# Section 1: Application of Decision Trees in Business (Theoretical Analysis)

• Why are decision trees useful in customer churn prediction?

**Customer behaviour** is hard to determine, since it depends on a combination of features. The features interact with each other and the relationship between features and outcomes is nonlinear. **Decision trees** automatically learn these complex interactions by making hierarchical splits of nodes, based on multiple conditions. From the provided dataset (customer_churn.csv), we can see that many important churn indicators are categorical. **Decision trees** can also split directly on categorical values without transforming the data.

**Decision trees** create a set of if-then rules at cutoff values that lead to a natural way to visualize why a customer is likely to churn. They rank features based on their importance. With this presentation, **business stakeholders** can see which features (age, number of complaints, subscription length, etc.) contribute to churn likelihood.

**Customer churn** prediction involves large datasets. There is a need for speed, because the model may be retrained multiple times with fine-tuned hyperparameters to improve a prediction's accuracy. The model's training must finish in a reasonable amount of time, without using too much computing power. **Decision trees** are an appropriate tool to handle this problem, because they can handle large datasets and are computationally efficient.

**Decision trees** also serve as building blocks for a more powerful model like Random Forests. Later in this report, we will verify whether they improve accuracy or not.

• What business actions can be taken based on the predictions of a decision tree model?

- If the decision tree predicts that customers with high monthly bills are likely to churn, provide them with a discount.
- If the churn risk is high for customers with frequent complaints, specifically provide better customer support for them.
- If the model finds that customers with multiple support tickets have a high churn rate, initiate proactive follow-ups to resolve issues quickly.
- If customers with month-to-month contracts churn more, encourage them to switch to annual plans by offering discounts or incentives.
- Introduce customized pricing to price-sensitive segments.

- If customers who use fewer features tend to churn, introduce in-app onboarding to educate them on more features.

- If the decision tree shows that young professionals prefer digital engagement, increase email & app-based communication.

- If the model reveals that customers who switch providers have a common churn reason (better pricing elsewhere), consider benchmarking competitors and adjusting pricing accordingly.

# Section 2: Python Implementation – Building the Model

The installed Python packages are: NumPy, Pandas, Matplotlib, Jupyter, Scikit-learn, Seaborn, Imbalanced-learn, Graphviz. If running Graphviz locally, run the installer for **graphviz-12.2.1** and check the option to "Add Graphviz to system PATH". (https://graphviz.org/download/)

***Task 1: Data Preparation and Exploration***

Modify the directory appropriately if using another testing environment. For example, in Google Colab, it is possible that the directory is as such: **"/content/customer_churn.csv".** We are using **Pandas** to read the CSV file because operations on the variable dataset require its type to be a DataFrame.

The summary statistics look like this:

```
...          CustomerID         Age  Subscription_Length_Months  Watch_Time_Hours  \
    count  1000.000000  1000.00000                 1000.000000       1000.000000
    mean    500.500000    43.81900                   18.218000        100.794546
    std     288.819436    14.99103                   10.177822         56.477606
    min       1.000000    18.00000                    1.000000          5.036738
    25%     250.750000    31.00000                    9.000000         50.383080
    50%     500.500000    44.00000                   18.000000        100.234954
    75%     750.250000    56.00000                   27.000000        150.445885
    max    1000.000000    69.00000                   35.000000        199.944192

           Number_of_Logins  Payment_Issues  Number_of_Complaints  \
    count       1000.000000     1000.000000           1000.000000
    mean          50.387000        0.154000              4.546000
    std           28.224171        0.361129              2.919316
    min            1.000000        0.000000              0.000000
    25%           26.000000        0.000000              2.000000
    50%           51.000000        0.000000              5.000000
    75%           75.000000        0.000000              7.000000
    max           99.000000        1.000000              9.000000

           Resolution_Time_Days        Churn
    count           1000.000000  1000.000000
    mean              15.268000     0.265000
    std                8.225317     0.441554
    min                1.000000     0.000000
    25%                9.000000     0.000000
    50%               15.000000     0.000000
    75%               22.000000     1.000000
    max               29.000000     1.000000
```

Surprisingly, we found no null values in this dataset.

