**Targeted Marketing Strategy for Investment Banking Acquisition**

Problem Statement:

**Business Use Case:** The bank is experiencing a drop in its income, and they have discovered that their customers are not depositing money as often as before. Investment Bankings are a type of savings account where people agree to keep their money in the bank for a specific period. This allows the bank to invest that money and make more profit. When customers have Investment Bankings, the bank can also try to convince them to buy other products like investment funds or insurance, which can make the bank even more money. So, the bank wants to find out which customers are more likely to agree to a Investment Banking and focus their advertising and marketing efforts on those customers to increase their income.

**Data Science Problem Statement**

Predict if the client will subscribe to a Investment Banking based on the analysis of the marketing campaigns the bank performed.

**Evaluation Metric**

We will be using ROC-AUC for evaluation.

Understanding the dataset

**Data Set Information**

The data is related to direct marketing campaigns of a banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank Investment Banking) would be subscribed ('yes') or not ('no') subscribed.

Goal: - The classification goal is to predict if the client will subscribe (yes/no) a Investment Banking (variable y).

**Features**

| Feature | Feature\_Type | Description |
| --- | --- | --- |
| age | numeric | age of a person |
| job | Categorical, nominal | type of job ('admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown') |
| marital | categorical, nominal | marital status ('divorced','married','single','unknown'; note: 'divorced' means divorced or widowed) |
| education | categorical, nominal | ('basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown') |
| default | Categorical, nominal | has credit in default? ('no','yes','unknown') |
| housing | categorical,  nominal | has housing loan? ('no','yes','unknown') |
| loan | categorical, nominal | has personal loan? ('no','yes','unknown') |
| contact | categorical, nominal | contact communication type ('cellular','telephone') |
| month | categorical, ordinal | last contact month of year ('jan', 'feb', 'mar', ..., 'nov', 'dec') |
| day\_of\_week | categorical, ordinal | last contact day of the week ('mon','tue','wed','thu','fri') |
| duration | numeric | last contact duration, in seconds . Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no') |
| campaign | numeric | number of contacts performed during this campaign and for this client (includes last contact) |
| pdays | numeric | number of days that passed by after the client was last contacted from a previous campaign (999 means client was not previously contacted) |
| previous | numeric | number of contacts performed before this campaign and for this client |
| poutcome | categorical, nominal | outcome of the previous marketing campaign ('failure','nonexistent','success') |

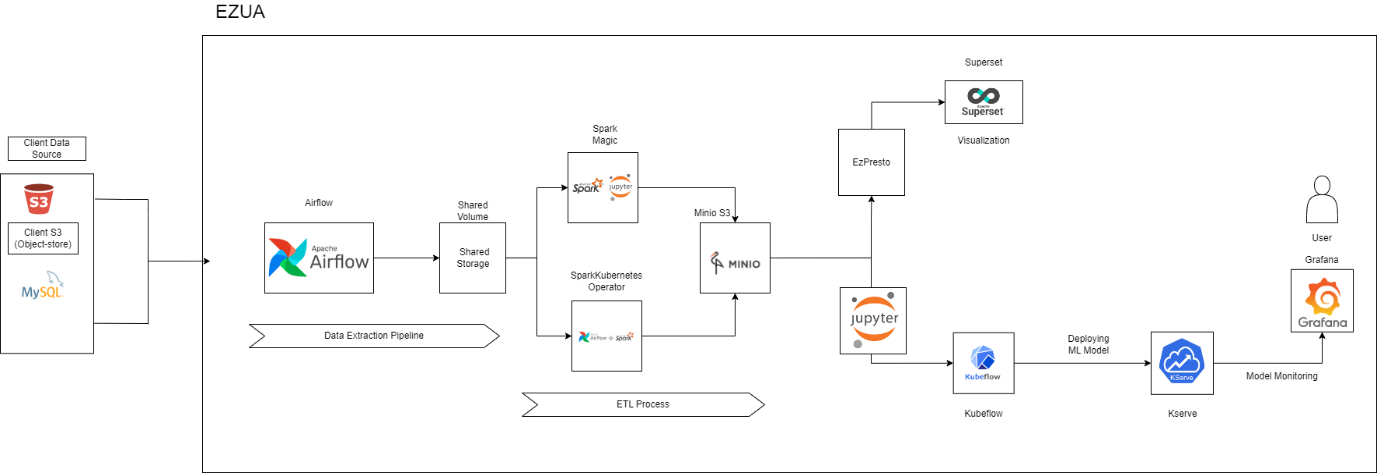
**Target variable (desired output):**

| Feature | Feature\_Type | Description |
| --- | --- | --- |
| y | binary | has the client subscribed a Investment Banking? ('yes','no') |

**Services & Tools:**

1. Visualization Tools: - Apache Superset
2. MLOps Tool: - Kubeflow
3. Programming Language: - Python
4. Machine Learning - Scikit Learn
5. Apache spark
6. Apache Airflow
7. Jupyter Notebook
8. Grafana
9. Ezpresto
10. Minio S3 storage

**Proposed Architecture:**

****

**Workflow:**

- Retrieve data from different client data sources, including S3 object store and SQL databases.  
- Utilize Airflow DAGs to extract data from client sources and store it in a shared storage system on our cluster.  
- Process the data using Spark, which can be invoked through Spark magic or within the Airflow DAG.  
- Upload the processed data to a Minio S3 object store.  
- Connect Minio to EzPresto for data visualization in Superset.  
- Perform model training and prediction using Kubeflow and its pipelines, executed within Jupyter notebooks.  
- Interpret the trained models using Kserve.  
- Visualize the data and results using Grafana.

