot precision-recall curves or ROC curves to visualize the trade-offs between precision, recall, and the threshold.

Feedback Loop:

Incorporate feedback from domain experts or stakeholders at PowerCo. They can provide insights into the reasons behind certain errors and suggest improvements.

Use customer feedback or additional data sources to refine the model and address specific shortcomings identified during error analysis.

Iterative Improvement:

Implement changes based on the findings of error analysis and retrain the model. This could involve feature engineering, adjusting hyperparameters, or using different algorithms.

Continuously monitor the model's performance over time and iterate on the error analysis process as needed.

Accuracy

$$\frac{TP + TN}{TP + FP + TN + FN}$$

The most commonly used metric to judge a model is actually not a clear indicator of the performance. The worst happens when classes are imbalanced.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Fold 1 Precision 0.714 Recall 0.018 Accuracy 0.904

Fold 2 Precision 0.800 Recall 0.028 Accuracy 0.905

Fold 3 Precision 0.900 Recall 0.032 Accuracy 0.906

Fold 4 Precision 0.875 Recall 0.025 Accuracy 0.905

Fold 5 Precision 0.700 Recall 0.025 Accuracy 0.904

Total Precision 0.800 Recall 0.025 Accuracy 0.905

Classification Report:

RandomForestClassifier Performance Metrics

	Precision	recall	f1-score	support
0	0.91	1.00	0.95	13187
1	0.80	0.03	0.05	1419
accuracy			0.90	14606
macro avg	0.85	0.51	0.50	14606
weighted avg	0.89	0.90	0.86	14606

Classification Report

Precision: Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives. A high precision indicates that the model has fewer false positives.

Recall: Recall, also known as sensitivity or true positive rate, measures the ability of the model to correctly identify positive instances out of all actual positive instances. It is calculated as the ratio of true positives to the sum of true positives and false negatives. A high recall indicates that the model has fewer false negatives**.**

F1-Score: The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, considering both false positives and false negatives. F1-score is calculated as 2 * (precision * recall) / (precision + recall). It ranges from 0 to 1, where higher values indicate better model performance.

Support: Support refers to the number of actual occurrences of each class in the dataset. It provides context for the precision, recall, and F1-score by indicating the distribution of each class**.**

Macro Average: The macro average is the unweighted mean of precision, recall, and F1-score across all classes. It gives equal weight to each class, regardless of its size. Macro average is useful for evaluating model performance when class imbalance exists.

Weighted Average: The weighted average is the average of precision, recall, and F1-score weighted by the support of each class. It provides a more balanced assessment of the model's overall performance, considering the distribution of classes in the dataset.

Impact of a 20% discount

Calculating Probability of Churn: First, predicts the probability of churn for each customer without applying the discount. This is done using a random forest classifier trained on the training data.

Applying the Discount: Next, a 20% discount is applied to the prices for each customer. This simulates the effect of offering a discount to prevent churn.

Predicting Churn with Discount: The random forest classifier is used again to predict the probability of churn for each customer after applying the discount.

Calculating Expected Profit: For each customer, the expected profit is calculated both with and without the discount. This is based on the customer's usage patterns and the discounted prices.

Analyzing Impact on Profit: The code identifies customers whose expected profit increases or decreases after applying the discount. It also calculates the total extra profit from customers whose expected profit increases and the total loss from customers whose expected profit decreases.

This analysis provides valuable insights into the potential impact of offering a discount on customer churn. By comparing the expected profit before and after the discount, businesses can make informed decisions about whether offering discounts is a cost-effective strategy for retaining customers.

main-project

BCG X

Boston Consulting Group

PowerCo's

1 A major gas and electricity utility that supplies to small and medium sized enterprises.

2 The energy market has had a lot of change in recent years and there are more options than ever for customers to choose from.

3 PowerCo are concerned about their customers leaving for better offers from other energy providers. When a customer leaves to use another service provider, this is called churn.

4 Project title

5 Customer churn Analysis

6 Abstract:

PowerCo is a major gas and electricity utility that supplies to corporate, SME (Small & Medium Enterprise), and residential customers. The power-liberalization of the energy market in Europe has led to significant customer churn, especially in the SME segment. In an increasingly competitive energy market, customer retention has become a critical focus for PowerCo's sustained growth and profitability.

PowerCo, facing the prospect of customer attrition, is operating in an industry characterized by evolving consumer preferences, regulatory changes, and the emergence of alternative energy solutions. To interpret this business context, a comprehensive analysis considers factors such as market dynamics, competitive landscape, customer behavior, and the company's existing strategies.

The study identifies the primary drivers contributing to customer churn, encompassing factors such as service quality, pricing structures, customer engagement, and technological advancements. Understanding these factors forms the basis for strategic interventions aimed at enhancing customer satisfaction and loyalty.

Recommendations include the implementation of targeted customer engagement initiatives, personalized communication strategies, and the adoption of innovative technologies to improve service delivery and customer experience. Additionally, the development of flexible pricing models and the introduction of loyalty programs are proposed to incentivize customer retention.

The analysis draws on industry best practices, case studies, and data-driven insights to formulate a holistic strategy tailored to PowerCo's specific challenges. The objective is to empower PowerCo to not only retain its existing customer base but also to position itself as a preferred choice in a rapidly evolving energy landscape.

By embracing a customer-centric approach and leveraging data analytics, PowerCo can proactively address the identified issues, foster customer loyalty, and ultimately secure its market position. The implementation of these recommendations is expected to enhance PowerCo's competitive advantage and ensure sustainable growth in the dynamic energy market.

[ ]:
```
from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

7 Data Set-Description

Data Set Name: client_data, price_data

Source: BCG X, From PowerCo

Purpose:

Format: CSV (Comma-Separated Values)

Size:
client_data :14606 rows × 26 columns (3,79,756 datas)
price_data :193002 rows × 8 columns (15,44,016 datas)

8 Columns:

client_data.csv

id = client company identifier
activity_new = category of the company's activity
channel_sales = code of the sales channel
cons_12m = electricity consumption of the past 12 months
cons_gas_12m = gas consumption of the past 12 months
cons_last_month = electricity consumption of the last month
date_activ = date of activation of the contract
date_end = registered date of the end of the contract
date_modif_prod = date of the last modification of the product
date_renewal = date of the next contract renewal
forecast_cons_12m = forecasted electricity consumption for next 12 months
forecast_cons_year = forecasted electricity consumption for the next calendar year

forecast_discount_energy = forecasted value of current discount
forecast_meter_rent_12m = forecasted bill of meter rental for the next 2 months
forecast_price_energy_off_peak = forecasted energy price for 1st period (off peak)
forecast_price_energy_peak = forecasted energy price for 2nd period (peak)
forecast_price_pow_off_peak = forecasted power price for 1st period (off peak)
has_gas = indicated if client is also a gas client
imp_cons = current paid consumption
margin_gross_pow_ele = gross margin on power subscription
margin_net_pow_ele = net margin on power subscription
nb_prod_act = number of active products and services
net_margin = total net margin
num_years_antig = antiquity of the client (in number of years)
origin_up = code of the electricity campaign the customer first subscribed to
pow_max = subscribed power
churn = has the client churned over the next 3 months

price_data.csv

id = client company identifier
price_date = reference date
price_off_peak_var = price of energy for the 1st period (off peak)
price_peak_var = price of energy for the 2nd period (peak)
price_mid_peak_var = price of energy for the 3rd period (mid peak)
price_off_peak_fix = price of power for the 1st period (off peak)
price_peak_fix = price of power for the 2nd period (peak)
price_mid_peak_fix = price of power for the 3rd period (mid peak)

[ ]:
```
import pandas as pd
pd.set_option('display.max_columns', 50)
#reading csv file
client_df=pd.read_csv("/content/drive/MyDrive/Bcg x/client_data (1).csv")
price_df=pd.read_csv("/content/drive/MyDrive/Bcg x/price_data (1).csv")
```

[ ]:
```
client_df.head()
```

[ ]:
```
                                 id                      channel_sales  \
0  24011ae4ebbe3035111d65fa7c15bc57  foosdfpfkusacimwkcsosbicdxkicaua
1  d29c2c54acc38ff3c0614d0a653813dd                            MISSING
2  764c75f661154dac3a6c254cd082ea7d  foosdfpfkusacimwkcsosbicdxkicaua
3  bba03439a292a1e166f80264c16191cb  lmkebamcaaclubfxadlmueccxoimlema
4  149d57cf92fc41cf94415803a877cb4b                            MISSING

   cons_12m  cons_gas_12m  cons_last_month  date_activ     date_end  \
0         0         54946                0  2013-06-15  2016-06-15
1      4660             0                0  2009-08-21  2016-08-30
2       544             0                0  2010-04-16  2016-04-16
3      1584             0                0  2010-03-30  2016-03-30
4      4425             0              526  2010-01-13  2016-03-07
```

```
   date_modif_prod  date_renewal  forecast_cons_12m  forecast_cons_year  \
0       2015-11-01   2015-06-23               0.00                   0
1       2009-08-21   2015-08-31             189.95                   0
2       2010-04-16   2015-04-17              47.96                   0
3       2010-03-30   2015-03-31             240.04                   0
4       2010-01-13   2015-03-09             445.75                 526

   forecast_discount_energy  forecast_meter_rent_12m  \
0                       0.0                     1.78
1                       0.0                    16.27
2                       0.0                    38.72
3                       0.0                    19.83
4                       0.0                   131.73

   forecast_price_energy_off_peak  forecast_price_energy_peak  \
0                        0.114481                    0.098142
1                        0.145711                    0.000000
2                        0.165794                    0.087899
3                        0.146694                    0.000000
4                        0.116900                    0.100015

   forecast_price_pow_off_peak has_gas  imp_cons  margin_gross_pow_ele  \
0                    40.606701       t      0.00                 25.44
1                    44.311378       f      0.00                 16.38
2                    44.311378       f      0.00                 28.60
3                    44.311378       f      0.00                 30.22
4                    40.606701       f     52.32                 44.91

   margin_net_pow_ele  nb_prod_act  net_margin  num_years_antig  \
0               25.44            2      678.99                3
1               16.38            1       18.89                6
2               28.60            1        6.60                6
3               30.22            1       25.46                6
4               44.91            1       47.98                6

                         origin_up  pow_max  churn
0  lxidpiddsbxsbosboudacockeimpuepw   43.648      1
1  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.800      0
2  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.856      0
3  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.200      0
4  kamkkxfxxuwbdslkwifmmcsiusiuosws   19.800      0
```

[ ]:
```
# Assuming client_df is your DataFrame
specific_columns = ['id', 'channel_sales', 'date_activ', 'date_end',␣
↪'date_modif_prod', 'date_renewal', 'origin_up']
client_df_specific = client_df[specific_columns].head()
```

```
print(client_df_specific)
```

```
                                 id                         channel_sales  \
0  24011ae4ebbe3035111d65fa7c15bc57  foosdfpfkusacimwkcsosbicdxkicaua
1  d29c2c54acc38ff3c0614d0a653813dd                             MISSING
2  764c75f661154dac3a6c254cd082ea7d  foosdfpfkusacimwkcsosbicdxkicaua
3  bba03439a292a1e166f80264c16191cb  lmkebamcaaclubfxadlmueccxoimlema
4  149d57cf92fc41cf94415803a877cb4b                             MISSING

   date_activ    date_end  date_modif_prod  date_renewal  \
0  2013-06-15  2016-06-15       2015-11-01    2015-06-23
1  2009-08-21  2016-08-30       2009-08-21    2015-08-31
2  2010-04-16  2016-04-16       2010-04-16    2015-04-17
3  2010-03-30  2016-03-30       2010-03-30    2015-03-31
4  2010-01-13  2016-03-07       2010-01-13    2015-03-09

                          origin_up
0  lxidpiddsbxsbosboudacockeimpuepw
1  kamkkxfxxuwbdslkwifmmcsiusiuosws
2  kamkkxfxxuwbdslkwifmmcsiusiuosws
3  kamkkxfxxuwbdslkwifmmcsiusiuosws
4  kamkkxfxxuwbdslkwifmmcsiusiuosws
```

[ ]:
```
price_df.head()
```

[ ]:
```
                                 id  price_date  price_off_peak_var  \
0  038af19179925da21a25619c5a24b745  2015-01-01            0.151367
1  038af19179925da21a25619c5a24b745  2015-02-01            0.151367
2  038af19179925da21a25619c5a24b745  2015-03-01            0.151367
3  038af19179925da21a25619c5a24b745  2015-04-01            0.149626
4  038af19179925da21a25619c5a24b745  2015-05-01            0.149626

   price_peak_var  price_mid_peak_var  price_off_peak_fix  price_peak_fix  \
0             0.0                 0.0           44.266931             0.0
1             0.0                 0.0           44.266931             0.0
2             0.0                 0.0           44.266931             0.0
3             0.0                 0.0           44.266931             0.0
4             0.0                 0.0           44.266931             0.0

   price_mid_peak_fix
0                 0.0
1                 0.0
2                 0.0
3                 0.0
4                 0.0
```

[ ]:
```
#check the size of Data frame
client_df.shape
```

[ ]: (14606, 26)

[ ]:
```
price_df.shape
```

[ ]: (193002, 8)

[ ]:
```

```

[ ]:
```
id                                object
channel_sales                     object
cons_12m                           int64
cons_gas_12m                       int64
cons_last_month                    int64
date_activ                        object
date_end                          object
date_modif_prod                   object
date_renewal                      object
forecast_cons_12m                float64
forecast_cons_year                 int64
forecast_discount_energy         float64
forecast_meter_rent_12m          float64
forecast_price_energy_off_peak   float64
forecast_price_energy_peak       float64
forecast_price_pow_off_peak      float64
has_gas                           object
imp_cons                         float64
margin_gross_pow_ele             float64
margin_net_pow_ele               float64
nb_prod_act                        int64
net_margin                       float64
num_years_antig                    int64
origin_up                         object
pow_max                          float64
churn                              int64
dtype: object
```

[ ]:
```
#convert datetime feature to datetime data type
for f in ['date_activ','date_end','date_modif_prod','date_renewal']:
    client_df[f]= pd.to_datetime(client_df[f])
```

[ ]:
```
client_df['contract_start_year'] = client_df['date_activ'].dt.year
client_df['contract_end_year'] = client_df['date_end'].dt.year
```

[ ]:
```
price_df.dtypes
```

[ ]:
```
id                      object
price_date              object
price_off_peak_var     float64
price_peak_var         float64
price_mid_peak_var     float64
price_off_peak_fix     float64
price_peak_fix         float64
price_mid_peak_fix     float64
dtype: object
```

[ ]:
```
#Convert datetime feature to datetime data type
price_df['price_date'] = pd.to_datetime(price_df['price_date'])
```

[ ]:
```
#check the count of null for each columns
client_df.isnull().sum()
```

[ ]:
```
id                                0
channel_sales                     0
cons_12m                          0
cons_gas_12m                      0
cons_last_month                   0
date_activ                        0
date_end                          0
date_modif_prod                   0
date_renewal                      0
forecast_cons_12m                 0
forecast_cons_year                0
forecast_discount_energy          0
forecast_meter_rent_12m           0
forecast_price_energy_off_peak    0
forecast_price_energy_peak        0
forecast_price_pow_off_peak       0
has_gas                           0
imp_cons                          0
margin_gross_pow_ele              0
margin_net_pow_ele                0
nb_prod_act                       0
net_margin                        0
num_years_antig                   0
origin_up                         0
pow_max                           0
churn                             0
contract_start_year               0
contract_end_year                 0
dtype: int64
```

[ ]:
```
price_df.isnull().sum()
```

[ ]:
```
id                    0
price_date            0
price_off_peak_var    0
price_peak_var        0
price_mid_peak_var    0
price_off_peak_fix    0
price_peak_fix        0
price_mid_peak_fix    0
dtype: int64
```

[ ]:
```
#describe method use for basic statistics for numerical data
client_df.describe(include='object')
```

[ ]:
```
                                     id                      channel_sales  \
count                             14606                              14606
unique                            14606                                  8
top     24011ae4ebbe3035111d65fa7c15bc57   foosdfpfkusacimwkcsosbicdxkicaua
freq                                  1                               6754

        has_gas                          origin_up
count     14606                              14606
unique        2                                  6
top           f   lxidpiddsbxsbosboudacockeimpuepw
freq      11955                               7097
```

But values of "channel_sales" and "origin_up" are not understandable, so I convert these features to numeric data type by label encoding.

[ ]:
```
#load packages
from sklearn.preprocessing import LabelEncoder
channel_encoder = LabelEncoder()
client_df['channel_sales'] = channel_encoder.
 ↪fit_transform(client_df['channel_sales'])
origin_encoder = LabelEncoder()
client_df['origin_up'] = origin_encoder.fit_transform(client_df['origin_up'])
```

[ ]:
```
price_df.describe(include='object')
```

[ ]:
```
                                     id
count                            193002
unique                            16096
top     038af19179925da21a25619c5a24b745
freq                                 12
```

[ ]:
```
client_df.describe()
```

[ ]:

	channel_sales	cons_12m	cons_gas_12m	cons_last_month \
count	14606.000000	1.460600e+04	1.460600e+04	14606.000000
mean	3.266945	1.592203e+05	2.809238e+04	16090.269752
min	0.000000	0.000000e+00	0.000000e+00	0.000000
25%	0.000000	5.674750e+03	0.000000e+00	0.000000
50%	4.000000	1.411550e+04	0.000000e+00	792.500000
75%	4.000000	4.076375e+04	0.000000e+00	3383.000000
max	7.000000	6.207104e+06	4.154590e+06	771203.000000
std	2.183223	5.734653e+05	1.629731e+05	64364.196422

	date_activ	date_end \
count	14606	14606
mean	2011-01-28 07:54:18.879912448	2016-07-27 20:48:26.422018560
min	2003-05-09 00:00:00	2016-01-28 00:00:00
25%	2010-01-15 00:00:00	2016-04-27 06:00:00
50%	2011-03-04 00:00:00	2016-08-01 00:00:00
75%	2012-04-19 00:00:00	2016-10-31 00:00:00
max	2014-09-01 00:00:00	2017-06-13 00:00:00
std	NaN	NaN

	date_modif_prod	date_renewal \
count	14606	14606
mean	2013-01-02 12:29:10.951663872	2015-07-21 06:59:00.353279488
min	2003-05-09 00:00:00	2013-06-26 00:00:00
25%	2010-08-12 00:00:00	2015-04-17 00:00:00
50%	2013-06-19 00:00:00	2015-07-27 00:00:00
75%	2015-06-16 00:00:00	2015-10-29 00:00:00
max	2016-01-29 00:00:00	2016-01-28 00:00:00
std	NaN	NaN

	forecast_cons_12m	forecast_cons_year	forecast_discount_energy \
count	14606.000000	14606.000000	14606.000000
mean	1868.614880	1399.762906	0.966726
min	0.000000	0.000000	0.000000
25%	494.995000	0.000000	0.000000
50%	1112.875000	314.000000	0.000000
75%	2401.790000	1745.750000	0.000000
max	82902.830000	175375.000000	30.000000
std	2387.571531	3247.786255	5.108289

	forecast_meter_rent_12m	forecast_price_energy_off_peak \
count	14606.000000	14606.000000
mean	63.086871	0.137283
min	0.000000	0.000000
25%	16.180000	0.116340
50%	18.795000	0.143166
75%	131.030000	0.146348