*Visualisation 1: Histograms (See Appendix 1 at page 8)*

We have dropped the column **'CustomerID'** from dataset_cols because it does not contribute to the analysis. We can already see a problem with our dataset **"customer_churn.csv"**. The distribution of Churn, which is the outcome we are trying to predict, is imbalanced. We will address this issue in Task 2. The distribution of **Payment_Issues** is fine, because the streaming service (StreamFlex) related to this data should be user-friendly (HOPEFULLY), which would drastically reduce the number of payment issues.

*Visualisation 2: Box Plots (See Appendix 2 at page 9)*

The box plots show us the value range of a feature. The bottom of the box corresponds to the minimum value. The top of the box corresponds to the maximum value. The line in the middle of the box corresponds to the mean.

*Visualisation 3: Correlation (See Appendix 3 at page 10)*

The reason why I converted the categorical columns into numbers for correlation_dataset is because the **Pandas corr()** function assumes that ALL inputs in the DataFrame are numbers.

I plotted the correlation values in a heatmap for visibility.

## ***Task 2: Building a Decision Tree Classifier***

For this assignment, I have decided to make the training and testing sets an 80/20 split of the dataset. To make **Task 2** and **Task 3** results consistent, I used random_state=42. The scoring metric used to determine the best model for both Task 2 and Task 3 will be the **f1-score**, because it shows the overall performance for Class 1 (churned) predictions.

Categorical features must be converted into numbers before training, because machine learning models typically work with numerical data. Encoding helps models interpret categorical variables correctly by converting the categories to numbers.

One-Hot Encoding was used for the columns **'Preferred_Content_Type'** and **'Payment_Method'**, because there is no order assumption for their data. Ordinal Encoding was used for the column **'Membership_Type'**, because there is an order assumption for its data. The order should be Basic = 0, Standard = 1 and Premium = 2.

I have decided to resample the training data, because the number of non-churned vs churned customers is extremely skewed in favor of the former (594 vs 206). If we keep this imbalance, the Decision Tree Classifier will tend to favor the majority class which will lead to biased predictions.

For this model, I used **SMOTE (Synthetic Minority Over-sampling Technique)** to rebalance the training data from a class distribution of 594:206 to 594:594.
The variable **param_grid** corresponds to the hyperparameter grid. We just need a wide range of hyperparameters to be able to fine-tune the model.

The **CV (cross-validation) fold value** determines how many times the dataset is split into training and validation sets. For a small dataset (1000 rows), cv=3 or cv=5 is ideal because it ensures enough data per fold while keeping variance in check.

Below is the process of training the model for a single set of hyperparameters in a grid:

Let cv = k.

- The model is trained on k-1 folds and tested on the remaining 1 fold.
- This process repeats k times, each time using a different fold for testing.
- The final performance is the average of all k test scores.

GridSearchCV is used to iterate through every hyperparameter combination in param_grid. For each iteration, the training data will be fit to the iteration's model, then the iteration's model's weighted F1-Score is registered. Repeat this until we have the model with the best weighted F1-Score, for a certain CV value.

We repeat this GridSearchCV process for multiple CV values.

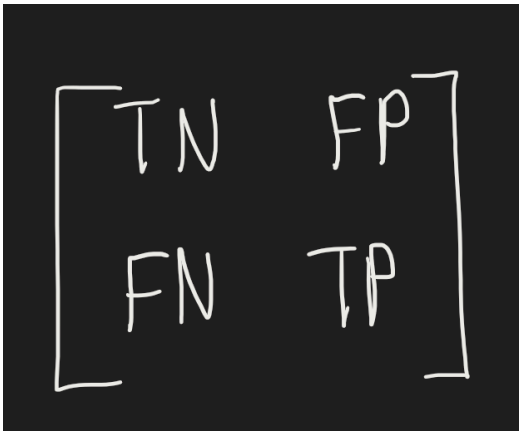
Along with the Python code and this report, I have also provided **"decision_tree.png"**. This PNG file has been generated using Graphviz, a package that lets us easily render a decision tree.

Given param_grid, the best Decision Tree Classifier model has these hyperparameters:
**{'class_weight': None, 'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'min_samples_leaf': 10, 'min_samples_split': 2, 'splitter': 'best'}**

Here are the Decision Tree Classifier model's performance values:
**Confusion Matrix:**
**[[121 20]**
**[ 46 13]]**
**Accuracy: 0.67**
**Precision: 0.39**
**Recall: 0.22**
**f1-score: 0.28**

The **Confusion Matrix**'s notation varies when looking at multiple sources. Since we are using scikit-learn for this assignment, we should use the notation in the documentation (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html).