**Airflow:**

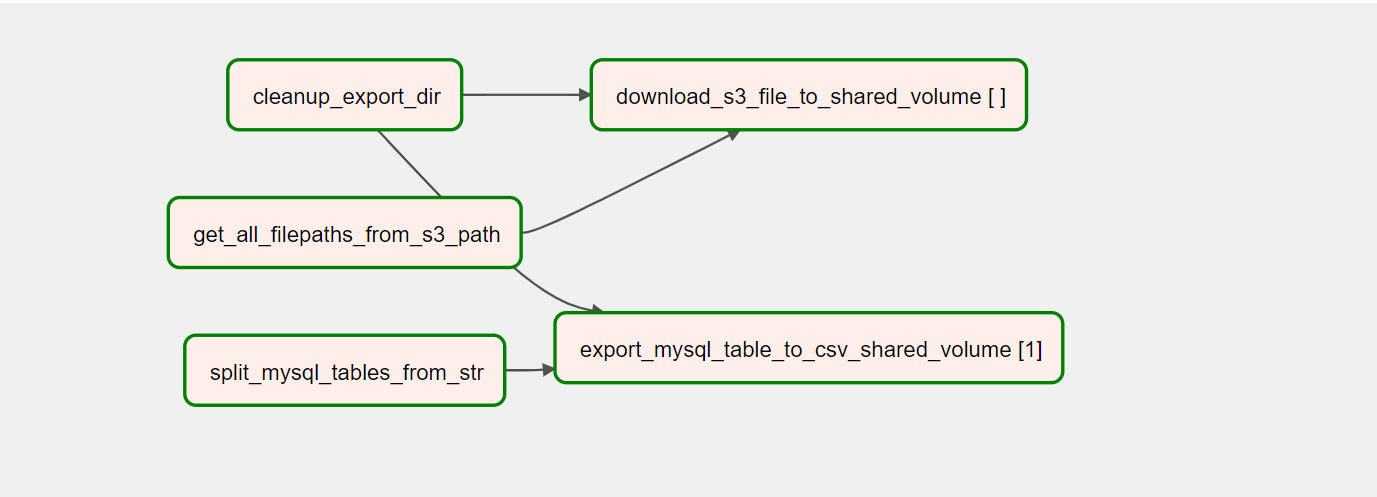
Airflow is a powerful tool used for data extraction and the ETL (Extract, Transform, Load) process. In the context of this experiment, two DAGs (Directed Acyclic Graphs) are employed. The first DAG is responsible for data extraction, while the second DAG invokes Spark for data processing.

**DAG 1: Data Extraction:**

This DAG focuses on fetching files from various data sources belonging to customers and pushing them to shared volumes in our cluster. All the tasks within this DAG are implemented as Python operators. The data sources involved are Minio S3 (an object store) and a MySQL database.

This DAG comprises 5 tasks:

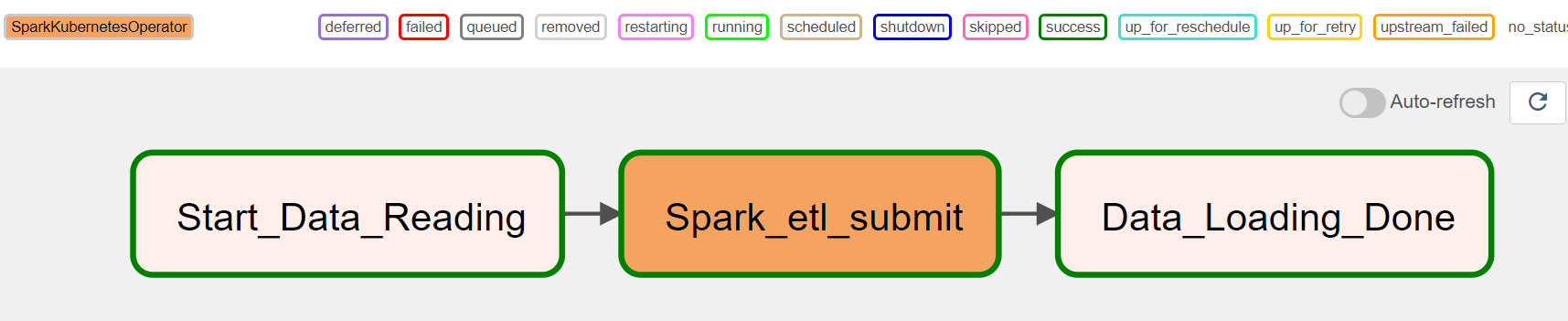
* **cleanup\_export\_dir:** This task clears the shared storage directory to ensure that any previously existing files are removed before the extraction process.
* **get\_all\_filepaths\_from\_s3\_path:** This task retrieves all the files from the customer's S3 bucket in Minio S3.
* **split\_mysql\_tables\_from\_str:** This task extracts the table name(s) from which data is to be fetched. It takes a string input containing comma-separated table names and splits it into individual table names.
* **download\_s3\_file\_to\_shared\_volume:** This task downloads the file from the S3 storage into the shared volume, making it locally available for further processing.
* **export\_mysql\_table\_to\_csv\_shared\_volume:** This task converts the specified SQL table into CSV format and stores it in the shared volume.