```
max                599.310000                         0.273963
std                 66.165783                         0.024623

       forecast_price_energy_peak  forecast_price_pow_off_peak      imp_cons  \
count                14606.000000                 14606.000000  14606.000000
mean                    0.050491                    43.130056    152.786896
min                     0.000000                     0.000000      0.000000
25%                     0.000000                    40.606701      0.000000
50%                     0.084138                    44.311378     37.395000
75%                     0.098837                    44.311378    193.980000
max                     0.195975                    59.266378  15042.790000
std                     0.049037                     4.485988    341.369366

       margin_gross_pow_ele  margin_net_pow_ele   nb_prod_act     net_margin  \
count          14606.000000        14606.000000  14606.000000   14606.000000
mean              24.565121           24.562517      1.292346     189.264522
min                0.000000            0.000000      1.000000       0.000000
25%               14.280000           14.280000      1.000000      50.712500
50%               21.640000           21.640000      1.000000     112.530000
75%               29.880000           29.880000      1.000000     243.097500
max              374.640000          374.640000     32.000000   24570.650000
std               20.231172           20.230280      0.709774     311.798130

       num_years_antig      origin_up       pow_max          churn  \
count     14606.000000  14606.000000  14606.000000  14606.000000
mean          4.997809      3.178899     18.135136      0.097152
min           1.000000      0.000000      3.300000      0.000000
25%           4.000000      2.000000     12.500000      0.000000
50%           5.000000      3.000000     13.856000      0.000000
75%           6.000000      4.000000     19.172500      0.000000
max          13.000000      5.000000    320.000000      1.000000
std           1.611749      0.887801     13.534743      0.296175

       contract_start_year  contract_end_year
count          14606.000000       14606.000000
mean            2010.577502        2016.064563
min             2003.000000        2016.000000
25%             2010.000000        2016.000000
50%             2011.000000        2016.000000
75%             2012.000000        2016.000000
max             2014.000000        2017.000000
std                1.653268           0.245761
```

[ ]:
```
price_df.describe()
```

[ ]:
```
                          price_date  price_off_peak_var  price_peak_var  \
count                         193002       193002.000000   193002.000000
```

```
mean   2015-06-16  12:50:49.933161216              0.141027       0.054630
min              2015-01-01 00:00:00              0.000000       0.000000
25%             2015-04-01 00:00:00              0.125976       0.000000
50%             2015-07-01 00:00:00              0.146033       0.085483
75%             2015-10-01 00:00:00              0.151635       0.101673
max             2015-12-01 00:00:00              0.280700       0.229788
std                             NaN              0.025032       0.049924

        price_mid_peak_var  price_off_peak_fix  price_peak_fix  \
count       193002.000000       193002.000000   193002.000000
mean             0.030496           43.334477       10.622875
min              0.000000            0.000000        0.000000
25%              0.000000           40.728885        0.000000
50%              0.000000           44.266930        0.000000
75%              0.072558           44.444710       24.339581
max              0.114102           59.444710       36.490692
std              0.036298            5.410297       12.841895

        price_mid_peak_fix
count       193002.000000
mean             6.409984
min              0.000000
25%              0.000000
50%              0.000000
75%             16.226389
max             17.458221
std              7.773592
```

[ ]:
```
client_df[client_df.dtypes[client_df.dtypes=='datetime64[ns]'].index.tolist()].
↪describe()
```

[ ]:
```
                          date_activ                       date_end  \
count                          14606                          14606
mean   2011-01-28 07:54:18.879912448  2016-07-27  20:48:26.422018560
min              2003-05-09 00:00:00            2016-01-28 00:00:00
25%              2010-01-15 00:00:00            2016-04-27 06:00:00
50%              2011-03-04 00:00:00            2016-08-01 00:00:00
75%              2012-04-19 00:00:00            2016-10-31 00:00:00
max              2014-09-01 00:00:00            2017-06-13 00:00:00

                     date_modif_prod                     date_renewal
count                          14606                          14606
mean   2013-01-02 12:29:10.951663872  2015-07-21  06:59:00.353279488
min              2003-05-09 00:00:00            2013-06-26 00:00:00
25%              2010-08-12 00:00:00            2015-04-17 00:00:00
50%              2013-06-19 00:00:00            2015-07-27 00:00:00
75%              2015-06-16 00:00:00            2015-10-29 00:00:00
```

```
max                 2016-01-29 00:00:00              2016-01-28 00:00:00
```

[ ]:
```
price_df['price_date'].describe()
```

[ ]:
```
count                           193002
mean     2015-06-16  12:50:49.933161216
min                 2015-01-01  00:00:00
25%                 2015-04-01  00:00:00
50%                 2015-07-01  00:00:00
75%                 2015-10-01  00:00:00
max                 2015-12-01  00:00:00
Name: price_date, dtype: object
```

[ ]:
```
client_df.head()
```

[ ]:
```
                                id  channel_sales  cons_12m  cons_gas_12m  \
0  24011ae4ebbe3035111d65fa7c15bc57              4         0         54946
1  d29c2c54acc38ff3c0614d0a653813dd              0      4660             0
2  764c75f661154dac3a6c254cd082ea7d              4       544             0
3  bba03439a292a1e166f80264c16191cb              5      1584             0
4  149d57cf92fc41cf94415803a877cb4b              0      4425             0

   cons_last_month  date_activ    date_end  date_modif_prod  date_renewal  \
0                0  2013-06-15  2016-06-15       2015-11-01    2015-06-23
1                0  2009-08-21  2016-08-30       2009-08-21    2015-08-31
2                0  2010-04-16  2016-04-16       2010-04-16    2015-04-17
3                0  2010-03-30  2016-03-30       2010-03-30    2015-03-31
4              526  2010-01-13  2016-03-07       2010-01-13    2015-03-09

   forecast_cons_12m  forecast_cons_year  forecast_discount_energy  \
0               0.00                   0                       0.0
1             189.95                   0                       0.0
2              47.96                   0                       0.0
3             240.04                   0                       0.0
4             445.75                 526                       0.0

   forecast_meter_rent_12m  forecast_price_energy_off_peak  \
0                     1.78                        0.114481
1                    16.27                        0.145711
2                    38.72                        0.165794
3                    19.83                        0.146694
4                   131.73                        0.116900

   forecast_price_energy_peak  forecast_price_pow_off_peak  has_gas  imp_cons  \
0                    0.098142                    40.606701        t      0.00
1                    0.000000                    44.311378        f      0.00
2                    0.087899                    44.311378        f      0.00
```

```
3                    0.000000                    44.311378        f      0.00
4                    0.100015                    40.606701        f     52.32

   margin_gross_pow_ele  margin_net_pow_ele  nb_prod_act  net_margin  \
0                 25.44               25.44            2  678.99
1                 16.38               16.38            1       18.89
2                 28.60               28.60            1        6.60
3                 30.22               30.22            1       25.46
4                 44.91               44.91            1       47.98

   num_years_antig  origin_up  pow_max  churn  contract_start_year  \
0                3          4   43.648      1                 2013
1                6          2   13.800      0                 2009
2                6          2   13.856      0                 2010
3                6          2   13.200      0                 2010
4                6          2   19.800      0                 2010

   contract_end_year
0               2016
1               2016
2               2016
3               2016
4               2016
```

[ ]:
```
# Assuming client_df is your DataFrame
specific_columns = ['id', 'channel_sales', 'date_activ', 'date_end',␣
↪'date_modif_prod', 'date_renewal',␣
↪'origin_up','contract_start_year','contract_end_year']
client_df_specific = client_df[specific_columns].head()
print(client_df_specific)
```

```
                                id  channel_sales  date_activ    date_end  \
0  24011ae4ebbe3035111d65fa7c15bc57              4  2013-06-15  2016-06-15
1   d29c2c54acc38ff3c0614d0a653813dd              0  2009-08-21  2016-08-30
2  764c75f661154dac3a6c254cd082ea7d              4  2010-04-16  2016-04-16
3  bba03439a292a1e166f80264c16191cb              5  2010-03-30  2016-03-30
4  149d57cf92fc41cf94415803a877cb4b              0  2010-01-13  2016-03-07

  date_modif_prod  date_renewal  origin_up  contract_start_year  \
0      2015-11-01    2015-06-23          4                 2013
1      2009-08-21    2015-08-31          2                 2009
2      2010-04-16    2015-04-17          2                 2010
3      2010-03-30    2015-03-31          2                 2010
4      2010-01-13    2015-03-09          2                 2010

   contract_end_year
0               2016
```

```
1                2016
2                2016
3                2016
4                2016
```

[ ]:
```
price_df.head()
```

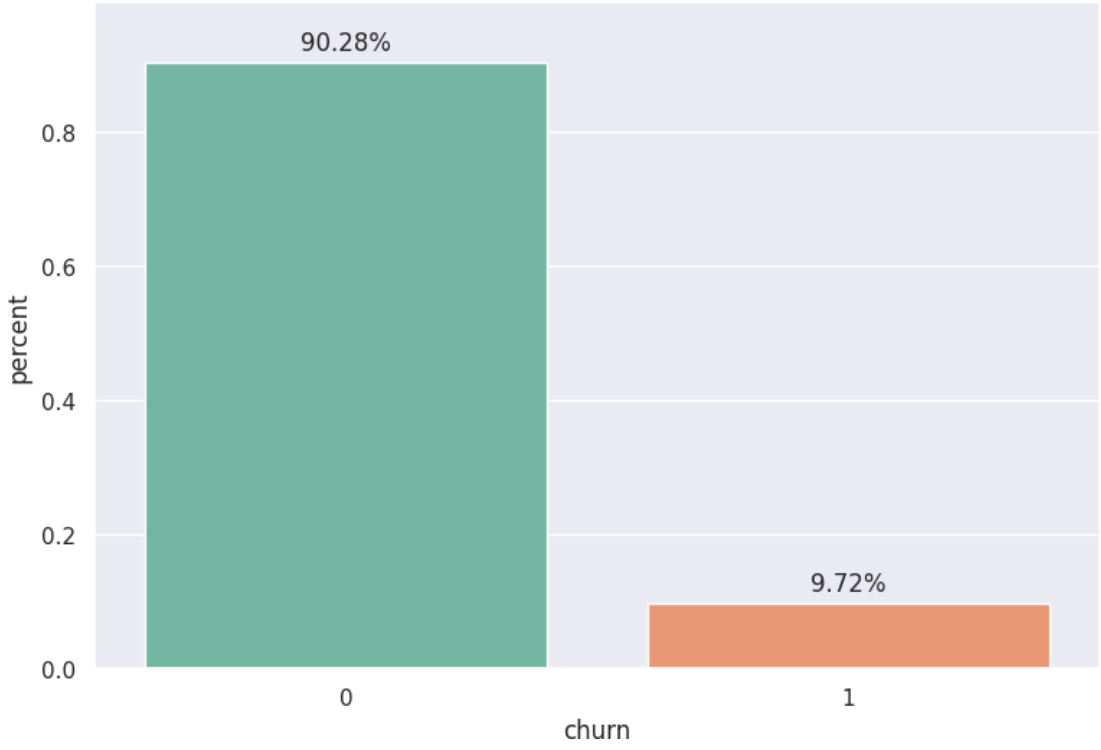
[ ]:
```
                                 id  price_date   price_off_peak_var    \
0  038af19179925da21a25619c5a24b745 2015-01-01             0.151367
1  038af19179925da21a25619c5a24b745 2015-02-01             0.151367
2  038af19179925da21a25619c5a24b745 2015-03-01             0.151367
3  038af19179925da21a25619c5a24b745 2015-04-01             0.149626
4  038af19179925da21a25619c5a24b745 2015-05-01             0.149626

   price_peak_var   price_mid_peak_var  price_off_peak_fix    price_peak_fix    \
0             0.0                  0.0           44.266931               0.0
1             0.0                  0.0           44.266931               0.0
2             0.0                  0.0           44.266931               0.0
3             0.0                  0.0           44.266931               0.0
4             0.0                  0.0           44.266931               0.0

   price_mid_peak_fix
0                 0.0
1                 0.0
2                 0.0
3                 0.0
4                 0.0
```

Data visualization
Import packages

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

%matplotlib inline

# Set plot style
sns.set(color_codes=True)
```

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

def plot_bar_chart_with_percent_label(df, target_variable, if_sort=False,␣
 ↪gap_label_bar=0, figsize=(9,6)):
    # prepare data
    plot_data = df[[target_variable]].value_counts().reset_index(name='count')
    plot_data['percent'] = plot_data['count'] / plot_data['count'].sum()
```

```
    if if_sort:
        x_order = plot_data.sort_values(by=['percent'],␣
↪ascending=False)[target_variable]
    else:
        x_order = plot_data.sort_values(by=[target_variable],␣
↪ascending=True)[target_variable]
    # plot
    fig = plt.figure(figsize=figsize)
    ax = sns.barplot(data=plot_data, x=target_variable, y='percent',␣
↪hue=target_variable, order=x_order, palette='Set2', legend=False) # Change␣
↪palette here
    # add label
    for p in ax.patches:
        x = p.get_x() + p.get_width()/2
        y = p.get_height() + gap_label_bar
        ax.annotate(text='{:.2f}%'.format(p.get_height()*100), xy=(x, y),␣
↪ha='center')
    ax.margins(y=0.1)
    plt.show()