In our case, a binary classification (non-churned or churned), the **Confusion Matrix** is formatted as such:

| | |
|---|---|
|  | TN (True Negatives) → Correctly predicted Class 0 (non-churned)<br><br>FP (False Positives) → Incorrectly predicted Class 0 (non-churned)<br><br>TP (True Positives) → Correctly predicted Class 1 (churned)<br><br>FN (False Negatives) → Incorrectly predicted Class 1 (churned) |

The accuracy shows us that 67% of the model's predictions on the test data were correct.
The precision signifies that only 39% of predicted Class 1 instances were correct.
The low recall (22%) suggests that the model is missing a lot of Class 1 cases.
The low f1-score (28%) implies that the overall performance for predicting Class 1 cases is weak.

We can conclude that the Decision Tree Classifier model is not distinguishing Class 1 (churned) well. We could consider a different model, like Random Forests (see **Task 3**).

***Task 3: Improving Performance with Random Forests***

Similarly to Task 2, we used One-Hot Encoding and Ordinal Encoding for the appropriate columns. However, after attempting to fine-tune the model's performance, we found that training it with the non-resampled data yielded better results. CV values 3 and 5 were used.

We decided to search through param_grid using RandomizedSearchCV, because a Random Forest Classifier model is much more computationally expensive to train than a Decision Tree Classifier model. RandomizedSearchCV finds a good combination of hyperparameters without having to test every possibility in the grid (unlike GridSearchCV).

Given param_grid, the best Random Forest Classifier model has these hyperparameters:
**{'n_estimators': 300, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_features': 'log2', 'max_depth': None, 'criterion': 'gini','class_weight': 'balanced', 'bootstrap': True}**

Here are the Random Forest Classifier model's performance values:
**Confusion Matrix:**
**[[129 12]**
**[ 52 7]]**

**Accuracy: 0.68**
**Precision: 0.37**
**Recall: 0.12**
**f1-score: 0.18**

The accuracy shows us that 68% of the model's predictions on the test data were correct.
The precision signifies that only 37% of predicted Class 1 instances were correct.
The low recall (12%) suggests that the model is missing a lot of Class 1 cases.
The f1-score indicates that the overall performance for Class 1 is weak.

Therefore, the Random Tree Classifier model is also not distinguishing Class 1 (churned) well.

| **Decision Tree Classifier** | **Random Forest Classifier** |
|---|---|
| **Confusion Matrix:**<br>**[[121 20]**<br>**[ 46 13]]**<br>**Accuracy: 0.67**<br>**Precision: 0.39**<br>**Recall: 0.22**<br>**f1-score: 0.28** | **Confusion Matrix:**<br>**[[129 12]**<br>**[ 52 7]]**<br>**Accuracy: 0.68**<br>**Precision: 0.37**<br>**Recall: 0.12**<br>**f1-score: 0.18** |

For predicting Class 0 (non-churned) cases, the Random Forest Classifier performed better than the Decision Tree Classifier. For predicting Class 1 (churned) cases, the Random Forest Classifier model performed worse than the Decision Tree Classifier. A few reasons may contribute to this.

Class 1 (churned) has fewer samples, so the model sees fewer examples of it. This leads to the model having difficulties in learning its patterns. Splitting imbalanced data into many Decision Trees (Random Forest) will amplify this issue, since a split's class distribution may be slightly different.

With only 1000 rows, each fold in cross-validation has even fewer training examples. The Random Forest Classifier relies on bootstrapping subsets using the small dataset. This leads to the subsets' data overlapping a lot and makes the trees less diverse.

Random Forests are used with large datasets. They reduce variance by averaging many trees. On a small dataset, this averaging can hurt performance, especially if the trees don't capture the minority class well. A single Decision Tree can be tuned to work well on small datasets, while a Random Forest is much harder to properly adjust for a small dataset.