**DAG2: Invoking spark through Airflow**

This DAG is responsible for invoking a Spark application on a Kubernetes cluster and running Spark jobs. The SparkKubernetesOperator is utilized for this purpose.

The main task within this DAG is **spark\_etl\_submit**. This task creates a new Spark Application object on the Kubernetes cluster and executes the required jobs. The Spark job fetches files from the shared volume, performs the necessary processing, and loads the transformed data into Minio S3.



* To establish a connection with the data source in Superset, **EzPresto** is configured with Minio storage.

**Superset Dashboard:**

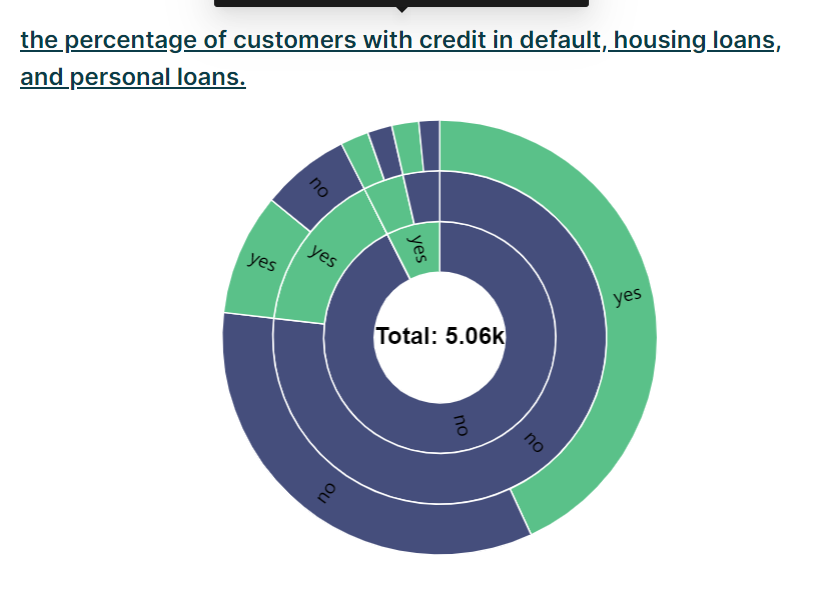
* **Proportion of Customers with Credit Default, Housing Loans, and Personal Loans**

1. Credit in Default: This refers to customers who have failed to repay their credit obligations on time, resulting in default. The percentage of customers with credit in default tells you how many people out of the total customer population have defaulted on their credit payments.

2. Housing Loans: This indicates whether customers have taken loans to finance their housing or real estate properties. The percentage of customers with housing loans tells you what portion of the customer base has borrowed money for housing purposes.

3. Personal Loans: This indicates whether customers have taken loans for personal use, such as for education, travel, or other personal expenses. The percentage of customers with personal loans tells you the proportion of customers who have taken loans for non-housing-related purposes.

By calculating the percentages, you can understand the prevalence of these financial arrangements among the customers in the dataset. For example, if 20% of customers have credit in default, it means that 20 out of every 100 customers in the dataset have failed to repay their credit obligations. Similarly, the percentages of housing loans and personal loans indicate the respective proportions of customers who have taken these types of loans.

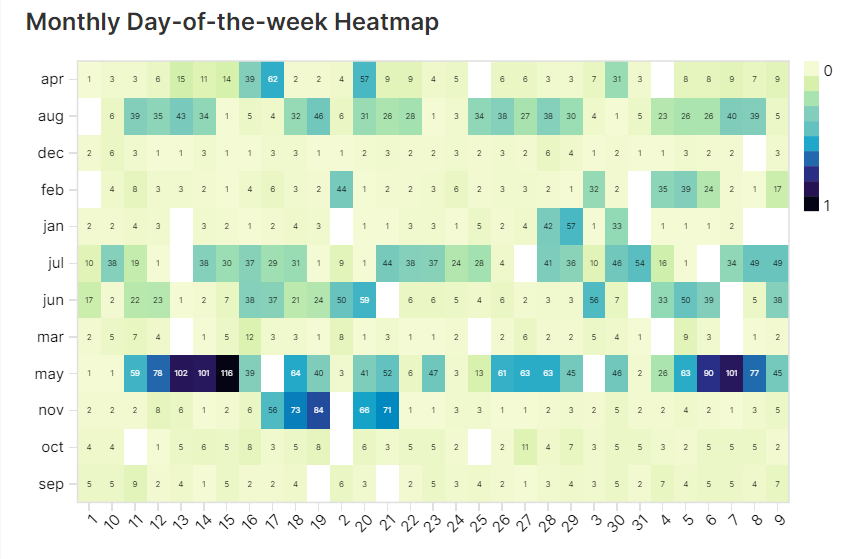


* **Monthly Day-of-the-week Heatmap**

A Heatmap is a graphical representation that uses color intensity to depict the frequency or density of data across two dimensions, such as months and days of the week. Each cell in the heatmap represents a combination of a month and a day, and the colour of the cell indicates the frequency or count of contacts that occurred on that specific day of the week within that month.

In simple words, the graph shows the busiest and least busy periods in terms of customer contacts. The darker or more intense colours represent higher frequencies, indicating that more contacts were made during those specific combinations of month and day of the week. On the other hand, lighter or less intense colours indicate lower frequencies, suggesting fewer contacts made on those days.