# Example usage
# Assuming client_df and 'churn' are defined
plot_bar_chart_with_percent_label(df=client_df, target_variable='churn',␣
↪gap_label_bar=0.02)
```

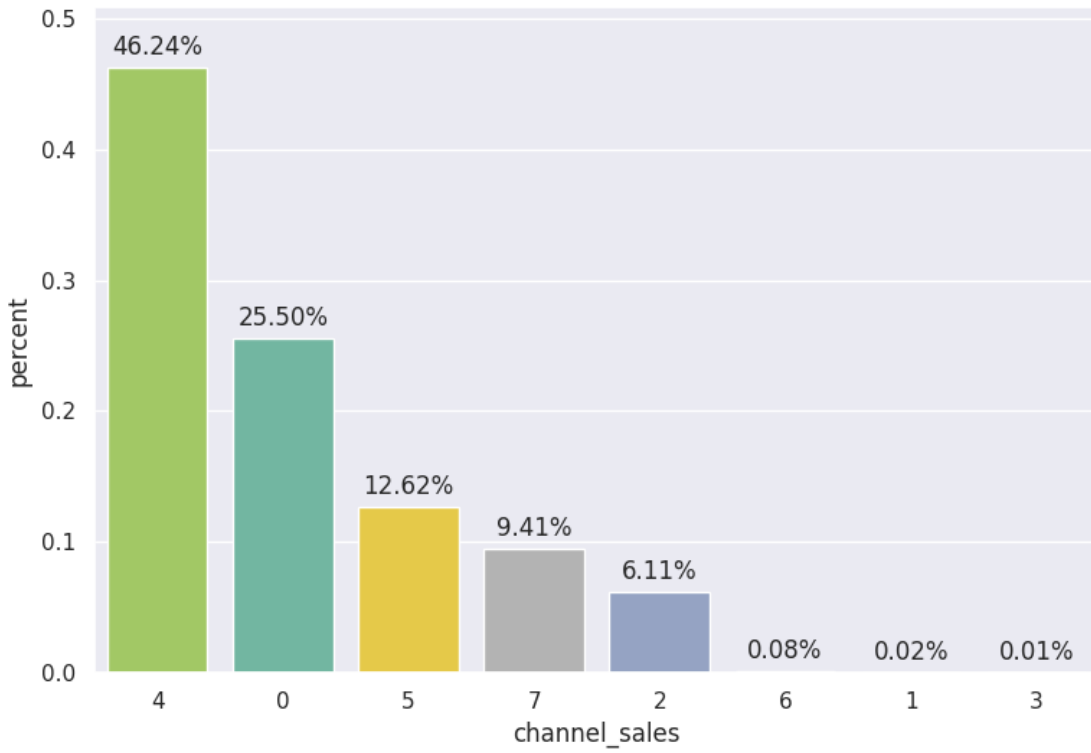


Around 9.72% of the customers change their providers.

Let's check categorical features' distribution. (Note: Discrete numeric features actually can be seen as categorial features representing the categories.)

channel_sales = code of the sales channel

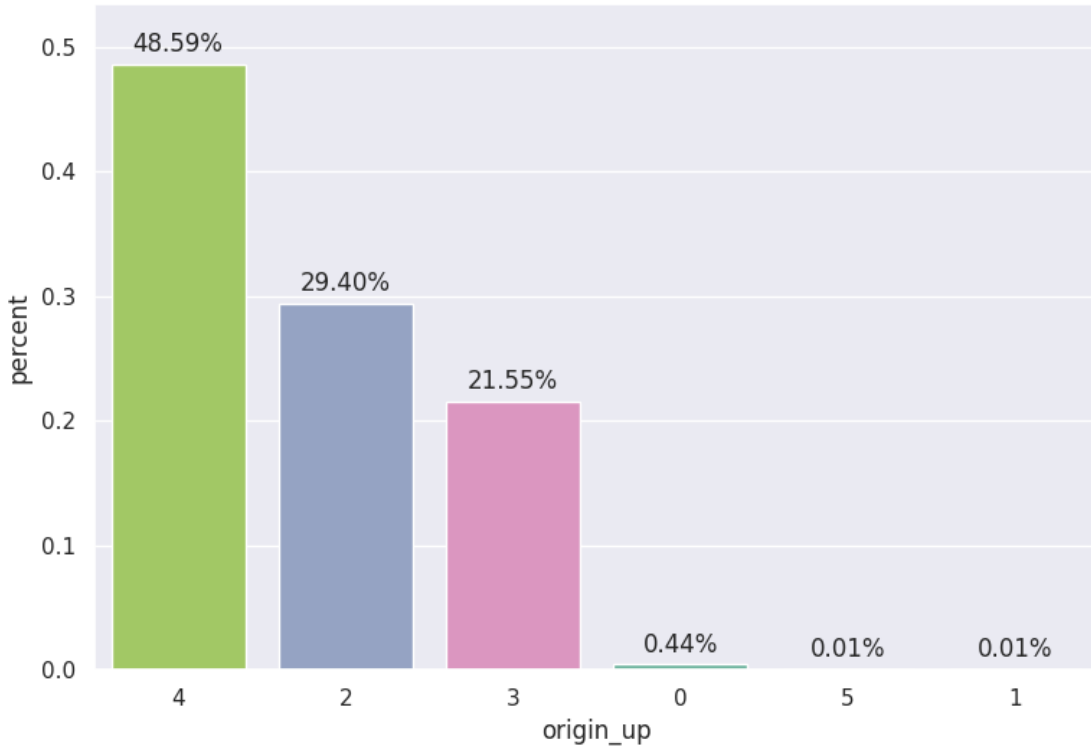
[ ]:
```
plot_bar_chart_with_percent_label(df=client_df,␣
↪target_variable='channel_sales', if_sort=True, gap_label_bar=0.01)
```



Type 4 is the most common sales channel. Type 1, 3 and 6 sales channels are very rare.

origin_up = code of the electricity campaign the customer first subscribed to

[ ]:
```
plot_bar_chart_with_percent_label(df=client_df, target_variable='origin_up',␣
↪if_sort=True, gap_label_bar=0.01)
```



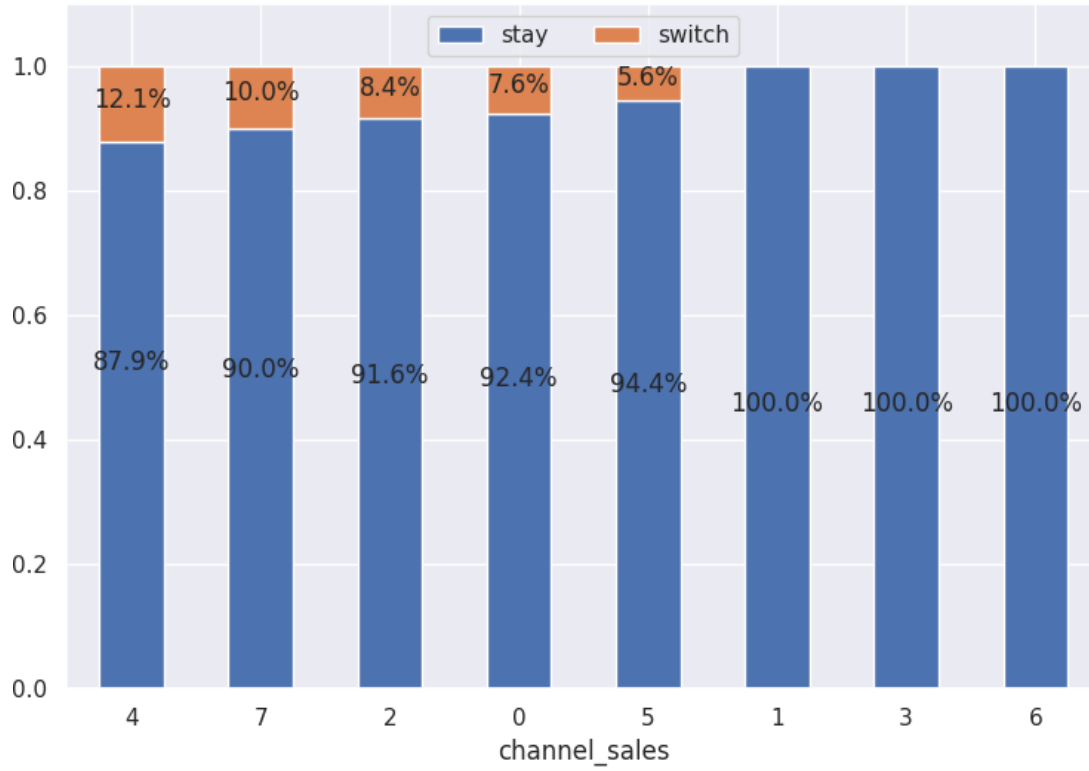
Type 4 is the most common electricty campaign that customers first subscribed to. Very few customers first subscribed to Type 0, 1 and 5 electricity campaign.

[ ]:
```
# plot stacked bar chart with percent label by churn
def plot_stacked_bar_with_percent_label_by_churn(df, target_variable,␣
↪if_sort=False, figsize=(9,6)):
    plot_data = df.groupby([target_variable,'churn']).size().
↪reset_index(name='count')
    plot_data = plot_data.pivot(index=target_variable, columns='churn',␣
↪values='count').fillna(0)
    plot_data = plot_data.div(plot_data.sum(axis=1), axis=0)
    if if_sort:
        plot_data = plot_data.sort_values(by=[0])
    ax = plot_data.plot(kind='bar', rot=False, stacked=True, figsize=figsize)
    plt.legend(['stay', 'switch'], loc='upper center', ncol=2)
    # add label
    for p in ax.patches:
        percentage = '{:.1f}%'.format(p.get_height()*100)
        if percentage!='0.0%':
            x = p.get_x() + p.get_width()/2
            y = 1 - p.get_height()/1.8
            ax.annotate(text=percentage, xy=(x, y), ha='center')
    ax.margins(y=0.1)
```

Plot stacked bar charts to show the feature distribution about the churning customers and non-churning customers.

channel_sales = code of the sales channel

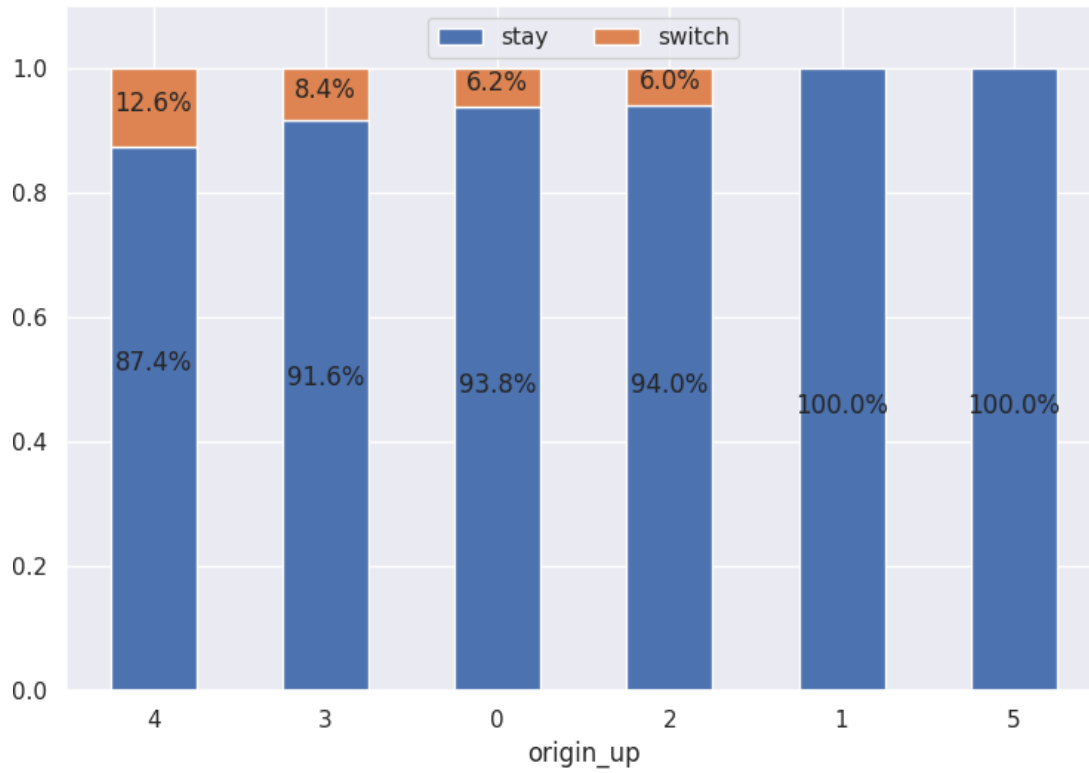
```
[ ]: plot_stacked_bar_with_percent_label_by_churn(df=client_df,␣
     ↪target_variable='channel_sales', if_sort=True)
```



For the customers from different sales channels, the proportion of churning customers ranges from 5.6% to 12.1% (excluding rare channels).

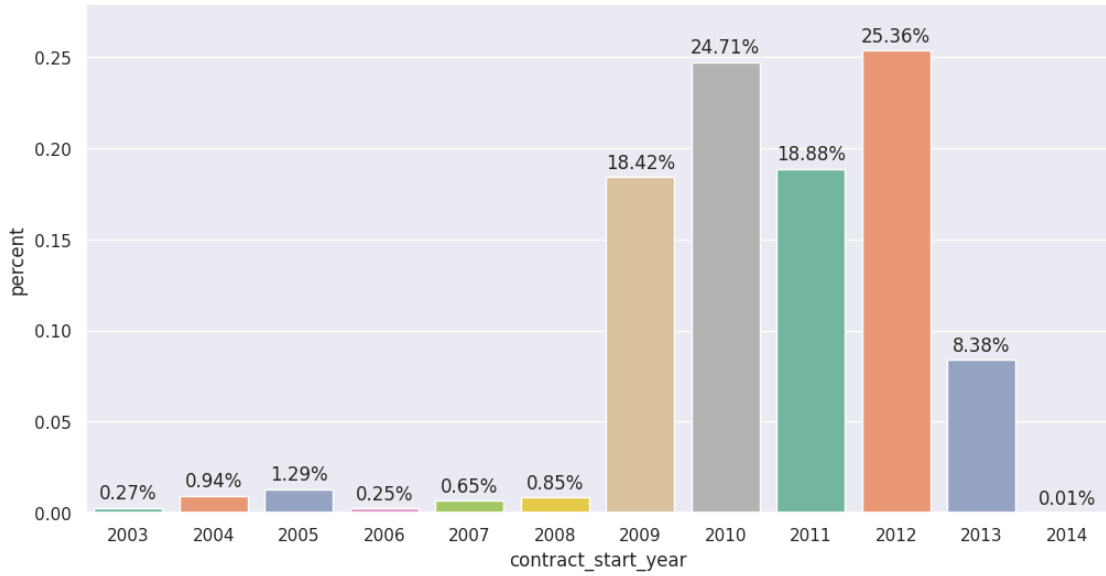
origin_up = code of the electricity campaign the customer first subscribed to

```
[ ]: plot_stacked_bar_with_percent_label_by_churn(df=client_df,␣
     ↪target_variable='origin_up', if_sort=True)
```



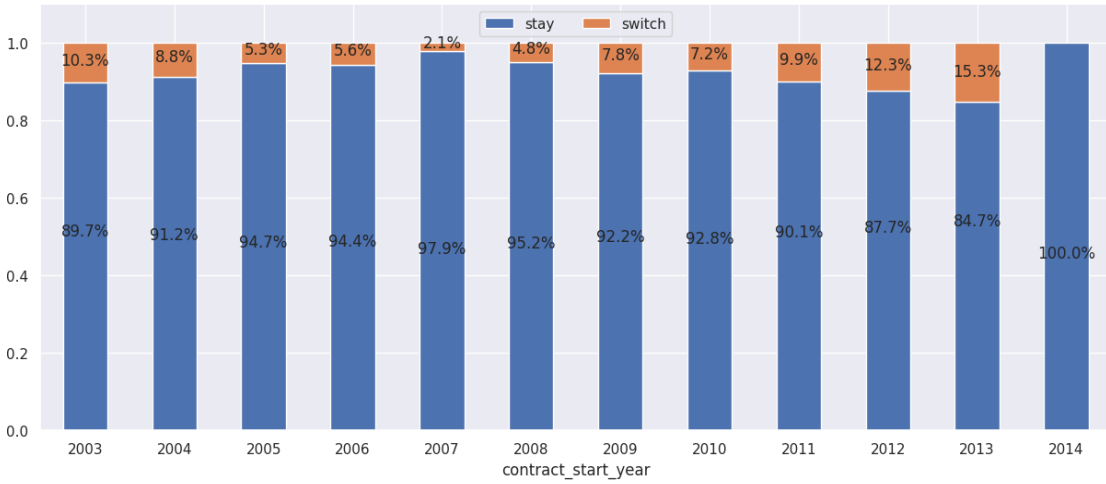
For the customers who first subscribed to different electricity campaigns, the proportion of churning customers ranges from 6.0% to 12.6% (excluding type 1 and type 5).

[ ]:
```
plot_bar_chart_with_percent_label(df=client_df,␣
↪target_variable='contract_start_year', if_sort=False, gap_label_bar=0.005,␣
↪figsize=(12,6))
```



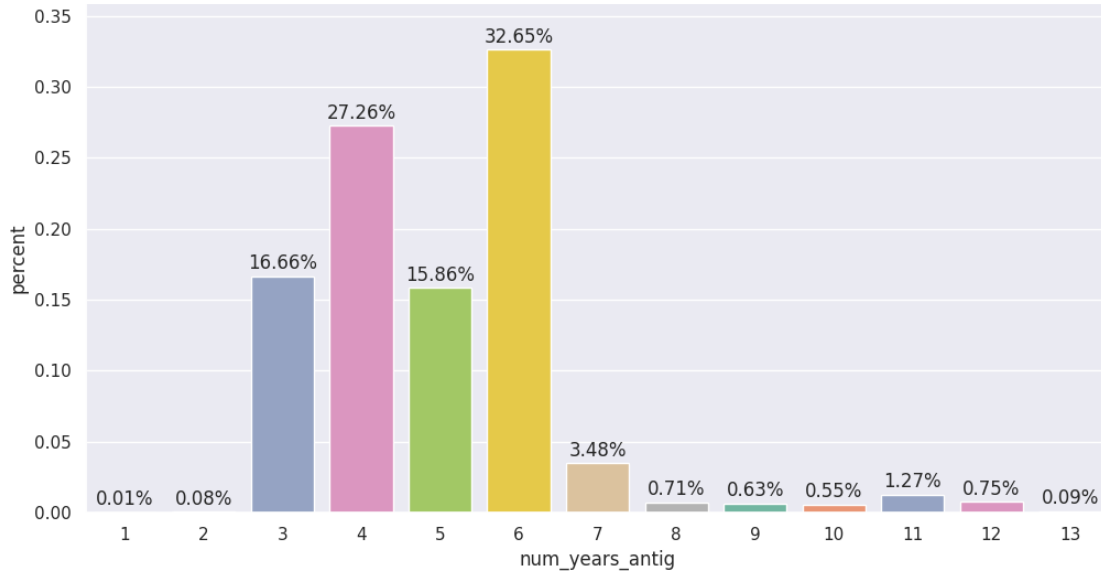
Most cutomers started their contracts from 2009 to 2013. Among these cutomers, it seems newer customers are more likely to switch to other providers.