Here are the feature importances from the best Random Forest Classifier model we trained:

- **Watch_Time_Hours (16.5%)** – Higher engagement (watch time) may correlate with retention or churn risk.
- **Number_of_Logins (15.4%)** – More logins could indicate a more active user, potentially reducing churn.
- **Subscription_Length_Months (13.8%)** – Longer subscriptions might suggest customer loyalty, reducing churn likelihood.
- **Age (13.6%)** – Could indicate different usage behaviors across age groups.
- **Resolution_Time_Days (12.8%)** – Might represent customer support response time, affecting user satisfaction.
- **Number_of_Complaints (9.8%)** – More complaints could correlate with dissatisfaction and potential churn.
- **Membership_Type (4.2%)** – Different membership tiers may affect retention differently.
- **Payment Methods (1.7–1.8%)** – The mode of payment has minimal impact.
- **Preferred Content Type (1.5–2%)** – Interest in TV shows, movies, sports, or documentaries has little effect on prediction.

***Task 4: Business Insights and Recommendations***

The characteristics that contribute the most to customer churn for the streaming service StreamFlex are **watch time (hours), number of logins and subscription length**.

First, StreamFlex could increase engagement through content recommendations. Personalized content recommendations based on viewing history may lead to users to spend more time on the platform. Second, StreamFlex could encourage frequent logins using notifications and gamification. Finally, StreamFlex could promote discounted long-term subscription plans for users that are most likely to churn. We suggest 3 business strategies:

**Loyalty & Reward Program**

- Introduce a point-based system where users earn rewards for consistent engagement (e.g., watching a set number of hours per week).
- Offer tiered benefits like early access to new releases or discounts on premium content.

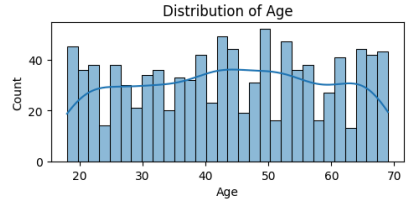
**AI-Driven Personalized Engagement**

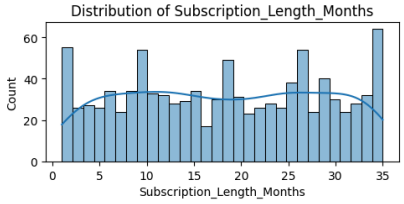
- Implement an AI-based recommendation engine that dynamically suggests content to users based on their watch history and preferences.
- Use machine learning to detect inactivity and trigger re-engagement campaigns with customized offers.

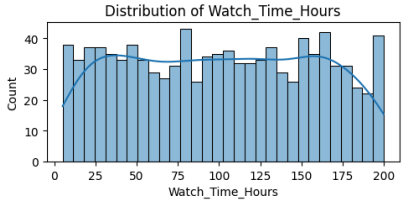
**Flexible & Incentivized Subscription Plans**

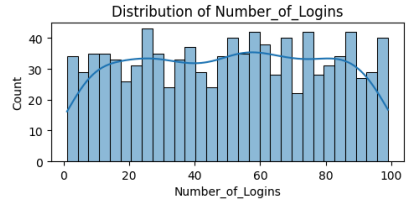
- Provide discounts for users switching from monthly to quarterly or annual plans.
- Introduce a "pause" option instead of cancellation, allowing users to return without feeling locked in.