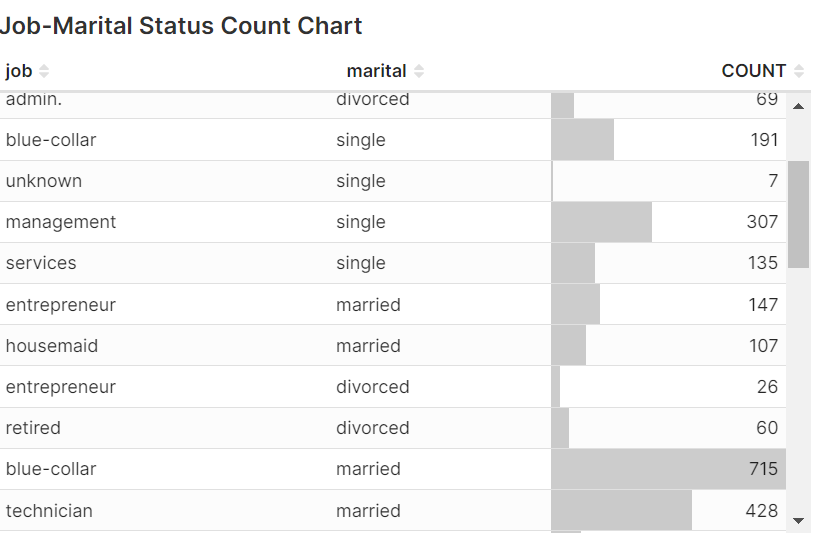
By looking at the graph, you can quickly identify patterns and trends in customer contacts over different months and days of the week. It helps in understanding when the bank had more interactions with customers and when the contact frequency was relatively lower. This information can be useful for scheduling future marketing campaigns, allocating resources, and making data-driven decisions based on the customer contact patterns.



* **Job-Marital Status Count Chart**

The chart that represents the count of customers grouped by their job and marital status.

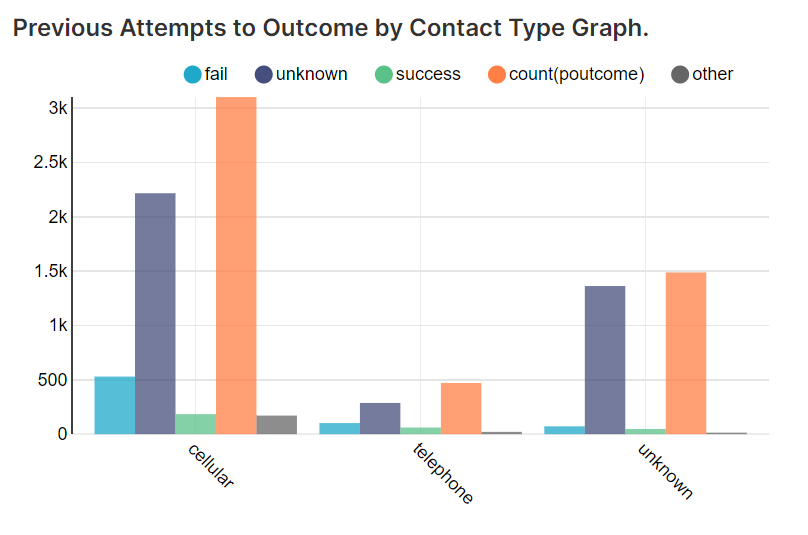
In simple words, this chart provides a visual representation of how many customers fall into different job categories based on their marital status. Each combination of job and marital status is represented by a separate column in the chart. The height of each bar represents the count or number of customers belonging to that specific job and marital status combination.  
  
For example, the chart can show the number of married customers who work in blue-collar jobs, the count of single customers who are students, or the quantity of divorced customers in management positions.  
  
The chart helps to understand the distribution and composition of customers across various job types and marital statuses. It allows you to compare and analyze the customer counts within different job and marital status categories, providing insights into the customer demographics and their relationship with job roles and marital status.



* **Previous Attempts to Outcome by Contact Type Graph:**

This graph illustrates how the outcome of previous marketing attempts is related to the number of previous attempts made, considering different contact types. It shows the success or failure of previous marketing campaigns based on the number of times customers were contacted before, categorized by the communication channel used (cellular or telephone).

By examining this graph, you can understand the effectiveness of previous marketing efforts based on the number of attempts and the contact type used. It helps in assessing whether certain contact types or a higher number of attempts have a positive impact on achieving successful outcomes from marketing campaigns.