```
[ ]: plot_stacked_bar_with_percent_label_by_churn(df=client_df,␣
     ↪target_variable='contract_start_year', if_sort=False, figsize=(15,6))
```



num_years_antig = antiquity of the client (in number of years)

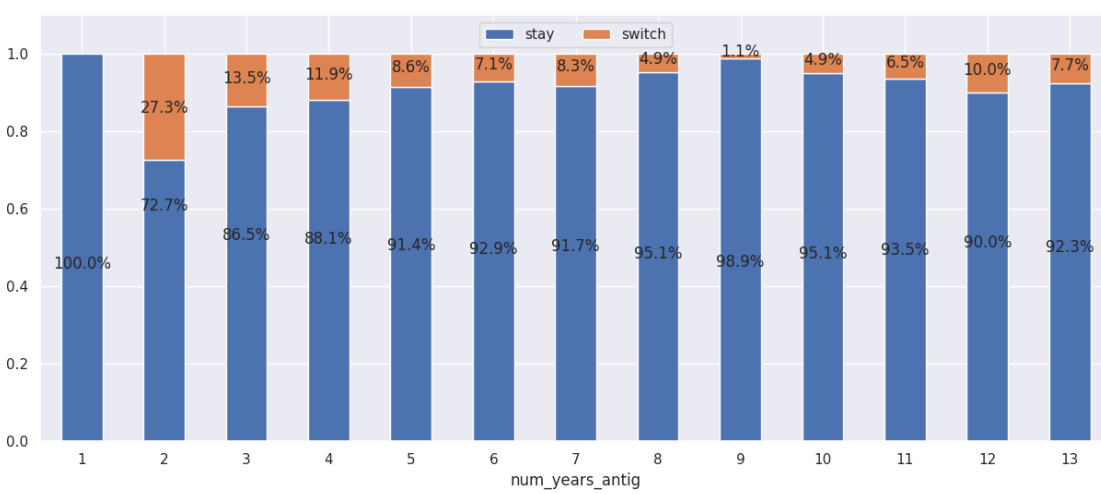
```
[ ]: plot_bar_chart_with_percent_label(df=client_df,␣
     ↪target_variable='num_years_antig', if_sort=False, gap_label_bar=0.005,␣
     ↪figsize=(12,6))
```



Most cutomers stayed with the current providers for 3 to 6 years. Among these customers, it seems newer customers are more likely to switch to other providers.

num_years_antig = antiquity of the client (in number of years)

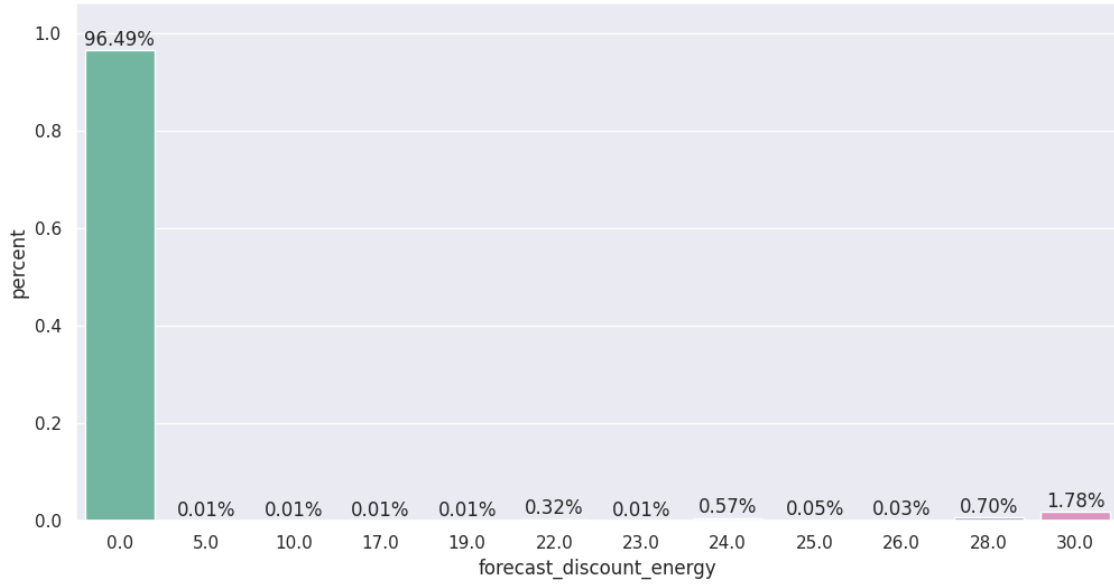
[ ]:
```
plot_stacked_bar_with_percent_label_by_churn(df=client_df,␣
↪target_variable='num_years_antig', if_sort=False, figsize=(15,6))
```



forecast_discount_energy = forecasted value of current discount

[ ]:

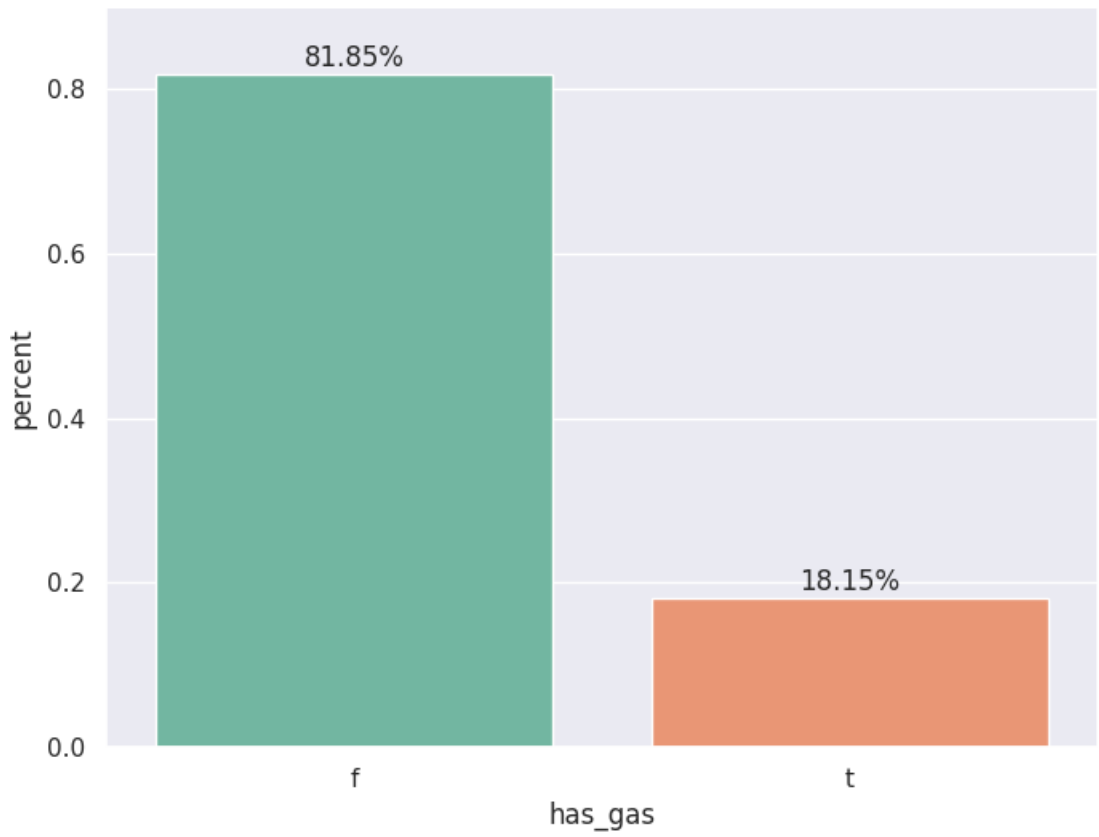
```
plot_bar_chart_with_percent_label(df=client_df,␣
↪target_variable='forecast_discount_energy', if_sort=False, gap_label_bar=0.
↪01, figsize=(12,6))
```



The vast majority of cutomers' forecasted value of current discount is 0.

has_gas = indicated if client is also a gas client

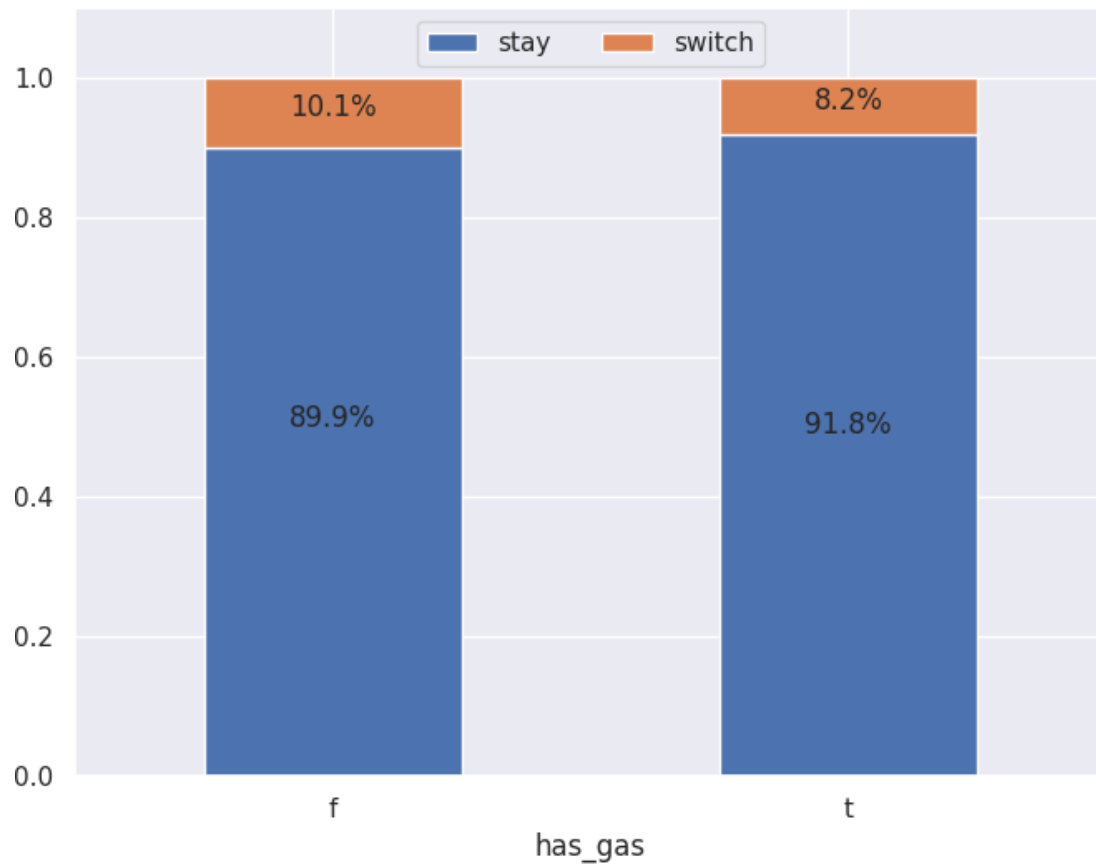
[ ]:
```
plot_bar_chart_with_percent_label(df=client_df, target_variable='has_gas',␣
↪if_sort=False, gap_label_bar=0.01, figsize=(8,6))
```



Only 18.15% of the clients were also gas clients.

has_gas = indicated if client is also a gas client

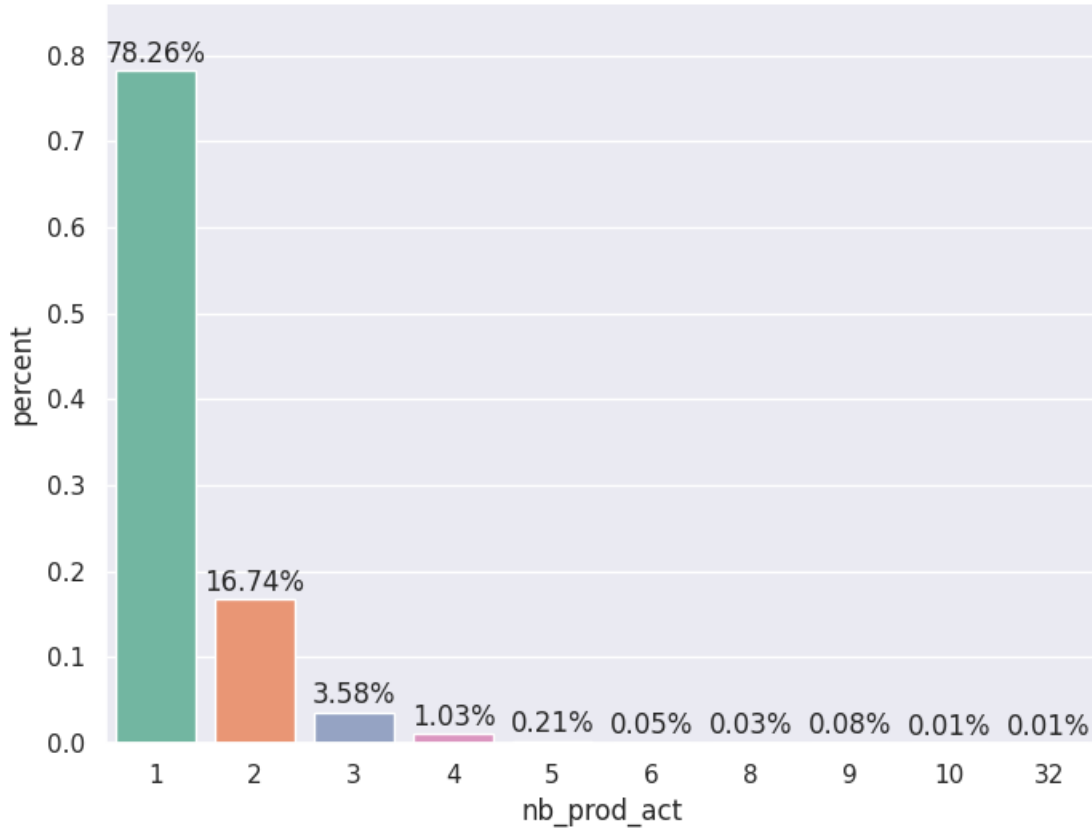
[ ]:
```
plot_stacked_bar_with_percent_label_by_churn(df=client_df,␣
↪target_variable='has_gas', if_sort=False, figsize=(8,6))
```



eLctricity clients are more likely to churn than electricity and gas clients.

nb_prod_act = number of active products and services

[ ]:
```
plot_bar_chart_with_percent_label(df=client_df, target_variable='nb_prod_act',␣
↪if_sort=False, gap_label_bar=0.01, figsize=(8,6))
```



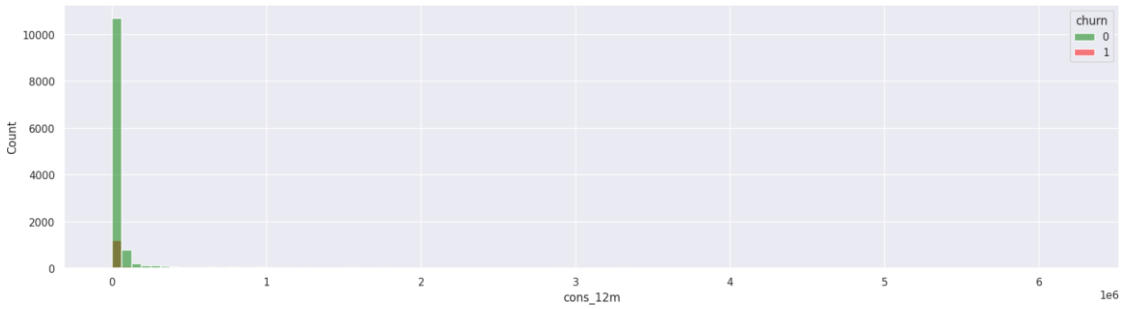
Most customers only had 1 or 2 active products and services.

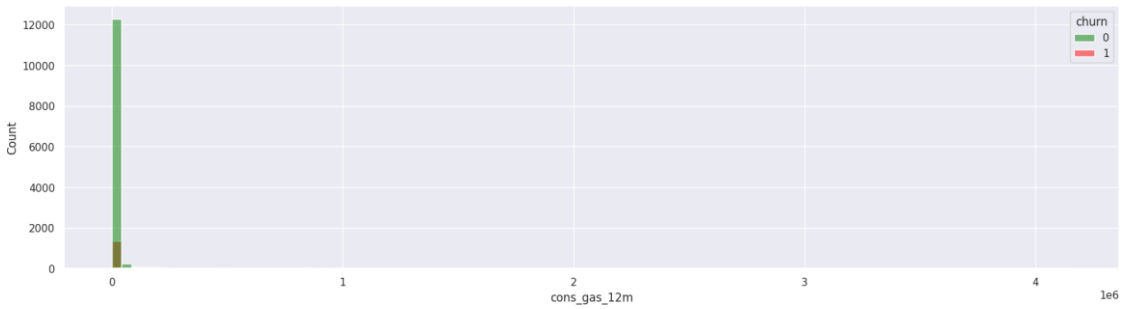
Plot histograms the distribution of continous numeric features.

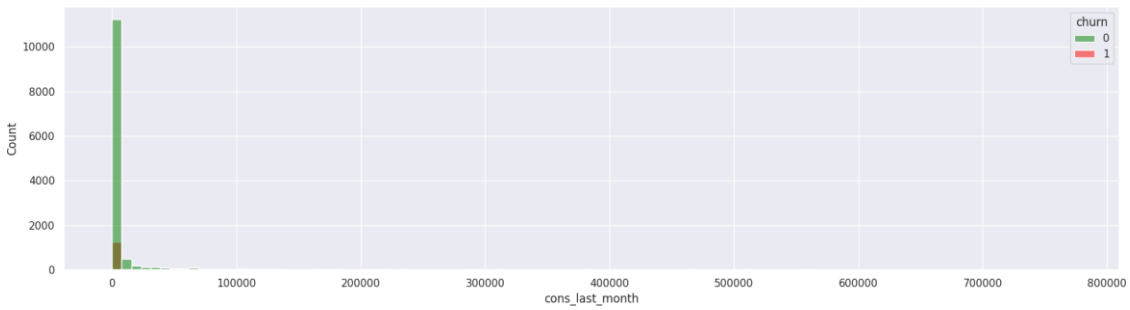
[ ]:
```
# plot histogram
def plot_histogram_by_churn(df, target_variable, figsize=(20,5), bins=100,␣
↪if_churn=True):
    fig = plt.figure(figsize=figsize)
    if if_churn:
        ax = sns.histplot(data=df, x=target_variable, bins=bins, hue='churn',␣
↪palette={0: 'green', 1: 'red'})
    else:
        ax = sns.histplot(data=df, x=target_variable, bins=bins)
    plt.show()
```

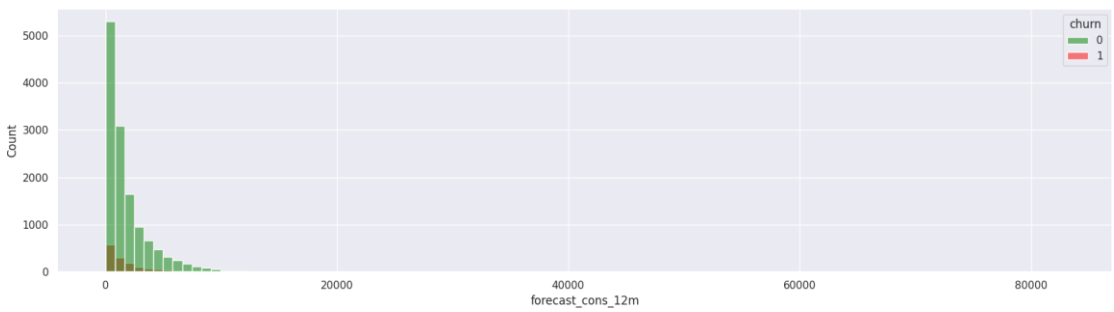
cons_12m = electricity consumption of the past 12 months
cons_gas_12m = gas consumption of the past 12 months
cons_last_month = electricity consumption of the last month
forecast_cons_12m = forecasted electricity consumption for next 12 months
forecast_cons_year = forecasted electricity consumption for the next calendar year

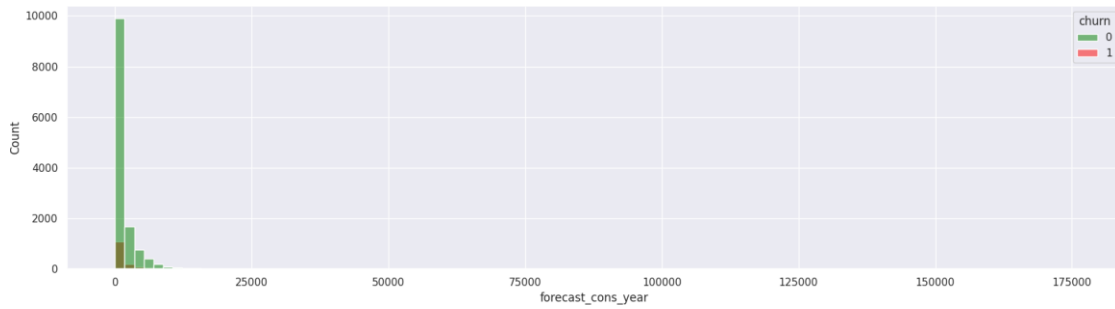
```
[ ]: for attr in ['cons_12m', 'cons_gas_12m', 'cons_last_month','forecast_cons_12m',␣
     ↪'forecast_cons_year']:
         plot_histogram_by_churn(df=client_df, target_variable=attr)
```











Plot histograms of energy consuption features. Each distribution has very long tail, which can be seen as the outliers.

Let's view price dataset

[ ]:
```
price_df['id'].value_counts().value_counts()
```

[ ]:
```
count
12    15990
11       83
10       11
9         6
8         3
7         3
Name: count, dtype: int64
```

We can see not all customers have the same number of price records. The vast majority of customers have 12 price reocrds and very few customers has less than 12 records.

[ ]:
```
price_df['price_date'].value_counts()
```

[ ]:
```
price_date
2015-08-01    16094
2015-12-01    16094
2015-07-01    16090
2015-11-01    16087
2015-06-01    16085
2015-10-01    16085
2015-02-01    16082
2015-09-01    16082
2015-05-01    16080
2015-04-01    16079
2015-03-01    16074
2015-01-01    16070
```

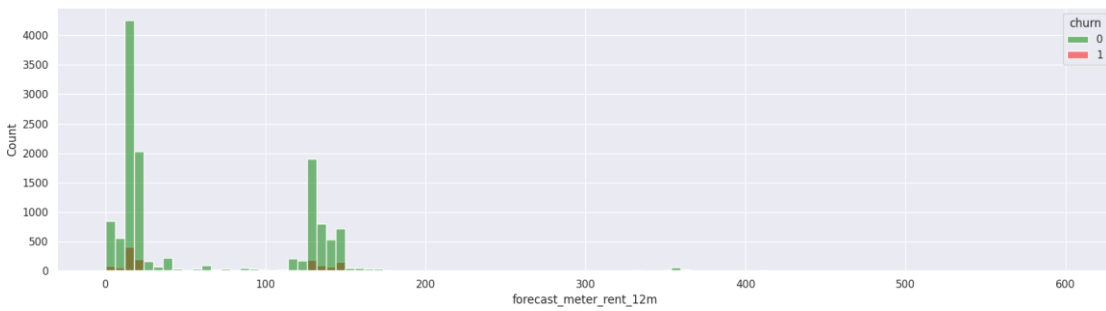
```
Name: count, dtype: int64
```

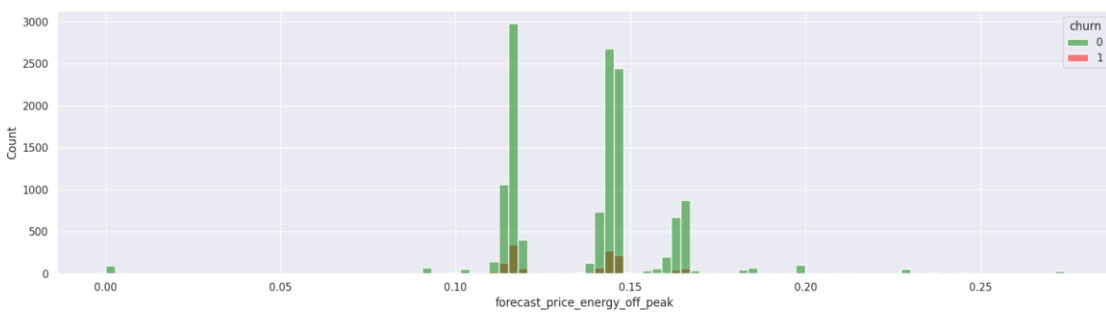