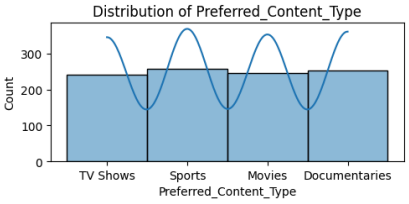
## ***Appendix 1: Histograms***

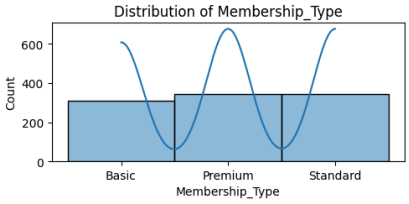


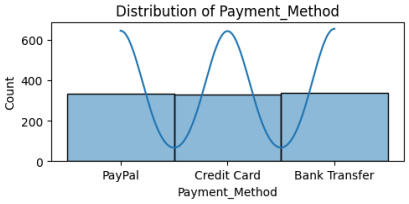



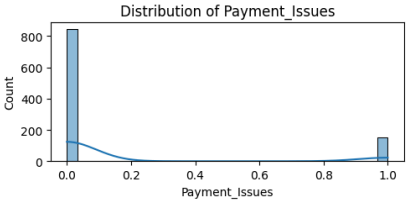



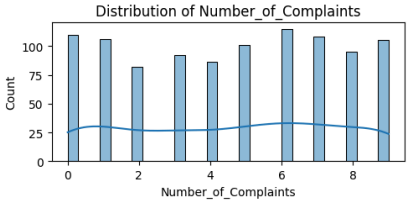



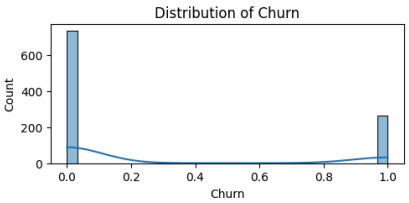



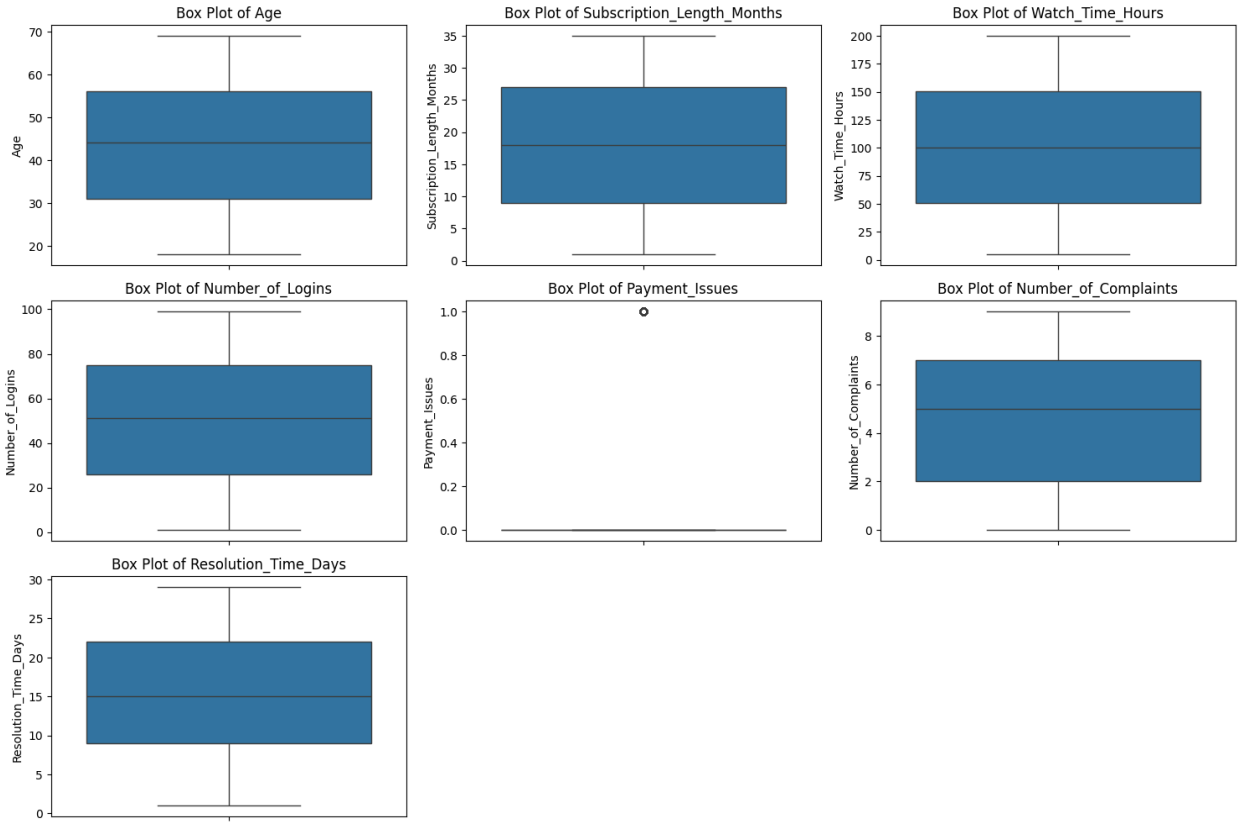







## ***Appendix 2: Box Plots***



## ***Appendix 3: Correlation Heatmap***