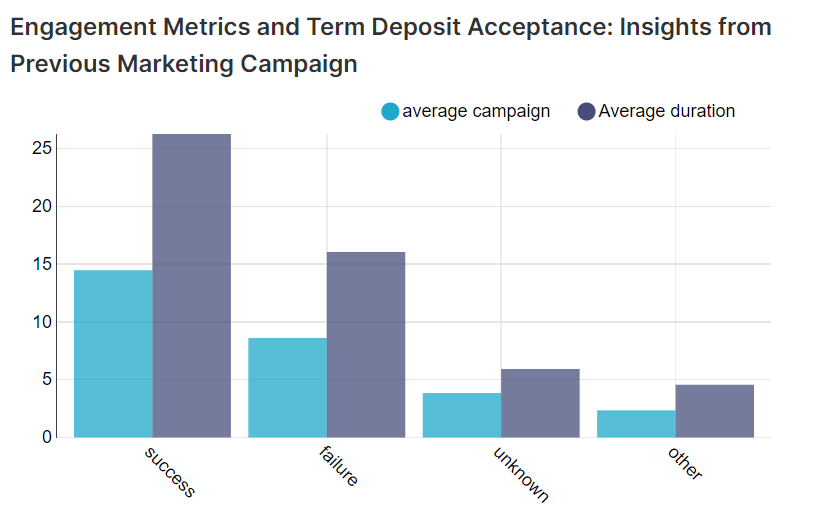


* [**Engagement Metrics and Investment Banking Acceptance: Insights from Previous Marketing Campaign**](https://superset.hpe-staging-ezaf.com/explore/?dashboard_page_id=289IRD9EZ2&slice_id=19):

The graph plots the average duration of calls and the number of contacts made. It categorizes the data based on the outcome of the previous marketing campaign, which could include categories like "Investment Banking Accepted" and "Investment Banking Not Accepted."

The significance of this graph lies in the insights it provides regarding the correlation between customer engagement during calls and their likelihood to agree to a Investment Banking. By analysing the data, the bank can identify patterns and trends in customer behaviour. For example, the graph may reveal that customers who had longer call durations were more likely to accept a Investment Banking, indicating a higher level of interest or engagement. Similarly, the number of contacts made can provide insights into customer responsiveness.

The graph helps the bank understand which customers are more receptive to Investment Bankings and can guide their advertising and marketing efforts. By focusing on customers who had longer call durations and/or higher numbers of contacts in the previous campaign, the bank can target these customers with tailored advertisements and personalized offers related to Investment Bankings. This targeted approach increases the chances of success in convincing these customers to agree to a Investment Banking, thereby potentially boosting the bank's income.



* [**Marital Status and Contact Strategy Analysis for Investment Bankings**](https://superset.hpe-staging-ezaf.com/explore/?dashboard_page_id=l9O0nZB-6&slice_id=20)

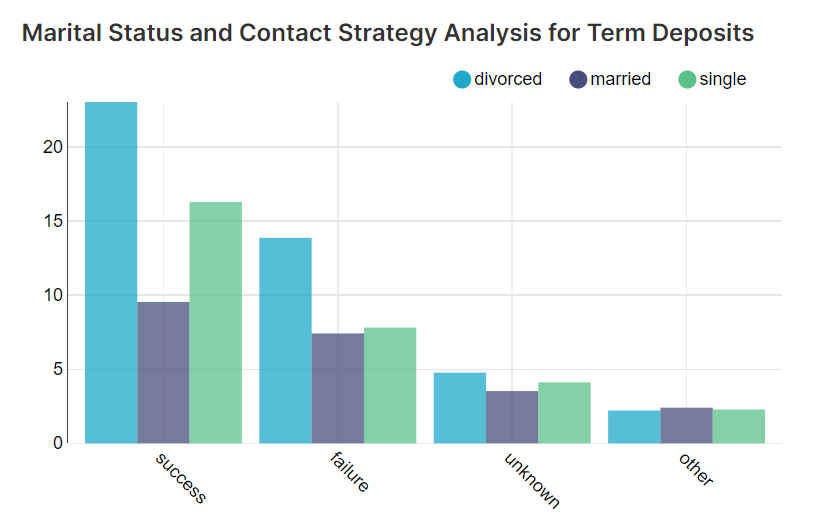
The significance of this graph is to understand the effectiveness of the bank's contact strategy in relation to the customers' marital status and their response to Investment Bankings. By analysing the data presented in this graph, the bank can gain insights into which marital status groups are more receptive to Investment Banking offers and which groups require more targeted marketing efforts.

Here's a breakdown of what this graph can help us understand:

Contact Strategy Evaluation: The graph allows the bank to evaluate its contact strategy's effectiveness in differentiating between marital status groups. It helps determine whether the bank is making an appropriate number of contacts with customers based on their marital status.

Marital Status Influence: The graph reveals how customers' marital status influences their response to Investment Banking offers. It helps identify any trends or patterns that may exist, such as whether married individuals are more likely to agree to a Investment Banking compared to unmarried individuals.

Targeted Marketing: Based on the graph's findings, the bank can tailor its advertising and marketing efforts to specific marital status groups that have shown a higher likelihood of agreeing to Investment Bankings. This approach can optimize the bank's resources by focusing on the customer segments with higher conversion rates.