Combined with the above findings, most customers have a 12-month price records each. Luckily, every customer has the last-month's reocrd (2015-12-01), this will be helpful in identifying price sensitivity.

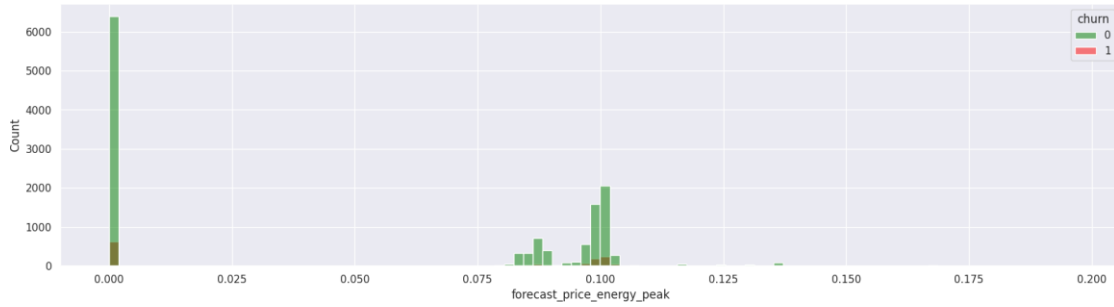
Plot histograms of price features. The values of each feature are centred around some points.

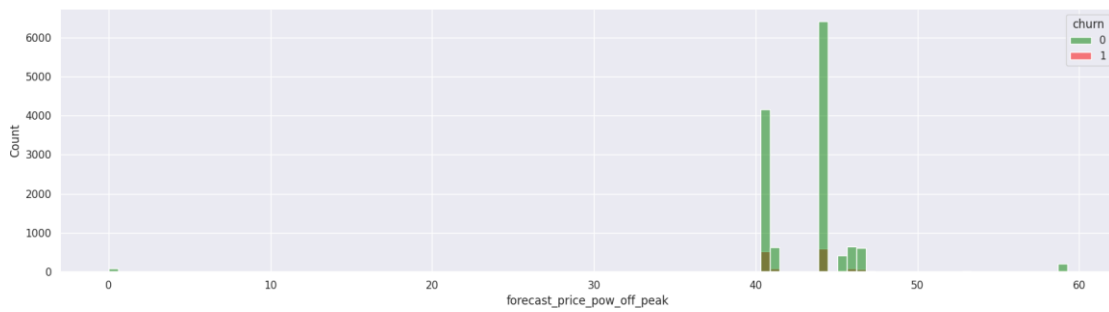
forecast_meter_rent_12m = forecasted bill of meter rental for the next 2 months
forecast_price_energy_off_peak = forecasted energy price for 1st period (off peak)
forecast_price_energy_peak = forecasted energy price for 2nd period (peak)
forecast_price_pow_off_peak = forecasted power price for 1st period (off peak)

[ ]:
```
for attr in ['forecast_meter_rent_12m', 'forecast_price_energy_off_peak',␣
↪'forecast_price_energy_peak', 'forecast_price_pow_off_peak',]:
    plot_histogram_by_churn(df=client_df, target_variable=attr)
```





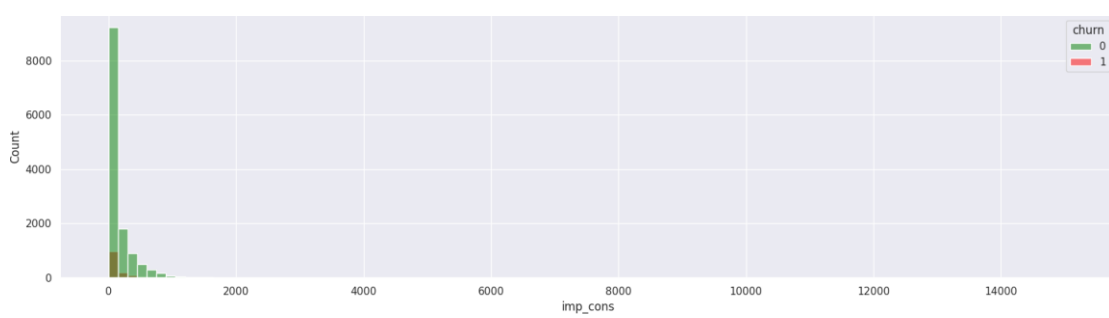


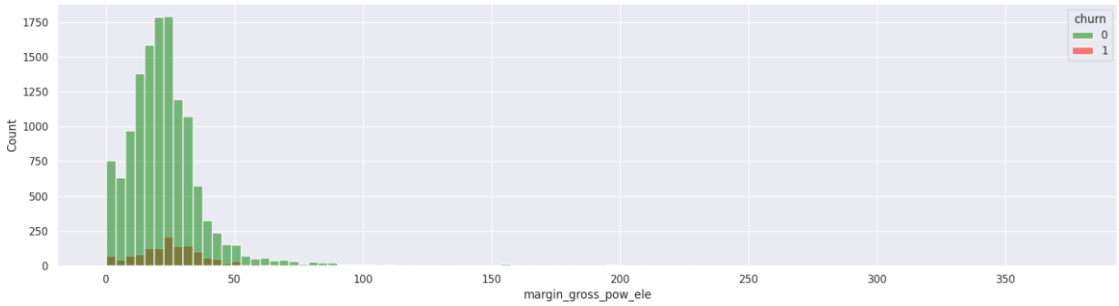


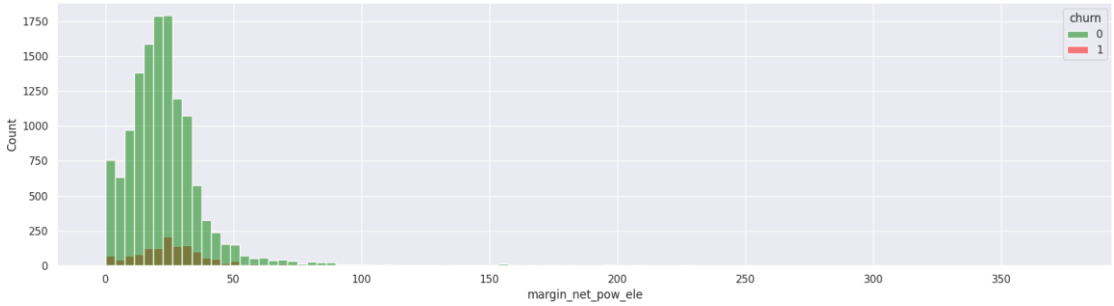
Other features also display significantly skewed distribution.

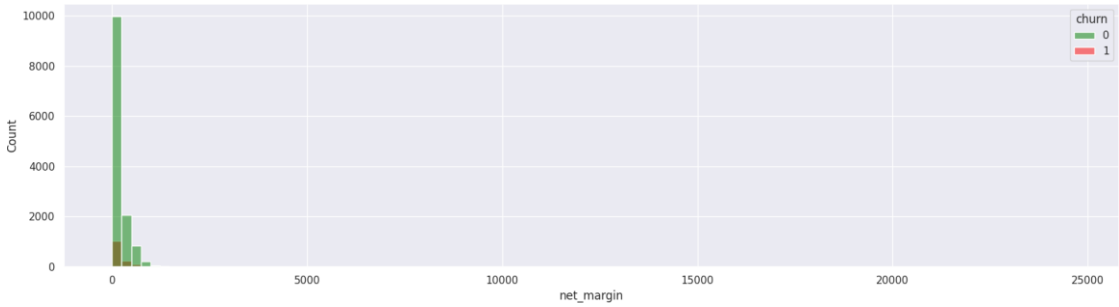
imp_cons = current paid consumption
margin_gross_pow_ele = gross margin on power subscription
margin_net_pow_ele = net margin on power subscription
pow_max = subscribed power

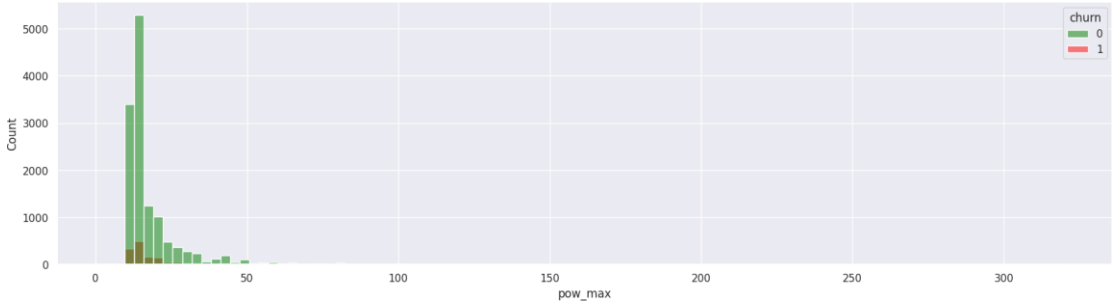
[ ]:
```
for attr in ['imp_cons', 'margin_gross_pow_ele',␣
↪'margin_net_pow_ele','net_margin','pow_max',]:
    plot_histogram_by_churn(df=client_df, target_variable=attr)
```









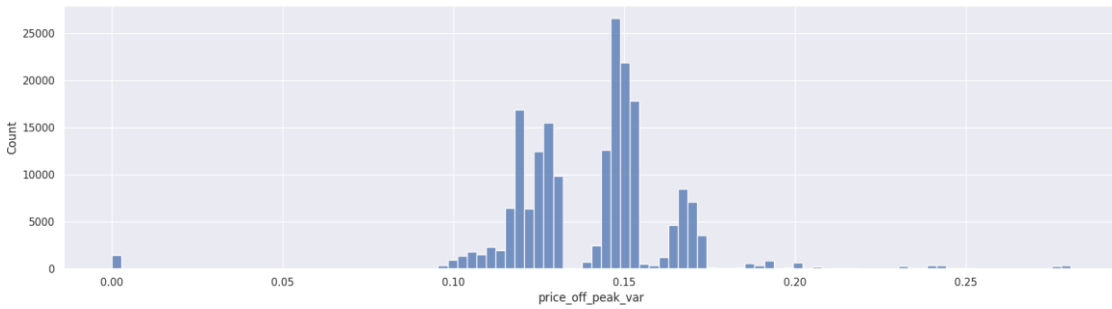


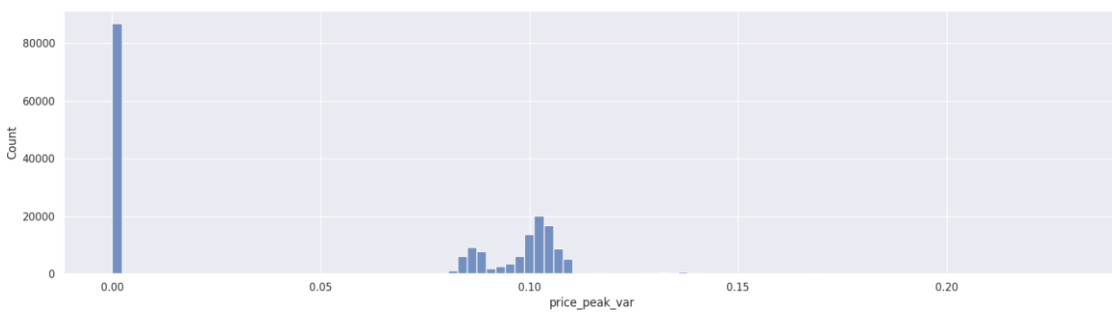
Let's have a look at distribution of price dataset. There are lots of 0 peak prices and mid-peak

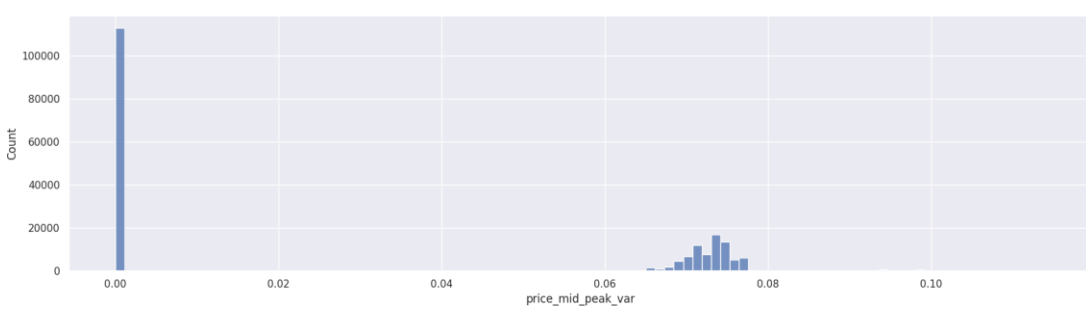
prices, but very few 0 off peak prices. So, I will analyse price sensivity only based on off-peak prices.

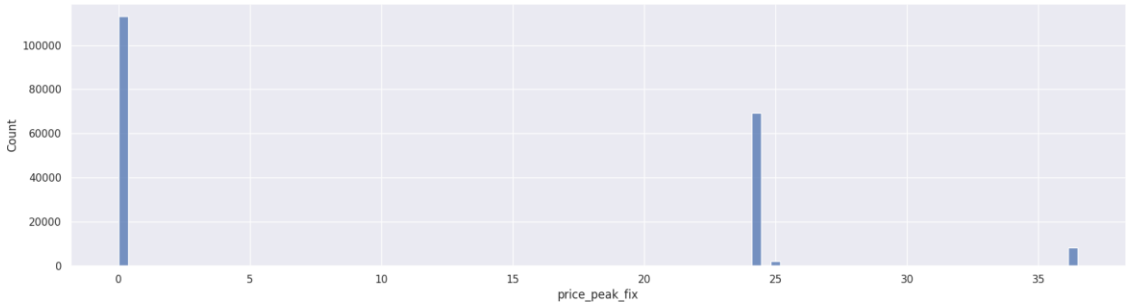
price_off_peak_var = price of energy for the 1st period (off peak)
price_peak_var = price of energy for the 2nd period (peak)
price_mid_peak_var = price of energy for the 3rd period (mid peak)
price_off_peak_fix = price of power for the 1st period (off peak)
price_peak_fix = price of power for the 2nd period (peak)
price_mid_peak_fix = price of power for the 3rd period (mid peak)

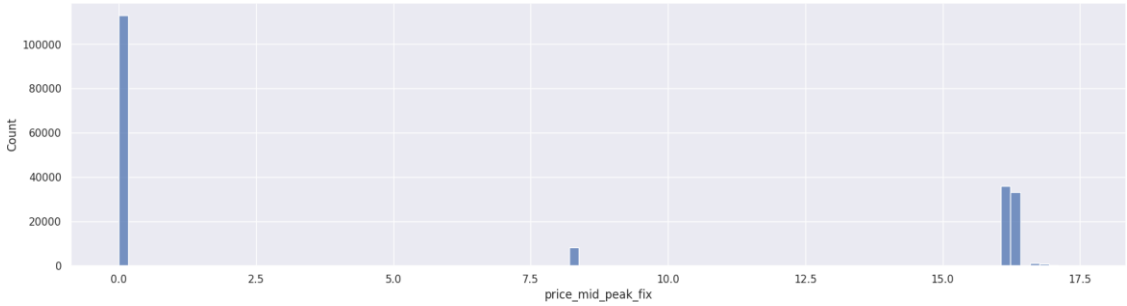
[ ]:
```
for attr in ['price_off_peak_var', 'price_peak_var', 'price_mid_peak_var',␣
↪'price_off_peak_fix', 'price_peak_fix', 'price_mid_peak_fix']:
    plot_histogram_by_churn(df=price_df, target_variable=attr, if_churn=False)
```











Now, let's check customers sensitivity to price. (Note: Normally, assuming other factors do not change, when the price increases and some cutomers switch, then we can say these customers are sensitive to price as the increase in prices lead the customer churn. But, if there is a decrease or no change in the price and cutomers still change their proveiders, I think in this case we cannot say these cutomers are sensitive to price.)

Calculate maximum, minimum and average prices over the whole year

price_off_peak_var = price of energy for the 1st period (off peak)
price_peak_var = price of energy for the 2nd period (peak)
price_mid_peak_var = price of energy for the 3rd period (mid peak)
price_off_peak_fix = price of power for the 1st period (off peak)
price_peak_fix = price of power for the 2nd period (peak)
price_mid_peak_fix = price of power for the 3rd period (mid peak)

[ ]:
```
stat_ = ['max', 'min', 'mean']
price_attr =␣
↪['price_off_peak_var','price_peak_var','price_mid_peak_var','price_off_peak_fix','price_pea
price_stat = price_df.drop(columns=['price_date']).groupby(['id']).
↪agg({'price_off_peak_var': stat_, 'price_peak_var': stat_,␣
↪'price_mid_peak_var': stat_,'price_off_peak_fix': stat_, 'price_peak_fix':␣
↪stat_, 'price_mid_peak_fix': stat_,})
# flatten the column names
price_stat.columns = ['_'.join(x) for x in zip(price_stat.columns.
↪get_level_values(0), price_stat.columns.get_level_values(1))]
price_stat = price_stat.reset_index()
price_stat.head(3)
```

[ ]:
```
                                id  price_off_peak_var_max  \
0  0002203ffbb812588b632b9e628cc38d                0.128067
1  0004351ebdd665e6ee664792efc4fd13                0.148405
2  0010bcc39e42b3c2131ed2ce55246e3c                0.205742

   price_off_peak_var_min  price_off_peak_var_mean  price_peak_var_max  \
0                0.119906                 0.124338            0.105842
1                0.143943                 0.146426            0.000000
2                0.150837                 0.181558            0.000000

   price_peak_var_min  price_peak_var_mean  price_mid_peak_var_max  \
0            0.101673             0.103794                0.073773
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_var_min  price_mid_peak_var_mean  price_off_peak_fix_max  \
0                0.070232                  0.07316               40.728885
1                0.000000                  0.00000               44.444710
2                0.000000                  0.00000               45.944710

   price_off_peak_fix_min  price_off_peak_fix_mean  price_peak_fix_max  \
0               40.565969                40.701732            24.43733
1               44.266931                44.385450             0.00000
2               44.444710                45.319710             0.00000

   price_peak_fix_min  price_peak_fix_mean  price_mid_peak_fix_max  \
0           24.339581            24.421038               16.291555
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_fix_min  price_mid_peak_fix_mean
0               16.226389                16.280694
1                0.000000                 0.000000
2                0.000000                 0.000000
```

Merge client dataset to the above dataset to add churn data

[ ]:
```
# add churn values
price_stat = price_stat.merge(client_df[['id','churn']], on=['id'], how='left')
# drop ids that are not inclued in the price dataset
price_stat = price_stat.dropna(subset=['churn']).reset_index(drop=True)
price_stat.head(3)
```

[ ]:
```
                                id  price_off_peak_var_max  \
0  0002203ffbb812588b632b9e628cc38d                0.128067
1  0004351ebdd665e6ee664792efc4fd13                0.148405
2  0010bcc39e42b3c2131ed2ce55246e3c                0.205742

   price_off_peak_var_min  price_off_peak_var_mean  price_peak_var_max  \
0                0.119906                 0.124338            0.105842
1                0.143943                 0.146426            0.000000
2                0.150837                 0.181558            0.000000

   price_peak_var_min  price_peak_var_mean  price_mid_peak_var_max  \
0            0.101673             0.103794                0.073773
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_var_min  price_mid_peak_var_mean  price_off_peak_fix_max  \
0                0.070232                  0.07316               40.728885
1                0.000000                  0.00000               44.444710
2                0.000000                  0.00000               45.944710

   price_off_peak_fix_min  price_off_peak_fix_mean  price_peak_fix_max  \
0               40.565969                40.701732           24.43733
1               44.266931                44.385450            0.00000
2               44.444710                45.319710            0.00000

   price_peak_fix_min  price_peak_fix_mean  price_mid_peak_fix_max  \
0           24.339581            24.421038               16.291555
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_fix_min  price_mid_peak_fix_mean  churn
0               16.226389                16.280694    0.0
1                0.000000                 0.000000    0.0
2                0.000000                 0.000000    0.0
```

Calculate the biggest changes in prices over the whole year.

[ ]:
```
for attr in price_attr:
    price_stat[f'diff_max_min_{attr}'] = price_stat[f'{attr}_max'] -␣
↪price_stat[f'{attr}_min']
```

```
price_stat.head(3)
```

[ ]:

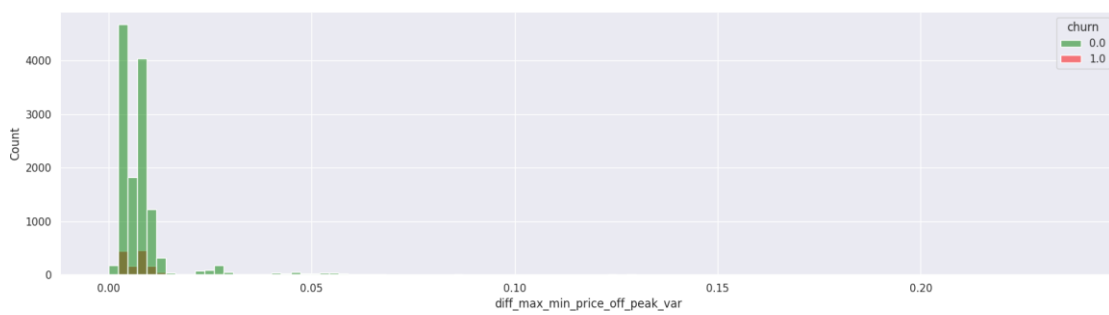
```
                                id  price_off_peak_var_max  \
0  0002203ffbb812588b632b9e628cc38d                0.128067
1  0004351ebdd665e6ee664792efc4fd13                0.148405
2  0010bcc39e42b3c2131ed2ce55246e3c                0.205742

   price_off_peak_var_min  price_off_peak_var_mean  price_peak_var_max  \
0                0.119906                 0.124338            0.105842
1                0.143943                 0.146426            0.000000
2                0.150837                 0.181558            0.000000

   price_peak_var_min  price_peak_var_mean  price_mid_peak_var_max  \
0            0.101673             0.103794                0.073773
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_var_min  price_mid_peak_var_mean  price_off_peak_fix_max  \
0                0.070232                  0.07316               40.728885
1                0.000000                  0.00000               44.444710
2                0.000000                  0.00000               45.944710

   price_off_peak_fix_min  price_off_peak_fix_mean  price_peak_fix_max  \
0               40.565969                40.701732           24.43733
1               44.266931                44.385450            0.00000
2               44.444710                45.319710            0.00000

   price_peak_fix_min  price_peak_fix_mean  price_mid_peak_fix_max  \
0           24.339581            24.421038               16.291555
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_fix_min  price_mid_peak_fix_mean   churn  \
0               16.226389                16.280694     0.0
1                0.000000                 0.000000     0.0
2                0.000000                 0.000000     0.0

   diff_max_min_price_off_peak_var  diff_max_min_price_peak_var  \
0                         0.008161                     0.004169
1                         0.004462                     0.000000
2                         0.054905                     0.000000

   diff_max_min_price_mid_peak_var  diff_max_min_price_off_peak_fix  \
0                         0.003541                         0.162916
1                         0.000000                         0.177779
2                         0.000000                         1.500000
```

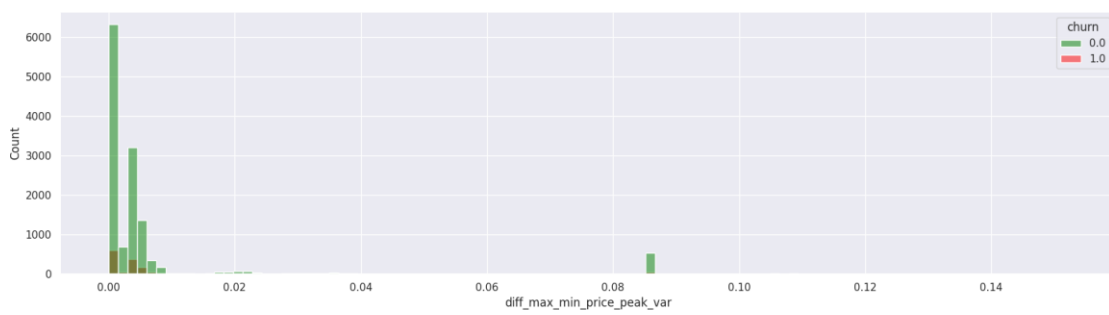
	diff_max_min_price_peak_fix	diff_max_min_price_mid_peak_fix
0	0.097749	0.065166
1	0.000000	0.000000
2	0.000000	0.000000

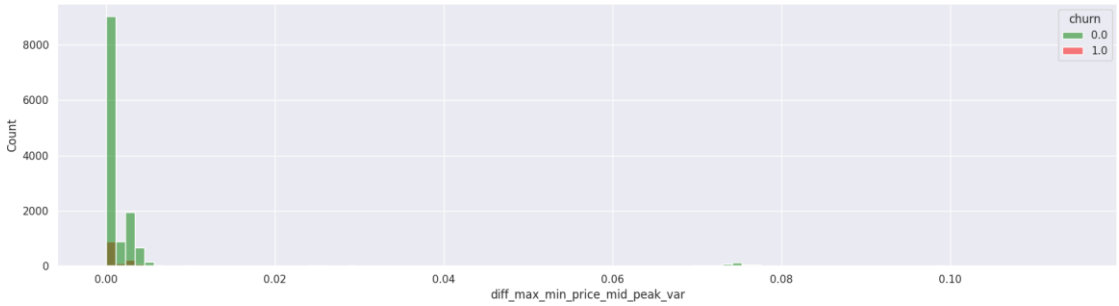
Most cutomers' prices did not change much and kept stable.

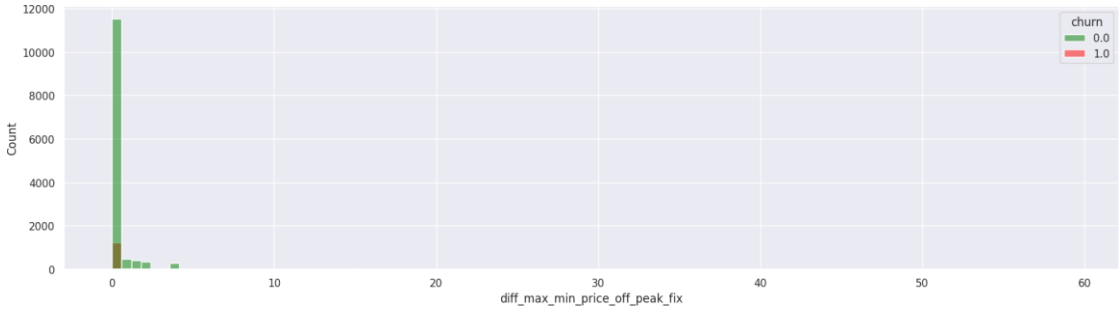
price_off_peak_var = price of energy for the 1st period (off peak)
price_peak_var = price of energy for the 2nd period (peak)
price_mid_peak_var = price of energy for the 3rd period (mid peak)
price_off_peak_fix = price of power for the 1st period (off peak)
price_peak_fix = price of power for the 2nd period (peak)
price_mid_peak_fix = price of power for the 3rd period (mid peak)

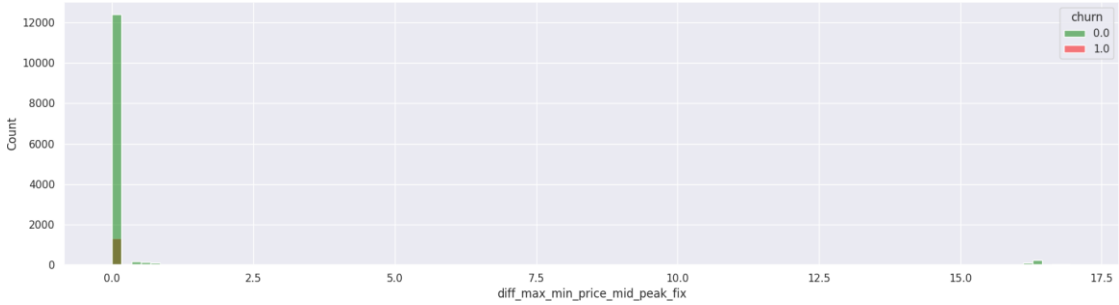
[ ]:
```
for attr in price_attr:
    plot_histogram_by_churn(df=price_stat,␣
↪target_variable=f"diff_max_min_{attr}", if_churn=True)
```











Calculate the difference between the price last month (2015-12-01) and mean price.

[ ]:
```
# add diff
for attr in price_attr:
    price_stat[f"diff_Dec_mean_{attr}"] = price_df[price_df['id'].
 ↪isin(price_stat['id'])].groupby(['id'])[attr].nth(-1).values -␣
 ↪price_stat[f"{attr}_mean"]
#
price_stat.head(3)
```

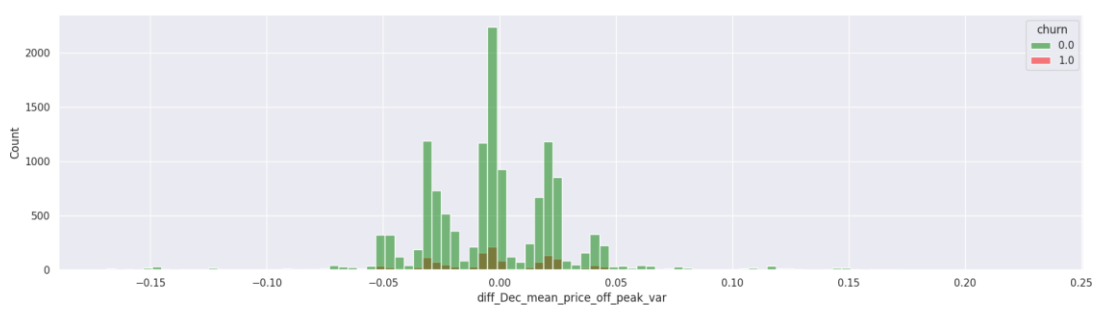
[ ]:
```
                                id  price_off_peak_var_max  \
0  0002203ffbb812588b632b9e628cc38d                0.128067
1  0004351ebdd665e6ee664792efc4fd13                0.148405
2  0010bcc39e42b3c2131ed2ce55246e3c                0.205742