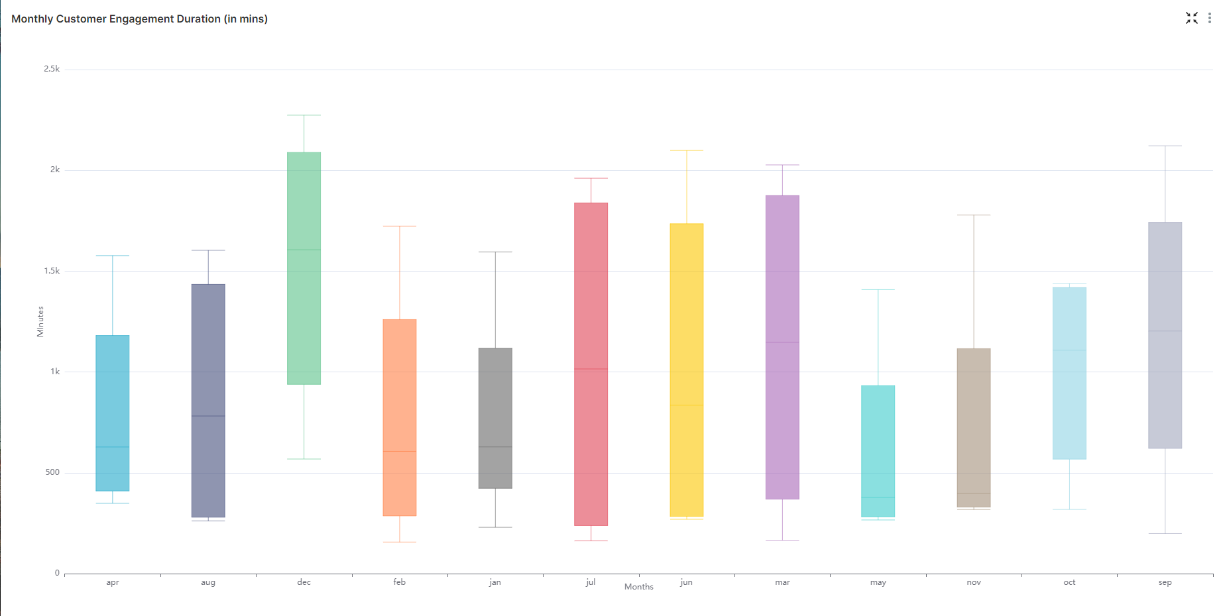


* [**Monthly Customer Engagement Duration (in mins)**](https://superset.hpe-staging-ezaf.com/explore/?dashboard_page_id=l9O0nZB-6&slice_id=21):

This graph refers to the measurement and analysis of the amount of time customers spend engaged with a particular product, service, or activity on a monthly basis, expressed in minutes.

This analysis looks at how long customers are actively involved or interacting with something over the course of a month. It measures the duration of customer engagement in terms of minutes spent.

By examining the monthly customer engagement duration, you can understand the level of interest or involvement customers have with a particular product, service, or activity. It helps in evaluating the extent to which customers are actively engaged and invested in what is being offered. This information can be valuable for identifying patterns, trends, or areas of improvement in customer engagement, and it can guide decisions related to marketing, product development, or customer satisfaction.



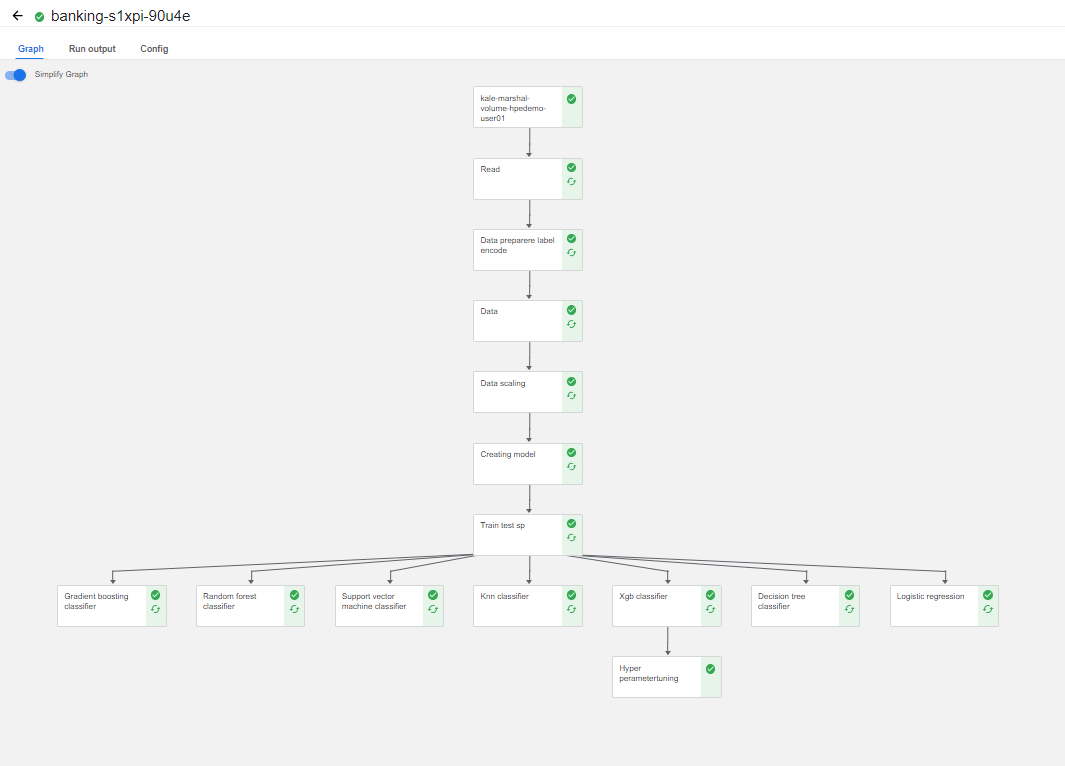
**Spark Application:**

* Importing Libraries: The necessary libraries from the pyspark.sql module are imported, including SparkSession and various functions for working with DataFrames and columns. The datetime class is imported from the Python standard library.
* Get Current Date: The current date is obtained and formatted as 'YYYY-MM-DD' using the datetime.today().strftime('%Y-%m-%d') function, and it is stored in the today variable.
* Creating a SparkSession: A SparkSession is created with the name "ETL". If a SparkSession already exists, it retrieves it; otherwise, it creates a new one. The application runs in local mode.
* Reading DataFrames from CSV Files: Two DataFrames (df1 and df2) are read from CSV files. The spark.read method is used with options such as header (to indicate the presence of headers in the CSV files) and inferSchema (to automatically infer the schema). The file paths are dynamic, using the today variable to read files with the current date in the filename.
* Displaying DataFrames: The contents of df1 and df2 are shown using the show() method. The schema of both DataFrames is printed using printSchema().
* Finding Duplicate Rows: The number of duplicate rows in df1 and df2 is calculated by subtracting the count of distinct rows from the total row count. This is achieved using dropDuplicates() and count() functions.
* Handling Null Values: Null values in both DataFrames are identified and counted using a combination of when, isnan, and isNull functions. The counts of null values for each column are displayed using show(). Additionally, for certain columns with null values, the most frequent value (mode) for that column is calculated and used to fill the null values using fillna().
* Concatenating DataFrames: The two DataFrames, df1 and df2, are concatenated (unioned) into a single DataFrame called df.
* Renaming a Column: The column 'y' in the df DataFrame is renamed to 'target' using the withColumnRenamed() method.
* Writing Data to CSV: The final DataFrame df is written to a CSV file on an S3 bucket with the current date appended to the filename. The data is written as a single partition using coalesce(1) to create a single output file.

**Jupyter Notebook:**

* Importing Libraries: The code begins by importing necessary libraries and modules required for data analysis and machine learning. Some of the essential libraries used in this code are pandas, numpy, scikit-learn, xgboost, matplotlib, seaborn, and others.
* S3 Access Setup: The code sets up access to a S3 bucket using the boto3 library. It assumes access to a specific S3 bucket and folder path.
* Data Retrieval and Cleaning: The code retrieves a CSV file from the specified AWS S3 bucket, loads it into a pandas DataFrame (df), and performs some initial data cleaning and preprocessing steps. This includes handling missing values, converting categorical variables to numerical using LabelEncoder, and dealing with outliers.
* Exploratory Data Analysis (EDA): The code performs exploratory data analysis using various data visualization techniques. It uses matplotlib and seaborn libraries to create plots and visualizations for understanding the data distribution, relationships between features, and how they relate to the target variable (customer churn).
* Feature Selection: The code applies feature selection techniques like VarianceThreshold and SelectKBest with ANOVA (f\_classif) to select important features for building the machine learning models.
* Model Building and Evaluation: The code builds several machine learning models, such as Logistic Regression, Random Forest Classifier, Decision Tree Classifier, Support Vector Machine (SVM) Classifier, K-Nearest Neighbors (KNN) Classifier, Gradient Boosting Classifier, and XGBoost Classifier. It splits the data into training and testing sets, trains the models on the training data, and evaluates their performance on the testing data using accuracy, confusion matrix, and classification reports.
* Cross-Validation: The code performs cross-validation using k-fold cross-validation to get an average accuracy score for each model.
* Hyperparameter Tuning: The code performs hyperparameter tuning for the XGBoost Classifier using GridSearchCV to find the best combination of hyperparameters that maximize the model's performance.
* Final Model: After hyperparameter tuning, the best-performing model (XGBoost Classifier) is selected and trained on the entire dataset. The model's performance metrics (accuracy, confusion matrix, classification report) are printed and displayed using visualizations.
* ROC-AUC Curve: The code plots the Receiver Operating Characteristic (ROC) curve and computes the Area Under the Curve (AUC) for the best model.
* Saving the Model: The final tuned XGBoost Classifier model is saved using the pickle library, allowing for future use or deployment.
* Customer Churn Analysis: The code loads the saved model and performs predictions on the test data. The results are then used to create a DataFrame containing the original and predicted churn labels for each customer.