   price_off_peak_var_min  price_off_peak_var_mean  price_peak_var_max  \
0                0.119906                 0.124338            0.105842
1                0.143943                 0.146426            0.000000
2                0.150837                 0.181558            0.000000

   price_peak_var_min  price_peak_var_mean  price_mid_peak_var_max  \
0            0.101673             0.103794                0.073773
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_var_min  price_mid_peak_var_mean  price_off_peak_fix_max  \
0                0.070232                  0.07316               40.728885
1                0.000000                  0.00000               44.444710
2                0.000000                  0.00000               45.944710

   price_off_peak_fix_min  price_off_peak_fix_mean  price_peak_fix_max  \
0               40.565969                40.701732            24.43733
1               44.266931                44.385450             0.00000
2               44.444710                45.319710             0.00000

   price_peak_fix_min  price_peak_fix_mean  price_mid_peak_fix_max  \
0           24.339581            24.421038               16.291555
1            0.000000             0.000000                0.000000
2            0.000000             0.000000                0.000000

   price_mid_peak_fix_min  price_mid_peak_fix_mean   churn  \
0               16.226389                16.280694     0.0
1                0.000000                 0.000000     0.0
2                0.000000                 0.000000     0.0

   diff_max_min_price_off_peak_var  diff_max_min_price_peak_var  \
0                         0.008161                     0.004169
1                         0.004462                     0.000000
2                         0.054905                     0.000000
```

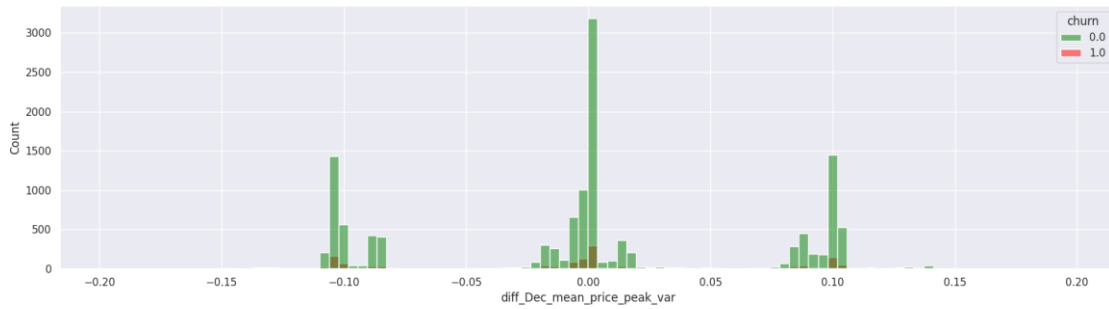
```
   diff_max_min_price_mid_peak_var  diff_max_min_price_off_peak_fix    \
0                         0.003541                         0.162916
1                         0.000000                         0.177779
2                         0.000000                         1.500000

   diff_max_min_price_peak_fix   diff_max_min_price_mid_peak_fix  \
0                     0.097749                         0.065166
1                     0.000000                         0.000000
2                     0.000000                         0.000000

   diff_Dec_mean_price_off_peak_var   diff_Dec_mean_price_peak_var  \
0                          0.021521                     -0.103794
1                         -0.027671                      0.101071
2                         -0.037615                      0.000000

   diff_Dec_mean_price_mid_peak_var    diff_Dec_mean_price_off_peak_fix   \
0                         -0.073160                          3.742978
1                          0.075096                         -3.656565
2                          0.000000                         -0.875000

   diff_Dec_mean_price_peak_fix   diff_Dec_mean_price_mid_peak_fix
0                    -24.421038                        -16.280694
1                     24.437330                         16.291555
2                      0.000000                          0.000000
```

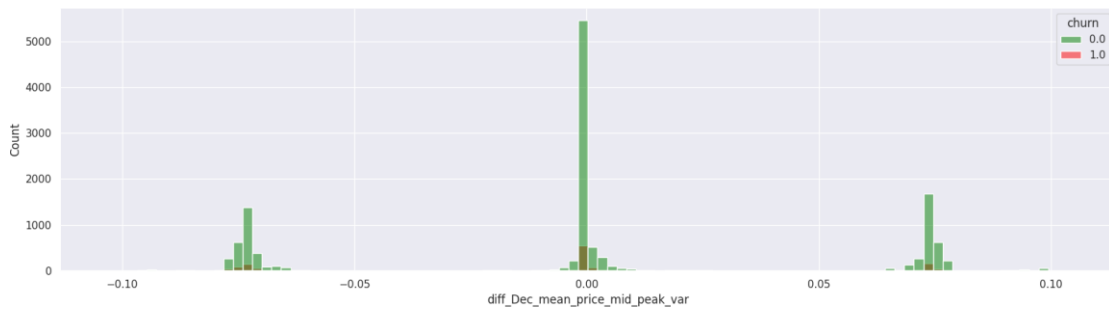
Most cutomers' prices last month were very close to the mean prices over the whole year.

price_off_peak_var = price of energy for the 1st period (off peak)
price_peak_var = price of energy for the 2nd period (peak)
price_mid_peak_var = price of energy for the 3rd period (mid peak)
price_off_peak_fix = price of power for the 1st period (off peak)
price_peak_fix = price of power for the 2nd period (peak)
price_mid_peak_fix = price of power for the 3rd period (mid peak)

[ ]:
```
for attr in price_attr:
    plot_histogram_by_churn(df=price_stat,␣
↪target_variable=f"diff_Dec_mean_{attr}", if_churn=True)
```







Check whether changes in prices would affect customer churn (only considering off peak prices)

```
[ ]: plot_bar_chart_with_percent_label(df=price_stat[price_stat['diff_Dec_mean_price_off_peak_var']
     ↪target_variable='churn', gap_label_bar=0.02)
```

Among the cutomers whose off-peak enery prices increased, around 10.35% of customers changed their providers.

```
[ ]: plot_bar_chart_with_percent_label(df=price_stat[price_stat['diff_Dec_mean_price_off_peak_var']
     ↪target_variable='churn', gap_label_bar=0.02)
```

Among the cutomers whose off-peak energy prices did not increase (decreased or did not change), also 9.34% of customers changed their providers.

```
[ ]: plot_bar_chart_with_percent_label(df=price_stat[price_stat['diff_Dec_mean_price_off_peak_fix']
     ↪target_variable='churn', gap_label_bar=0.02)
```

Among the cutomers whose off-peak power prices increased, around 9.81% of customers changed their providers.

```
[ ]: plot_bar_chart_with_percent_label(df=price_stat[price_stat['diff_Dec_mean_price_off_peak_fix']
     ↪target_variable='churn', gap_label_bar=0.02)
```

Among the cutomers whose off-peak enery prices did not increase (decreased or did not change), also 9.57% of customers changed their providers.

In order to confirm this conlusion, I continue to compare prices of December with average prices of the last 6 months and 3 months.

[ ]:
```
price_stat_6_month = price_df[price_df['price_date']>'2015-06-01'].
↪drop(columns=['price_date']).groupby(['id']).agg({'price_off_peak_var':␣
↪stat_, 'price_off_peak_fix': stat_,})
# flatten the column names
price_stat_6_month.columns = ['_'.join(x) for x in zip(price_stat_6_month.
↪columns.get_level_values(0), price_stat_6_month.columns.get_level_values(1))]
price_stat_6_month = price_stat_6_month.reset_index()
# add churn values
price_stat_6_month = price_stat_6_month.merge(client_df[['id','churn']],␣
↪on=['id'], how='left')
# drop ids that are not inclued in the price dataset
price_stat_6_month = price_stat_6_month.dropna(subset=['churn']).
↪reset_index(drop=True)
# add diff
for attr in ['price_off_peak_var','price_off_peak_fix']:
    price_stat_6_month[f'diff_Dec_mean_{attr}'] = price_df[price_df['id'].
↪isin(price_stat_6_month['id'])].groupby(['id'])[attr].nth(-1).values -␣
↪price_stat_6_month[f'{attr}_mean']
price_stat_6_month.head(5)
```

If only considering the last 6 months, changes in prices still did not affect customer churn.

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_6_month[price_stat_6_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_6_month[price_stat_6_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_6_month[price_stat_6_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_6_month[price_stat_6_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

[ ]:
```
price_stat_3_month = price_df[price_df['price_date']>'2015-09-01'].
↪drop(columns=['price_date']).groupby(['id']).agg({'price_off_peak_var':␣
↪stat_, 'price_off_peak_fix': stat_,})
# flatten the column names
price_stat_3_month.columns = ['_'.join(x) for x in zip(price_stat_3_month.
↪columns.get_level_values(0), price_stat_3_month.columns.get_level_values(1))]
price_stat_3_month = price_stat_3_month.reset_index()
# add churn values
```

```
price_stat_3_month = price_stat_3_month.merge(client_df[['id','churn']],␣
↪on=['id'], how='left')
# drop ids that are not inclued in the price dataset
price_stat_3_month = price_stat_3_month.dropna(subset=['churn']).
↪reset_index(drop=True)
# add diff
for attr in ['price_off_peak_var','price_off_peak_fix']:
    price_stat_3_month[f'diff_Dec_mean_{attr}'] = price_df[price_df['id'].
↪isin(price_stat_3_month['id'])].groupby(['id'])[attr].nth(-1).values -␣
↪price_stat_3_month[f'{attr}_mean']
price_stat_3_month.head(5)
```

If only considering the last 3 months, changes in prices did not affect customer churn either.

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_3_month[price_stat_3_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_3_month[price_stat_3_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_3_month[price_stat_3_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

[ ]:
```
plot_bar_chart_with_percent_label(df=price_stat_3_month[price_stat_3_month['diff_Dec_mean_pric
↪target_variable='churn', gap_label_bar=0.02)
```

From all results above, we can conclude customers were not sensitive to prices.

Key findings

1. About_____of customers changed providers.
2. Numeric variables on consumption are highly skewed.
3. Changes in prices does not affect customer churn.

[ ]:
```
# load packages
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, recall_score, accuracy_score,␣
↪confusion_matrix, ConfusionMatrixDisplay, classification_report
```

[ ]:
```
# drop useless columns
train_data = client_df.copy()
train_data['year_modif_prod'] = train_data['date_modif_prod'].dt.year
train_data['year_renewal'] = train_data['date_renewal'].dt.year
```

```
train_data = train_data.
↪drop(columns=['date_activ','date_end','date_modif_prod','date_renewal'])
#
has_gas_encoder = LabelEncoder()
train_data['has_gas'] = has_gas_encoder.fit_transform(train_data['has_gas'])
# add new feature "diff_dec_jan_off_peak_var" and "diff_dec_jan_off_peak_fix".␣
↪I just calculate the differnece between last price and first price for each␣
↪customer.
diff_dec_jan_off_peak_var = price_df.sort_values(by=['price_date']).
↪groupby(['id'])['price_off_peak_var'].max() - price_df.
↪sort_values(by=['price_date']).groupby(['id'])['price_off_peak_var'].min()
diff_dec_jan_off_peak_var = diff_dec_jan_off_peak_var.
↪reset_index(name='diff_dec_jan_off_peak_var')
diff_dec_jan_off_peak_fix = price_df.sort_values(by=['price_date']).
↪groupby(['id'])['price_off_peak_fix'].max() - price_df.
↪sort_values(by=['price_date']).groupby(['id'])['price_off_peak_fix'].min()
diff_dec_jan_off_peak_fix = diff_dec_jan_off_peak_fix.
↪reset_index(name='diff_dec_jan_off_peak_fix')
train_data = train_data.merge(diff_dec_jan_off_peak_var, on='id', how='left')
train_data = train_data.merge(diff_dec_jan_off_peak_fix, on='id', how='left')
# # also add the above differences of other prices
for attr in␣
↪['price_peak_var','price_peak_fix','price_mid_peak_var','price_mid_peak_fix']:
↪
    diff_dec_jan_temp = price_df.sort_values(by=['price_date']).
↪groupby(['id'])[attr].max() - price_df.sort_values(by=['price_date']).
↪groupby(['id'])[attr].min()
    diff_dec_jan_temp = diff_dec_jan_temp.
↪reset_index(name=f'diff_dec_jan_{attr}')
    train_data = train_data.merge(diff_dec_jan_temp, on='id', how='left')
# # add price changing trends
train_data = train_data.
↪merge(price_stat[['id','diff_Dec_mean_price_off_peak_var','diff_Dec_mean_price_off_peak_fix
↪'diff_Dec_mean_price_peak_var','diff_Dec_mean_price_peak_fix',
↪'diff_Dec_mean_price_mid_peak_var','diff_Dec_mean_price_mid_peak_fix',
                                        ]], on='id', how='left')

# #
train_data
```

Create training data and training label.

Note: for tree-based models

there is no need to perform data scaling, unlike other machine learning models.

do not apply one-hot encoding to processing categorical variables, just use label-encoding to convert categorical variables to numeric variables.

[ ]:
```
X = train_data.drop(columns=['id','churn'])
y = train_data['churn']
X.shape, y.shape
```

According to the task requirement, build a RandomForest model.

As there is no testing set, the "out-of-fold" cross validation strategy is used to evaluate the model's performace. Besides, the training set is a very imbalanced dataset, so I use precision and recall as the metrics, instead of accuracy (results of accuracy are still shown, but not for evaluating the model's performance). As the goal of this task is not finding the optimal parameters, I do not tune the model's parameters.

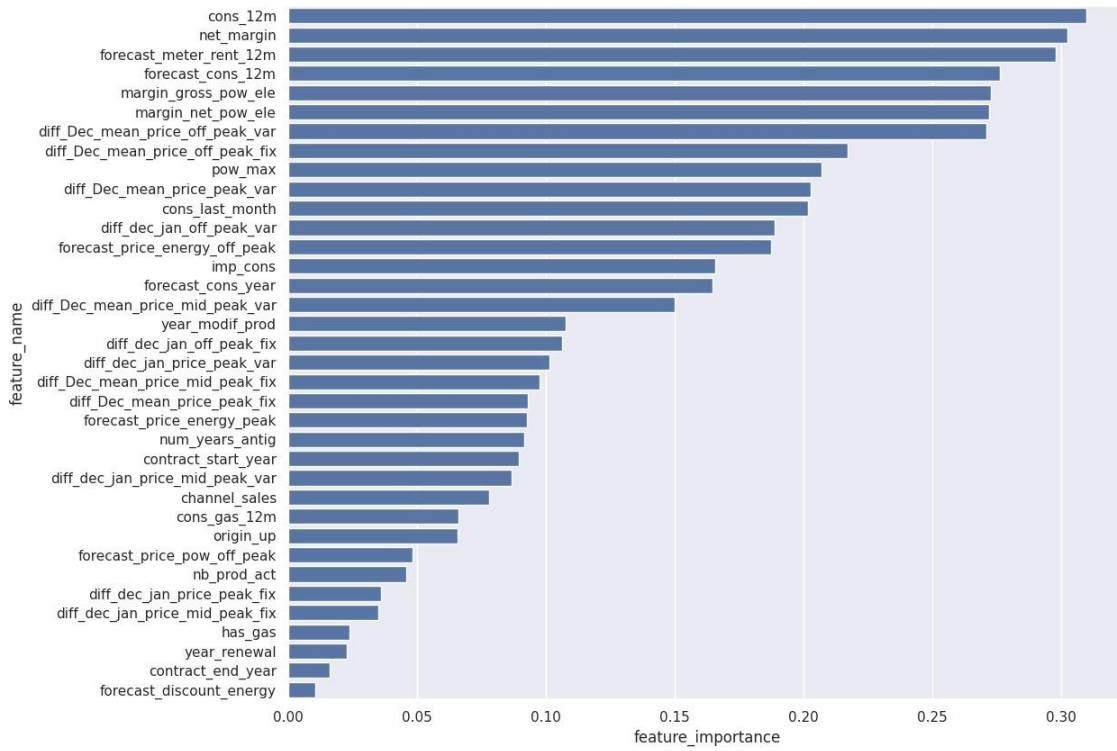
[ ]:
```
import numpy as np
```

[ ]:
```
pred_train_labels = np.zeros(shape=(X.shape[0], 2))    # pred training labels
feature_importance_df = pd.DataFrame(data={'feature_name':X.columns,␣
↪'feature_importance':[0]*len(X.columns)})
# create cv dataset
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=29)
fold_counter = 1
for train_index, test_index in kfold.split(X, y):
    X_train, X_test = X.loc[train_index], X.loc[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # build model
    rf = RandomForestClassifier(random_state=56)
    # train model
    rf.fit(X_train, y_train)
    pred_train_labels[test_index] = rf.predict_proba(X_test)
    feature_importance_df['feature_importance'] =␣
↪feature_importance_df['feature_importance'] + (rf.feature_importances_)
    print(f"Fold {fold_counter} Precision {precision_score(y_test, rf.
↪predict(X_test)):.3f} Recall {recall_score(y_test, rf.predict(X_test)):.3f}␣
↪Accuracy {accuracy_score(y_test, rf.predict(X_test)):.3f}")
    fold_counter = fold_counter + 1
# predicted labels
pred_y = pred_train_labels.argmax(axis=-1)
print(f"Total   Precision {precision_score(y, pred_y):.3f} Recall␣
↪{recall_score(y, pred_y):.3f} Accuracy {accuracy_score(y, pred_y):.3f}")
```

```
Fold 1 Precision 0.714 Recall 0.018 Accuracy 0.904
Fold 2 Precision 0.800 Recall 0.028 Accuracy 0.905
Fold 3 Precision 0.900 Recall 0.032 Accuracy 0.906
Fold 4 Precision 0.875 Recall 0.025 Accuracy 0.905
Fold 5 Precision 0.700 Recall 0.025 Accuracy 0.904
Total   Precision 0.800 Recall 0.025 Accuracy 0.905
```

The below figure displays how important each feature is when growing trees.