**Kube Flow:**

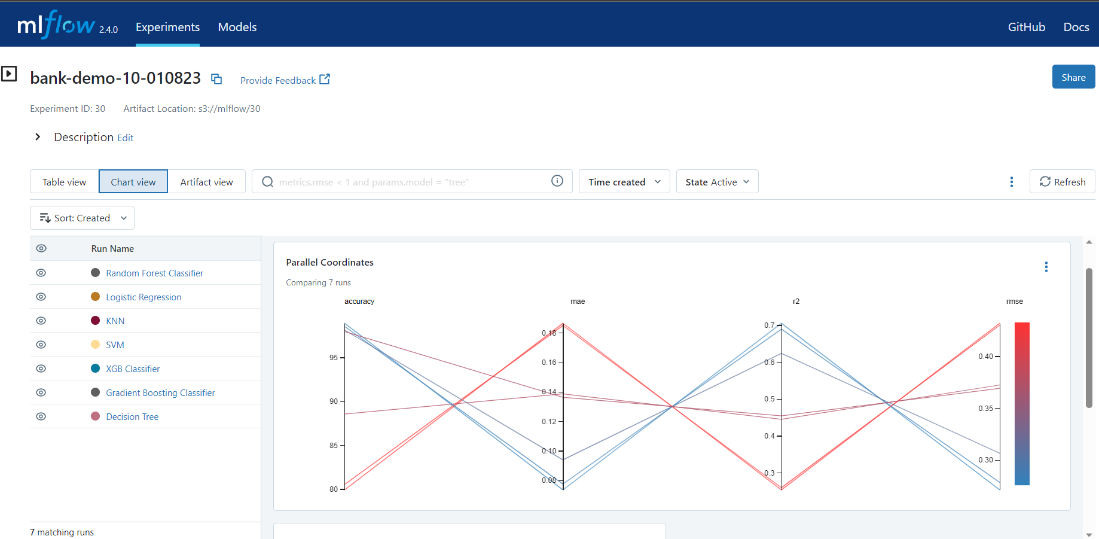


Kubeflow can be leveraged to streamline the various stages involved in building and deploying machine learning models. Let's break down the pipeline steps and explain how Kubeflow can be utilized for each stage:

* Read Data:  
  In this step, the pipeline reads the input data from a specified source, such as a database or a file system. Kubeflow can be used to define a data ingestion component that fetches the data and stores it in a suitable format for further processing.
* Data Preparation:  
  Data preparation involves cleaning, transforming, and pre-processing the raw data to make it suitable for model training. Kubeflow can be used to define a data pre-processing component that performs tasks like handling missing values, outlier detection, and feature engineering.
* Label Encoding:  
  Label encoding is a process of converting categorical variables into numerical representations. Kubeflow can be utilized to define a label encoding component that applies appropriate encoding techniques to convert categorical data into numerical form.
* Data Scaling:  
  Data scaling is a crucial step in machine learning pipelines to ensure that all features are on a similar scale. Kubeflow can be used to define a data scaling component that applies techniques like standardization or normalization to scale the input data.
* Creating Model:  
  In this step, the pipeline defines the machine learning model architecture. Kubeflow can be utilized to define a model creation component that specifies the model structure, including the layers, activation functions, and other parameters.
* Train-Test Split:  
  To evaluate the model's performance, the dataset is typically split into training and testing subsets. Kubeflow can be used to define a train-test split component that partitions the data into training and testing sets based on a specified ratio.
* Model Training:  
  Kubeflow can be leveraged to define a model training component that trains the machine learning model using the training dataset. This component can specify the training algorithm, loss function, and optimization parameters. Gradient Boosting Classifier, Random Forest Classifier, Support Vector Machine Classifier, KNN Classifier, XGB Classifier, Decision Tree Classifier, Logistic Regression Classifier: These are different classification algorithms that can be used to build machine learning models. Kubeflow can be utilized to define separate components for each classifier, specifying the algorithm, hyperparameters, and evaluation metrics.
* Hyperparameter Tuning:  
  Hyperparameter tuning involves finding the optimal values for the model's hyperparameters to improve its performance. Kubeflow can be used to define a hyperparameter tuning component that performs an automated search for the best hyperparameter values using techniques like grid search or random search.

By utilizing Kubeflow, the investment banking pipeline can be orchestrated and managed efficiently, allowing for reproducibility, scalability, and easy experimentation with different models and hyperparameters.

**ML Flow:**



It provides a comprehensive set of tools and APIs to track experiments, log parameters and metrics, and deploy models. MLflow promotes reproducibility, collaboration, and easy experimentation with different models and hyperparameters.

MLflow offers the following key features:

* Tracking: MLflow allows you to track experiments by logging parameters, metrics, and artifacts. It provides a unified interface to record and compare different runs, making it easy to track model performance and experiment iterations.
* Projects: MLflow enables you to package your code into reproducible projects. These projects can be easily shared and executed on different platforms, ensuring consistent results across different environments.
* Model Registry: MLflow provides a centralized model registry to manage and version your trained models. The model registry allows you to easily deploy and serve models in a production environment.
* Model Serving: MLflow supports model serving through its REST API, allowing you to deploy models as web services. This makes it easy to integrate MLflow models into your existing applications or infrastructure.
* Integration: MLflow integrates with popular machine learning libraries and frameworks, such as scikit-learn, TensorFlow, PyTorch, and XGBoost. It seamlessly integrates with your existing workflows and tools.

**Grafana:**

**Knative Serving - Revision CPU and Memory Usage**

The Knative Sandbox Grafana dashboard offers insights into Knative application performance. It visualizes CPU and memory usage metrics for different application revisions, aiding in resource monitoring. By displaying graphs and data, it enables tracking of resource consumption patterns. This helps optimize efficiency by identifying potential bottlenecks and allowing informed resource allocation decisions. In essence, the dashboard provides a quick overview of how CPU and memory are utilized by various application versions, facilitating efficient management, and ensuring optimal performance.



**Knative Serving - Revision HTTP Requests**

The Grafana dashboard designed by Knative for tracking HTTP requests across application revisions offers a visual insight into performance. It presents metrics like request count, average response time, and error rates across revisions for a holistic view. A comparison section allows side-by-side analysis of different revisions, highlighting response time trends and error rates. Through graphs depicting response times and error percentages, developers can identify anomalies or slowdowns. The dashboard showcases throughput graphs illustrating request rates, aiding in load distribution understanding. Geographical data representation might indicate usage hotspots. Custom metrics cater to specific app needs. Alerts notify when metrics cross predefined thresholds. Interactive filters permit focus on specific revisions or time frames. Overall, this Grafana dashboard empowers proactive decision-making, enhancing application performance and user experience within the Knative environment.