```
[ ]: fig = plt.figure(figsize=(12,10))
ax = sns.barplot(data=feature_importance_df.
 ↪sort_values(by=['feature_importance'], ascending=False), y='feature_name',␣
 ↪x='feature_importance')
```



The below table shows the main classification metrics.

```
[ ]: print(classification_report(y, pred_y))
```

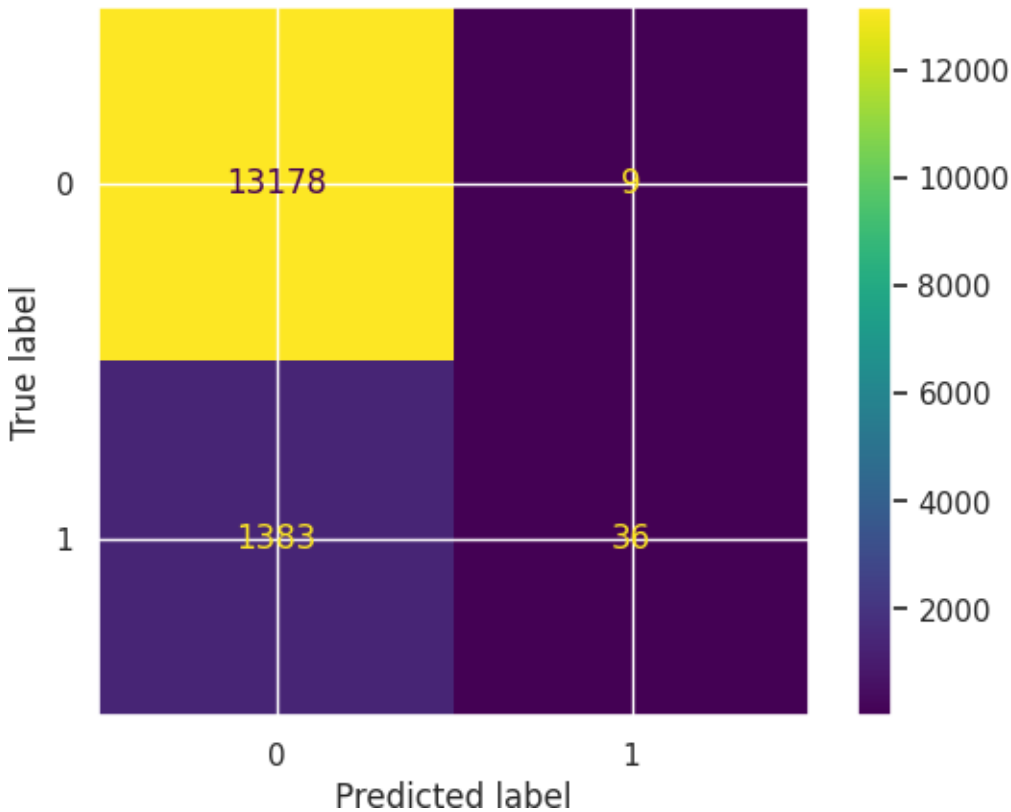
	precision	recall	f1-score	support
0	0.91	1.00	0.95	13187
1	0.80	0.03	0.05	1419
accuracy			0.90	14606
macro avg	0.85	0.51	0.50	14606
weighted avg	0.89	0.90	0.86	14606

The below figure shows the confusion matrix.

```
[ ]: cm = confusion_matrix(y, pred_y, labels=rf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=rf.classes_)
```

```
disp.plot()
```

[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x795e7782f6a0>



task: identify the impact of a 20% discount¶ This is a tricky problem.

first, intuitively, we have concluded that changes in prices do not have a significant impact on customer churn. So, a discount should be unable to prevent customer churn.
second, we cannot really identify whether a discount can prevent customer churn, as we do not exactly know what will happen in the future.
Therefore, I try to calculate the expected profit based on the probability of customer churn and evaluate the impact of the discount.

[ ]:
```
prob_no_discount = pd.DataFrame(data = pred_train_labels, columns=['0', '1'])
prob_no_discount['id'] = train_data['id']
prob_no_discount = prob_no_discount[['id','0','1']]
prob_no_discount
```

[ ]:

	id	0	1
0	24011ae4ebbe3035111d65fa7c15bc57	0.69	0.31

```
1       d29c2c54acc38ff3c0614d0a653813dd    0.96  0.04
2       764c75f661154dac3a6c254cd082ea7d    0.88  0.12
3       bba03439a292a1e166f80264c16191cb    0.97  0.03
4       149d57cf92fc41cf94415803a877cb4b    0.99  0.01
...                                    ...    ...   ...
14601   18463073fb097fc0ac5d3e040f356987    0.75  0.25
14602   d0a6f71671571ed83b2645d23af6de00    0.45  0.55
14603   10e6828ddd62cbcf687cb74928c4c2d2    0.81  0.19
14604   1cf20fd6206d7678d5bcafd28c53b4db    0.92  0.08
14605   563dde550fd624d7352f3de77c0cdfcd    0.87  0.13

[14606 rows x 3 columns]
```

We need to predcit the probability of churn when applying a 20% discount. Note: I still use the data on this year, not the next year.

[ ]:
```
# drop useless columns
test_data = client_df.copy()
test_data['year_modif_prod'] = test_data['date_modif_prod'].dt.year
test_data['year_renewal'] = test_data['date_renewal'].dt.year
test_data = test_data.
↪drop(columns=['date_activ','date_end','date_modif_prod','date_renewal'])
#
has_gas_encoder = LabelEncoder()
test_data['has_gas'] = has_gas_encoder.fit_transform(test_data['has_gas'])
# apply a 20% discount
diff_dec_jan_off_peak_var = price_df.sort_values(by=['price_date']).
↪groupby(['id'])['price_off_peak_var'].max()*0.8 - price_df.
↪sort_values(by=['price_date']).groupby(['id'])['price_off_peak_var'].min()*0.
↪8
diff_dec_jan_off_peak_var = diff_dec_jan_off_peak_var.
↪reset_index(name='diff_dec_jan_off_peak_var')
diff_dec_jan_off_peak_fix = price_df.sort_values(by=['price_date']).
↪groupby(['id'])['price_off_peak_fix'].max()*0.8 - price_df.
↪sort_values(by=['price_date']).groupby(['id'])['price_off_peak_fix'].min()*0.
↪8
diff_dec_jan_off_peak_fix = diff_dec_jan_off_peak_fix.
↪reset_index(name='diff_dec_jan_off_peak_fix')
test_data = test_data.merge(diff_dec_jan_off_peak_var, on='id', how='left')
test_data = test_data.merge(diff_dec_jan_off_peak_fix, on='id', how='left')
# # also add the above differences of other prices
for attr in␣
↪['price_peak_var','price_peak_fix','price_mid_peak_var','price_mid_peak_fix']:
↪
    diff_dec_jan_temp = price_df.sort_values(by=['price_date']).
↪groupby(['id'])[attr].max()*0.8 - price_df.sort_values(by=['price_date']).
↪groupby(['id'])[attr].min()*0.8
```

```
    diff_dec_jan_temp = diff_dec_jan_temp.
 ↪reset_index(name=f"diff_dec_jan_{attr}")
    test_data = test_data.merge(diff_dec_jan_temp, on='id', how='left')

# add price changing trends
price_attr =␣
 ↪['price_off_peak_var','price_peak_var','price_mid_peak_var','price_off_peak_fix','price_pea
price_stat_test = price_df.drop(columns=['price_date']).groupby(['id']).
 ↪agg({'price_off_peak_var': ['mean'], 'price_peak_var': ['mean'],␣
 ↪'price_mid_peak_var': ['mean'],'price_off_peak_fix': ['mean'],␣
 ↪'price_peak_fix': ['mean'], 'price_mid_peak_fix': ['mean'],})
# flatten the column names
price_stat_test.columns = ['_'.join(x) for x in zip(price_stat_test.columns.
 ↪get_level_values(0), price_stat_test.columns.get_level_values(1))]
price_stat_test = price_stat_test.reset_index()
# add diff
for attr in price_attr:
    price_stat_test[f"diff_Dec_mean_{attr}"] = price_df[price_df['id'].
 ↪isin(price_stat_test['id'])].groupby(['id'])[attr].nth(-1).values*0.8 -␣
 ↪price_stat_test[f"{attr}_mean"]*0.8
#
test_data = test_data.
 ↪merge(price_stat_test[['id','diff_Dec_mean_price_off_peak_var','diff_Dec_mean_price_off_pea

 ↪'diff_Dec_mean_price_peak_var','diff_Dec_mean_price_peak_fix',

 ↪'diff_Dec_mean_price_mid_peak_var','diff_Dec_mean_price_mid_peak_fix',
                                        ]], on='id', how='left')

# #
test_data.head()
```

[ ]:

```
                                id  channel_sales  cons_12m  cons_gas_12m  \
0  24011ae4ebbe3035111d65fa7c15bc57              4         0         54946
1  d29c2c54acc38ff3c0614d0a653813dd              0      4660             0
2  764c75f661154dac3a6c254cd082ea7d              4       544             0
3  bba03439a292a1e166f80264c16191cb              5      1584             0
4  149d57cf92fc41cf94415803a877cb4b              0      4425             0

   cons_last_month  forecast_cons_12m  forecast_cons_year  \
0                0               0.00                   0
1                0             189.95                   0
2                0              47.96                   0
3                0             240.04                   0
4              526             445.75                 526
```

	forecast_discount_energy	forecast_meter_rent_12m \
0	0.0	1.78
1	0.0	16.27
2	0.0	38.72
3	0.0	19.83
4	0.0	131.73

	forecast_price_energy_off_peak	forecast_price_energy_peak \
0	0.114481	0.098142
1	0.145711	0.000000
2	0.165794	0.087899
3	0.146694	0.000000
4	0.116900	0.100015

	forecast_price_pow_off_peak	has_gas	imp_cons	margin_gross_pow_ele \
0	40.606701	1	0.00	25.44
1	44.311378	0	0.00	16.38
2	44.311378	0	0.00	28.60
3	44.311378	0	0.00	30.22
4	40.606701	0	52.32	44.91

	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig	origin_up \
0	25.44	2	678.99	3	4
1	16.38	1	18.89	6	2
2	28.60	1	6.60	6	2
3	30.22	1	25.46	6	2
4	44.91	1	47.98	6	2

	pow_max	churn	contract_start_year	contract_end_year	year_modif_prod \
0	43.648	1	2013	2016	2015
1	13.800	0	2009	2016	2009
2	13.856	0	2010	2016	2010
3	13.200	0	2010	2016	2010
4	19.800	0	2010	2016	2010

	year_renewal	diff_dec_jan_off_peak_var	diff_dec_jan_off_peak_fix \
0	2015	0.022843	2.960768
1	2015	0.004267	0.142224
2	2015	0.003736	0.142223
3	2015	0.003638	0.142223
4	2015	0.006529	0.130332

	diff_dec_jan_price_peak_var	diff_dec_jan_price_peak_fix \
0	0.014784	19.549864
1	0.068386	0.000000
2	0.001025	0.000000
3	0.000000	0.000000

```
4                       0.003335                       0.078199

   diff_dec_jan_price_mid_peak_var   diff_dec_jan_price_mid_peak_fix    \
0                         0.059098  13.033244
1                         0.000000                          0.000000
2                         0.000000                          0.000000
3                         0.000000                          0.000000
4                         0.002833                          0.052133

   diff_Dec_mean_price_off_peak_var   diff_Dec_mean_price_off_peak_fix   \
0                         -0.003905                          -0.170704
1                         -0.001096                           1.706668
2                         -0.042333                          -2.925252
3                         -0.002582                          -0.106668
4                          0.018415                           3.005243

   diff_Dec_mean_price_peak_var  diff_Dec_mean_price_peak_fix   \
0                      0.000739                       1.668256
1                     -0.005699                       0.000000
2                      0.009192                      19.549864
3                      0.000000                       0.000000
4                     -0.082911                     -19.530314

  diff_Dec_mean_price_mid_peak_var  diff_Dec_mean_price_mid_peak_fix
0                         0.005751                          1.112172
1                         0.000000                          0.000000
2                         0.059149                         13.033244
3                         0.000000                          0.000000
4                        -0.058292                        -13.020211
```

[ ]:
```
testing_set = test_data.drop(columns=['id','churn'])
testing_set.shape
```

[ ]: (14606, 36)

[ ]:
```
pred_train_labels = np.zeros(shape=(X.shape[0], 2))    # pred training labels
pred_test_labels = np.zeros(shape=(testing_set.shape[0], 2))
# create cv dataset
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=29)
fold_counter = 1
for train_index, test_index in kfold.split(X, y):
    X_train, X_test = X.loc[train_index], X.loc[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # build model
    rf = RandomForestClassifier(random_state=56)
    # train model
    rf.fit(X_train, y_train)
```

```
    pred_train_labels[test_index] = rf.predict_proba(X_test)
    pred_test_labels += rf.predict_proba(testing_set)/5
    print(f"Fold {fold_counter} Precision {precision_score(y_test, rf.
↪predict(X_test)):.3f} Recall {recall_score(y_test, rf.predict(X_test)):.3f}␣
↪Accuracy {accuracy_score(y_test, rf.predict(X_test)):.3f}")
    fold_counter = fold_counter + 1
#
print(f"Total   Precision {precision_score(y, pred_y):.3f} Recall␣
↪{recall_score(y, pred_y):.3f} Accuracy {accuracy_score(y, pred_y):.3f}")
```

[ ]:
```
prob_discount = pd.DataFrame(data=pred_test_labels, columns=['0_dis','1_dis'])
customer_prob = prob_no_discount.join(prob_discount)
customer_prob.head()
```

Then, let's calculate the expected profit from customers.

[ ]:
```
# calculate expected profit of a single customer with discounted prices and␣
↪without discounted prices
def expect_from_a_customer(customer_id):
    # expected values without discount
    average_yearly_price_var =␣
↪price_stat[price_stat['id']==customer_id]['price_off_peak_var_mean'].
↪values[0]
    total_yearly_usage_var =␣
↪client_df[client_df['id']==customer_id]['cons_12m'].values[0]
    average_yearly_price_fix =␣
↪price_stat[price_stat['id']==customer_id]['price_off_peak_fix_mean'].
↪values[0]
    total_yearly_usage_fix =␣
↪client_df[client_df['id']==customer_id]['cons_gas_12m'].values[0]
    total_profit = average_yearly_price_var*total_yearly_usage_var +␣
↪average_yearly_price_fix*total_yearly_usage_fix
    #
    prob_stay = customer_prob[customer_prob['id']==customer_id]['0'].values[0]
    expected_profit = prob_stay*total_profit
    # expected values with discount
    total_profit_dis = total_profit*0.8
    #
    prob_stay_dis = customer_prob[customer_prob['id']==customer_id]['0_dis'].
↪values[0]
    expected_profit_dis = prob_stay_dis*total_profit_dis
    return expected_profit, expected_profit_dis
```

[ ]:
```
# Calculate expected profit from each customer
expected_no_dis, expected_dis = [],[]
for cus_id in customer_prob['id']:
    temp = expect_from_a_customer(customer_id=cus_id)
```

```
    expected_no_dis.append(temp[0])
    expected_dis.append(temp[1])
#
customer_prob["expected_no_dis"] = expected_no_dis
customer_prob["expected_dis"] = expected_dis
customer_prob["diff_discount"] = customer_prob["expected_dis"] -␣
↪customer_prob["expected_no_dis"]
customer_prob
```

Expected profit from only 349 customers (2%) rise after being offered discounted prices.

[ ]:
```
customer_prob[customer_prob["diff_discount"]>0]
```

[ ]:
```
# 2% of all cutomers
customer_prob[customer_prob["diff_discount"]>0].shape[0]/len(customer_prob)
```

Among the 104 predcited churning customers, after being offered discounted prices,

the expected profit from 61 customers increase and total expected profit increase by 2250001.

the expected profit from 43 customers decrease and total expected profit decrease by 1138753.

So, it is expected that offering a discount to the predcited churning customers can bring extra profit of 1111248.

[ ]:
```
# number of churning customers
customer_prob[(customer_prob["1"]>0.5)].shape[0]
```

[ ]:
```
# number of customers whose expected profit increase
customer_prob[(customer_prob["diff_discount"]>0)&(customer_prob["1"]>0.5)].
↪shape[0]
```

[ ]:
```
# number of customers whose expected profit decreas
customer_prob[(customer_prob["diff_discount"]<=0)&(customer_prob["1"]>0.5)].
↪shape[0]
```

[ ]:
```
# total extra profit from customers whose expected profit increase
customer_prob[(customer_prob["diff_discount"]>0)&(customer_prob["1"]>0.
↪5)]["diff_discount"].sum()
```

[ ]:
```
# total extra profit from customers whose expected profit decrease
customer_prob[(customer_prob["diff_discount"]<=0)&(customer_prob["1"]>0.
↪5)]["diff_discount"].sum()
```

[ ]:
```
# total extra profit from predicted churning customers
customer_prob[customer_prob["1"]>0.5]["diff_discount"].sum()
```

Based on the results above, it seems offering a discount to the predicted churning customers can increase the expected profit from these customers.